bhv®

Александр Леоненков

Нечеткое моделирование в среде MATLAB и fuzzyTECH

- Основы теории нечетких множеств и нечеткой логики
- Построение нечетких моделей в среде MATLAB Fuzzy Logic Toolbox
- Создание проектов в пакете fuzzyTECH

Александр Леоненков

Нечеткое моделирование в среде MATLAB и fuzzyTECH

Санкт-Петербург
«БХВ-Петербург»
2003

УДК 681.3.068
ББК 32.973.26-018.2
Л47

Леоненков А. В.

Л47 Нечеткое моделирование в среде MATLAB и fuzzyTECH. — СПб.: БХВ-Петербург, 2003. — 736 с.: ил.

ISBN 5-94157-087-2

В книге рассматриваются основы нечеткого моделирования — нового направления применения наукоемких технологий для решения практических задач. Подробно описываются базовые понятия теории нечетких множеств и нечеткой логики, необходимые для построения нечетких моделей систем в технике и экономике (в т. ч. бизнесе). Исследуются особенности нечеткого моделирования в средах MATLAB и fuzzyTECH. Изложение сопровождается примерами разработки отдельных нечетких моделей и иллюстрациями выполнения всех необходимых операций с нечеткими множествами.

Для системных аналитиков, программистов и студентов вузов

УДК 681.3.068
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Григорий Добин*
Редактор	*Анатолий Хрипов*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Оформление серии	*Via Design*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.04.03.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 59,34.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в ФГУП ордена Трудового Красного Знамени "Техническая книга"
Министерства Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций.
198005, Санкт-Петербург, Измайловский пр., 29.

ISBN 5-94157-087-2

© Леоненков А. В., 2003
© Оформление, издательство "БХВ-Петербург", 2003

Содержание

Предисловие **1**

Структура книги 3
Рекомендации по изучению материала книги 5
Благодарности 6

Часть I. Основы теории нечетких множеств и нечеткой логики **9**

Глава 1. Введение **11**

1.1. История развития теории и приложений нечетких множеств и нечеткой логики 12
Первые промышленные приложения в Европе 12
Япония — лидер в области промышленных приложений 13
Европа и США преследуют Японию 14
1.2. Методология системного моделирования 15
Анализ проблемной ситуации 19
Структуризация предметной области и построение модели 20
Выполнение вычислительных экспериментов с моделью 21
Применение результатов вычислительных экспериментов 22
Коррекция или доработка модели 23
1.3. Методология нечеткого моделирования 24
1.4. Анализ нечеткого и вероятностного подходов к моделированию неопределенности 26
Стохастическая неопределенность 27
Лингвистическая неопределенность 28
Моделирование лингвистической неопределенности 29
Нечеткая логика в сравнении с теорией вероятностей 30

Глава 2. Основные понятия теории нечетких множеств **33**

2.1 Определение нечеткого множества 33
2.2. Основные характеристики нечетких множеств 43
2.3. Основные типы функций принадлежности 52
Кусочно-линейные функции принадлежности 52
Z-образные и *S*-образные функции принадлежности 54
П-образные функции принадлежности 60
2.4. Некоторые рекомендации по построению функций принадлежности нечетких множеств 63
Прямые методы построения функций принадлежности 64
Косвенные методы построения функций принадлежности 65

Глава 3. Операции над нечеткими множествами 67

3.1. Равенство и доминирование нечетких множеств 68
3.2. Операции пересечения, объединения и разности нечетких множеств 70
3.3. Альтернативные операции пересечения и объединения нечетких множеств 79
Нечеткие операторы 89
3.4. Некоторые дополнительные операции над нечеткими множествами 93

Глава 4. Нечеткие отношения 99

4.1. Нечеткое отношение и способы его задания 99
Способы задания нечетких отношений 101
4.2. Основные характеристики нечетких отношений 110
4.3. Операции над нечеткими отношениями 114
Композиция бинарных нечетких отношений 118
4.4. Нечеткое отображение 124
Принцип обобщения в теории нечетких множеств 125
4.5. Свойства бинарных нечетких отношений, заданных на одном универсуме 126
Операция транзитивного замыкания бинарного нечеткого отношения 128
4.6. Некоторые специальные виды нечетких бинарных отношений, заданных на одном базисном множестве 131

Глава 5. Нечеткая и лингвистическая переменные. Нечеткие величины, числа и интервалы 134

5.1. Определения нечеткой и лингвистической переменных 134
5.2. Нечеткие величины, числа и интервалы 137
Операции над нечеткими числами и интервалами 139
5.3. Нечеткие числа и интервалы в форме $(L\text{-}R)$-функций 141
Операции над нечеткими числами и интервалами $(L\text{-}R)$-типа 145
5.4. Треугольные нечеткие числа и трапециевидные нечеткие интервалы 148
Операции над треугольными нечеткими числами и трапециевидными нечеткими интервалами 152

Глава 6. Основы нечеткой логики 158

6.1. Понятие нечеткого высказывания и нечеткого предиката 159
Нечеткие предикаты 161
6.2. Основные логические операции с нечеткими высказываниями 162
Логическое отрицание нечетких высказываний 162
Логическая конъюнкция нечетких высказываний 163
Логическая дизъюнкция нечетких высказываний 164
Нечеткая импликация 165
Нечеткая эквивалентность 167
6.3. Правила нечетких продукций 167
Прямой и обратный методы вывода заключений в системах нечетких продукций 171

Глава 7. Системы нечеткого вывода ... 178

7.1. Базовая архитектура систем нечеткого вывода ... 178
Нечеткие лингвистические высказывания ... 179
Правила нечетких продукций в системах нечеткого вывода ... 181
Механизм или алгоритм вывода в системах нечеткого вывода ... 185
7.2. Основные этапы нечеткого вывода ... 185
Формирование базы правил систем нечеткого вывода ... 187
Фаззификация (Fuzzification) ... 189
Агрегирование (Aggregation) ... 191
Активизация (Activation) ... 192
Аккумуляция (Accumulation) ... 195
Дефаззификация (Defuzzification) ... 197
Метод центра тяжести ... 197
Метод центра тяжести для одноточечных множеств ... 198
Метод центра площади ... 199
Метод левого модального значения ... 200
Метод правого модального значения ... 200
7.3. Основные алгоритмы нечеткого вывода ... 201
Алгоритм Мамдани (Mamdani) ... 202
Алгоритм Цукамото (Tsukamoto) ... 202
Алгоритм Ларсена (Larsen) ... 203
Алгоритм Сугено (Sugeno) ... 204
Упрощенный алгоритм нечеткого вывода ... 205
7.4. Примеры использования систем нечеткого вывода в задачах управления ... 205
Нечеткая модель управления смесителем воды при принятии душа ... 208
Содержательная постановка задачи ... 208
Построение базы нечетких лингвистических правил ... 209
Фаззификация входных переменных ... 209
Нечеткая модель управления кондиционером воздуха в помещении ... 212
Содержательная постановка задачи ... 212
Построение базы нечетких лингвистических правил ... 214
Фаззификация входных переменных ... 215
Нечеткая модель управления контейнерным краном ... 218
Содержательная постановка задачи ... 218
Формирование базы правил систем нечеткого вывода ... 219
Фаззификация входных переменных ... 220

Глава 8. Язык нечеткого управления — FCL ... 222

8.1. Концептуальные основы нечеткого управления ... 222
Интеграция программируемых контроллеров ... 226
Перенос программ нечеткого управления ... 228
История разработки и стандартизации языка FCL ... 229
8.2. Базовая нотация языка нечеткого управления FCL ... 230
Основные элементы языка FCL ... 230
Нотация правил продукций ... 230
Ключевые слова языка FCL ... 232
Интерфейс функционального блока (Function Block interface) ... 234
Фаззификация (Fuzzification) ... 235
Дефаззификация (Defuzzification) ... 237

Блок правил (Rule block) ... 239
Простой пример записи модели нечеткого управления с использованием нотации языка FCL ... 243
Необязательные параметры (Optional parameters) ... 244
Согласованность классов языка FCL ... 244
Список проверки данных ... 248
8.3. Пример разработки и записи нечетких моделей на языке FCL ... 250
Нечеткая модель управления смесителем воды при принятии душа ... 250
Нечеткая модель управления кондиционером воздуха в помещении ... 251
Нечеткая модель управления контейнерным краном ... 253

Глава 9. Основы общей теории нечеткой меры ... 256

9.1. Нечеткие меры и их основные свойства ... 256
Общее определение нечеткой меры ... 257
Меры доверия и правдоподобия ... 258
Меры возможности, необходимости и вероятности ... 259
λ-нечеткие меры ... 261
Классификация пространств с нечеткими мерами ... 262
9.2. Нечеткий интеграл и примеры его вычисления ... 263

Глава 10. Нечеткие сети Петри ... 267

10.1. Базовый формализм классических сетей Петри ... 268
Свойства сетей Петри и задачи их анализа ... 276
10.2. Основные подклассы нечетких сетей Петри ... 280
Нечеткие сети Петри типа V_f ... 280
Нечеткие сети Петри типа C_f ... 286
Обобщенные нечеткие временные сети Петри типа C_{PT}^f ... 291
Свойства нечетких сетей Петри ... 298
Классификация нечетких сетей Петри ... 300
10.3. Использование нечетких сетей Петри для представления правил нечетких продукций ... 303

Часть II. Нечеткое моделирование в среде MATLAB ... 309

Глава 11. Общая характеристика программы MATLAB ... 311

11.1. Основные элементы системы MATLAB ... 311
Особенности инсталляции системы MATLAB на компьютер пользователя ... 312
Запуск системы MATLAB и элементы ее графического интерфейса ... 313
Встроенная справочная система и документация, поставляемая с системой MATLAB ... 319
11.2. Основные приемы работы в системе MATLAB ... 321
Назначение операций главного меню ... 322
Назначение операций панели инструментов ... 326
Основные приемы работы в окне команд ... 327
11.3. Графические возможности системы MATLAB ... 334

Глава 12. Процесс нечеткого моделирования в среде MATLAB 343

12.1. Процесс разработки системы нечеткого вывода в интерактивном режиме 343
Редактор систем нечеткого вывода FIS 344
Редактор функций принадлежности 349
Редактор правил системы нечеткого вывода 352
Программа просмотра правил системы нечеткого вывода 354
Программа просмотра поверхности системы нечеткого вывода 356
12.2. Пример разработки системы нечеткого вывода в интерактивном режиме 358
12.3. Процесс разработки системы нечеткого вывода в режиме командной строки 371

Глава 13. Нечеткая кластеризация в Fuzzy Logic Toolbox 379

13.1. Общая характеристика задач кластерного анализа 379
13.2. Задача нечеткой кластеризации и алгоритм ее решения 380
Общая формальная постановка задачи нечеткого кластерного анализа 381
Уточненная постановка задачи нечеткой кластеризации 383
Алгоритм решения задачи нечеткой кластеризации методом нечетких с-средних 385
13.3. Средства решения задачи нечеткой кластеризации в пакете Fuzzy Logic Toolbox 387
Решение задачи нечеткой кластеризации в командном режиме 387
Решение задачи нечеткой кластеризации с использованием средств графического интерфейса 392
Решение задачи определения числа кластеров для нечеткой кластеризации в системе MATLAB 395

Глава 14. Основы программирования в среде MATLAB 399

14.1. Основы языка программирования системы MATLAB 399
Операторы управления последовательностью выполнения команд 404
Условный оператор if…elseif…else…end 404
Оператор выбора switch…case…otherwise…end 405
Оператор цикла for…end 406
Оператор цикла while…end 407
Оператор continue 408
Оператор break 408
Оператор return 409
Защищенный блок try...catch…end 410
Текстовые комментарии 411
14.2. Основные приемы работы с редактором/отладчиком m-файлов 411
Назначение операций главного меню 413
Назначение операций панели инструментов 418
14.3. Пример программы, расширяющей возможности пакета нечеткой логики Fuzzy Logic Toolbox 420

Глава 15. Основы нечетких нейронных сетей ... 426

15.1. Общая характеристика ANFIS — адаптивных систем нейро-нечеткого вывода ... 427
Понятие нейронной сети и основные способы ее задания ... 427
Гибридная сеть как адаптивная система нейро-нечеткого вывода ... 432
15.2. Реализация ANFIS в среде MATLAB ... 432
15.3. Пример решения задачи нейро-нечеткого вывода ... 442

Глава 16. Примеры разработки нечетких моделей управления в среде MATLAB ... 451

16.1. Нечеткая модель управления кондиционером воздуха в помещении ... 451
16.2. Нечеткая модель управления контейнерным краном ... 457

Глава 17. Примеры разработки нечетких моделей принятия решений в среде MATLAB ... 464

17.1 Оценивание финансовой состоятельности клиентов при предоставлении банковских кредитов ... 464
Содержательная постановка задачи оценивания финансовой состоятельности клиентов ... 464
Описание входных и выходных переменных рассматриваемой задачи ... 465
Нечеткая модель оценивания финансовой состоятельности клиентов ... 468
Фаззификация входных и выходных переменных ... 469
Формирование базы правил систем нечеткого вывода ... 471
Построение нечеткой модели средствами Fuzzy Logic Toolbox и анализ полученных результатов ... 473
17.2. Анализ и прогнозирование валютных цен на финансовом рынке ... 479

Часть III. Нечеткое моделирование в среде fuzzyTECH ... 489

Глава 18. Общая характеристика программы fuzzyTECH ... 491

18.1. Общая характеристика нечеткого проекта в среде fuzzyTECH ... 492
18.2. Основные элементы рабочего интерфейса программы fuzzyTECH ... 497
Встроенная справочная система программы fuzzyTECH ... 503
18.3. Назначение операций главного меню и панели инструментов программы fuzzyTECH ... 506
Назначение операций главного меню ... 506
Назначение операций панели инструментов ... 516
18.4. Графические средства визуализации результатов нечеткого вывода в программе fuzzyTECH ... 518
Графическое окно просмотра поверхности нечеткого вывода на плоскости ... 518
Графическое окно просмотра трехмерной поверхности нечеткого вывода ... 521
Графическое окно просмотра временных графиков значений лингвистических переменных ... 524

Глава 19. Процесс нечеткого моделирования в среде fuzzyTECH 527

19.1. Основные средства редактирования и анализа систем нечеткого вывода в fuzzyTECH 527
Графический редактор лингвистической переменной и функций принадлежности их термов 528
Графические редакторы правил системы нечеткого вывода 533
Графические средства анализа результатов нечеткого вывода 539
19.2. Основные средства разработки проектов и компонентов систем нечеткого вывода в fuzzyTECH 543
Мастер нечеткого проекта 543
Мастер лингвистической переменной 549
Мастер блока правил 554

Глава 20. Примеры разработки и анализа нечетких моделей в среде fuzzyTECH 559

20.1. Пример разработки системы нечеткого вывода для задачи «Чаевые в ресторане» 559
20.2. Нечеткая модель управления контейнерным краном 568
20.3. Нечеткая модель оценивания финансовой состоятельности клиентов при предоставлении банковских кредитов 573

Часть IV. Приложения 583

Приложение 1. Основы классической теории множеств и отношений 585

Множество и способы его задания 585
Основные теоретико-множественные операции 593
Булеан или множество всех подмножеств 599
Мультимножество или комплект 600
Отношения и способы их задания 601
Операции над бинарными отношениями 607
Отображение 609
Свойства бинарных отношений, заданных на одном базисном множестве 610
Некоторые специальные виды бинарных отношений, заданных на одном базисном множестве 612
Отношение строгого частичного порядка 613
Отношение толерантности 613
Отношение эквивалентности 614
Мультиотношение 614

Приложение 2. Основы математической логики 616

Классическая логика высказываний 617
Основные понятия логики высказываний 617
Основные логические операции над высказываниями 619
Формальные теории 624
Исчисление высказываний как формальная теория 625

Логика предикатов 628
Основные понятия логики предикатов первого порядка 629
Логические операции над предикатами 630
Кванторы логики предикатов 631
Исчисление предикатов первого порядка как формальная теория 632
Продукционные системы 635
Прямой и обратный методы вывода заключений в продукционных системах 637

Приложение 3. Справочник функций пакета Fuzzy Logic Toolbox системы MATLAB 641

Приложение 4. Пример файла проекта для программы fuzzyTECH 705

Глоссарий 713

Литература 717

Предисловие

Книга посвящена рассмотрению теоретических основ и прикладных методов новой современной технологии — нечеткого моделирования — в контексте решения практических задач с использованием специализированных программных средств MATLAB и fuzzyTECH. Основная цель предлагаемого вниманию читателей учебного пособия — привлечь внимание студентов, аспирантов, преподавателей, инженеров, молодых научных сотрудников и программистов к нечеткой проблематике и дать доступное введение в одну из интереснейших и перспективных областей современных высоких технологий — нечеткое моделирование.

Чем же обусловлена актуальность этой новой технологии и в чем проявляется ее преимущество перед известными и ставшими уже классическими концепциями моделирования и управления?

Прежде всего — это тенденция увеличения сложности математических и формальных моделей реальных систем и процессов управления, связанная с желанием повысить их адекватность и учесть все большее число различных факторов, оказывающих влияние на процессы принятия решений.

С одной стороны, традиционные методы построения моделей не приводят к удовлетворительным результатам, когда исходное описание подлежащей решению проблемы заведомо является неточным или неполным. С другой стороны, стремление получить всю исчерпывающую информацию для построения точной математической модели сколь-нибудь сложной реальной ситуации может привести к потере времени и средств, поскольку это может быть в принципе невозможно.

В подобных случаях наиболее целесообразно воспользоваться такими методами, которые специально ориентированы на построение моделей, учитывающих неполноту и неточность исходных данных. Именно в таких ситуациях технология нечеткого моделирования оказывается наиболее конструктивной, поскольку за последнее десятилетие на ее основе были решены сотни практических задач управления и принятия решений.

Сейчас уже не вызывает сомнения тот факт, что важнейшей особенностью жизнеспособности той или иной теоретической концепции является ее реализация и поддержка в соответствующих программных инструментах. Появление и успешное развитие коммерческих программных средств, которые специально ориентированы на решение задач нечеткого моделирования, объективно свидетельствуют в пользу того, что теория нечетких множеств и нечеткая логика могут и должны быть эффективно использованы для решения широкого круга практических задач. При этом наиболее интересными программными средствами, в которых реализована технология нечеткого моделирования, по мнению автора, являются рассмотренные в книге система MATLAB и программа fuzzyTECH.

К сожалению, существующие издания по нечеткой проблематике либо излишне упрощены и поверхностны, что характерно, в первую очередь, для информации, представленной в Интернете, либо содержат абстрактное изложение отдельных, зачастую весьма узких аспектов теории нечетких множеств и различных ее направлений, что характерно для академических работ последних трех десятилетий.

С одной стороны, упрощенное изложение теории нечетких множеств и нечеткой логики на уровне картинок создает несерьезное отношение к ней со стороны профессиональных математиков и программистов, препятствуя внедрению соответствующих идей в процесс их подготовки и обучения. С другой стороны, тенденция перевести все идеи современной математики на язык теории нечетких множеств привела к появлению целого ряда работ, содержащих абстрактное обобщение тех или иных математических конструкций, которые оказались оторванными от проблематики реальных практических задач системного моделирования. По целому ряду причин эта тенденция также не нашла широкого признания в среде математиков и программистов.

В настоящей книге представлен материал, который тщательно отобран из большого многообразия идей и работ, получивших развитие в последние три десятилетия. При отборе материала автор руководствовался, главным образом, возможностью конструктивного применения соответствующих идей на практике. При этом изложение материала не является поверхностным в ущерб математической строгости, ибо, по мнению автора, теория нечетких множеств продолжает оставаться разделом математики, а нечеткая логика — разделом математической логики. Нечеткая математика не может и не должна излагаться нечетким языком. С этой целью в книге приводится теоретический материал, который необходим для адекватного понимания всех основных идей нечеткого моделирования.

В то же время конструктивное восприятие идей нечеткого моделирования возможно посредством построения и анализа нечетких моделей конкретных практических задач. С этой целью в книге рассматривается достаточное количество прикладных задач, которые не только иллюстрируют особенности реализации тех или иных идей нечеткого моделирования, но и могут быть эффективно решены с использованием соответствующих программных инструментов — MATLAB и fuzzyTECH. При этом целый ряд представленных задач имеет оригинальный характер, в чем читатель сможет убедиться самостоятельно при чтении книги.

При изложении материала невольно возникает проблема унификации терминологии и обозначений, традиционно применяемых в различных работах по теории нечетких множеств. Как представляется автору, целый ряд таких обозначений не являются вполне удобными и не отражают внутреннюю логику рассматриваемых понятий, хотя исторически применялись в целом ряде оригинальных работ. Это относится, например, к использованию символов нижнего подчеркивания, суммы и интеграла для обозначения нечетких множеств. Поскольку эти обозначения способны привести к путанице и трудностям у начинающих читателей, они не используются в книге, а заменены на более привычные теоретико-множественные, которые наиболее точно отражают обобщенный характер нечетких понятий по сравнению с понятиями классической математики.

Структура книги

В основу книги положены две основные идеи. С одной стороны, познакомить читателя с теоретическими основами новой концепции нечеткого моделирования сложных систем, которая может быть конструктивно использована для построения нечетких моделей и без понимания которой вряд ли возможно адекватно использовать богатейший потенциал возможностей соответствующих программных инструментов.

С другой стороны, рассмотреть основные программные инструменты, которые за последние несколько лет оказались наиболее эффективными при решении практических задач с использованием технологии нечеткого моделирования.

Материал книги делится на три части. Первая часть знакомит с основными теоретическими понятиями, которые необходимы для правильного понимания базовой терминологии нечеткого моделирования и возможностей соответствующей технологии в контексте решения прикладных задач. Здесь представлен теоретический материал, необходимый для уяснения всех основных понятий теории нечетких множеств и нечеткой логики. При этом материал излагается независимо от программных инструментов, следуя логической и исторической традиции развития соответствующих научных направлений.

Вторая часть посвящена рассмотрению особенностей процесса нечеткого моделирования с использованием возможностей одной из наиболее мощных и универсальных систем компьютерной математики — системы MATLAB. Третья часть содержит описание особенностей процесса нечеткого моделирования с использованием специального программного инструментария — программы fuzzyTECH.

В *главе 1* рассматриваются особенности современного состояния нечеткого моделирования в контексте общих концепций системного моделирования, приводится исторический обзор развития методологии нечеткого моделирования сложных систем. Здесь можно познакомиться с сущностью проблемы неопределенности и основными подходами ее количественного анализа.

В *главе 2* рассматриваются основные понятия теории нечетких множеств и их связь с определением классического множества, описываются все основные типы функций принадлежности, их аналитическое и графическое представления, а также приводятся примеры различных нечетких множеств.

В *главе 3* рассматриваются операции над нечеткими множествами и основные способы их определения в контексте классических теоретико-множественных операций. При этом результаты выполнения операций над нечеткими множествами иллюстрируются целым рядом примеров.

В *главе 4* рассматриваются нечеткие отношения и операции над нечеткими отношениями, а также описывается ряд свойств нечетких отношений, которые позволяют определить нечеткое разбиение и нечеткий порядок. Приводятся примеры конкретных нечетких отношений, возникающих в экономике, бизнесе и в быту.

В *главе 5* рассматриваются нечеткая и лингвистическая переменные, которые используются в дальнейшем при определении понятий нечеткой логики, а также описываются основные операции над нечеткими числами и интервалами, которые иллюстрируются различными примерами.

В *главе 6* излагаются теоретические основы нечеткой логики, рассматриваются основные операции с нечеткими высказываниями в контексте классической логики, приводятся примеры нечетких высказываний и выполнения нечетко-логических операций.

В *главе 7* рассматриваются нечеткий вывод и системы нечеткого вывода, определяются различные способы вычисления степени истинности нечеткой импликации, приводятся примеры использования нечеткого вывода в системном моделировании.

В *главе 8* приводится описание базовой нотации языка нечеткого управления FCL в соответствии со Стандартом IEC 1131-7. Здесь также рассматриваются основные понятия теории нечеткого управления и приводятся примеры записи нечетких моделей управления в нотации языка FCL.

В *главе 9* рассматриваются основы теории возможностей, которая представляет одно из перспективных направлений развития технологии нечеткого моделирования. Приводится общее определение нечеткой меры и ее разновидностей — мер возможности, необходимости и вероятности в контексте адекватного представления неопределенности.

В *главе 10* рассматривается аппарат нечетких сетей Петри, который также представляется перспективным направлением развития технологии нечеткого моделирования, описываются особенности формального и графического представлений, приводятся примеры задач и построение нечетких моделей систем с использованием формализма нечетких сетей Петри.

В *главе 11* представлена общая характеристика системы компьютерной математики MATLAB и ее пакета расширения — Fuzzy Logic Toolbox, предназначенного для построения систем нечеткого вывода, а также рассматриваются графический интерфейс и основные функции режима команд системы MATLAB.

В *главе 12* описывается процесс построения нечетких моделей в среде MATLAB и соответствующие графические средства пакета Fuzzy Logic Toolbox, приводятся простейшие примеры нечеткого моделирования.

В *главе 13* рассматриваются теоретические основы нечеткой кластеризации и особенности реализации соответствующих алгоритмов в системе MATLAB, приводятся примеры решения задач нечеткой кластеризации в среде Fuzzy Logic Toolbox.

В *главе 14* излагаются основы программирования в среде MATLAB с использованием языка разработки m-файлов, рассматривается синтаксис этого языка и его основные конструкции, приводится пример разработки m-файла для реализации операций с нечеткими множествами, который расширяет возможности системы MATLAB.

В *главе 15* рассматривается новое направление нечеткого моделирования — адаптивные системы нейро-нечеткого вывода, приводится исходное определение нейронной сети и описываются особенности реализации адаптивных систем ANFIS в среде MATLAB.

В *главах 16* и *17* приводятся конкретные примеры построения нечетких моделей в среде MATLAB, предназначенные для решения задач управления и принятия решений, среди которых — нечеткие модели управления кондиционером воздуха в помещении, управления контейнерным краном в порту, оценивания финансовой состоятельности клиентов при предоставлении банковских кредитов и прогнозирования валютных цен на финансовом рынке.

В *главе 18* представлена общая характеристика программы fuzzyTECH, описаны особенности ее инсталляция и графического интерфейса, рассматриваются основные характеристики нечеткого проекта в среде fuzzyTECH.

В *главе 19* рассматриваются все специальные средства программы fuzzyTECH, предназначенные для разработки и анализа нечетких моделей, а также даются рекомендации по выполнению отдельных этапов нечеткого моделирования в среде fuzzyTECH.

В *главе 20* приводятся конкретные примеры построения нечетких моделей в среде fuzzyTECH, предназначенные для решения отдельных задач управления и принятия решений, среди которых — нечеткие модели поведения в ресторане, управления контейнерным краном в порту и оценивания финансовой состоятельности клиентов при предоставлении банковских кредитов.

В *приложении 1* рассматриваются основы классической теории множеств, которые необходимы для понимания базовой концепции теории нечетких множеств, являющейся обобщением и дальнейшим развитием описываемых здесь понятий.

В *приложении 2* рассматриваются основы математической логики, которые необходимы для понимания концепции систем нечеткого вывода, также являющейся обобщением и дальнейшим развитием описываемых здесь понятий.

В *приложении 3* приводится справочник функций пакета Fuzzy Logic Toolbox системы MATLAB для выполнения различных операций с системами нечеткого вывода.

В *приложении 4* приводится текст файла проекта системы нечеткого вывода в формате FTL для среды fuzzyTECH, который иллюстрирует дополнительные возможности спецификации нечетких проектов в среде fuzzyTECH.

Рекомендации по изучению материала книги

Представленный в книге материал охватывает всю основную проблематику методологии нечеткого моделирования и технологии решения практических задач с использованием наиболее эффективных программных инструментов. В то же

время для всестороннего понимания особенностей разработки и применения нечетких моделей, как правило, недостаточно общей эрудиции и наличия того или иного программного инструментария. Как показывает практический опыт, для творческого овладения методологией нечеткого моделирования необходима определенная математическая подготовка и знание некоторых общих принципов моделирования и управления, разработанных в рамках прикладного системного анализа.

Читатели, впервые приступающие к изучению нечеткого моделирования и ставящие перед собой цель в совершенстве овладеть данным предметом, могут последовательно знакомиться с материалом отдельных глав, обращаясь к приложениям по мере необходимости. При этом теоретический материал, изложенный в первой части книги, используется в дальнейшем при изложении конкретных нечетких моделей. Те из читателей, кто знаком с понятиями и проблематикой теории нечетких множеств и нечеткой логики, могут сразу перейти к рассмотрению особенностей реализации нечетких моделей в средах MATLAB и fuzzyTECH, обращаясь к основному теоретическому материалу по мере необходимости. Наконец, читатели, которые интересуются прикладными аспектами отдельных направлений нечеткого моделирования, такими, как адаптивные системы нейро-нечеткого вывода или методы нечеткой кластеризации, могут выборочно обратиться к материалу соответствующих глав.

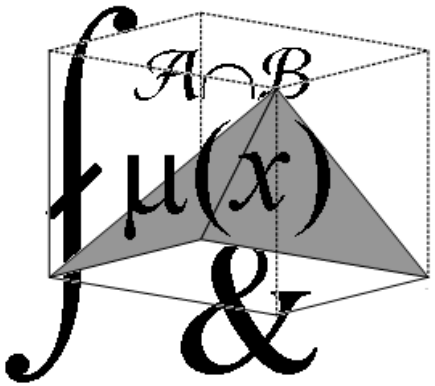
Материал книги может быть использован для постановки соответствующего учебного курса в вузах с целью подготовки специалистов математического, экономического и технического профиля. В этом случае автор надеется, что как студенты, так и преподаватели найдут в книге интересный для размышления материал, который позволит понять целый ряд особенностей и перспектив профессионального образования в области современных информационных технологий.

Благодарности

Автор искренне благодарит К. Н. Ильинского, Е. В. Кондукову, Е. В. Строганову, А. М. Коновалова и И. А. Корнеева за предоставленные в разное время материалы, которые были использованы при написании книги, а также доцента А. Н. Павлова за конструктивное обсуждение материала отдельных глав книги. Автор искренне признателен директору Школы IT-менеджмента АНХ при Правительстве РФ **(www.itmane.ru)** И. Ю. Прокиной, а также Л. А. Ермакову, В. А. Перекрестову и В. В. Фамильнову за оказанную поддержку в процессе работы над книгой. В предоставлении персональной лицензии и фирменной документации на систему MATLAB неоценимую помощь оказали Борис Манзон (SoftLine) и сотрудник компании MathWorks — Courtney Esposito.

Написание современной книги немыслимо без использования ресурсов Интернета. В этой связи хотелось бы выразить особую признательность директору Междисциплинарного Центра СПбГУ **(www.icape.nw.ru)** профессору Н. В. Борисову за предоставленную возможность электронной коммуникации.

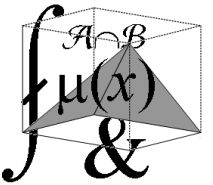
В заключение следует специально отметить одно немаловажное обстоятельство, которое усложняет понимание и распространение идей нечеткого моделирования среди отечественных математиков, инженеров и программистов. Речь идет о неустановившейся терминологии в этой области и о неоднозначности перевода отдельных терминов, имеющих зачастую многозначное толкование в том или ином конкретном контексте. С этой целью названия наиболее важных понятий и их краткая характеристика отдельно приводятся в конце книги. В любом случае автор будет признателен за все отзывы и конструктивные предложения, связанные с содержанием книги и проблематикой нечеткого моделирования, которые можно отправлять по адресу: **fuzzy@itmane.ru**.



ЧАСТЬ I

Основы теории нечетких множеств и нечеткой логики

Глава 1



Введение

Теория нечетких множеств, основные идеи которой были предложены американским математиком Лотфи Заде (Lotfi Zadeh) более 35 лет назад, позволяет описывать качественные, неточные понятия и наши знания об окружающем мире, а также оперировать этими знаниями с целью получения новой информации. Основанные на этой теории методы построения информационных моделей существенно расширяют традиционные области применения компьютеров и образуют самостоятельное направление научно-прикладных исследований, которое получило специальное название — *нечеткое моделирование*.

В последнее время нечеткое моделирование является одной из наиболее активных и перспективных направлений прикладных исследований в области управления и принятия решений. Нечеткое моделирование оказывается особенно полезным, когда в описании технических систем и бизнес-процессов присутствует неопределенность, которая затрудняет или даже исключает применение точных количественных методов и подходов.

В области управления техническими системами нечеткое моделирование позволяет получать более адекватные результаты по сравнению с результатами, которые основываются на использовании традиционных аналитических моделей и алгоритмов управления. Диапазон применения нечетких методов с каждым годом расширяется, охватывая такие области, как проектирование промышленных роботов и бытовых электроприборов, управление доменными печами и движением поездов метро, автоматическое распознавание речи и изображений.

Нечеткая логика, которая служит основой для реализации методов нечеткого управления, более естественно описывает характер человеческого мышления и ход его рассуждений, чем традиционные формально-логические системы. Именно поэтому изучение и использование математических средств для представления нечеткой исходной информации позволяет строить модели, которые наиболее адекватно отражают различные аспекты неопределенности, постоянно присутствующей в окружающей нас реальности.

1.1. История развития теории и приложений нечетких множеств и нечеткой логики

Первой публикацией по теории нечетких множеств принято считать работу профессора из Университета Беркли (шт. Калифорния, США) Лотфи Заде, которая относится к 1965 г. Понятие нечеткого множества в смысле Л. Заде положило начало новому импульсу в области математических и прикладных исследований, в рамках которых за короткий срок были предложены нечеткие обобщения всех основных теоретико-множественных и формально-логических понятий.

Наиболее значимыми из работ в этой области следует отметить публикации Л. Заде, Д. Дюбуа (D. Dubois) и А. Прада (H. Prade) по теории нечеткой меры и меры возможности, М.Сугено (M. Sugeno) по нечеткому выводу и нечеткому интегралу, Дж. Бездека (J. Bezdek) по нечеткой кластеризации и распознаванию образов, Р. Ягера (R. R.Yager) по нечеткой логике.

Однако, несмотря на большое количество теоретических работ, прикладное значение нечетких моделей долгое время ставилось под сомнение. Даже сегодня, когда имеется информация о многих десятках успешных применений нечетких моделей, некоторые ученые все еще скептически относятся к возможностям нечеткого моделирования.

Первые промышленные приложения в Европе

Первые реализации нечетких моделей в промышленности относятся к середине 1970-х гг. Именно в этот период в Великобритании Эбрахим Мамдани (Ebrahim Mamdani) использовал нечеткую логику для управления парогенератором. Решение этой задачи обычными методами было сопряжено с целым рядом трудностей вычислительного характера. Предложенный Э. Мамдани алгоритм, основанный на нечетком логическом выводе, позволил избежать чрезмерно большого объема вычислений и был по достоинству оценен специалистами. В этот же период нечеткие модели были применены при управлении печью для обжига цемента. Тем не менее, эти немногие приложения, использовавшие нечеткую логику, по существу скрывали этот факт, поскольку в них нечеткая логика называлась "многозначной логикой" или "непрерывной логикой".

В начале 1980-х гг. нечеткая логика и теория нечетких множеств получили свое дальнейшее развитие в целом ряде программных средств поддержки принятия решений и в экспертных системах анализа данных. Хотя многие из этих программных инструментариев так и не вышли за пределы научно-исследовательских лабораторий и институтов, в ходе их разработки были получены важные эмпирические результаты по моделированию с помощью нечеткой логики процессов человеческих рассуждений и принятия решений.

Япония — лидер в области промышленных приложений

После первых промышленных приложений в Европе Япония за короткий период времени вышла на первое место в мире по количеству устройств и механизмов, в которых были реализованы нечеткие технологии. Появление микропроцессоров и микроконтроллеров инициировало резкое увеличение бытовых приборов и промышленных установок с алгоритмами управления на основе нечеткой логики. В настоящее время в Японии запатентовано более чем 3000 соответствующих устройств в этой области. Слово "*фаззи*" (fuzzy) стало символом популярности и коммерческого успеха новых промышленных изделий в этой стране.

Имеется целый ряд обстоятельств, которые объясняют причины столь впечатляющей популярности нечеткой логики в Японии. Во-первых, нечеткая логика поддерживает разработку быстрого прототипа технического устройства с последующим усложнением его функциональности, что характерно для стиля работы японских инженеров. Во-вторых, нечеткая логическая модель более проста для понимания, чем аналогичная математическая модель на основе дифференциальных или разностных уравнений. В-третьих, нечеткие модели оказываются более простыми для своей аппаратной реализации по сравнению с классическими алгоритмами управления техническими системами.

В результате этого нечеткие технологии нашли свое применение в самых различных технических устройствах и бытовых приборах, выпускаемых японскими фирмами. Фотоаппараты и видеокамеры используют нечеткую логику, чтобы реализовать опыт фотографа в управлении этими устройствами. Например, компании Fisher и Sanyo производят нечеткие логические видеокамеры, в которых применяется нечеткая фокусировка и стабилизация изображения.

Компания Matsushita выпускает стиральную машину, в которой используются датчики и микропроцессоры с нечеткими алгоритмами управления. Датчики определяют цвет и вид одежды, количество твердых частиц, степень загрязнения, а нечеткий микропроцессор выбирает наиболее подходящую программу стирки из 600 доступных комбинаций температуры воды, количества стирального порошка и времени производственного цикла быстрого или медленного вращения и промывки.

Компания Mitsubishi объявила о выпуске первого в мире автомобиля, где управление каждой системой основано на нечеткой логике. При этом Mitsubishi также производит "нечеткий" кондиционер, который управляет изменением температуры и влажности в помещении согласно человеческому восприятию степени комфорта. Компания Nissan разработала "нечеткую" автоматическую трансмиссию и "нечеткую" противоскользящую тормозную систему и реализовала их в одном из своих последних автомобилей повышенной комфортности.

Японский город Сендай имеет метрополитен с 16 станциями, который управляется нечетким компьютером. При этом нечеткий компьютер регулирует процессы ускорения и торможения поездов метро, делая на 70% меньше ошибок, чем соответствующий человек-оператор.

На фондовом рынке Токио используется несколько трейдерных систем, основанных на нечеткой логике, которые превосходят по скоростным и динамическим характеристикам традиционные информационные системы. В Японии имеются также "нечеткие" системы управления уличным движением, "нечеткие" тостеры, "нечеткие" рисовые печи, "нечеткие" пылесосы и многие другие бытовые и технические устройства.

Европа и США преследуют Японию

Только в начале 1990-х гг. ведущие европейские корпорации поняли, что они практически уступили Японии одну из ключевых современных технологий. С этого времени были предприняты серьезные усилия наверстать упущенные возможности в этой области. Именно в этот период в Европе появилось более 200 видов промышленных изделий и устройств, в которых были реализованы нечеткие модели. Это были, главным образом, бытовые приборы, которые характеризовались более эффективной экономией электроэнергии и водопотребления без дополнительного увеличения цены изделия. Другие промышленные приложения относились к автоматизации производства, включая управление химическими и биологическими процессами, управление станками и сборочными конвейерами, а также различные интеллектуальные датчики.

Поскольку этим приложениям сопутствовал коммерческий успех, в настоящее время нечеткая логика рассматривается как стандартный метод проектирования и получила широкое признание среди инженеров и проектировщиков. К нечетким технологиям проявляют все больший интерес компании из США, особенно те из них, кто испытывает жесткую конкуренцию со стороны фирм из Азии и Европы. Тем не менее, для американских корпораций остались открытыми целые сегменты потребительского рынка.

Например, нечеткая логика оказалась превосходным инструментом для разработки систем управления внутренними компонентами персональных компьютеров, а также алгоритмов компрессии речи и видео. Так, например, в системной плате MSI K7T Pro 266 Master-R используется система интеллектуального разгона микропроцессора Fuzzy Logic™3, которая автоматически выбирает частоту системной шины и процессора в зависимости от температуры и рабочей нагрузки базовых компонентов персонального компьютера.

Известны приложения из области теле- и радиосвязи, направленные на устранение влияния отраженных ТВ-сигналов и радиосигналов. Предложены и реализованы программные алгоритмы для сетевой маршрутизации и распознавания речи на основе нечеткой логики. Следует учитывать и другое важное обстоятельство — в настоящее время в США развернуты серьезные исследования по нейро-сетевым технологиям. Все эксперты соглашаются с тем, что комбинация нейронных сетей и нечеткой логики будет следующим серьезным шагом в дальнейшем прогрессе высоких технологий.

Сегодня количество технических изделий и программных средств, включая новые патенты, продолжает быстро расти. Поэтому, чтобы остаться конкурентоспособными, многие американские компании начинают свои собственные внут-

ренние нечеткие проекты. Хотя информации о подобных проектах недостаточно, можно отметить ассигнования Министерства обороны США на исследования в области построения систем управления вооружением и тренажеров для обучения пилотов истребителей на основе нечетких технологий. Национальное управление по аэронавтике и космонавтике (НАСА) предполагает использовать нечеткие модели для решения специальных задач в космосе.

Таким образом, можно сделать вывод, что область приложений теории нечетких множеств и нечеткой логики с каждым годом продолжает неуклонно расширяться. При этом процесс разработки и применения нечетких моделей тесно взаимосвязан с концепцией системного моделирования как наиболее общей методологией построения и использования информационных моделей сложных систем различной физической природы. Именно поэтому изложению методов нечеткого моделирования предшествует рассмотрение основных особенностей методологии системного моделирования, в контексте которой возможна разработка наиболее адекватных и эффективных информационных моделей сложных систем.

1.2. Методология системного моделирования

Системный анализ и системное моделирование имеют более давнюю историю, чем теория нечетких множеств. Центральным понятием системного моделирования является само понятие *система*, под которой понимается совокупность объектов, компонентов или элементов произвольной природы, образующих некоторую целостность в том или ином контексте. Определяющим принципом рассмотрения некоторой совокупности объектов как системы является появление у нее новых свойств, которых не имеют составляющие ее элементы. Значимость этого принципа проявляется в том, что он получил даже специальное название — принцип *эмерджентности* (от *англ.* emergence — появление, выявление).

Системы различной физической природы окружают нас повсеместно — это и конкретные предметы и объекты: солнечная система, человек, персональный компьютер, автомобиль, самолет, аэропорт. Это и более абстрактные сущности, такие как компьютерная программа, естественный язык, коммерческая фирма, культура, политика, наука, экономика. Наиболее ортодоксальная точка зрения предполагает, что все окружающие нас предметы являются системами.

При рассмотрении той или иной системы исходным этапом ее изучения является определение ее *границы*. Речь идет о необходимости разделения всех элементов на два класса: принадлежащих и не принадлежащих системе. При этом те сущности или объекты, которые собственно принадлежат системе, и будут являться ее элементами. Напротив, не принадлежащие системе объекты, но оказывающие на нее то или иное влияние, образуют *среду* или внешнюю по отношению к системе предметную область. Традиционно одним из принципов системного анализа являлось предположение о том, что граница системы *четко* разделяет элементы системы и ее внешнюю среду (рис. 1.1).

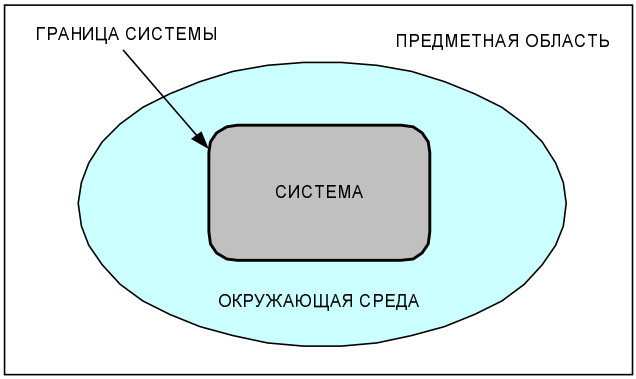


Рис. 1.1. Общее представление системы и окружающей среды в контексте традиционного системного анализа

Важнейшими характеристиками любой системы являются ее структура и процесс функционирования. Под *структурой системы* понимают устойчивую во времени совокупность взаимосвязей между ее элементами или компонентами. Именно структура системы связывает воедино все элементы и препятствует распаду системы на отдельные компоненты. Структура системы может отражать самые различные взаимосвязи, в том числе, и вложенность элементов одной системы в другую. В этом случае принято называть более мелкую или вложенную систему *подсистемой*, а более крупную систему — *метасистемой*.

Процесс функционирования системы тесно связан с изменением свойств системы или отдельных ее элементов во времени. При этом важной характеристикой системы является ее *состояние*, под которым понимается совокупность свойств или признаков системы, которые в каждый момент времени отражают наиболее существенные особенности поведения системы.

Рассмотрим следующий пример. В качестве системы рассмотрим такой объект, как "Автомобиль". Границы этой системы четко ограничены теми компонентами, которые размещаются в корпусе отдельного автомобиля. При этом такой объект, как двигатель является элементом системы "Автомобиль". С другой стороны, двигатель сам является системой, которая состоит из отдельных компонентов, таких как блок цилиндров, свечи зажигания и др. Поэтому система "Двигатель" в свою очередь является подсистемой системы "Автомобиль". Система охлаждения двигателя и система электрооборудования также будут являться подсистемами "Автомобиль".

Структура системы может быть описана с разных точек зрения. Наиболее общее представление о структуре дает схема устройства той или иной системы. При этом взаимодействие элементов может носить не только механический, электрический или биологический характер, но и информационный, что характерно для современных организационно-технических систем. Состояние системы также можно рассматривать с различных точек зрения, наиболее общей из которых

является рассмотрение особенностей функционирования или эксплуатации той или иной системы.

Процесс функционирования системы отражает поведение системы во времени и может быть представлен как последовательное изменение ее состояний. Если система изменяет одно свое состояние на другое состояние, то принято говорить, что система *переходит* из одного состояния в другое. Совокупность признаков или условий изменения состояний системы в этом случае называется *переходом*. Для системы с дискретными состояниями процесс функционирования может быть представлен в виде последовательности состояний с соответствующими переходами.

При рассмотрении движущегося по трассе автомобиля можно выделить различные характеристики его состояния. Это, прежде всего, скорость движения автомобиля, угловое положение передних колес относительно продольной оси, температура охлаждающей жидкости, количество топлива в баке и другие. Изменение значений этих характеристик могут привести к изменению состояний автомобиля, в частности, к изменению его скорости и направления движения.

Методология системного моделирования служит концептуальной основой системно-ориентированной структуризации предметной области. В этом случае исходными компонентами концептуализации являются системы и взаимосвязи между ними. Результатом системного моделирования является построение некоторой модели системы и соответствующей предметной области, которая описывает важнейшие с точки зрения решаемой проблемы аспекты системы.

Под *моделью* будем понимать некоторое представление о системе, отражающее наиболее существенные закономерности ее структуры и процесса функционирования и зафиксированное на некотором языке или в некоторой форме. Применительно к теме нашего рассмотрения нас будут интересовать только такие аспекты построения моделей, которые связаны с информационным или логическим моделированием систем.

Примерами моделей являются не только известные физические модели (аэродинамическая модель гоночного автомобиля или проектируемого самолета), но и логические модели различных систем (математическая модель колебательной системы, аналитическая модель системы электроснабжения региона, информационная модель избирательной компании и др.).

Общим свойством всех моделей является их подобие некоторому реальному объекту или системе-оригиналу. Важность построения моделей заключается в возможности их использования для получения информации о свойствах или поведении системы-оригинала. При этом сам процесс построения и последующего применения моделей для получения информации о системе-оригинале является основным содержанием процесса *системного моделирования*.

Наиболее общей информационной моделью системы является так называемая модель "*черного ящика*". В этом случае система представляется в виде прямоугольника, внутреннее устройство которого скрыто от системного аналитика или вообще неизвестно. Однако система не является полностью изолированной от внешней среды, поскольку последняя оказывает на систему некоторые ин-

формационные или материальные воздействия. Такие воздействия получили название *входных воздействий* или *входных параметров*, *входных переменных*. Среди входных воздействий выделяют специальный класс — так называемых *управляющих воздействий (переменных)*. Последние предназначены для того, чтобы оказывать на систему целенаправленное воздействие, предназначенное для достижения системой некоторой цели (целей) или желаемого поведения. В свою очередь система также оказывает на среду или другие системы определенные информационные или материальные воздействия, которые получили название *выходных воздействий (параметров, переменных)*. Графически данная модель может быть изображена следующим образом (рис. 1.2).

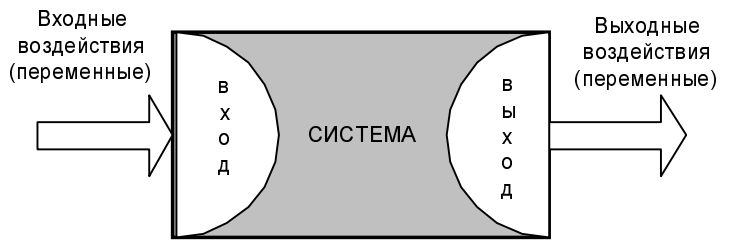


Рис. 1.2. Графическое изображение модели системы в виде "черного ящика"

Ценность моделей, подобных модели "черного ящика", весьма условна. Основное ее назначение состоит в том, чтобы структурировать исходную информацию относительно самой системы и внешней по отношению к ней среды. Поэтому эта модель, прежде всего, фиксирует упоминавшиеся выше границы системы. В дополнение к этому модель специфицирует воздействия, на которые реагирует система, и как проявляется эта реакция на окружающие объекты и системы. При этом в случае количественного описания входных (выходных) воздействий их иногда называют входными (выходными) переменными. В рамках системного моделирования разработаны определенные методологические средства, позволяющие выполнить дальнейшую структуризацию или концептуализацию этой наиболее общей модели системы.

В самом общем случае процесс системного моделирования может быть представлен в форме взаимосвязанных этапов, на каждом из которых выполняются определенные действия, направленные на построение и последующее использование информационно-логических моделей систем (рис. 1.3). Характерной особенностью данного процесса является его циклический или итеративный характер, который отражает современные требования к анализу и проектированию сложных систем.

Таким образом, отдельными этапами процесса системного моделирования являются:

1. Анализ проблемной ситуации.
2. Структуризация предметной области и построение модели.

3. Выполнение вычислительных экспериментов с моделью.
4. Применение результатов вычислительных экспериментов.
5. Коррекция или доработка модели.

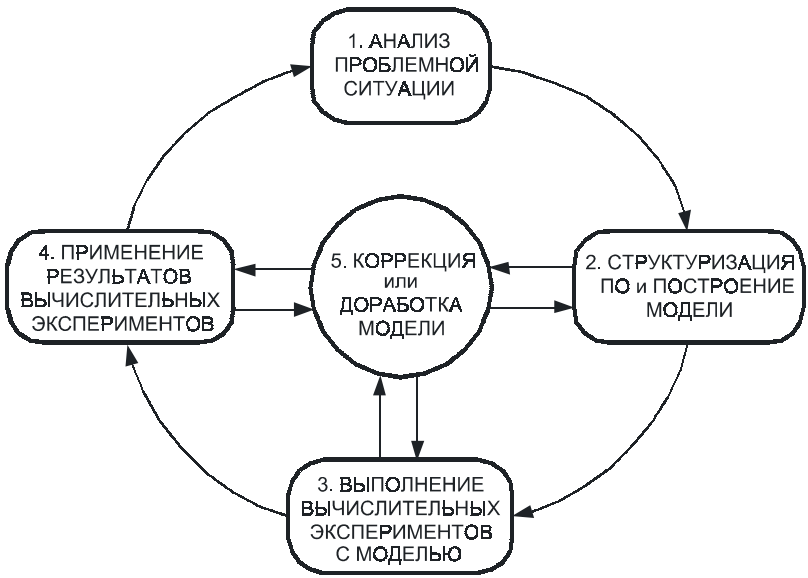


Рис. 1.3. Общая концептуальная схема процесса системного моделирования

Ниже дается краткая характеристика каждого из этапов, конкретное содержание которых зависит от специфических особенностей решаемых задач в той или иной проблемной области. При этом каждый отдельный цикл процесса системного моделирования инициируется этапом анализа проблемной ситуации, в чем проявляется реализация требования проблемно-ориентированного подхода к построению и использованию информационно-логических моделей систем.

Анализ проблемной ситуации

Одним из основных принципов системного моделирования является проблемная ориентация процессов построения и использования моделей. Другими словами, та или иная модель конкретной системы строится в контексте решения некоторой проблемы или достижения некоторой цели. Главное назначение первого этапа — логическое осмысление решаемой проблемы в контексте методологии системного моделирования. При этом выполняется анализ всех доступных ресурсов (материальных, финансовых, информационных и др.), необходимых для построения модели, ее использования и реализации полученных результатов с целью решения имеющейся проблемы. В случае отсутствия требуемых ресурсов на данном

этапе может быть принято решение либо о сужении (уменьшении масштаба) решаемой проблемы, либо вообще об отказе от использования средств системного моделирования. На этом этапе также выполняется анализ требований, предъявляемых в той или иной форме к результату решения проблемы.

Первоначальный анализ решаемой проблемы и соответствующей проблемной области является наименее формализуемым с точки зрения применения известных аналитических подходов и средств. Поэтому на данном этапе рекомендуется применять так называемые *эвристические* или неформальные методы системного анализа. К ним относятся:

- методы построения логистических сценариев на естественном языке для анализа возможных способов и альтернативных путей решения проблемы;
- методы мозговой атаки (штурма) для генерации новых идей и нестандартных подходов к решению проблемы;
- методы морфологического и концептуального анализа для достижения требуемой полноты рассмотрения исходной проблемы;
- методы построения и анализа дерева целей и задач, которые позволяют разбить исходную проблему на ряд более частных или более простых подпроблем.

Структуризация предметной области и построение модели

Целью данного этапа является построение адекватной модели системы и соответствующей предметной области в наиболее общем контексте решения исходной проблемы. Структуризация проблемной области предполагает определение и последующее уточнение ее границ, а также установление границ и состава систем, которые потенциально могут участвовать в решении исходной проблемы. Соответствующая информация представляется в форме модели системы или проблемной области в целом на некотором формально-логическом языке.

Речь идет о том, что вся доступная информация о решении проблемы должна быть зафиксирована в виде некоторой информационно-логической модели системы. При этом модель должна удовлетворять принципу адекватности отражения основных особенностей системы-оригинала. Другими словами, модель не должна быть ни поверхностной (неполной), которая не учитывает существенные аспекты структуры или поведения системы-оригинала, ни излишне сложной или избыточной, в рамках которой разработчики пытаются учесть даже несущественные с точки зрения исходной проблемы детали системы-оригинала.

Данный этап построения информационно-логической модели предполагает выполнение следующей последовательности действий:

1. Построение концептуальной или информационной модели системы и проблемной области, которая содержит наиболее общую информацию и отражает структурные взаимосвязи ситемы-оригинала с другими объектами окружающей среды.

2. Построение аналитической или математической модели системы, которая детализирует отдельные аспекты структуры и поведения системы-оригинала в форме текста с использованием специальной математической нотации (символики).
3. Построение имитационной или программной модели системы, которая непосредственно реализует информационно-логическую модель в форме, специально предназначенной для ее исследования с использованием компьютеров.

Примечание

Один из принципов системного моделирования заключается в том, что для построения адекватной модели сложной системы может потребоваться не одна, а несколько моделей системы-оригинала. В этом случае каждая из подобных моделей будет являться отдельным представлением сложной системы, а полная модель системы будет состоять из комплекса взаимосвязанных моделей. Этот принцип получил специальное название — принцип *многомодельности* системного моделирования. С точки зрения системного аналитика все частные модели системы равноправны, поэтому корректно вести речь лишь об их адекватности. При этом выбор типа модели должен зависеть от характера решаемой проблемы, а не от профессиональной специализации прикладных математиков и системных аналитиков, участвующих в решении проблемы.

Процесс разработки адекватных моделей и их последующего конструктивного применения требует не только знания общей методологии системного анализа, но и наличия соответствующих изобразительных средств или языков для фиксации результатов моделирования и их документирования. Очевидно, что естественный язык не вполне подходит для этой цели, поскольку обладает неоднозначностью и неопределенностью. Поэтому для построения моделей используются формально-теоретические методы, основанные на дальнейшем развитии математических и логических средств моделирования. Для этой цели также предложены различные графические нотации и языки моделирования, в той или иной степени отражающие специфику решаемых задач на основе применения соответствующих программных инструментариев.

Выполнение вычислительных экспериментов с моделью

Модель системы разрабатывается для получения некоторой новой информации о системе-оригинале с целью решения исходной проблемы. В этом случае базовым объектом для получения такой информации является программная модель сложной системы, реализованная на одном из языков программирования или построенная с использованием соответствующих программных инструментариев.

Реализация данного этапа в контексте методологии системного моделирования означает выполнение серии экспериментов с программной моделью системы на той или иной вычислительной платформе. При этом возможна следующая по-

следовательность действий, отражающая содержание собственно процесса планирования экспериментов:

1. Формирование конкретных значений наборов исходных данных (входных переменных), которые характеризуют отдельный вычислительный эксперимент с программной моделью системы.
2. Выполнение расчетов или, в общем случае, выполнение отдельной итерации с имитационной моделью системы с целью получения конкретных значений выходных параметров (переменных) модели.
3. Оценка точности и верификация полученных результатов на основе проверки согласованности отдельных компонентов вычислительных расчетов с использованием аналитической модели.
4. Интерпретация полученных результатов в форме управляющих воздействий или альтернатив решения исходной проблемы.
5. Оценка потенциальной возможности реализации полученных результатов применительно к системе-оригиналу.

Применение результатов вычислительных экспериментов

Содержанием данного этапа является материальное или информационное воздействие на систему-оригинал с целью решения исходной проблемы. При этом может потребоваться планирование организационных мероприятий по реализации подобных воздействий и контроль их выполнения.

После реализации рекомендаций выполненных исследований, что оказывается возможным только после окончания этапа вычислительных экспериментов с моделью, вообще говоря, может сложиться одна из двух ситуаций.

- Исходная проблема полностью решена — тем самым цели системного моделирования достигнуты. В этом случае можно перейти к решению очередной проблемы из данной предметной области, что характеризует начало нового цикла системного моделирования.
- Исходная проблема не решена или решена не полностью — тем самым цели системного моделирования не достигнуты. В этом случае необходимо тщательно проанализировать сложившуюся ситуацию и причины неудачи. После этого можно перейти либо к коррекции исходной модели системы, либо вообще отказаться от построенной модели и реализовывать цикл системного моделирования заново.

Примечание

Следует заметить, что процесс системного моделирования при решении сложных проблем занимает достаточно продолжительное время, в течение которого, вообще говоря, может измениться как само содержание исходной проблемы, так и наличие необходимых для ее решения ресурсов. Эти особенности

зачастую не учитываются при реализации сложных проектов, что является источником их неудачного завершения. Именно для исключения или ослабления негативного влияния данных факторов на схеме системного моделирования должен быть предусмотрен отдельный этап — коррекция или доработка модели, который может начать выполняться с любого момента изменения исходной ситуации или в результате возникновения признаков неадекватности модели на любом из рассмотренных выше этапов.

Коррекция или доработка модели

Цель данного этапа неявно была уже сформулирована выше, а именно — внесение изменений в существующую модель, которые направлены на обеспечение ее адекватности решаемой проблеме. Речь может идти как о включении в состав исходной модели дополнительных компонентов, так и о радикальном изменении структуры и содержания модели. Важно отметить проблемно-ориентированный характер этих изменений, т. е. коррекция или доработка модели должны выполняться в непосредственном контексте с решаемой проблемой.

Упоминавшиеся выше сложные системы, исследование которых представляет наибольший интерес в рамках методологии системного моделирования, образуют отдельный подкласс систем. При этом *сложность* системы и, соответственно, ее модели могут быть рассмотрены с различных точек зрения. Прежде всего, можно выделить сложность структуры системы, которая характеризуется большим количеством элементов системы и различными типами взаимосвязей между этими элементами.

Так, например, если количество элементов системы превышает некоторое пороговое значение, которое, вообще говоря, не является строго фиксированным, то такая система может быть названа сложной. Например, если программная система управления базой данных насчитывает более 100 отдельных форм ввода и вывода информации, то многие программисты сочтут ее сложной. Транспортные и энергетические системы современных мегаполисов, макроэкономика государства или отдельных отраслей также могут служить примерами сложных систем, состоящих из десятков и сотен отдельных подсистем или элементов с нетривиальной структурой взаимосвязей между ними.

Вторым аспектом сложности является сложность процесса функционирования системы или отдельных ее подсистем. Это может быть связано как с непредсказуемым характером поведения системы, так и невозможностью формального представления правил преобразования входных воздействий в выходные. Этот важный аспект сложности системы может быть связан с наличием неопределенности в описании процесса поведения системы-оригинала.

Так, например, процесс поведения участников некоторого рынка товаров или услуг в определенной степени непредсказуем или характеризуется неопределенностью состояний своих элементов. Процесс функционирования современных операционных систем также характеризуется сложностью поведения, поскольку их надежность и безопасность не всегда удовлетворяют требованиям различных категорий пользователей.

При анализе структуры и поведения сложных систем, как правило, присутствуют различные факторы неопределенности, которые могут быть учтены и адекватно представлены в процессе построения информационно-логических моделей в рамках нового направления системного моделирования — нечеткого моделирования.

1.3. Методология нечеткого моделирования

Прежде всего, следует заметить, что методология нечеткого моделирования не заменяет и не исключает рассмотренную выше методологию системного моделирования, а конкретизирует последнюю применительно к процессу построения и использования нечетких моделей сложных систем. Процесс нечеткого моделирования представляет аналогичную последовательность взаимосвязанных этапов, как и процесс системного моделирования (см. рис. 1.3). При этом каждый из этапов выполняется с целью построения и использования нечеткой модели системы для решения исходной проблемы.

В общем случае под *нечеткой моделью* понимается информационно-логическая модель системы, построенная на основе теории нечетких множеств и нечеткой логики.

Таким образом, отдельными этапами процесса нечеткого моделирования являются:

1. Анализ проблемной ситуации.
2. Структуризация предметной области и построение нечеткой модели.
3. Выполнение вычислительных экспериментов с нечеткой моделью.
4. Применение результатов вычислительных экспериментов.
5. Коррекция или доработка нечеткой модели.

Примечание

Поскольку к настоящему времени предложены нечеткие обобщения для самых различных разделов математики и логики, каждое из них потенциально может служить основой для построения соответствующей нечеткой модели. Однако чтобы исключить возможные противоречия при столь широком толковании нечеткой модели, ее содержание будет ограничено лишь рассмотренными в первой части книги нечеткими понятиями. Соответственно понятие нечеткой модели будет уточняться по мере изложения последующего материала.

Как было отмечено ранее, одним из характерных признаков сложности построения модели является неопределенность в представлении структуры или поведения системы-оригинала. При этом сама категория неопределенности может быть рассмотрена с различных точек зрения. В рамках современной методологии сис-

темного моделирования неопределенность может характеризовать следующие аспекты модельных представлений.

- Неясность или нечеткость границы системы. Так, например, использование дихотомических признаков "высокий-низкий", "большой-маленький", "дорогой-дешевый", "молодой-старый", "опытный-неопытный", "быстрый-медленный" и подобных им для определения состава элементов системы сталкивается с принципиальной трудностью представления структуры модели системы. Характерный пример этого аспекта неопределенности — собственно класс сложных систем в контексте ответа на вопрос: "Какие системы следует считать сложными?" Другим примером может служить проблема распознавания рукописного текста компьютером, которая и сейчас не решена в полном объеме.
- Неоднозначность семантики отдельных терминов, которые используются при построении концептуальных моделей систем. Речь идет о присущей естественным языкам *полисемии* или неоднозначности смысла понятий (*модель* прически и математическая *модель*, игральный *автомат* и *автомат* как стрелковое оружие, географическая *карта* местности и игральная *карта*, *стрела* башенного крана и *стрела*, пущенная из лука, *замок* двери и средневековый *замок*).
- Неполнота модельных представлений о некоторой сложной системе, особенно в связи с решением слабо формализуемых проблем. В этом случае сама попытка построить адекватную модель сложной системы или предметной области сталкивается с принципиальной невозможностью учесть все релевантные особенности решаемой проблемы.
- Противоречивость отдельных компонентов модельных представлений или требований, которым должна удовлетворять модель сложной системы. Так, например, требование решить проблему за минимальное время и с минимальными финансовыми затратами содержит в себе элемент противоречия. Элементы противоречий содержатся в законодательных актах и являются предметом юридической практики.
- Неопределенность наступления тех или иных событий, относящихся к возможности нахождения системы-оригинала в том или ином состоянии в будущем. Речь идет о том, что анализ процесса поведения системы не дает оснований для однозначного ответа на вопрос: "Будет ли находиться система-оригинал в некотором состоянии в момент времени, который относится к ее будущему?" Этот аспект неопределенности часто называют стохастическим, поскольку он традиционно исследовался средствами теории вероятностей и математической статистики.

Возвращаясь к характеристике методологии нечеткого моделирования, следует отметить, что исходной предпосылкой ее развития являлась разработка адекватных модельных средств для представления *первого* аспекта неопределенности, связанного, прежде всего, с неясностью или нечеткостью описания границы системы или отдельных ее состояний. Тем не менее, появление и последующее развитие концепции нечеткой меры и теории возможностей позволяет утверждать то, что и другие аспекты неопределенности могут быть подвергнуты нечеткому анализу.

Таким образом, нечеткая модель системы-оригинала, или нечеткая система в первую очередь характеризуется неопределенностью типа неясности (нечеткости) границы системы, а также, возможно, отдельных ее состояний, входных и выходных воздействий. В этом случае исходная структуризация нечеткой системы может быть изображена графически в виде фигуры с расплывчатыми границами (рис. 1.4).

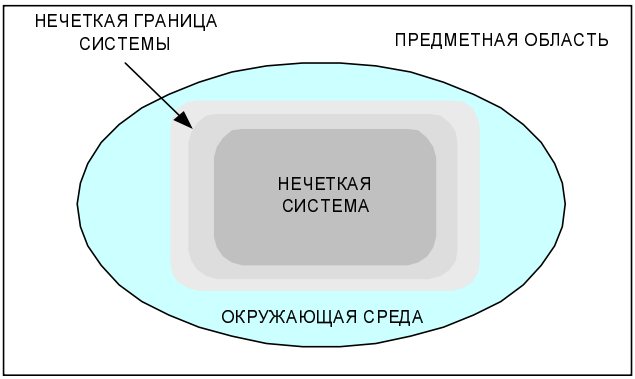


Рис. 1.4. Графическая иллюстрация нечеткой системы как системы с нечеткой границей

Как было отмечено выше, базовой методологией построения нечетких моделей являются собственно теория нечетких множеств и нечеткая логика, которые, в свою очередь, являются обобщением классической теории множеств и классической формальной логики. В связи с этим в *приложениях 1* и *2* рассматриваются те из понятий классической теории множеств и формальной логики, которые в той или иной степени используются далее для соответствующего нечеткого обобщения. Читатели, которые знакомы с соответствующей терминологией, могут непосредственно перейти к рассмотрению теории нечетких множеств *(см. главу 2)*, а к материалу *приложений 1* и *2* обращаться по мере необходимости.

1.4. Анализ нечеткого и вероятностного подходов к моделированию неопределенности

В связи с рассмотренными выше различными аспектами неопределенности, перечень которых, в свою очередь, не претендует на полноту, следует отметить дискуссию, которая возникла по вопросу: "Является ли нечеткость разновидностью вероятности или она имеет некое самостоятельное содержание?" Эта дискуссия была инициирована адептами стохастического подхода к анализу неопределенности и время от времени дополняется новой аргументацией в пользу того, что,

по их мнению, нечеткость не вносит ничего нового в процесс анализа неопределенности. Хотя ниже будет строго математически показано, что концепция нечеткой меры включает как частный случай вероятностную меру, уже сейчас можно увидеть качественное отличие в рассмотренных выше аспектах неопределенности. Наличие других ее аспектов, таких как неуверенность, несогласованность, ненадежность, недостаточность, могут послужить предметом дальнейших размышлений заинтересованных читателей по данной проблематике.

Исторически изучением и разработкой моделей, учитывающих неопределенность того или иного вида, занимаются многие математические дисциплины, такие как теория вероятностей, теория информации, математическая статистика, теория игр, теория массового обслуживания и теория нечетких множеств. Один из способов показать различия нечеткого и стохастического подходов — классифицировать тип неопределенности, которая изучается этими дисциплинами. С этой целью рассмотрим два наиболее характерных типа неопределенности — стохастическую и лингвистическую неопределенности.

Стохастическая неопределенность

Стохастическая неопределенность имеет место в ситуациях, когда некоторое хорошо описанное событие может произойти, а может не произойти. При этом с течением времени степень неопределенности, связанная с этим событием, может измениться. Дополнительно необходимо принять некоторые предположения относительно условий, при которых рассматривается данное событие. Эти условия, как правило, характеризуют так называемый *идеальный эксперимент*.

Рассмотрим следующее высказывание: "*Вероятность того, что при бросании монеты выпадет орел (герб), равна* 0.5".

В этом высказывании неявно предполагается, что монета и поверхность идеально правильной формы, процесс бросания идеален с точки зрения субъектов эксперимента, а потенциальная возможность того, что монета окажется в вертикальном положении, исключается полностью. По прошествии некоторого времени неопределенность исчезает, поскольку после подбрасывания монеты она окажется в одном из двух возможных состояний: либо орлом сверху, либо решкой.

Таким образом, рассматриваемое высказывание имеет смысл только по отношению к событию в будущем. Изменение условий эксперимента может привести к изменению содержания этого высказывания. Поскольку обеспечить идеальные условия на практике не всегда возможно, вольно или невольно мы вынуждены считаться с некоторой потенциально присутствующей ошибкой в количественной оценке вероятности событий. Предельные теоремы теории вероятностей как раз и предназначены для оценки этой погрешности при частотной интерпретации вероятности события в длинной серии испытаний.

Исторически теория вероятностей была первой математической дисциплиной для представления неопределенности в математических моделях. По этой причине любая неопределенность долгое время считалась стохастической по своей

природе и наделялась, иногда искусственно, свойствами случайной неопределенности. Что касается вероятностного процесса, результат любой частной реализации которого является исключительно вопросом случая, предсказать последовательность событий просто невозможно. Для вероятностных процессов оказывается возможным лишь точное описание статистических оценок некоторых усредненных характеристик этого процесса.

Рассмотрим другое высказывание: "*Вероятность того, что завтра пойдет дождь, равна* 0.8".

В этом высказывании неявно предполагается, что событие "*пойдет дождь*" хорошо описано. Тем не менее, совершенно очевидно, что это событие недостаточно хорошо определено: не ясно, то ли дождь будет идти целый день, или дождь будет идти 80% от следующих по времени суток? Более того, следует ли считать дождем мелкий дождь или только ливень? Таким образом, при кажущейся очевидности этого высказывания при более детальном его анализе мы обнаруживаем некоторый другой тип неопределенности, который содержательно отличается от стохастического. Эта неопределенность скорее относится к лингвистическому описанию ситуации или события, а не к количественной оценке того, произойдет это событие в будущем или не произойдет.

Лингвистическая неопределенность

Реальный мир сложен, причем эта сложность зачастую проявляется как неопределенность в форме неоднозначности или неточности.

Этот тип неопределенности связан с неточностью обычного человеческого языка, с ним мы постоянно сталкиваемся в повседневной жизни. Достаточно рассмотреть фразы типа "высокие люди", "горячие пирожки", "красивое лицо", "хороший автомобиль", "устойчивая валюта", "дождливый день", "неважное самочувствие", "трудный день", чтобы понять, что вряд ли возможно дать им точные количественные определения.

Действительно, высокие и низкие люди будут иметь свои собственные представления о том, каких людей следует считать высокими. Более того, если мы формально установим считать высокими всех людей выше 180 см, будет ли человек с ростом 179.999 см высоким или нет? Контекст фраз тоже имеет значение, поскольку оценка высоких людей, находящихся на сцене театра и в зрительном зале, будет различной.

Для изучения подобных субъективных оценок предназначена отдельная наука — психолингвистика. В рамках этой науки принято считать, что в рассмотренных фразах люди используют слова в качестве некоторых *субъективных категорий*. Эти субъективные категории дают нам возможность классифицировать объекты, которые характеризуются такими свойствами, как "высота", "длина", "вес", "температура", "цвет". Даже при том, что большинство используемых категорий точно не определено, люди могут использовать их для весьма комплексных оценок и решений, которые основаны на учете многих различных факторов.

Рассмотрим высказывание: "*Вероятно, мы будем иметь успешный финансовый год*".

Это высказывание имеет существенные отличия от рассмотренных ранее высказываний. Во-первых, само событие точно не определено. Для некоторых компаний успешный финансовый год может означать, что им удастся избежать банкротства. Для других это может означать превышение прибыли за предшествующий год. Даже для отдельно взятой компании трудно предложить некоторое количественное значение прибыли, чтобы определить, будет ли для нее бюджетный год, как рассматривается, успешным или нет. Следовательно, понятие "успешный финансовый год" является субъективной категорией.

Другая особенность последнего высказывания заключается в определении выражения вероятности. В то время как в предыдущих двух высказываниях вероятность была выражена количественно, данное высказывание не определяет количество вероятности. Следовательно, выражение вероятности в последнем высказывании также является субъективной категорией так же, как "высокие люди" и "горячие пирожки".

Моделирование лингвистической неопределенности

Высказывания, аналогичные последнему высказыванию и использующие субъективные категории людей, играют важную роль в процессе повседневного принятия решения. Даже при том, что эти высказывания не имеют количественного содержания, люди успешно используют их для комплексных оценок. В некоторых случаях неопределенность, которая присутствует в значении тех или иных слов, сознательно используется нами в разговоре для придания ему дополнительной гибкости. Достаточно представить себе диалоги в ситуациях с поиском высокооплачиваемой работы или приобретением недвижимости.

Чтобы адекватно использовать логику, присутствующую в человеческих рассуждениях, для решения технических проблем необходимо разработать соответствующую математическую модель. Именно с этой целью была разработана нечеткая логика, которая позволяет представить процессы принятия решений и оценки ситуаций человеком в некоторой алгоритмической форме. Хотя возможности человеческого мышления и фантазии безграничны, пределы того, что позволяет моделировать нечеткая логика, существуют.

Каким образом люди могут рассуждать относительно реальных систем, когда законченное описание реальной системы часто требует более детальных данных, чем человек в состоянии получить и интерпретировать? Ответ состоит в том, что люди имеют способность рассуждать приблизительно, возможность, которой компьютеры в настоящее время не обладают. При общении людей использование фраз типа "высокий человек" и "высокооплачиваемая работа" не приводит к возникновению концептуальных проблем, поскольку передает семантически понятную информацию участвующим в разговоре личностям. При необходимости всегда можно уточнить используемые субъективные категории.

В то же время компьютеры или микропроцессоры используют в своей работе исключительно бинарную логику. Для понимания соответствующих фраз компьютером необходимо, чтобы конкретное значение высоты или заработной платы сравнилось с заданным пороговым значением для рассматриваемых формальных категорий "высокий человек" и "высокооплачиваемая работа". Основное достоинство теории нечетких множеств заключается в возможности использовать лингвистические переменные вместо количественных, нечеткую логику вместо бинарной логики для формального представления подобных неточных субъективных категорий.

При рассмотрении сложной системы люди рассуждают относительно ее структуры и поведения приблизительно или неточно. Тем самым достигается некоторое универсальное понимание содержания проблемы. К счастью, эта общность и неточность, приобретаемая в форме опыта с течением времени, зачастую оказываются достаточными для человеческого понимания сложных явлений и адекватного принятия решений в бытовых ситуациях. Именно в рамках теории нечетких множеств оказывается возможным включить в описание проблемы этот опыт и интуицию отдельного человека.

Нечеткая логика в сравнении с теорией вероятностей

Рассмотренные выше примеры высказываний иллюстрируют тот факт, что стохастическая и лингвистическая неопределенности имеют различный характер. Стохастическая неопределенность имеет дело с неопределенностью того, произойдет ли некоторое хорошо описанное событие в будущем, а теория вероятностей позволяет дать на этот вопрос тот или иной ответ.

Напротив, лингвистическая неопределенность связана с неточностью описания самой ситуации или события независимо от времени их рассмотрения. Теория вероятностей не может использоваться для решения подобных проблем, поскольку представления о субъективных категориях, присутствующих в процессах мышления человека, в полной мере не согласуются с ее аксиомами.

Тем не менее, некоторые из специалистов, интенсивно работающие с теорией вероятностей и математической статистикой, долгое время отрицали саму возможность применения нечеткой логики в приложениях. Эти специалисты зачастую утверждали, что все виды неопределенности могут быть выражены в понятиях теории вероятностей. В то же время даже из рассмотренных выше примеров становится очевидным, что как теорию вероятностей, так и теорию нечетких множеств целесообразно использовать для моделирования различных аспектов неопределенности, отличающихся по своей природе.

В заключение приведем еще один наглядный пример, который хорошо иллюстрирует семантическое различие между стохастической и лингвистической неопределенностью. Представим себе ситуацию, когда путник после длительного путешествия, испытывая чувство жажды, находит две бутылки с неизвестной жидкостью внутри каждой из них (рис. 1.5, *а*). Естественным желанием путника

является утолить свою жажду. Однако никаких этикеток с указанием напитка найденные бутылки не содержат (кроме, возможно, пометок *А* и *Б*).

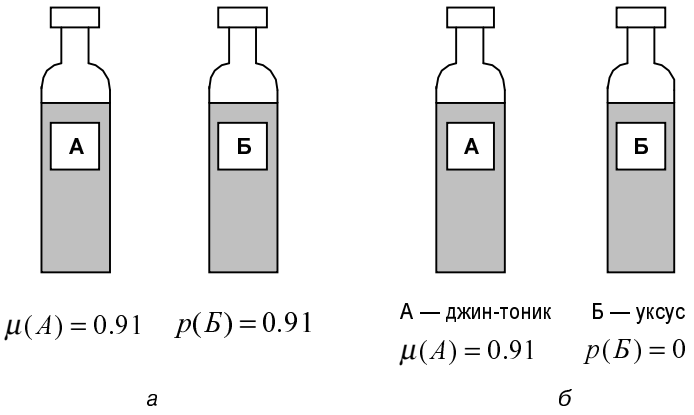


Рис. 1.5. Пример с неизвестными напитками в бутылках

Предположим, что дополнительно известна степень принадлежности содержимого бутылки *А* к жидкостям, пригодным для питья, и эта степень принадлежности равна 0.91. Известна также вероятность того, что содержимое бутылки *Б* пригодно для питья, и эта вероятность также равна 0.91. Если путник органичен в выборе напитков этими двумя бутылками, какую из них ему следует выбрать для утоления жажды?

Если путник знаком с теорией нечетких множеств и теорией вероятностей, то его выбор может основываться на следующем рассуждении. Анализируя информацию о содержимом бутылки *А*, он может предположить, что в ней находится не совсем пригодная для питья жидкость, например, болотная вода. При этом естественно считать, что чистая вода имела бы степень принадлежности равную 1. В то же время в этой бутылке не может находиться ядовитая жидкость, скажем, серная кислота, поскольку в этом случае степень принадлежности содержимого бутылки *А* к жидкостям, пригодным для питья, была бы равна 0.

Анализируя информацию о содержимом бутылки *Б*, путник, естественно, будет апеллировать к частотной интерпретации вероятности содержащейся в ней жидкости. В этом случае резонно предположить, что если бы путник имел возможность многократного выбора бутылки *Б*, то приблизительно в 9 случаях из 10 он смог бы благополучно утолить свою жажду. При этом содержимое бутылки *Б* должно было бы быть по качеству близким к чистой воде. Что же должно произойти в том единственном случае из рассматриваемых 10 — остается непонятным. Возможно, результатом может оказаться самый печальный исход для путника или его серьезное недомогание. Очевидно, что это не может случиться, если в бутылке *Б* находится пиво или квас. Значит в одном случае из 10 в этой бутылке может находиться нечто совсем неприемлемое для питья, например, соляная кислота.

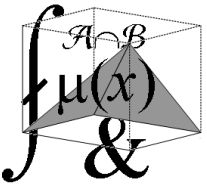
Вывод для путника напрашивается очевидный — при наличии только указанной информации выбрать содержимое бутылки *А*. Во всяком случае это позволит избежать серьезного отравления или летального исхода. Но этот выбор может измениться, если изменится информация о содержимом бутылок. Так, совсем не очевидно, какую из бутылок предпочесть, если степень принадлежности для бутылки *А* и вероятность для бутылки *Б* станут равными, скажем, 0.5.

Посмотрим, как изменятся эти количественные характеристики после того, как путнику станет известно, что в бутылке *А* содержится джин-тоник, а в бутылке *Б* — уксус (рис. 1.5, *б*). Очевидно, что обе ситуации возможны при рассмотренных исходных условиях. Степень принадлежности жидкости в бутылке *А* останется без изменений, если исключить явное предпочтение этого напитка перед всеми остальными. А вот апостериорная вероятность жидкости в бутылке *Б* станет равной 0, поскольку вряд ли уксус следует считать жидкостью, пригодной для питья без опасности печальных последствий. С другой стороны, если бы в бутылке *Б* оказался джин-тоник, то у нас были бы все основания считать соответствующую апостериорную вероятность равной 1.

Таким образом, понятие нечеткого множества способно обеспечить нас адекватной информацией относительно неточного описания тех или иных ситуаций. По существу, этот подход наиболее применим для решения таких проблем, в которых неопределенность характеризуется отсутствием хорошо определенных критериев, позволяющих однозначно судить о принадлежности элементов тому или иному классу. Именно в этом проявляется различие между нечеткостью и случайностью. В то же время нечеткие модели не являются заменой моделей, разработанных в теории вероятностей. Как будет видно из последующего изложения, каждое четкое множество является нечетким, но обратное утверждение не верно. Поэтому нечеткие модели обобщают традиционные и более знакомые нам математические модели. Иногда они работают лучше, а иногда нет. В конце концов, эффективность модели проявляется в ее способности адекватно решить ту или иную конкретную проблему.

Как правило, сложная проблема в той или иной степени связана с неопределенностью. Искусство и профессионализм системного аналитика как раз и проявляются в том, чтобы предложить для ее решения такую модель, которая наиболее адекватно учитывает тот или иной тип неопределенности. Достигнутые в последнее время впечатляющие успехи в приложении нечетких технологий для решения самых разнообразных практических задач позволяют утверждать, что нечеткое моделирование реальных сложных систем — эффективная альтернатива традиционным математическим моделям и методам.

Глава 2



Основные понятия теории нечетких множеств

Настоящая глава во многих отношениях является базовой, поскольку в ней представлены определения всех основных свойств нечетких множеств, которые используются на всем протяжении книги. Хотя из общих методологических рассуждений *главы 1* может сложиться впечатление о неформальном характере теории нечетких множеств, это впечатление обманчиво. В действительности данная теория в математическом смысле является строго формализованной. К настоящему времени предложены самые разнообразные определения нечетких теоретико-множественных понятий. Однако в книгу вошел только тот материал, который непосредственно применяется для решения различных практических задач и в той или иной степени реализован в соответствующих инструментальных средствах.

2.1 Определение нечеткого множества

Нечеткое множество. *Нечеткое множество* (fuzzy set) представляет собой совокупность элементов произвольной природы, относительно которых нельзя с полной определенностью утверждать — принадлежит ли тот или иной элемент рассматриваемой совокупности данному множеству или нет. Другими словами, нечеткое множество отличается от обычного множества тем, что для всех или части его элементов не существует однозначного ответа на вопрос: "Принадлежит или не принадлежит тот или иной элемент рассматриваемому нечеткому множеству?" Можно этот вопрос задать и по-другому: "Обладают или нет его элементы некоторым характеристическим свойством, которое может быть использовано для задания этого нечеткого множества?"

Для построения нечетких моделей систем само понятие нечеткого множества следует определить более строго, чтобы исключить неоднозначность толкования тех или иных его свойств. Оказалось, что существуют несколько вариантов формального определения нечеткого множества, которые по сути отличаются между собой способом задания характеристической функции данных множеств. Среди этих вариантов наиболее естественным и интуитивно понятным является задание области значений подобной функции как интервал действительных чисел, заключенных между 0 и 1 (включая и сами эти значения).

Математическое определение нечеткого множества. Формально нечеткое множество $\mathcal{A}$ определяется как множество упорядоченных пар или кортежей вида: $<x, \mu_{\mathcal{A}}(x)>$, где x является элементом некоторого универсального множества или универсума $\boldsymbol{X}$, а $\mu_{\mathcal{A}}(x)$ — *функция принадлежности*, которая ставит в соответствие каждому из элементов $x \in \boldsymbol{X}$ некоторое действительное число из интервала [0, 1], т. е. данная функция определяется в форме отображения:

$$\mu_{\mathcal{A}} : \boldsymbol{X} \to [0, 1]. \qquad (2.1)$$

При этом значение $\mu_{\mathcal{A}}(x)=1$ для некоторого $x \in \boldsymbol{X}$ означает, что элемент x *определенно принадлежит* нечеткому множеству $\mathcal{A}$, а значение $\mu_{\mathcal{A}}(x)=0$ означает, что элемент x *определенно не принадлежит* нечеткому множеству $\mathcal{A}$.

Формально конечное нечеткое множество будем записывать в виде: $\mathcal{A}=\{<x_1, \mu_{\mathcal{A}}(x_1)>, <x_2, \mu_{\mathcal{A}}(x_2)>,\ldots, <x_n, \mu_{\mathcal{A}}(x_n)>\}$, а в общем случае — в виде: $\mathcal{A}=\{<x, \mu_{\mathcal{A}}(x)>\}$.

Примечание

В литературе по теории нечетких множеств, которая исчисляется огромным количеством работ, можно встретить не только различные определения, но и разнообразные обозначения для нечетких множеств. Наиболее общие из определений нечеткого множества предполагают, что в качестве области значений функции принадлежности могут выступать другие нечеткие множества или произвольные вполне упорядоченные множества. Следует также отметить, что в ранних работах отечественных авторов по данной тематике нечеткие множества иногда назывались *расплывчатыми*. Кроме принятых нами обозначений конечные нечеткие множества часто записываются в форме: $A = \{(\mu_A(x_1), x_1), (\mu_A(x_2), x_2), \ldots, (\mu_A(x_n), x_n)\}$, $A = \{x_1/\mu_A(x_1) + x_2/\mu_A(x_2) + \ldots + x_n/\mu_A(x_n)\}$ или $A = \{\frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \ldots + \frac{\mu_A(x_n)}{x_n}\}$. При этом косая и горизонтальная черта служат просто разделителем, а знак "+" обозначает не арифметическую сумму, а теоретико-множественное объединение отдельных элементов. Бесконечные нечеткие множества иногда записывают со знаком интеграла в виде: $A = \int \mu_A(x) / x$. Все это скорее дань традиции, чем нечто имеющее содержательный смысл. Тем более, что сам знак интеграла может быть воспринят как нечеткий интеграл, чем он здесь никак не является. Желая подчеркнуть или явно указать, что множество A является нечетким, многие авторы часто записывают нечеткое множество со знаком *тильда* "~" внизу или вверху, т. е. в форме: $\underset{\sim}{A}$ или $\widetilde{A}$.

Поскольку существующие различия в формах записи не имеют принципиального значения, в последующем тексте нечеткие множества для удобства будут обозначаться рукописными прописными буквами: $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$. С другой стороны, для записи классических (не нечетких, crisp) множеств будут по-прежнему использоваться общепринятые обозначения в форме: A, B, C, D *(см. приложение 1)*. Что касается других определений и обозначений нечетких множеств или нечетких подмножеств, то заинтересованный читатель может познакомиться с ними, обратившись к дополнительной литературе, приводимому в конце книги.

Из всех нечетких множеств выделим два частных случая, которые по сути совпадают со своими классическими аналогами и используются в дальнейшем при определении других нечетких понятий.

Пустое нечеткое множество. В теории нечетких множеств сохраняют свой смысл некоторые специальные классические множества. Так, например, *пустое нечеткое* множество или множество, которое не содержит ни одного элемента, по-прежнему обозначается через $\varnothing$ и формально определяется как такое нечеткое множество, функция принадлежности которого тождественно равна нулю для всех без исключения элементов: $\mu_{\varnothing} = 0$. В этой связи уместно упомянуть о том, что характеристическая функция обычного пустого множества также тождественно равна нулю для каких бы то ни было элементов: $\chi_{\varnothing} = 0$ *(см. приложение 1)*.

Универсум. Что касается другого специального множества, то так называемый *универсум*, обозначаемый через $\boldsymbol{X}$, уже был использован выше в качестве обычного множества, содержащего в рамках некоторого контекста все возможные элементы. Формально удобно считать, что функция принадлежности универсума как нечеткого множества тождественно равна единице для всех без исключения элементов: $\mu_X = 1$. При этом характеристическая функция обычного универсального множества также тождественно равна единице для каких бы то ни было элементов: $\chi_X = 1$ *(см. приложение 1)*.

Примечание

Как не трудно заметить, рассмотренные понятия пустого множества и универсума, используемые в теории нечетких множеств, по своему содержанию полностью идентичны соответствующим понятиям классической теории множеств. Поэтому говоря о них, мы не будем использовать определение "нечеткое", поскольку в произвольном контексте они всегда являются формально определенными.

Для того чтобы определить конечные и бесконечные нечеткие множества, необходимо ввести в рассмотрение одно из основных понятий, которое используется для характеристики произвольного нечеткого множества, а именно — понятие носителя нечеткого множества.

Носитель нечеткого множества. *Носителем* нечеткого множества $\mathcal{A}$ называется обычное множество A_S, которое содержит те и только те элементы универсума, для которых значения функции принадлежности соответствующего нечеткого множества отличны от нуля. Математически носитель нечеткого множества определяется следующим условием:

$$A_S = \{x \in \boldsymbol{X} \mid \mu_{\mathcal{A}}(x) > 0\} \qquad \forall x \in \boldsymbol{X}. \tag{2.2}$$

Примечание

Иногда носитель нечеткого множества обозначают через $supp(\mathcal{A})$, где *supp* — первые буквы английского слова *support*. Мы не будем использовать это обозначение, поскольку оно может быть ошибочно ассоциировано с обозначением рассматриваемой ниже функции $\sup(x)$.

Очевидно, пустое нечеткое множество имеет пустой носитель, поскольку $\mu_{\varnothing}=0$ для любого его элемента. Носитель универсума, рассматриваемого как нечеткое множество, совпадает с самим универсумом. Для удобства и сокращения записи произвольного нечеткого множества часто указывают лишь значения его функции принадлежности для элементов носителя, неявно предполагая, что все остальные значения функции принадлежности равны нулю.

В зависимости от количества элементов в нечетком множестве по аналогии с обычными множествами можно определить конечные и бесконечные нечеткие множества.

Конечные нечеткие множества. Нечеткое множество называется *конечным*, если его носитель является конечным множеством. При этом вполне уместно говорить, что такое нечеткое множество имеет конечную *мощность*, которая численно равна количеству элементов его носителя как обычного множества *(см. приложение 1)*. В этом случае для обозначения мощности произвольного нечеткого множества $\mathcal{A}$ можно также использовать символ $card(\mathcal{A})$. Удобно считать мощность пустого множества равной 0.

Бесконечные нечеткие множества. Аналогичным образом можно определить и *бесконечные* нечеткие множества как такие нечеткие множества, носитель которых не является конечным множеством. При этом *счетным* нечетким множеством будем называть нечеткое множество со счетным носителем, т. е. носитель которого имеет счетную мощность $\aleph_0$ *в обычном смысле (см. приложение 1)*. *Несчетным* нечетким множеством будем называть нечеткое множество с несчетным носителем, т. е. носитель которого имеет несчетную мощность или мощность *континуума* **c** (или $\aleph$) в обычном смысле.

Очевидно, данное выше определение носителя нечеткого множеств корректно, поскольку как для конечных, так и для бесконечных нечетких множеств выражение (2.2) имеет смысл.

Чтобы привести некоторые примеры нечетких множеств и приступить к определению их основных свойств, следует рассмотреть основные способы, которыми формально могут быть заданы произвольные нечеткие множества.

Нечеткие множества могут быть заданы двумя основными способами:

1. В форме *списка с явным перечислением* всех элементов и соответствующих им значений функции принадлежности, образующих рассматриваемое нечеткое множество. При этом зачастую элементы с нулевыми значениями функции принадлежности просто не указываются в данном списке. Этот способ подходит для задания нечетких множеств с конечным дискретным носителем и небольшим числом элементов. В этом случае нечеткое множество удобно записывать в виде: $\mathcal{A}=\{<x_1, \mu_{\mathcal{A}}(x_1)>, <x_2, \mu_{\mathcal{A}}(x_2)>,\ldots, <x_n, \mu_{\mathcal{A}}(x_n)>\}$, где n — рассматриваемое число элементов нечеткого множества $\mathcal{A}$ (его носителя).

   Например, возьмем в качестве универсума $\boldsymbol{X}=\{1, 2, 3,...\}$ — множество натуральных чисел. Тогда нечеткое множество $\mathcal{A}$, представляющее в некотором контексте "*небольшое натуральное число*", можно задать следующим образом: $\mathcal{A}=\{<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>\}$. При этом элементы, для которых $\mu_{\mathcal{A}}(x) = 0$, отсутствуют в этом списке.

2. *Аналитически* в форме математического выражения для соответствующей функции принадлежности. Этот способ может быть использован для задания произвольных нечетких множеств как с конечным, так и с бесконечным носителем. В этом случае нечеткое множество удобно записывать в виде: $\mathcal{A}=\{<x, \mu_{\mathcal{A}}(x)>\}$ или $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$, где $\mu_{\mathcal{A}}$ — некоторая функция, заданная аналитически в форме математического выражения $f(x)$ или графически в форме некоторой кривой. Наиболее часто используемые виды функций принадлежности будут рассмотрены ниже в этой главе.

Для формальной строгости при задании нечетких множеств необходимо явно указывать соответствующий универсум $\boldsymbol{X}$ элементов, из которых формируется то или иное конкретное нечеткое множество. В общем случае никаких предположений относительно элементов этого множества не делается. Однако с практической точки зрения целесообразно ограничить универсум элементами рассматриваемой предметной области или решаемой задачи. Поскольку при построении нечетких моделей систем используются количественные переменные, то наиболее часто в качестве универсума $\boldsymbol{X}$ используется некоторое подмножество действительных чисел $\mathbb{R}$, например, множество неотрицательных действительных чисел $\mathbb{R}_+$ или натуральных чисел $\mathbb{N}$.

Рассмотрим некоторые конкретные примеры нечетких множеств.

П р и м е р 2.1. Предположим, необходимо построить некоторое нечеткое множество, которое содержательно описывало бы выходные (нерабочие) дни обычной семидневной недели. В терминологии классических множеств ситуация тривиальная, а именно, дни недели с понедельника по пятницу являются рабочими, а суббота и воскресенье — выходными. Заметим, что речь идет о традиционной календарной неделе, а рабочие дни считаются без учета сменности и других особенностей трудозатрат. Таким образом, обычное не нечеткое множество выходных дней A состоит из двух элементов: $A=\{$*суббота*, *воскресенье*$\}$. Эта точка зрения является общепринятой для бухгалтерии при расчете заработной платы сотрудникам.

Что же касается определения соответствующего нечеткого множества $\mathcal{A}$, попытаемся субъективно оценить степень нашего эмоционального отношения к различным дням недели, рассматривая их с точки зрения выходных и психологии возможного отдыха. Для большинства из нас ситуация уже не будет казаться столь простой, как в предыдущем случае.

Что касается дней с понедельника по четверг, то отношение к ним как к рабочим дням вряд ли изменится. А вот пятница, особенно ее вечер, для многих ассоциируется с полноценным отдыхом и высокой степенью положительных эмоций. Суббота является безусловно выходным днем, в течение которого могут быть забыты все служебные заботы, особенно в субботу вечером, а для многих — и ночью. А вот что касается воскресенья, то ближе к вечеру ситуация меняется — нередко на ум приходит мысль: "*Завтра нужно рано вставать и приступать к работе*", и настроение уже нельзя считать столь безоблачным.

Таким образом, рассматриваемое нечеткое множество $\mathcal{A}$, описывающее выходные дни недели, может быть задано, например, в виде: $\mathcal{A}=\{<$*понедельник*, 0>, <*вторник*, 0>, <*среда*, 0>, <*четверг*, 0>, <*пятница*, 0.5>, <*суббота*, 1.0>,

<*воскресенье*, 0.8>}. Здесь в качестве универсума выступают все дни недели: X={*понедельник, вторник, среда, четверг, пятница, суббота, воскресенье*}, а функция принадлежности задается перечислением своих значений. При этом чем ближе ее значение к 1, тем больше соответствует тот или иной день недели нашему отношению к нему как к выходному дню.

Попробуем представить это нечеткое множество графически. Очевидно, обычный способ изображения множеств с помощью диаграмм Венна *(см. приложение 1)* здесь не подходит, поскольку границы данного нечеткого множества не являются четко очерченными. Однако, помня, что каждое нечеткое множество вполне определяется своей функцией принадлежности, изобразим графически функцию принадлежности этого нечеткого множества. Для этого на горизонтальной оси отметим отдельные значения элементов универсума (в нашем случае — элементы множества X), а на вертикальной оси — значения соответствующей функции принадлежности $\mu_{\mathcal{A}}(x)$ (рис. 2.1).

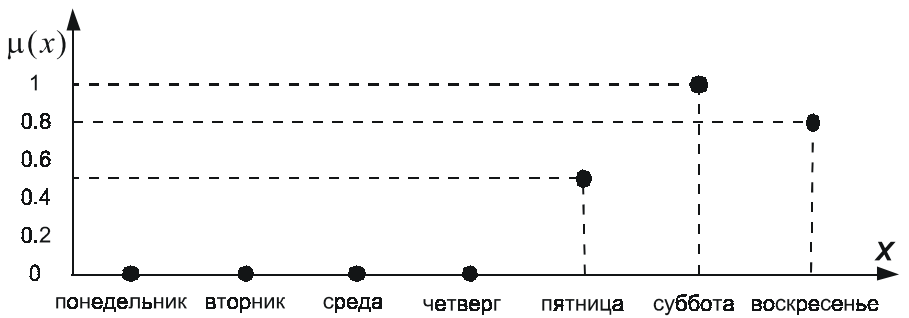


Рис. 2.1. Графическое представление конечного нечеткого множества $\mathcal{A}$, описывающего выходные дни недели, в форме значений функции принадлежности этого нечеткого множества

Примечание

Даже этот простой пример показывает, что однозначно определить то или иное нечеткое множество не представляется возможным, а иногда — и принципиально невозможным. Если кто-то решит, что его субъективная оценка выходных дней отличается от рассмотренной выше, то он/она будут по-своему правы. Соответственно, в качестве нечеткого множества $\mathcal{A}$ могли бы выступать множества: $\mathcal{A}$={<*понедельник*, 0>, <*вторник*, 0>, <*среда*, 0>, <*четверг*, 0.1>, <*пятница*, 0.6>, <*суббота*, 1.0>, <*воскресенье*, 0.7>} или $\mathcal{A}$={<*понедельник*, 0>, <*вторник*, 0.1>, <*среда*, 0>, <*четверг*, 0.1>, <*пятница*, 0.5>, <*суббота*, 0.9>, <*воскресенье*, 0.8>}. Важно представлять себе, что с формальной точки зрения все они должны удовлетворять лишь исходному определению нечеткого множества в форме (2.1).

Продолжим рассмотрение предыдущего примера с целью его расширения на случай бесконечного нечеткого множества. Поскольку наше отношение к выходным дням недели может изменяться в течение времени суток, а горизонталь-

ная ось на рис. 2.1 легко преобразуется к непрерывной оси времени, то и соответствующее нечеткое множество $\mathcal{A}$ допускает естественное обобщение. А именно, каждый из дней недели будем представлять как отдельные сутки с переходом в 0 часов к следующему дню недели. Тогда функция принадлежности нечеткого множества $\mathcal{A}$ может быть задана аналитически в форме некоторой кривой, которая в максимальной степени соответствует нашему эмоциональному отношению к выходным дням в течение всех суток.

В простейшем случае мы могли бы аппроксимировать представленную ранее функцию принадлежности (рис. 2.1) некоторой кривой. Один из возможных вариантов такой функции принадлежности изображен на рис. 2.2, на котором горизонтальная ось соответствует посуточному представлению семидневной недели.

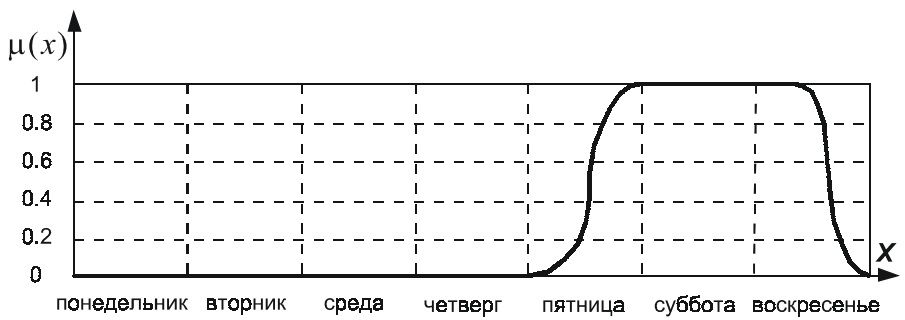


Рис. 2.2. Графическое представление бесконечного нечеткого множества $\mathcal{A}$, описывающего выходные дни недели, в форме кривой его функции принадлежности

Для сравнения рассмотрим представление обычного (не нечеткого) множества выходных дней недели $A=\{суббота, воскресенье\}$ в форме бесконечного множества. В этом случае характеристическая функция $\chi_A(x)$ данного множества может быть записана в виде кусочно-непрерывной функции, принимающей только два значения — 0 и 1 на множестве значений универсума $\boldsymbol{X}$ (рис. 2.3).

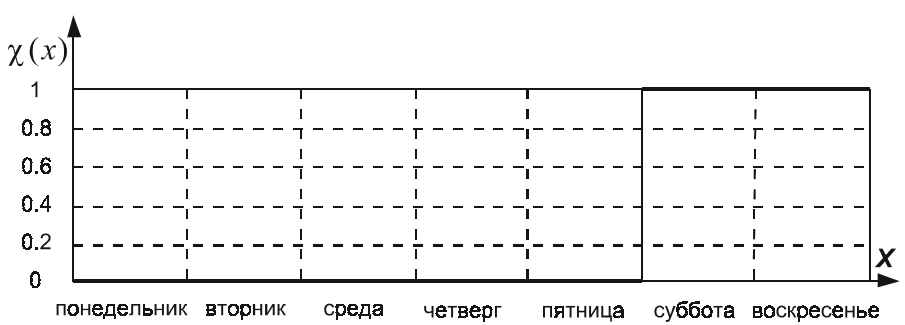


Рис. 2.3. Графическое представление обычного множества выходных дней A в форме значений соответствующей характеристической функции

Примечание

Поскольку изображенная на рис. 2.3 функция принадлежности имеет разрывы в точках смены суток с пятницы на субботу и с воскресенья на понедельник, то строго формально следует определить ее значение в данных точках. Эта неопределенность содержательно связана с неопределенностью соответствующих моментов времени на временной оси (24 часа в пятницу и 0 часов в субботу). Не ограничивая общности изложения, математически можно считать, что значения функции принадлежности в этих точках равны нулю (или единице) и соответствующим образом откорректировать рисунок. Поскольку в нашем контексте это не имеет принципиального значения, мы оставим данный рисунок без изменения.

Из рассмотрения данного примера видно, что характеристическую функцию χ_A обычного множества A в том или ином контексте удобно считать специальным случаем функции принадлежности $\mu_{\mathcal{A}}$ соответствующего нечеткого множества $\mathcal{A}$. Этот факт позволяет рассматривать произвольное нечеткое множество $\mathcal{A}$ как обобщение обычного множества A, а множество A — как сужение или частный случай соответствующего нечеткого множества $\mathcal{A}$.

П р и м е р 2.2. В качестве второго примера рассмотрим типичную бытовую ситуацию, с которой сталкиваются многие из нас при попытке дать характеристику температуры того или иного напитка. Подобная характеристика обычно основывается исключительно на субъективных ощущениях, например, горячий кофе или чай, холодный квас или кола. Хотя в этом случае неявно используется некоторая шкала температуры, при этом, как правило, не применяется никаких измерительных инструментов.

Применительно к данной ситуации рассмотрим нечеткое множество $\mathcal{B}$, которое будет характеризовать "*горячий кофе*". В этом случае в качестве универсума естественно взять шкалу температуры, измеренной в градусах Цельсия и заключенной в открытом интервале (0 °C, 100 °C), т. е. $X=\{x \mid 0\ °C < x < 100\ °C\}$. Выбор этого интервала вполне оправдан с физической точки зрения, поскольку именно в этом диапазоне температур кофе потенциально может существовать как напиток. Очевидно, что отдельная чашка кофе, скажем x_1, с температурой 10 °C не может быть признана горячей, поэтому для нее значение функции принадлежности рассматриваемому множеству $\mathcal{B}$ будет равно нулю, т. е. $\mu_{\mathcal{B}}(x_1)=0$. С другой стороны, другая чашка кофе x_2 с температурой 90 °C вполне может быть признана горячей, поэтому для нее значение функции принадлежности рассматриваемому множеству $\mathcal{B}$ будет равно 1, т. е. $\mu_{\mathcal{B}}(x_2)=1$.

Что касается значений температур, заключенных между этими крайними значениями, то ситуация представляется уже не столь однозначной. Более того, она по своей сути является исключительно субъективной и неопределенной, поскольку чашка кофе с температурой 55 °C для одного индивидуума может оказаться горячей, а для другого — не слишком горячей. Именно в этом и проявляется *нечеткость* задания соответствующего множества. Тем не менее, мы можем быть вполне уверены в общем виде функции принадлежности, а именно — в том, что

соответствующая функция принадлежности является *монотонно возрастающей* (или более строго — монотонно неубывающей).

Таким образом, в качестве множества $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$, описывающего горячий кофе, можно рассматривать, например, такое нечеткое множество, для которого функция принадлежности имеет следующий вид (см. рис. 2.4, *а* и/или 2.4, *б*).

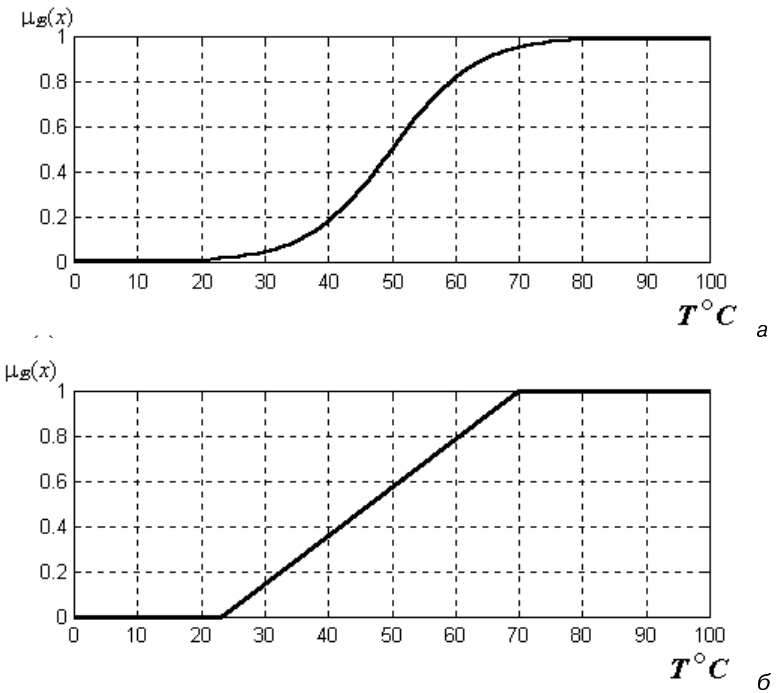


Рис. 2.4. Графики вариантов функций принадлежности для нечеткого множества $\mathcal{B}$, описывающего "*горячий кофе*"

Примечание

Рассмотренный пример допускает обобщение на другие ситуации, связанные с представлением аналогичной нечеткой информации. В частности, целый ряд свойств технических устройств, бытовых приборов и социальных явлений могут инициировать похожие нечеткие множества. Например, такие фразы, как "*скоростной автомобиль*", "*высокооплачиваемая работа*", "*благоустроенная квартира*", "*щедрые чаевые*", "*престижный район*", "*вкусный ужин*" порождают нечеткие множества, аналогичные рассмотренному в примере 2.2. При этом общий вид функций принадлежности таких множеств будет подобен изображенным на рис. 2.4, *а*, *б*.

П р и м е р 2.3. Следующий пример связан с распознаванием букв некоторого алфавита и десятичных цифр, что является весьма актуальной задачей при сканировании текстовых документов. Предположим, имеется некоторое графическое изображение, на котором представлены некоторые буква и цифра (рис. 2.5).

Рис. 2.5. Графическое изображение некоторой буквы (*а*) и некоторой десятичной цифры (*б*)

Первое изображение порождает на множестве всех прописных букв (например, русского) алфавита X={А, Б, В,..., Я} некоторое конечное нечеткое множество C={<А, μ_C(А)>, <Б, μ_C(Б)>,..., <Я, μ_C(Я)>}. Это нечеткое множество содержательно описывает соответствие изображения, представленного на рис. 2.5, *а*, той или иной букве русского алфавита. Таким множеством может быть, например следующее нечеткое множество: C={<А, 0>, <Б, 0>,...,<И, 1.0>, <Й, 0.9>, <К, 0.4>, <Л, 0>, <М, 1.0>, <Н, 1.0>, <О, 0>,...,<Х, 0.3>,...,<Я, 0>}. Пропущенные элементы соответствуют нулевым значениям функции принадлежности для остальных букв алфавита.

Второе изображение порождает на множестве всех десятичных цифр X={0, 1, 2, 3, 4, 5, 6, 7, 8, 9} конечное нечеткое множество $\mathcal{D}$={<0, $\mu_{\mathcal{D}}$(0)>, <1, $\mu_{\mathcal{D}}$(1)>,..., <9, $\mu_{\mathcal{D}}$(9)>}. Это нечеткое множество содержательно описывает соответствие изображения, представленного на рис. 2.5, *б*, той или иной десятичной цифре. В частном случае таким нечетким множеством может быть, например, следующее: C={<0, 0.8>, <1, 0>, <2, 0>, <3, 0.9>, <4, 0>, <5, 0.2>, <6, 1.0>, <7, 0>, <8, 1.0>, <9, 0.9>}. Здесь указаны все значения функции принадлежности для элементов универсума.

Примечание

Рассмотренные выше примеры иллюстрируют характерные аспекты неопределенности, которые встречаются в практике нечеткого моделирования. Во-первых, каждое из нечетких множеств допускает в общем случае неоднозначное представление, что отражает субъективную точку зрения на моделирование соответствующих практических ситуаций. Другими словами, если кто-то не согласен с конкретным вариантом задания нечетких множеств $\mathcal{A}$, $\mathcal{B}$, и C, то он/она могут предложить свои варианты значений функций принадлежности. И формально все будут по-своему правы, поскольку адекватность этих представлений обуславливается их последующим практическим использованием для решения той или иной задачи. Во-вторых, эти примеры хорошо иллюстри-

руют концептуальное различие между теорией нечетких множеств и теорией вероятностей, поскольку рассмотренные варианты неопределенности имеют не стохастический характер. И, наконец, в-третьих, выбор аналитической функции или вида кривой для той или иной функции принадлежности с целью задания соответствующего нечеткого множества зачастую определяется соображениями удобства и простоты.

Перейдем к рассмотрению основных характеристик нечетких множеств, которые используются для их более детального описания и систематического изучения.

2.2. Основные характеристики нечетких множеств

Пусть $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$ — произвольное нечеткое множество (конечное или бесконечное) с элементами из универсума $\boldsymbol{X}$ и функцией принадлежности $\mu_{\mathcal{A}}(x)$.

М н о ж е с т в о α-у р о в н я. Обобщением носителя нечеткого множества является понятие *множества α-уровня*, под которым понимается обычное множество A_α, удовлетворяющее следующему условию: $A_\alpha=\{x\in\boldsymbol{X} \mid \mu_{\mathcal{A}}(x) \geq \alpha\}$, где α — некоторое действительное число из интервала [0, 1], т. е. $\alpha\in[0, 1]$.

Примечание

Иногда можно встретить также определение *множества строгого α-уровня*, которое отличается строгим неравенством в соответствующем условии: $A_\alpha=\{x\in\boldsymbol{X} \mid \mu_{\mathcal{A}}(x) >\alpha\}$. Очевидно, в этом случае носитель произвольного нечеткого множества есть его множество строгого 0-уровня, т. е. справедливо формальное равенство: $A_0 = A_S$.

В качестве примера рассмотрим определенное выше нечеткое множество $\mathcal{A}$, представляющее в некотором контексте "*небольшое натуральное число*" и равное: $\mathcal{A}=\{<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>\}$. Тогда некоторые из его множеств α-уровня равны: $A_{0.8}=\{1, 2, 3, 4\}$, $A_{0.5}=\{1, 2, 3, 4, 5, 6\}$, $A_{0.1}=\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Графически множества α-уровня для конечного нечеткого множества удобно представить с помощью вложенных диаграмм Венна. В этом случае каждая из окружностей будет соответствовать отдельному множеству α-уровня, а элементы каждого из множеств α-уровня размещаются внутри соответствующей окружности (рис. 2.6).

В случае бесконечных нечетких множеств для построения множеств α-уровня можно поступить следующим образом. На графике соответствующей функции принадлежности следует провести прямую линию $y=\alpha$. После чего выделить на оси X те точки или интервалы, для которых отдельные части графика расположены выше этой линии.

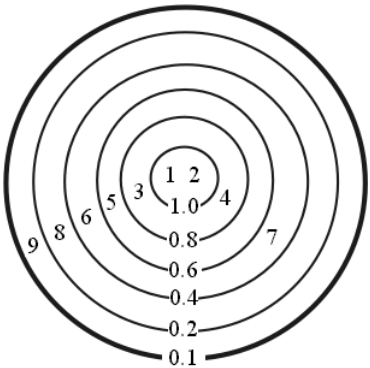


Рис. 2.6. Графическое изображение различных множеств α-уровня с помощью вложенных диаграмм Венна для конечного нечеткого множества "*небольшое натуральное число*"

Так, если в качестве примера рассмотреть бесконечное нечеткое множество $\mathcal{B}$, которое представляет "*действительное число, приближенно равное нулю*", с функцией принадлежности, график которой изображен на рис. 2.7, *а*, то описанным выше способом можно получить, например, его множество 0.5-уровня (рис. 2.7, *б*). Как можно заметить, в данном случае $B_{0.5} = [-0.5, 0.5]$.

Очевидно, для множеств α-уровня произвольного нечеткого множества $\mathcal{A}$ справедливо следующее свойство: если $\alpha_1 \geq \alpha_2$, то $A_{\alpha 1} \subseteq A_{\alpha 2}$.

Высота нечеткого множества. Величина $h_{\mathcal{A}} = \sup\{\mu_{\mathcal{A}}(x)\}$, где супремум берется по всем значениям функции принадлежности для $x \in \boldsymbol{X}$, называется *высотой* нечеткого множества $\mathcal{A}$. Согласно этому определению, нечеткое множество $\mathcal{A}$ пусто, если его высота в точности равна 0, т. е. $h_{\mathcal{A}} = 0$.

Например, высота конечного нечеткого множества $\mathcal{A}$ "*небольшое натуральное число*" равна 1 и соответствует двум элементам универсума: 1 и 2. Высота нечеткого множества $\mathcal{B}$, которое представляет "*действительное число, приближенно равное нулю*", также равна 1 и $\mu_{\mathcal{B}}(0)=1$.

Рассмотрим в качестве еще одного примера бесконечное нечеткое множество C, которое представляет "*большое действительное число*", с функцией принадлежности, заданной следующим математическим выражением:

$\mu_C(x) = 0$ для $x \in [0, 1)$ и $\mu_C(x) = \dfrac{x-1}{x}$ для $x \in \mathbb{R}_+ \backslash [0, 1)$. Высота этого нечеткого множества также равна 1, однако среди элементов универсума $\boldsymbol{X}=\mathbb{R}_+$ отсутствуют числа, для которых $\mu_C(x) = 1$ (рис. 2.8). Действительно, какое бы число мы не рассмотрели, соответствующее значение функции принадлежности всегда будет строго меньше 1.

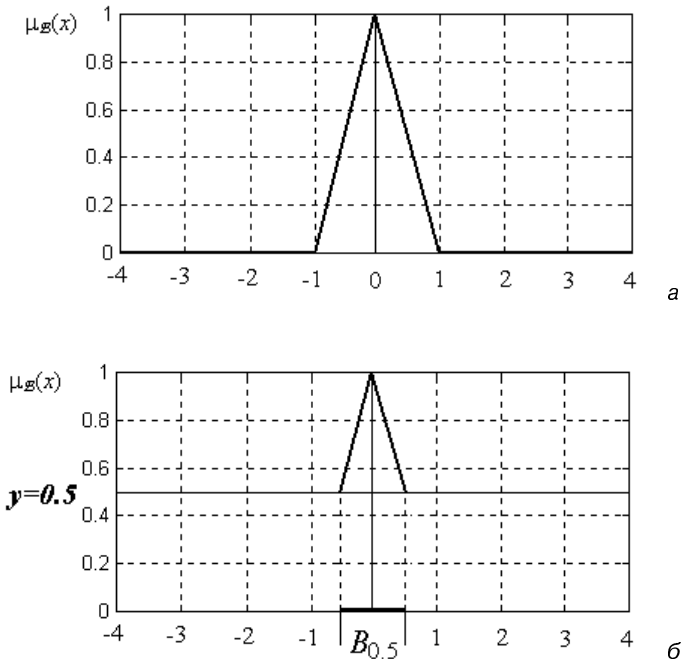


Рис. 2.7. Графическое изображение функции принадлежности бесконечного нечеткого множества "*действительное число, приближенно равное нулю*" (*а*) и его множества 0.5-уровня (*б*)

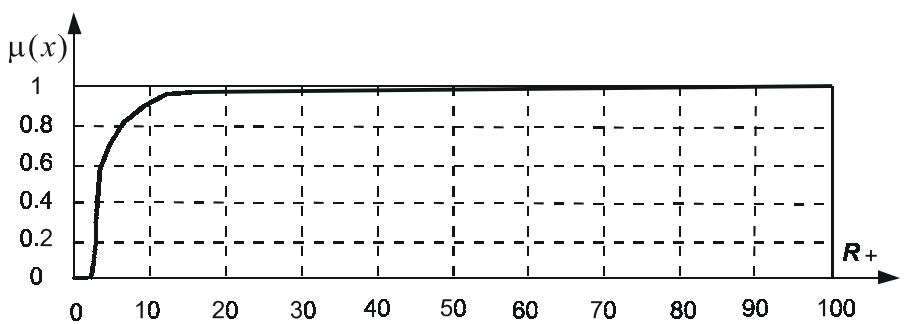


Рис. 2.8. График функции принадлежности бесконечного нечеткого множества *C*, которое представляет "*большое вещественное число*"

Особенность определения высоты заключается в том, что высота нечеткого множества всегда существует и равна некоторому действительному числу из интервала [0, 1], которому может соответствовать несколько элементов универсума. Действительно, для конечных нечетких множеств высота всегда равна мак-

симальному значению их функций принадлежности. Для бесконечных нечетких множеств область значений соответствующих функций принадлежности всегда является компактным множеством, т. к. является подмножеством интервала [0, 1]. А поскольку для произвольного компактного множества всегда существует наименьшая верхняя грань, то она и принимается по определению за высоту нечеткого множества.

Примечание

При определении высоты нечеткого множества использована специальная функция $y=sup(f)$, которая получила свое название от латинского *supremum* — наивысшее и называется *верхней гранью* (или *наименьшей верхней гранью*). Как будет видно из дальнейших рассуждений, эта функция отличается от похожей на нее функции max(*f*). Формально функция y=sup(*f*) определяется для обычных множеств следующим образом. Рассмотрим произвольное отображение $f: D \to \mathbb{R}$, где D — область определения этого отображения ($D \subseteq \boldsymbol{X}$). Отображение f называется *ограниченным сверху (снизу)* на множестве D, если существует конечное число $k \in \mathbb{R}$, такое что выполняется условие: $f(x) \le k$ (соответственно, $f(x) \ge k$) ($\forall x \in D$). При этом отображение $f(x)$ называется *ограниченным* на множестве D, если оно одновременно ограничено на D сверху и снизу. Далее рассмотрим некоторое ограниченное сверху отображение $f(x)$, для которого $D \subseteq \mathbb{R}$, т. е. ограниченную сверху функцию $f(x)$. В общем случае числовые значения $k \in \mathbb{R}$, для которых выполняется условие: $f(x) \le k$, $\forall x \in D$, образуют некоторое числовое подмножество $U \subseteq \mathbb{R}$, при этом очевидно, что $U \neq \varnothing$. Если среди всех $k \in U$ найдется некоторое наименьшее значение, обозначим его через k_s, то оно называется *наименьшей верхней гранью* функции $f(x)$ на множестве $D \subseteq \mathbb{R}$ и обозначается через k_s= sup(*f*) (читается "супремум *f* на множестве *D*"). Очевидно, если $k_s \in Im_f$, то это значение одновременно является и *максимумом* функции $f(x)$ на множестве $D \subseteq \mathbb{R}$, т. е. k_s= max(*f*) и тогда значения этих двух функций sup(*f*) и max(*f*) совпадают. С другой стороны, может оказаться, что в множестве Im_f не существует такого k_s, для которого выполнялось бы условие: $f(x) \le k_s$, $\forall x \in D$, т е. $k_s \notin Im_f$. Введение в рассмотрение функции sup(*f*) всегда гарантирует существование такого k_s в множестве действительных чисел $\mathbb{R}$, поскольку последнее является непрерывным и вполне упорядоченным множеством. Следует также отметить, что функции sup(*f*) и max(*f*) всегда можно определить на множестве значений рассматриваемых функций *f*, т. е. на Im_f. Таким образом, поскольку область значений любой функции принадлежности ограничена интервалом [0, 1], высота произвольного нечеткого множества всегда существует и это числовое значение принадлежит интервалу [0, 1].

Приведем простой пример. Рассмотрим конкретную числовую функцию — параболу, которую запишем в традиционной нотации: $y=x^2$, а в качестве области определения возьмем два интервала: замкнутый D_1=[–1, 1] и открытый D_2= = (–1, 1). Очевидно, что $\sup\limits_{x \in D_1}(y) = 1$ и $\sup\limits_{x \in D_2}(y) = 1$, при этом $\max\limits_{x \in D_1}(y) = 1$ (достигается при x_1 = –1, x_2 =1 и эти $x_{1,2} \in D_1$), а $\max\limits_{x \in D_2}(y)$ не существует. Действительно,

в открытом интервале (–1, 1) нет такого числа $x \in D_2$, для которого выполнялось бы равенство: $x^2 = 1$.

Нормальное нечеткое множество. Нечеткое множество $\mathcal{A}$ называется *нормальным*, если максимальное значение его функции принадлежности равно 1. Формально это означает, что для нормального нечеткого множества необходимо выполнение следующего условия:

$$\mu_{\mathcal{A}}(x) = 1, \qquad (\exists\, x \in \boldsymbol{X}) \tag{2.3}$$

Например, нечеткое множество $\mathcal{A}$ "*небольшое натуральное число*" является нормальным, поскольку его высота равна 1 и соответствует двум его элементам: 1 и 2. Нечеткое множество $\mathcal{B}$ "*действительное число, приближенно равное нулю*" также является нормальным, поскольку его высота равна 1 и $\mu_{\mathcal{B}}(0)=1$. Напротив, нечеткое множество C "*большое действительное число*" не является нормальным.

Субнормальное нечеткое множество. Если высота нечеткого множества равна единице ($h_{\mathcal{A}}=1$), но условие (2.3) не выполняется, то такое нечеткое множество будем называть *субнормальным*.

Очевидно, нечеткое множество C "*большое действительное число*" является субнормальным.

Другими словами, для субнормального нечеткого множества необходимо лишь, чтобы его высота была равна 1, т. е. выполнялось бы условие: $h_{\mathcal{A}}=1$. Это определение корректно, поскольку в этом случае всякое нормальное нечеткое множество является субнормальным с дополнительным условием (2.3).

Примечание

Ситуация с понятием нормального нечеткого множества не является столь однозначной, поскольку в литературе можно встретить и другие определения понятий нормального и субнормального нечеткого множества. Так, например, нормальным нечетким множеством иногда называют такое, для которого выполняется лишь условие: $\sup(\mu_{\mathcal{A}}(x))=1$ $(\forall x \in \boldsymbol{X})$, а субнормальным нечетким множеством называют нечеткое множество, для которого выполняется условие: $\sup(\mu_{\mathcal{A}}(x))<1$ $(\forall x \in \boldsymbol{X})$. Как нетрудно заметить, для рассматриваемых нами нечетких множеств в смысле определения (2.1) нестрогая форма второго условия выполняется всегда, а значит, подобное определение субнормальности в какой-то мере теряет свой конструктивизм. С другой стороны, в большинстве работ, в которых рассматриваются нечеткие числа и интервалы, определение нормальности последних основано на выполнении условия: $\max(\mu_{\mathcal{A}}(x))=1$ $(\exists x \in \boldsymbol{X})$. Поскольку это противоречит общему определению нормального нечеткого множества, было решено использовать более частное его определение в смысле (2.3).

Унимодальное нечеткое множество. Нечеткое множество $\mathcal{A}$ называется *унимодальным (строго унимодальным)*, если его функция принадлежности $\mu_{\mathcal{A}}(x)$ является унимодальной (строго унимодальной).

В свою очередь произвольная функция принадлежности $\mu(x)$ называется *унимодальной на интервале* $[a, b] \subset \boldsymbol{R}$, если она непрерывна на $[a, b]$, а также существует

некоторый непустой $[c, d] \subset [a, b]$, такой что $a \le c \le d \le b$ и выполняются следующие условия:

- функция $\mu(x)$ строго монотонно возрастает на интервале $[a, c]$ при $a<c$;
- функция $\mu(x)$ строго монотонно убывает на интервале $[d, b]$ при $d<b$;
- функция $\mu(x)$ принимает свое максимальное значение на интервале $[c, d]$, т. е. любая точка $x_m \in [c, d]$ является точкой максимума функции принадлежности относительно интервала $[a, b]$:

$$x_m = \arg \max_{x \in [a, b]} \{\mu(x)\}. \qquad (2.4)$$

В этом случае любая точка $x_m \in \mathcal{A}$ нечеткого множества $\mathcal{A}$, удовлетворяющая условию (2.4), называется *модальным значением* или *модой* нечеткого множества $\mathcal{A}$.

Если в этом определении интервал $[c, d]$ вырождается в точку, т. е. $c=d$, то соответствующая функция принадлежности называется *строго унимодальной на интервале* $[a, b]$.

Функция принадлежности $\mu_{\mathcal{A}}(x)$ называется *унимодальной (строго унимодальной)*, если она унимодальна (строго унимодальна) на носителе соответствующего нечеткого множества $\mathcal{A}$.

Например, рассмотренное выше в примере 2.1 нечеткое множество $\mathcal{A}$ с функцией принадлежности, изображенной на рис. 2.2, является унимодальным, но не является строго унимодальным. Нечеткое множество $\mathcal{B}$ из примера 2.2 является унимодальным на интервале [25 °C, 99 °C], поскольку оно задано на универсуме $\boldsymbol{X} = \{x \mid 0\ °C < x < 100\ °C\}$, но не является строго унимодальным на этом интервале.

Что касается дискретного нечеткого множества C из примера 2.3, то относительно его унимодальности ничего сказать нельзя. Рассматриваемые ниже функции принадлежности трапециевидной формы являются унимодальными, а треугольной формы — строго унимодальными.

Примечание

Следует заметить, что рассмотренное выше определение унимодальности непрерывной функции может быть распространено на случай некоторой дискретной топологии. Действительно, если в качестве интервалов использовать вполне упорядоченные множества, то условие (2.4) остается справедливым. Поскольку это условие сохраняет свой смысл и в случае нечетких множеств с конечным числовым носителем, соответствующее определение унимодальности может быть применено к нечетким множествам, заданным на некотором конечном подмножестве действительных или целых чисел.

Ядро нечеткого множества. *Ядром* нечеткого множество $\mathcal{A}$ называется такое обычное множество A_1, элементы которого удовлетворяют условию: $A_1 = \{x \in \boldsymbol{X} \mid \mu_{\mathcal{A}}(x) = 1\}$.

Например, ядро нечеткого множества $\mathcal{A}$ "*небольшое натуральное число*" равно двухэлементному множеству $A_1 = \{1, 2\}$. Ядро нечеткого множества $\mathcal{B}$ "*действи-*

тельное число, приближенно равное нулю" равно одноэлементному множеству (singleton) $B_1 = \{0\}$. Нечеткое множество C "*большое действительное число*" имеет пустое ядро.

Не трудно заметить, что если произвольное нечеткое множество не является нормальным, то ядро такого нечеткого множества будет пустым. Таким образом, имеет место следующая фундаментальная т е о р е м а. Для того чтобы некоторое нечеткое множество было нормальным, необходимо и достаточно, чтобы оно имело непустое ядро.

Поскольку, как было показано выше, высота нечеткого множества всегда существует, то произвольное непустое нечеткое множество $\mathcal{A}$ всегда можно преобразовать по меньшей мере к субнормальному нечеткому множеству $\mathcal{A}'$ по следующей формуле:

$$\mu_{A'}(x) = \frac{\mu_A(x)}{h_A}. \qquad (2.5)$$

Более того, если в исходном нечетком множестве $\mathcal{A}$ найдется хотя бы один элемент $x \in \mathcal{A}$, для которого значение функции принадлежности равно высоте этого нечеткого множества, т. е. $\mu_{\mathcal{A}}(x) = h_{\mathcal{A}}$, то полученное после преобразования (2.5) нечеткое множество $\mathcal{A}'$ будет нормальным.

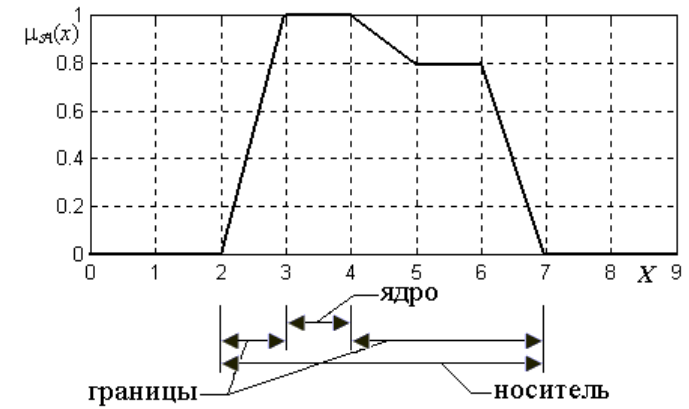
В частности, если исходное нечеткое множество $\mathcal{A}$ является нормальным или субнормальным, то преобразование (2.5) приводит к тривиальному результату. Рассмотрим случай, когда исходное нечеткое множество $\mathcal{A}$ не является пустым и субнормальным. Это означает, что его высота равна некоторому значению из открытого интервала (0, 1), т. е. $h_{\mathcal{A}} \in (0, 1)$. При этом, если $h_{\mathcal{A}} = \mu_{\mathcal{A}}(x)$ для некоторого элемента $x \in \boldsymbol{X}$, то для этого элемента $x \in \boldsymbol{X}$ значение функции принадлежности $\mu_{\mathcal{A}'}(x)$, рассчитанное по формуле (2.5), будет равно 1. Это означает, что нечеткое множество $\mathcal{A}'$ будет нормальным.

Если же $h_{\mathcal{A}} > \mu_{\mathcal{A}}(x)$ для всех элементов $x \in \boldsymbol{X}$, то значение функции принадлежности $\mu_{\mathcal{A}'}(x)$, рассчитанное по формуле (2.5), всегда будет меньше 1. Однако, по свойству наименьшей верхней грани числового множества, высота результирующего нечеткого множества будет равна единице: $h_{\mathcal{A}'} = \sup\{\mu_{\mathcal{A}'}(x)\} = 1$. А это означает, что нечеткое множество $\mathcal{A}'$ будет субнормальным.

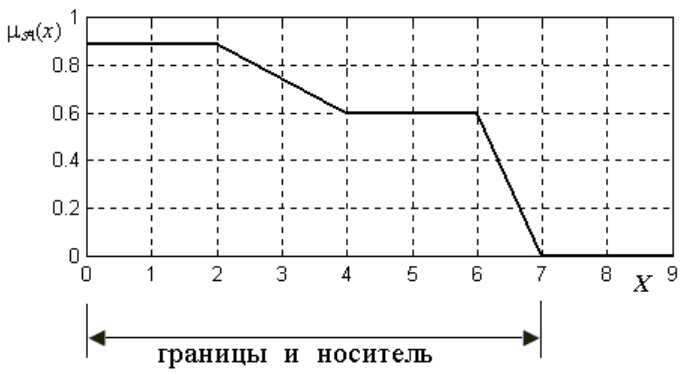
Г р а н и ц ы н е ч е т к о г о м н о ж е с т в а. *Границами* нечеткого множества называются такие элементы универсума, для которых значения функции принадлежности отличны от 0 и 1. Другими словами, границы нечеткого множества $\mathcal{A} = \{x, \mu_{\mathcal{A}}(x)\}$ включают те и только те элементы универсума $x \in \boldsymbol{X}$, для которых выполняется условие: $0 < \mu_{\mathcal{A}}(x) < 1$.

Т о ч к и п е р е х о д а н е ч е т к о г о м н о ж е с т в а. Элементы нечеткого множества $y \in \mathcal{A}$, для которых выполняется условие: $\mu_{\mathcal{A}}(y) = 0.5$, называются *точками перехода* этого нечеткого множества $\mathcal{A}$.

В общем случае введенные в рассмотрение понятия можно проиллюстрировать графически следующим образом (рис. 2.9).



а



б

Рис. 2.9. Ядро, носитель и границы нечетких множеств, одно из которых является нормальным (*а*), а другое — не является нормальным (*б*)

В дополнение к этому рассмотренное в примере 2.1 конечное нечеткое множество $\mathcal{A}$ выходных дней (см. рис. 2.1) имеет непустой носитель $A=\{$*пятница*, *суббота*, *воскресенье*$\}$, является нормальным, поскольку $\mu_{\mathcal{A}}($*суббота*$)=1$.

Рассмотренное в этом же примере 2.1 бесконечное нечеткое множество $\mathcal{A}$ выходных дней (см. рис. 2.2) имеет непустой носитель A_S, которому будет соответствовать открытый интервал действительных чисел, для которых график функции принадлежности лежит выше оси абсцисс. Оно также является нормальным, поскольку $\mu_{\mathcal{A}}(x)=1$.

Б л и ж а й ш е е ч е т к о е м н о ж е с т в о. Часто оказывается полезным понятие четкого множества A, *ближайшего* к нечеткому множеству $\mathcal{A}$. Характеристическая функция такого множества может быть определена следующим выражением:

$$\chi_A(x) = \begin{cases} 0, & \text{если } \mu_A(x) < 0.5 \\ 1, & \text{если } \mu_A(x) > 0.5 \\ 0 \text{ или } 1, & \text{если } \mu_A(x) = 0.5 \end{cases}. \qquad (2.6)$$

Для характеристики нечетких множеств используют также понятие выпуклости, которое ассоциируется с соответствующим графическим изображением функции принадлежности.

В ы п у к л о е н е ч е т к о е м н о ж е с т в о. Нечеткое множество $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$ с универсумом $\boldsymbol{X}$ называют *выпуклым*, если его функция принадлежности $\mu_{\mathcal{A}}(x)$ удовлетворяет следующему неравенству:

$$\mu_{\mathcal{A}}(x) \geq \min\{\mu_{\mathcal{A}}(a), \mu_{\mathcal{A}}(b)\} \qquad (2.7)$$

для любых значений x, a, $b \in \boldsymbol{X}$, при которых $a<x<b$ и $a \neq b$.

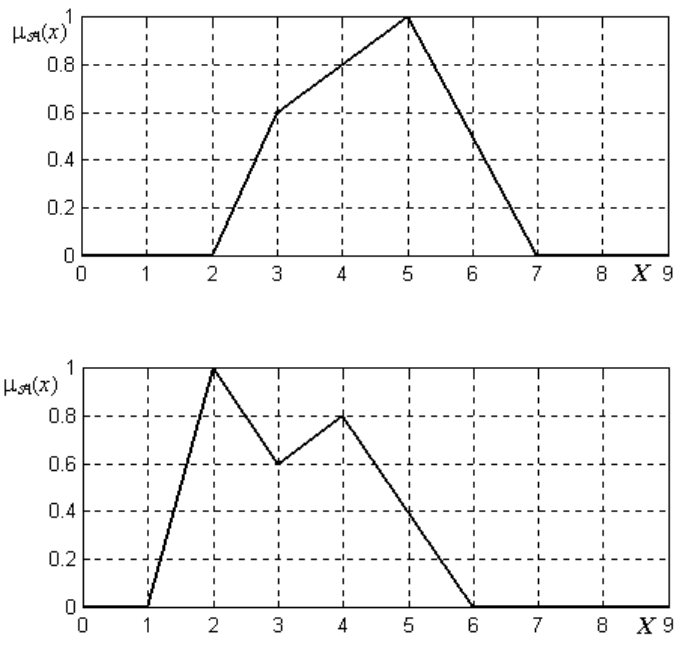


Рис. 2.10. Графики функций принадлежности выпуклого (*а*) и невыпуклого (*б*) нечеткого множества

Примечание

Определение выпуклости для нечетких множеств отличается от известного в анализе, поскольку имеет более общий математический контекст. Тем не менее, его весьма удобно использовать на практике, поскольку кроме непрерывных функций принадлежности оно применимо к конечным нечетким множествам, а также ко множествам, функция принадлежности которых не является непрерывной кривой.

На рис. 2.10 изображены графики двух функций принадлежности, первая из которых является выпуклой, а вторая — не является выпуклой. В связи с рассмотрением этого примера следует заметить, что первая функция принадлежности является строго унимодальной с модой x_m=5, а вторая — не является унимодальной и имеет две моды: x_m=2 и x_m=4.

2.3. Основные типы функций принадлежности

Формальное определение нечеткого множества (2.1) не накладывает никаких ограничений на выбор конкретной функции принадлежности для его представления. Однако на практике удобно использовать те из них, которые допускают аналитическое представление в виде некоторой простой математической функции. Это упрощает не только соответствующие численные расчеты, но и сокращает вычислительные ресурсы, необходимые для хранения отдельных значений этих функций принадлежности. Необходимость типизации отдельных функций принадлежности также обусловлена наличием реализаций соответствующих функций в рассматриваемых далее инструментальных средствах.

Кусочно-линейные функции принадлежности

В качестве первого типа функций принадлежности рассмотрим функции, которые, как следует из их названия, состоят из отрезков прямых линий, образуя непрерывную или кусочно-непрерывную функцию. Наиболее характерным примером таких функций являются "*треугольная*" (рис. 2.11, *а*) и "*трапециевидная*" (рис. 2.11, *б*) функции принадлежности. В нашем случае каждая из этих функций задана на универсуме $\boldsymbol{X}$=[0, 10], в качестве которого выбран замкнутый интервал действительных чисел. В общем случае выбор универсума может быть произвольным и не ограничен никакими правилами.

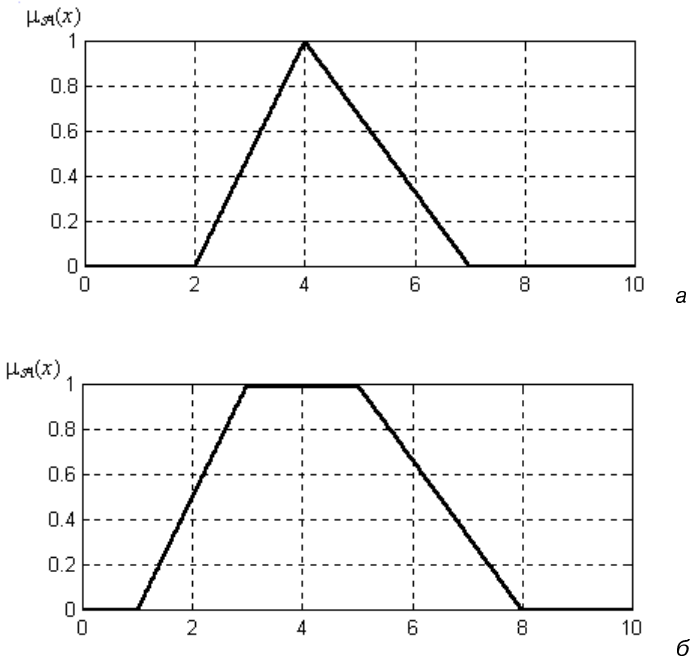


Рис. 2.11. Графики функций принадлежности треугольной (*а*) и трапециевидной (*б*) формы

Первая из этих функций принадлежности в общем случае может быть задана аналитически следующим выражением:

$$f_{\Delta}(x;a,b,c)=\begin{Bmatrix} 0, & x\le a \\ \dfrac{x-a}{b-a}, & a\le x\le b \\ \dfrac{c-x}{c-b}, & b\le x\le c \\ 0, & c\le x \end{Bmatrix}, \qquad (2.8)$$

где a, b, c — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a\le b\le c$.

Применительно к конкретной функции, изображенной на рис. 2.11, *а*, значения параметров равны: $a=2$, $b=4$, $c=7$. Как нетрудно заметить, параметры a и c характеризуют основание треугольника, а параметр b — его вершину. Как можно заметить, эта функция принадлежности порождает нормальное выпуклое унимодальное нечеткое множество с носителем — интервалом (a, c), границами $(a, c)\backslash\{b\}$, ядром $\{b\}$ и модой b.

Трапециевидная функция принадлежности в общем случае может быть задана аналитически следующим выражением:

$$f_{\mathrm{T}}(x;\, a,b,c,d)=\begin{Bmatrix} 0, & x\le a \\ \dfrac{x-a}{b-a}, & a\le x\le b \\ 1, & b\le x\le c \\ \dfrac{d-x}{d-c}, & c\le x\le d \\ 0, & d\le x \end{Bmatrix}, \qquad (2.9)$$

где a, b, c, d — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a\le b\le c\le d$.

Применительно к конкретной функции, изображенной на рис. 2.11, *б*, значения параметров равны: a=1, b=3, c=5, d=8. Как нетрудно заметить, параметры a и d характеризуют нижнее основание трапеции, а параметры b и c — верхнее основание трапеции. При этом данная функция принадлежности порождает нормальное выпуклое нечеткое множество с носителем — интервалом (a, d), границами $(a, b)\cup(c, d)$ и ядром $[b, c]$.

Эти функции используются для задания таких свойств множеств, которые характеризуют неопределенность типа: "*приблизительно равно*", "*среднее значение*", "*расположен в интервале*", "*подобен объекту*", "*похож на предмет*" и др. Они также служат для представления нечетких чисел и интервалов, которые будут рассмотрены в *главе 5*.

Z-образные и *S*-образные функции принадлежности

Эти функции принадлежности также получили свое название по виду кривых, которые представляют их графики. Первая из функций этой группы называется *Z-образной* кривой или *сплайн-функцией* и в общем случае может быть задана аналитически следующим выражением:

$$f_{Z_1}(x;\, a,b)=\begin{Bmatrix} 1, & x<a \\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-a}{b-a}\pi\right), & a\le x\le b \\ 0, & x>b \end{Bmatrix}, \qquad (2.10)$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$. График этой функции для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}$=[0, 10] изображен на

рис. 2.12, *а*, при этом значения параметров соответственно равны a=3, b=6. *Сплайн-функция* может быть также задана другим выражением:

$$f_{Z_2}(x; a, b) = \begin{cases} 1, & x \le a \\ 1-2\left(\dfrac{x-a}{b-a}\right)^2, & a < x \le \dfrac{a+b}{2} \\ 2\left(\dfrac{b-x}{b-a}\right)^2, & \dfrac{a+b}{2} < x < b \\ 0, & b \le x \end{cases}, \qquad (2.11)$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$. График этой функции для некоторого нечеткого множества $\mathcal{A}$ и универсума X=[0, 10] изображен на рис. 2.12, *б*, при этом значения параметров соответственно равны a=3, b=6.

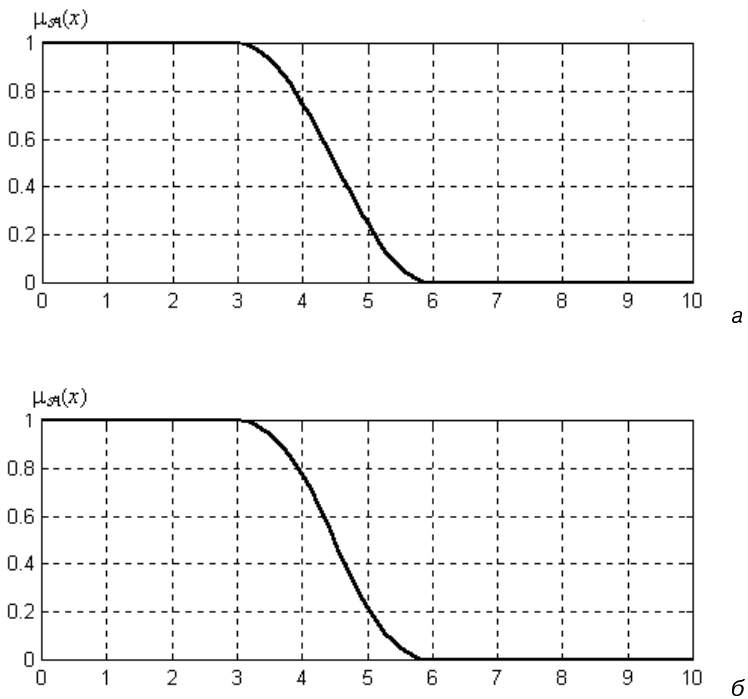


Рис. 2.12. Графики Z-образных функций принадлежности f_{Z1} и f_{Z2} для значений параметров a=3, b=6

Примечание

Хотя на первый взгляд различие между этими функциями едва уловимо, тем не менее оно существует, в чем можно убедиться посредством совмещения их графиков на одном рисунке.

Данные функции принадлежности порождают нормальные выпуклые нечеткие множества с ядром $(-\infty, a]$ и носителем $(-\infty, b)$.

Эти функции используются для представления таких свойств нечетких множеств, которые характеризуются неопределенностью типа: "*малое количество*", "*небольшое значение*", "*незначительная величина*", "*низкая себестоимость продукции*", "*низкий уровень цен или доходов*", "*низкая процентная ставка*" и многих других. Общим для всех таких ситуаций является слабая степень проявления того или иного качественного или количественного признака. Особенность нечеткого моделирования при этом заключается в представлении соответствующих нечетких множеств с помощью невозрастающих (монотонно убывающих) функций принадлежности.

Вторая из функций рассматриваемой группы называется *S-образной* кривой или *сплайн-функцией* и в общем случае может быть задана аналитически следующим выражением:

$$f_{S_1}(x; a, b) = \begin{Bmatrix} 0, & x < a \\ \frac{1}{2} + \frac{1}{2}\cos(\frac{x-b}{b-a}\pi), & a \le x \le b \\ 1, & x > b \end{Bmatrix}, \qquad (2.12)$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$. График этой функции для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}=[0, 10]$ изображен на рис. 2.13, *а*, при этом значения параметров соответственно равны $a=3$, $b=6$. *Сплайн-функция* может быть также задана другим выражением:

$$f_{S_2}(x; a, b) = \begin{Bmatrix} 0, & x \le a \\ 2\left(\frac{x-a}{b-a}\right)^2, & a < x \le \frac{a+b}{2} \\ 1-2\left(\frac{b-x}{b-a}\right)^2, & \frac{a+b}{2} < x < b \\ 1, & b \le x \end{Bmatrix}, \qquad (2.13)$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$. График этой функции для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}=[0, 10]$ изображен на рис. 2.13, *б*, при этом значения параметров соответственно равны $a=3$, $b=6$.

Данные функции принадлежности порождают нормальные выпуклые нечеткие множества с ядром $[b, +\infty)$ и носителем $(a, +\infty)$.

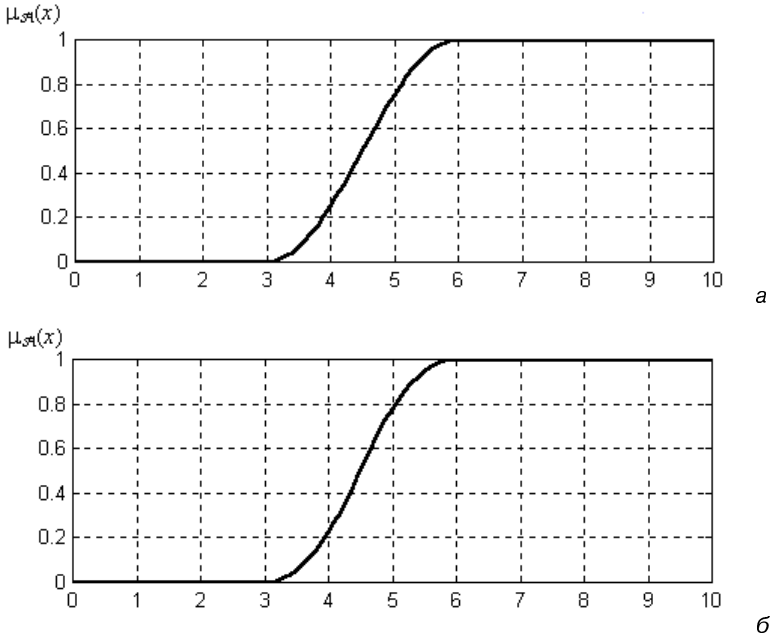


Рис. 2.13. Графики S-образных функций принадлежности f_{S1} и f_{S2} для значений параметров a=3, b=6

К типу *S*-образных и одновременно *Z*-образных функций принадлежности может быть отнесена так называемая *сигмоидальная* функция (сигмоид), которая в общем случае задается аналитически следующим выражением:

$$f_{S_3}(x;a,b)=\frac{1}{1+e^{-a(x-b)}}, \qquad (2.14)$$

здесь a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$, а e — основание натуральных логарифмов, которое инициирует задание соответствующей экспоненциальной функции. При этом в случае $a>0$ может быть получена *S*-образная функция принадлежности, а в случае $a<0$ — *Z*-образная функция принадлежности.

Графики этой функции для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}$=[0, 10] изображены на рис. 2.14. При этом *S*-образной функции принадлежности соответствуют значения параметров a=3, b=6 (рис. 2.14, *а*), а *Z*-образной функции принадлежности соответствуют значения параметров a= –3, b=6 (рис. 2.14, *б*).

Данные функции принадлежности порождают субнормальные выпуклые нечеткие множества с носителем и границей $\mathbb{R}$ и точкой перехода b.

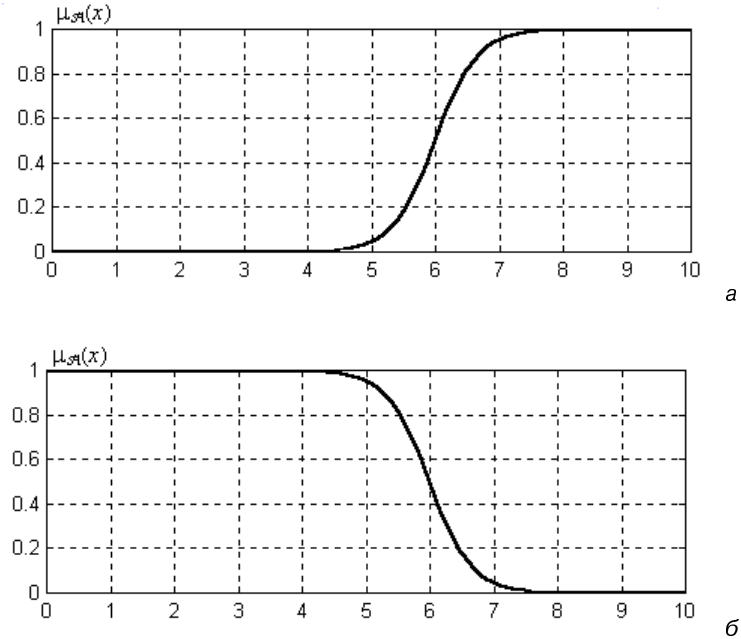


Рис. 2.14. Графики сигмоидальной функции принадлежности f_{S3} для значений параметров a=3, b=6 (*а*) и a= −3, b=6 (*б*)

Рассмотренные *S*-образные функции используются для представления таких нечетких множеств, которые характеризуются неопределенностью типа: "*большое количество*", "*большое значение*", "*значительная величина*", "*высокий уровень доходов и цен*", "*высокая норма прибыли*", "*высокое качество услуг*", "*высокий сервис обслуживания*" и многих других. Общим для всех таких ситуаций является высокая степень проявления того или иного качественного или количественного признака. Особенность нечеткого моделирования при этом заключается в представлении соответствующих нечетких множеств с помощью неубывающих (монотонно возрастающих) функций принадлежности.

В качестве частных случаев *Z*- и *S*-образных кривых удобно рассматривать так называемую *линейную Z-образную* функцию (рис. 2.15, *а*) и *линейную S-образную* функцию (рис. 2.15, *б*). Первая из этих функций в общем случае может быть задана аналитически следующим выражением:

$$f_{\downarrow}(x;a,b)=\begin{Bmatrix}1, & x\le a\\ \dfrac{b-x}{b-a}, & a<x<b\\ 0, & b\le x\end{Bmatrix}, \qquad (2.15)$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$. График этой функции

для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}$=[0, 10] изображен на рис. 2.15, *а*, при этом значения параметров соответственно равны a=3, b=6.

Вторая из этих функций в общем случае может быть задана аналитически следующим выражением:

$$f_{\uparrow}(x;a,b)=\begin{Bmatrix} 0, & x\le a \\ \dfrac{x-a}{b-a}, & a<x<b \\ 1, & b\le x \end{Bmatrix}, \qquad (2.16)$$

где a, b — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$. График этой функции для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}$=[0, 10] изображен на рис. 2.15, *б*, при этом значения параметров соответственно также равны a=3, b=6.

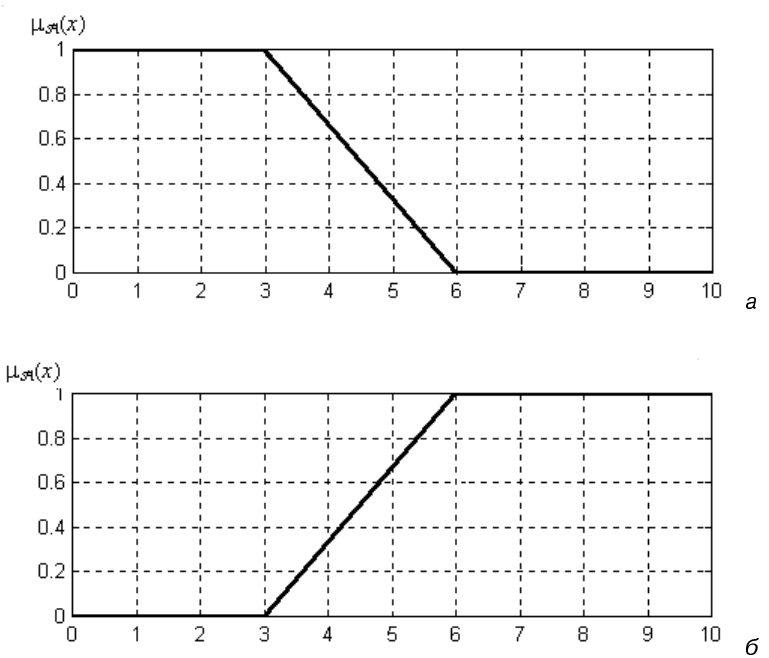


Рис. 2.15. Графики линейной *Z*-образной функции (*а*) и линейной *S*-образной функции (*б*) принадлежности для значений параметров a=3, b=6

Данные функции принадлежности порождают нормальные выпуклые нечеткие множества с границами (a, b).

Следует заметить, что данные линейные *Z*- и *S*-образные функции могут быть использованы для построения рассмотренных выше треугольной и трапециевид-

ной функций принадлежности (см. рис. 2.11). В частности треугольная функция принадлежности получается как композиция линейной Z-образной и линейной S-образной функций по следующей формуле:

$$f_{\Delta}(x; a, b, c) = \min_{x \in X}\{f_{\uparrow}(x; a, b), f_{\downarrow}(x; b, c)\}, \quad (2.17)$$

где a, b, c — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a \le b \le c$. В выражении (2.17) используется операция взятия минимума (обозначенная знаком min) из всех значений, указанных в фигурных скобках через запятую. При этом если соответствующие функциональные значения зависят от некоторой независимой переменной (в нашем случае от x), то под знаком минимума явно указывается диапазон или множество значений этой переменной (в нашем случае — универсум).

Трапециевидная функция принадлежности получается как композиция двух линейных Z-образной и S-образной функций по следующей формуле:

$$f_{T}(x; a, b, c, d) = \min_{x \in X}\{f_{\uparrow}(x; a, b), f_{\downarrow}(x; c, d)\}, \quad (2.18)$$

где a, b, c, d — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a \le b < c \le d$.

П-образные функции принадлежности

К данному типу функций принадлежности можно отнести целый класс кривых, которые по своей форме напоминают колокол, сглаженную трапецию или букву "П".

Первая из подобных функций так и называется — *П-образная функция*, и в общем случае задается аналитически следующим выражением:

$$f_{\Pi}(x; a, b, c, d) = f_{S}(x; a, b) \cdot f_{Z}(x; c, d), \quad (2.19)$$

где a, b, c, d — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a \le b < c \le d$, а знак "·" обозначает обычное арифметическое произведение значений соответствующих функций.

При этом могут быть использованы любые из рассмотренных выше Z- и S-образных функций. В частности, если использовать функции f_{S1} и f_{Z1}, то получим Π-функцию $f_{\Pi 1}$, график которой для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}$=[0, 10] изображен на рис. 2.16, *а*. При этом значения параметров для функции f_{S1} равны a=1, b=4, а для функции f_{Z1} — c=5, d=9. Если же использовать функции f_{S1} и f_{Z2}, то получим Π-функцию $f_{\Pi 2}$, график которой для некоторого нечеткого множества $\mathcal{A}$ и универсума $\boldsymbol{X}$=[0, 10] изображен на рис. 2.16, *б* для тех же значений параметров.

Очевидно, этот тип функций принадлежности порождает нормальные выпуклые нечеткие множества с носителем (a, d) и ядром [b, c].

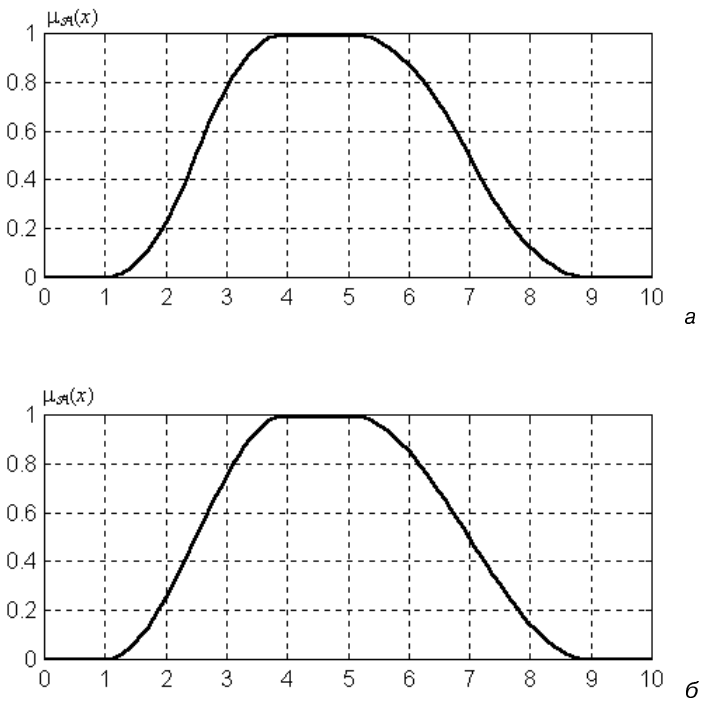


Рис. 2.16. Графики *П*-образных функций принадлежности $f_{П1}$ (*а*) и $f_{П2}$ (*б*) для значений параметров a=1, b=4, c=5, d=9

Следующая функция этого типа *П-образных функций* определяется как произведение двух сигмоидальных функций и в общем случае может быть задана аналитически следующим выражением:

$$f_{П_3}(x; a, b, c, d) = f_{S_3}(x; a, b) \cdot f_{S_3}(x; c, d), \tag{2.20}$$

где a, b, c, d — некоторые числовые параметры, принимающие произвольные действительные значения, причем $a>0$, $c<0$, и упорядоченные отношением: $a \le b < |c| \le d$. Знак "·" обозначает арифметическое произведение значений соответствующих функций, а функция $|x|$ — модуль действительного числа.

К *П*-образным функциям относится также так называемая *колоколообразная* (bell-shaped) *функция*, которая в общем случае задается аналитически следующим выражением:

$$f_{П_4}(x; a, b, c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}, \tag{2.21}$$

где a, b, c — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b<c$, причем параметр $b>0$. Здесь функция $|x|$ обозначает модуль действительного числа.

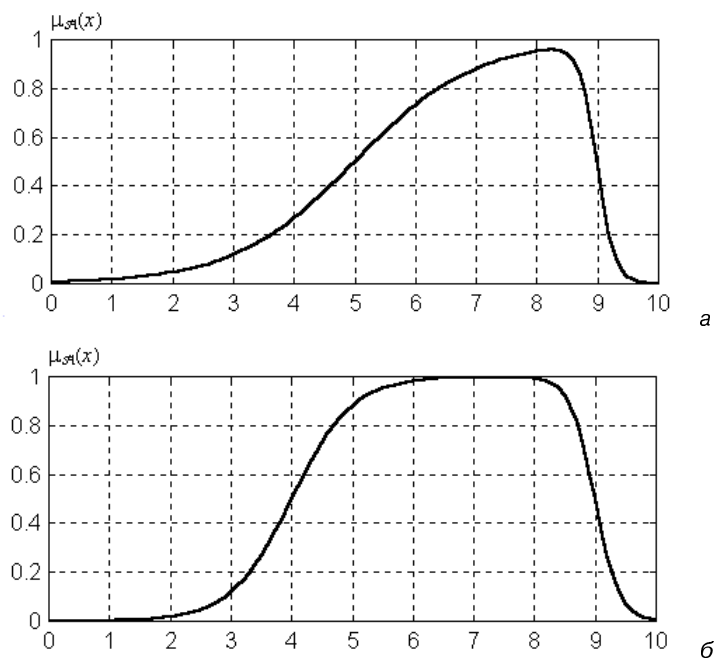


Рис. 2.17. Графики *П*-образных функций принадлежности $f_{П3}$ для значений параметров a=1, b=5, c=-7, d=9 (*а*) и для значений параметров a=2, b=4, c=-5, d=9 (*б*)

Наконец, последней из рассматриваемых функций данного типа является хорошо известная в теории вероятностей функция *плотности нормального распределения* в предположении, что $\sqrt{2\pi}\sigma = 1$, и которая в нашем случае задается аналитически следующим выражением:

$$f_{П_5}(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}. \qquad (2.22)$$

Здесь σ и c — числовые параметры, при этом квадрат первого из них σ^2 в теории вероятностей называется *дисперсией* распределения, а второй параметр c — *математическим ожиданием*.

Очевидно, эти последние типы функций принадлежности порождают нормальные выпуклые нечеткие множества, при этом плотность нормального распределения обеспечивает унимодальность соответствующего нечеткого множества.

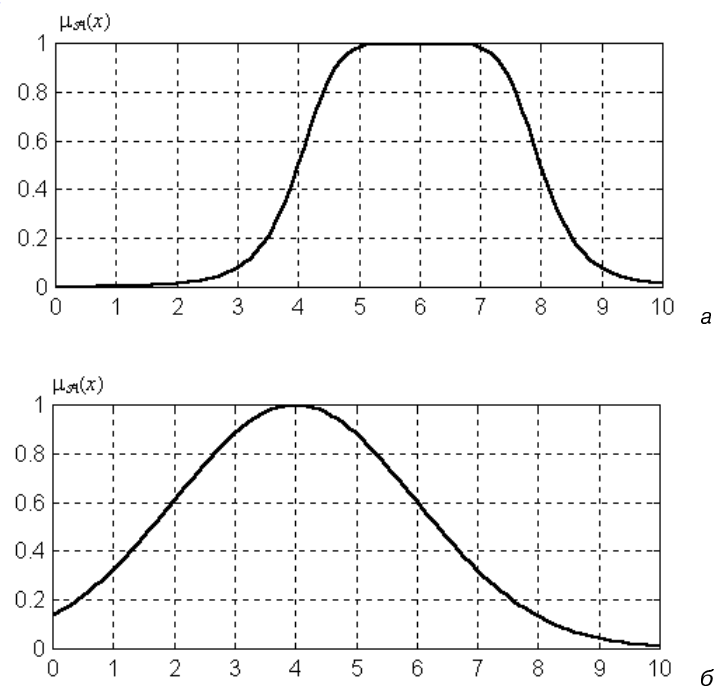


Рис. 2.18. Графики *П*-образных функций принадлежности $f_{П4}$ для значений параметров a=2, b=3, c=6 (*а*) и $f_{П5}$ для значений параметров σ=2, c=4 (*б*)

2.4. Некоторые рекомендации по построению функций принадлежности нечетких множеств

При построении функций принадлежности для нечетких множеств следует придерживаться некоторых правил, которые предопределяются характером неопределенности, имеющей место при построении конкретных нечетких моделей.

С практической точки зрения с каждым нечетким множеством удобно ассоциировать некоторое свойство, признак или атрибут, которые характеризуют рассматриваемую совокупность объектов универсума. При этом по аналогии с классическими множествами рассматриваемое свойство может порождать некоторый предикат (*см. приложение 2)*, который вполне естественно назвать нечетким предикатом. Данный нечеткий предикат может принимать не одно из двух значений истинности ("*истина*" или "*ложь*"), а целый континуум значений истинности, которые для удобства выбираются из интервала [0, 1]. При этом значению "*истина*" по-прежнему соответствует число 1, а значению "*ложь*" — число 0.

Содержательно это означает следующее. Чем в большей степени элемент $x \in \boldsymbol{X}$ обладает рассматриваемым свойством, тем более близко к 1 должно быть значение истинности соответствующего нечеткого предиката. И наоборот, чем в меньшей степени элемент $x \in \boldsymbol{X}$ обладает рассматриваемым свойством, тем более близко к 0 должно быть значение истинности этого нечеткого предиката. Если элемент $x \in \boldsymbol{X}$ *определенно* не обладает рассматриваемым свойством, то соответствующий нечеткий предикат принимает значение "*ложь*" (или число 0). Если же элемент $x \in \boldsymbol{X}$ *определенно* обладает рассматриваемым свойством, то соответствующий нечеткий предикат принимает значение "*истина*" (или число 1).

Тогда в общем случае задание нечеткого множества с использованием специального свойства эквивалентно заданию такой функции принадлежности, которая содержательно представляет степень истинности соответствующего одноместного нечеткого предиката. Более подробно эта взаимосвязь нечетких множеств и нечеткой логики будет рассмотрена в *главе 6*.

Наибольшее распространение при построении функций принадлежности нечетких множеств получили прямые и косвенные методы.

Прямые методы построения функций принадлежности

В прямых методах эксперт либо группа экспертов просто задают для каждого $x \in \boldsymbol{X}$ значение функции принадлежности $\mu_{\mathcal{A}}(x)$. Как правило, прямые методы построения функций принадлежности используются для таких свойств, которые могут быть измерены в некоторой количественной шкале. Например, такие физические величины, как скорость, время, расстояние, давление, температура и другие имеют соответствующие единицы и эталоны для своего измерения. При этом целесообразно ограничить рассмотрение только теми значениями величин, которые имеют физический смысл в контексте решаемой задачи.

При прямом построении функций принадлежности следует учитывать то обстоятельство, что теория нечетких множеств не требует абсолютно точного задания функций принадлежности. Зачастую бывает достаточно зафиксировать лишь наиболее характерные значения и вид (тип) функции принадлежности.

Так, например, если необходимо построить нечеткое множество, которое представляет свойство "*скорость движения автомобиля около 50 км/ч*", на начальном этапе может оказаться достаточным представить соответствующее нечеткое множество треугольной функцией принадлежности f_{Δ} с параметрами a = 40 км/ч, b = 50 км/ч и c = 60 км/ч. Аналогично, в случае построения нечеткого множества для представления свойства "*скорость движения автомобиля находится приблизительно в пределах 50—60 км/ч*", на начальном этапе может оказаться достаточным представить соответствующее нечеткое множество трапециевидной функцией принадлежности f_{T} с параметрами a = 45 км/ч, b = 50 км/ч, c = 60 км/ч и d = = 65 км/ч. В последующем функция принадлежности может быть уточнена опытным путем на основе анализа результатов решения конкретных задач.

Процесс построения или задания нечеткого множества на основе некоторого известного заранее количественного значения измеримого признака получил даже специальное название — *фаззификация* или приведение к нечеткости. Речь идет о том, что хотя иногда нам бывает известно некоторое значение измеримой величины, мы признаем тот факт, что это значение известно неточно, возможно с погрешностью или случайной ошибкой. При этом, чем меньше мы уверены в точности измерения признака, тем большим будет интервал носителя соответствующего нечеткого множества. Следует помнить, что в большинстве практических случаев абсолютная точность измерения является лишь удобной абстракцией для построения математических моделей.

Именно по этой причине фаззификация позволяет более адекватно представить объективно присутствующую неточность результатов физических измерений. Более подробно особенности этого процесса будут рассмотрены далее в *главе 7*.

Косвенные методы построения функций принадлежности

Как правило, косвенные методы определения значений функции принадлежности используются в тех случаях, когда отсутствуют очевидные измеримые свойства, которые могут быть использованы для построения нечетких моделей рассматриваемой предметной области.

Среди косвенных методов наиболее известен так называемый метод попарных сравнений. Этот метод используется для конечных нечетких множеств и основан на следующем предположении. Если бы значения искомой функции принадлежности были известны и равны значениям $\mu_{\mathcal{A}}(x_i)$ для $i \in \{1, 2, ..., n\}$, то попарные сравнения соответствующих элементов носителя нечеткого множества $\mathcal{A}$ можно было бы представить в виде матрицы $\mathbf{A}$ с элементами a_{ij}, при этом элементы этой матрицы равны: $a_{ij} = \mu_{\mathcal{A}}(x_i)/\mu_{\mathcal{A}}(x_j)$, где символ "/" обозначает операцию деления.

На практике бывает проще вначале построить матрицу $\mathbf{A}$ в предположении, что ее диагональные элементы должны быть равны 1, а симметричные относительно главной диагонали элементы должны быть взаимно обратными, т. е. $a_{ij} = 1/a_{ji}$. Последнее условие означает, что если степень принадлежности одного из элементов оценивается в a раз сильнее степени принадлежности другого, то степень принадлежности второго элемента должна быть в $1/a$ раз сильнее степени принадлежности первого элемента.

В этом случае задача построения функции принадлежности сводится к нахождению такого вектора $\boldsymbol{w}$, который является решением следующего уравнения: $\mathbf{A} \cdot \boldsymbol{w} = \lambda_{max}\mathbf{w}$, где λ_{max} — наибольшее собственное значение матрицы $\mathbf{A}$. Поскольку все значения элементов матрицы $\mathbf{A}$ положительны по построению, решение данного уравнения существует и является положительным.

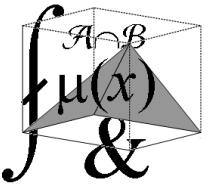
Собственно процесс попарного сравнения элементов может быть основан на субъективной интуиции или на выполнении некоторой последовательности алгоритмических или логических действий. При этом отдельные элементы универ-

сума могут использоваться в качестве эталонов или все элементы могут быть разделены на группы с последующим сравнением этих групп между собой.

Из алгоритмических процедур наибольшую известность получили методы итеративного уточнения значений функций принадлежности, основанные на нейронных сетях и генетических алгоритмах. Логические процедуры используют методы индуктивного обучения и построения нечетких метаправил. Иногда применяются методы обработки статистических данных, факторного и дискриминантного анализа с целью выделения значимых признаков для последующего сравнения элементов рассматриваемого универсума. Заинтересованный читатель более подробное изложение этих вопросов может найти в дополнительной литературе.

В заключение следует отметить, что в случае недостатка информации об особенностях функций принадлежности нечетких переменных рекомендуется начинать построение нечеткой модели с использования наиболее простых форм функции принадлежности, а именно — кусочно-линейных функций. В последствии их характер может быть уточнен и учтен на этапе коррекции нечеткой модели.

Глава 3



Операции над нечеткими множествами

Прежде чем приступить к рассмотрению операций над нечеткими множествами следует привести некоторые важные соображения, которые необходимо принимать во внимание при определении нечетких аналогов обычных теоретико-множественных понятий.

Во-первых, следует иметь в виду, что то или иное нечеткое множество является *обобщением* классического множества. Поскольку в общем случае можно предложить самые различные варианты подобного обобщения, это приводит к принципиальной *неоднозначности* тех или иных определений, имеющих аналогию в классической теории множеств и представляющих практический интерес. Применительно к операциям над нечеткими множествами это означает, что любое определение той или иной операции должно быть справедливым в том частном случае, когда вместо нечетких множеств используются обычные множества. Другими словами, подобные определения должны превращаться в известные определения теоретико-множественных операций, если участвующие в них функции принадлежности заменить характеристическими функциями множеств.

Во-вторых, если при рассмотрении классических множеств *(см. приложение 1)* понятие универсума можно мыслить в форме "*все что угодно*", то сравнение нечетких множеств и выполнение над ними различных операций становится возможным, только когда соответствующие нечеткие множества определены на одном и том же универсуме.

Наконец, в-третьих, поскольку каждое нечеткое множество вполне определяется своей функцией принадлежности, последнее понятие зачастую используется как синоним нечеткого множества. При этом следует помнить, что в общем случае одна и та же функция принадлежности может описывать качественно различные нечеткие множества. С другой стороны, хотя одно и то же нечеткое множество или точнее — то или иное свойство в форме нечеткого множества, может быть представлено различными функциями принадлежности, отражающими неоднозначность субъективных или иных представлений, с формальной точки зрения все из них следует различать и говорить о различных нечетких множествах.

Поэтому, говоря о соответствии нечетких множеств и функций принадлежности, мы будем понимать это соответствие в форме математического изоморфизма.

Именно наличие подобного изоморфизма нечетких множеств, заданных одной и той же функцией принадлежности, позволяет рассматривать формальные определения на требуемом уровне строгости.

3.1. Равенство и доминирование нечетких множеств

По аналогии с обычными множествами, прежде всего, определим два простейших обычных отношения, которые могут иметь место между двумя произвольными нечеткими множествами $\mathcal{A}$ и $\mathcal{B}$, заданными на одном и том же универсуме $\boldsymbol{X}$. Первое из них — равенство двух нечетких множеств.

Равенство нечетких множеств. Два нечетких множества $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$ и $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$ считаются *равными*, если их функции принадлежности принимают равные значения на всем универсуме $\boldsymbol{X}$:

$$\mu_{\mathcal{A}}(x)=\mu_{\mathcal{B}}(x) \text{ для любого } x\in\boldsymbol{X}. \qquad (3.1)$$

Равенство множеств в данном случае записывается как $\mathcal{A}=\mathcal{B}$.

Примечание

Вообще говоря, с практической точки зрения не совсем корректно говорить о равенстве двух нечетких множеств в смысле выполнения только условия (3.1), если речь идет о содержательно различных множествах. Возможно, строго формально следовало бы говорить о математическом изоморфизме таких нечетких множеств. Тем не менее, далее понятие равенства нечетких множеств будет использоваться в смысле (3.1).

Следующим простейшим отношением является понятие нечеткого подмножества (или нечеткого доминирования) произвольных нечетких множеств. Формально это определение также записывается с помощью соответствующих функций принадлежности.

Нечеткое подмножество. Нечеткое множество $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$ является *нечетким подмножеством* нечеткого множества $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$ (записывается как и $\mathcal{A}\subseteq\mathcal{B}$) тогда и только тогда, когда значения функции принадлежности первого не превосходят соответствующих значений функции принадлежности второго, т. е. выполняется следующее условие:

$$\mu_{\mathcal{A}}(x)\le \mu_{\mathcal{B}}(x) \qquad (\forall x\in\boldsymbol{X}). \qquad (3.2)$$

Так же как и для обычных множеств, для обозначения нечеткого подмножества используется символ "$\subseteq$". При этом в случае $\mathcal{A}\subseteq\mathcal{B}$ часто говорят, что нечеткое множество $\mathcal{B}$ *доминирует* нечеткое множество $\mathcal{A}$, а нечеткое множество $\mathcal{A}$ *содержится* в нечетком множестве $\mathcal{B}$.

По аналогии с классическими множествами среди нечетких множеств можно различать два различных варианта доминирования. Рассмотренное выше опре-

деление характерно для так называемого *несобственного* подмножества, когда не исключается случай возможного равенства двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$. Если же в определении нечеткого подмножества исключается равенство соответствующих нечетких множеств в форме (3.1), то в этом случае $\mathcal{A}$ называется *собственным* нечетким подмножеством $\mathcal{B}$ и обозначается: $\mathcal{A} \subset \mathcal{B}$. При этом часто говорят, что нечеткое множество $\mathcal{B}$ *строго* доминирует нечеткое множество $\mathcal{A}$, а нечеткое множество $\mathcal{A}$ *строго* содержится в нечетком множестве $\mathcal{B}$.

Из определения нечеткого подмножества следует, что пустое множество является собственным подмножеством любого нечеткого множества, не являющегося в свою очередь пустым. Другими словами, для любого нечеткого множества $\mathcal{A}$, такого что $\mathcal{A} \neq \varnothing$, всегда справедливо утверждение: $\varnothing \subset \mathcal{A}$, где знак "$\subset$ " понимается в нечетком смысле, поскольку справа от него стоит нечеткое множество. Из этого определения также следует, что любое нечеткое множество, не являющееся в свою очередь универсумом, является собственным подмножеством универсума. То есть для любого нечеткого множества $\mathcal{A}$, такого что $\mathcal{A} \neq \boldsymbol{X}$, всегда справедливо утверждение: $\mathcal{A} \subset \boldsymbol{X}$.

Если для двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных на одном универсуме, не выполняется ни отношение $\mathcal{A} \subseteq \mathcal{B}$, ни отношение $\mathcal{B} \subseteq \mathcal{A}$, то в этом случае говорят, что нечеткие множества $\mathcal{A}$ и $\mathcal{B}$ *несравнимые*.

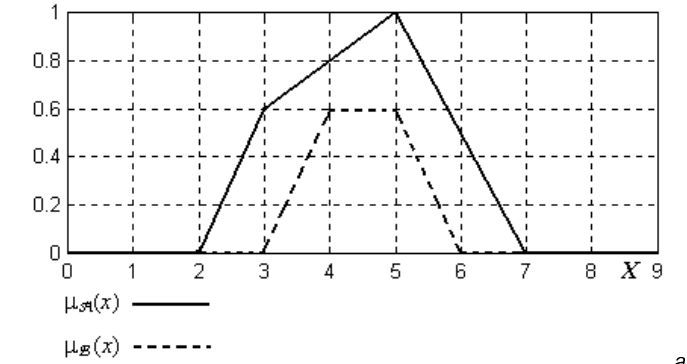
Так, например, для конечных нечетких множеств $\mathcal{A}_1$ и $\mathcal{A}_2$, каждое из которых представляет в некотором контексте "*небольшое натуральное число*", и равные: $\mathcal{A}_1 = \{<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>\}$ и $\mathcal{A}_2 = \{<1, 1.0>, <2, 0.9>, <3, 0.8>, <4, 0.7>, <5, 0.5>, <6, 0.4>, <7, 0.3>, <8, 0.2>, <9, 0.1>\}$, справедливо следующее отношение доминирования: $\mathcal{A}_2 \subseteq \mathcal{A}_1$.

Нечеткое доминирование или факт включения элементов одного нечеткого множества в другое нечеткое множество можно изобразить графически в декартовой системе координат на плоскости. С этой целью изобразим прямоугольную систему координат, на оси абсцисс которой в том или ином порядке расположим элементы универсума $\boldsymbol{X}$, а на оси ординат — соответствующие им значения функции принадлежности рассматриваемого нечеткого множества.

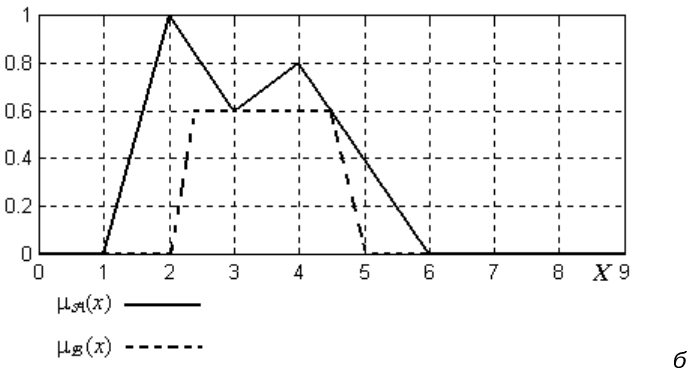
Примечание

Подобное графическое изображение уже было использовано нами в *главе 2* при рассмотрении функций принадлежности нечетких множеств. Очевидно, этот способ наиболее удобен, когда в качестве универсума выступает некоторое подмножество действительных чисел $\mathbb{R}$.

Для случая $\mathcal{B} \subseteq \mathcal{A}$ график функции принадлежности нечеткого множества $\mathcal{B}$ будет расположен по вертикальной оси не выше графика функции принадлежности нечеткого множества $\mathcal{A}$ (рис. 3.1, *а*, *б*). Более того, как на рис. 3.1, *а*, так и на рис. 3.1, *б*, изображены случаи строгого доминирования $\mathcal{B} \subset \mathcal{A}$.



а



б

Рис. 3.1. Различные варианты отношения доминирования $\mathcal{B} \subseteq \mathcal{A}$ двух нечетких множеств

3.2. Операции пересечения, объединения и разности нечетких множеств

Пусть $\mathcal{A}$ и $\mathcal{B}$ — произвольные (конечные или бесконечные) нечеткие множества, заданные на одном и том же универсуме X.

П е р е с е ч е н и е. *Пересечением* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ будем называть некоторое третье нечеткое множество C, заданное на этом же универсуме X, функция принадлежности которого определяется по следующей формуле:

$$\mu_C(x) = \min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)\} \quad (\forall x \in X). \tag{3.3}$$

Операция пересечения нечетких множеств по аналогии с обычными множествами обозначается знаком "$\cap$". В этом случае результат операции пересечения двух нечетких множеств записывается в виде: $C = \mathcal{A} \cap \mathcal{B}$.

В этом случае $C=\{x|\mu_C(x)\}$ — нечеткое множество с функцией принадлежности $\mu_C(x)$, которая определяется по формуле (3.3). Как нетрудно заметить, пересечение $\mathcal{A}\cap\mathcal{B}$ есть наибольшее нечеткое подмножество C, которое содержится одновременно в нечетких множествах $\mathcal{A}$ и $\mathcal{B}$.

Операцию пересечения нечетких множеств в смысле (3.3) иногда называют min-пересечением или $\wedge$-пересечением. Последнее обозначение связано с определением логической операции "И", которая в математической логике обозначается знаком "$\wedge$" *(см. приложение 2)*. Соответственно функция принадлежности пересечения $\mu_C(x)$ в этом случае записывается в виде: $\mu_C(x)=\mu_{\mathcal{A}}(x)\wedge\mu_{\mathcal{B}}(x)$ $(\forall x\in\boldsymbol{X})$. При этом знак "$\wedge$" используется в качестве синонима операции нахождения минимального значения. Поскольку в практике нечеткого моделирования эта операция используется наиболее часто, в дальнейшем, говоря о пересечении нечетких множеств, если явно не указано другое, мы будем иметь в виду min-пересечение ($\wedge$-пересечение).

Операция min-пересечения нечетких множеств корректна в том смысле, что она сохраняет свое определение для случая обычных множеств. А именно, если в качестве нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ взять обычные множества A и B как их частный случай, то определение операции пересечения (3.3) превратится в определение операции пересечения (П1.4) для характеристических функций последних.

В качестве примера рассмотрим конечное нечеткое множество $\mathcal{A}$, которое представляет в некотором контексте свойство "*небольшое натуральное число*", и равно: $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}, и конечное нечеткое множество $\mathcal{B}$, которое представляет свойство "*натуральное число, приближенно равное двум*", и равно: $\mathcal{B}$ ={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}. Тогда нечеткое множество C как результат операции пересечения $C=\mathcal{A}\cap\mathcal{B}$ будет равно: C={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}. Содержательно нечеткое множество C может представлять в этом же контексте "*небольшое натуральное число, приближенно равное двум*".

Результат операции пересечения двух и большего числа нечетких множеств, заданных на одном и том же универсуме $\boldsymbol{X}$, также можно изобразить графически в декартовой системе координат на плоскости. Этот способ особенно удобен для визуализации операций с бесконечными нечеткими множествами. В данном случае каждое из нечетких множеств изображается соответствующей функцией принадлежности, а функция принадлежности результата операции пересечения изображается утолщенной линией. Для дополнительной наглядности область, расположенная ниже значений результирующей функции принадлежности, изображается затемненной.

Для случая пересечения двух нечетких множеств $\mathcal{A}\cap\mathcal{B}$, заданных различными функциями принадлежности, результат операции изображен на рис. 3.2, *а*, *б*. При этом линейные Z-образная и S-образная функции принадлежности имеют параметры a=3, b=6, а Π-образные функции принадлежности — a=1, b=3, c=4, d=7 и a=3, b=6, c=7, d=9 соответственно.

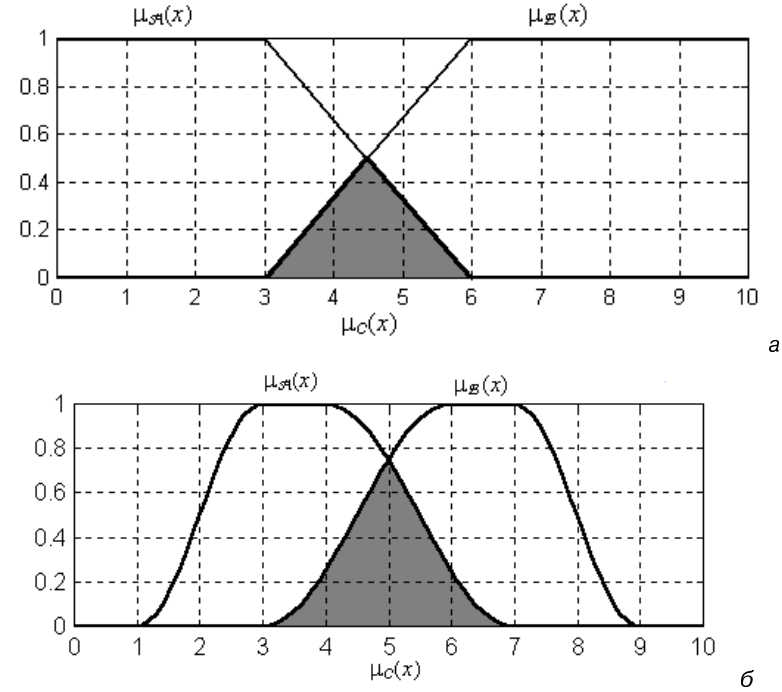


Рис. 3.2. Графическое представление операции пересечения двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Для выпуклых нечетких множеств имеет место следующее свойство. Если нечеткие множества $\mathcal{A}$ и $\mathcal{B}$ — выпуклые, что их пересечение $\mathcal{A} \cap \mathcal{B}$ также является выпуклым нечетким множеством. Рис. 3.2 поясняет и данное свойство, поскольку изображенные на нем исходные нечеткие множества и результат операции пересечения являются выпуклыми.

О б ъ е д и н е н и е. *Объединением* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ называется некоторое третье нечеткое множество $\mathcal{D}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{D}}(x) = \max\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)\} \qquad (\forall x \in \boldsymbol{X}). \tag{3.4}$$

Операция объединения нечетких множеств по аналогии с обычными множествами обозначается знаком "$\cup$". В этом случае результат операции объединения двух нечетких множеств записывается в виде: $\mathcal{D} = \mathcal{A} \cup \mathcal{B}$.

В этом случае $\mathcal{D} = \{x | \mu_{\mathcal{D}}(x)\}$ — нечеткое множество с функцией принадлежности $\mu_{\mathcal{D}}(x)$, которая определяется по формуле (3.4). Как нетрудно заметить, объеди-

нение $\mathcal{A} \cup \mathcal{B}$ есть наименьшее нечеткое множество $\mathcal{D}$, которое доминирует одновременно как $\mathcal{A}$, так и $\mathcal{B}$.

Операцию объединения нечетких множеств в смысле (3.4) иногда называют max-объединением или $\vee$-объединением. Последнее обозначение связано с определением логической операции "ИЛИ" (неисключающего ИЛИ), которая в математической логике обозначается знаком "$\vee$" *(см. приложение 2)*. Соответственно функция принадлежности объединения $\mu_{\mathcal{D}}(x)$ в этом случае часто записывается в виде: $\mu_{\mathcal{D}}(x)=\mu_{\mathcal{A}}(x) \vee \mu_{\mathcal{B}}(x)$ $(\forall x \in \boldsymbol{X})$. При этом знак "$\vee$" используется в качестве синонима операции максимума. Поскольку в практике нечеткого моделирования эта операция используется наиболее часто, в дальнейшем, говоря об объединении нечетких множеств, если явно не указано другое, мы будем иметь в виду их max-объединение ($\vee$-объединение).

Операция max-объединения нечетких множеств также корректна в том смысле, что она сохраняет свое определение для случая обычных множеств. А именно, если в качестве нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ взять обычные множества A и B как их частный случай, то определение операции объединения (3.4) превратится в определение операции объединения (П1.5) для характеристических функций обычных множеств.

В качестве примера рассмотрим нечеткое множество $\mathcal{A}$, которое, как и выше, представляет в некотором контексте "*небольшое натуральное число*", и равно: $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}, и нечеткое множество $\mathcal{B}$, которое представляет "*натуральное число, приближенно равное двум*", и равно: $\mathcal{B}$ ={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}. Тогда нечеткое множество $\mathcal{D}$ как результат операции объединения $\mathcal{D}=\mathcal{A} \cup \mathcal{B}$ будет равно: $\mathcal{D}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}. Содержательно нечеткое множество $\mathcal{D}$ может представлять в этом же контексте "*небольшое натуральное число или натуральное число, приближенно равное двум*".

Результат операции объединения двух и большего числа нечетких множеств, заданных на одном и том же универсуме $\boldsymbol{X}$, можно изобразить графически в декартовой системе координат на плоскости. Для случая объединения двух нечетких множеств $\mathcal{A} \cup \mathcal{B}$, заданных различными функциями принадлежности, результат операции изображен на рис. 3.3, *а, б*.

Р а з н о с т ь. *Разностью* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ называется некоторое третье нечеткое множество $\mathcal{E}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{E}}(x) = \max\{\mu_{\mathcal{A}}(x) - \mu_{\mathcal{B}}(x),\ 0\} \qquad (\forall x \in \boldsymbol{X}), \tag{3.5}$$

где под знаком максимума используется обычная операция арифметической разности двух чисел. Операция разности двух нечетких множеств по аналогии с обычными множествами обозначается знаком "\". В этом случае результат операции разности двух нечетких множеств можно записать в виде: $\mathcal{E}=\mathcal{A} \backslash \mathcal{B}$.

Так, если, как и выше, рассмотреть нечеткое множество $\mathcal{A}$, равное: $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}, и нечеткое множество $\mathcal{B}$, равное: $\mathcal{B}$ ={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}, то разность $\mathcal{A}\backslash\mathcal{B}$ будет равна: $\mathcal{A}\backslash\mathcal{B}$={<1, 0.5>, <2, 0>, <3, 0.3>, <4, 0.4>, <5, 0.4>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}. Содержательно нечеткое множество $\mathcal{A}\backslash\mathcal{B}$ может представлять в том же контексте "*небольшое натуральное число, не являющееся приближенно равным двум*". Для этих двух нечетких множеств разность $\mathcal{B}\backslash\mathcal{A}$ будет равна пустому множеству, поскольку все значения функции принадлежности результата будут равны нулю. Содержательно нечеткое множество $\mathcal{B}\backslash\mathcal{A}$ может представлять в том же контексте "*натуральное число, приближенно равное двум и не являющееся небольшим*".

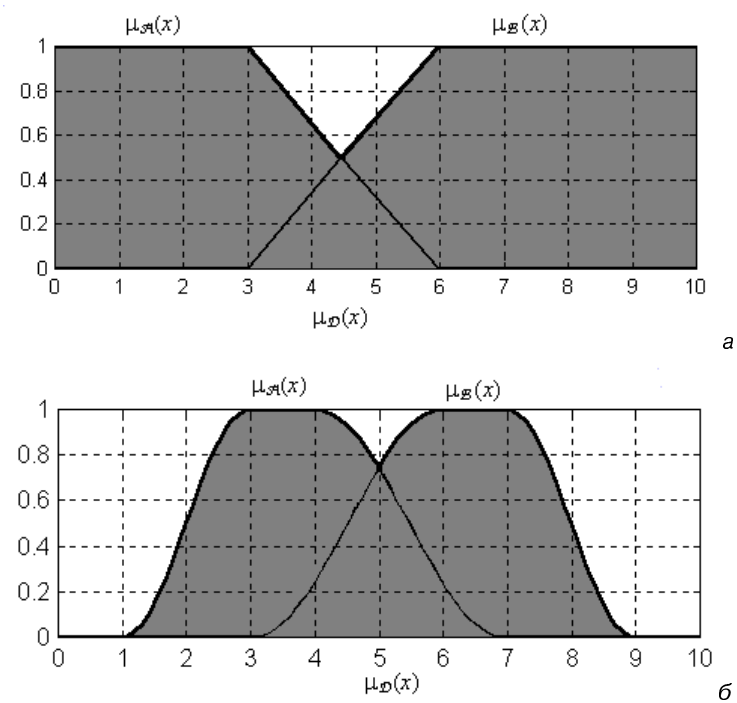


Рис. 3.3. Графическое представление операции объединения двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Результат операции разности двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных на одном и том же универсуме X различными функциями принадлежности, изображен на рис. 3.4, *а, б*.

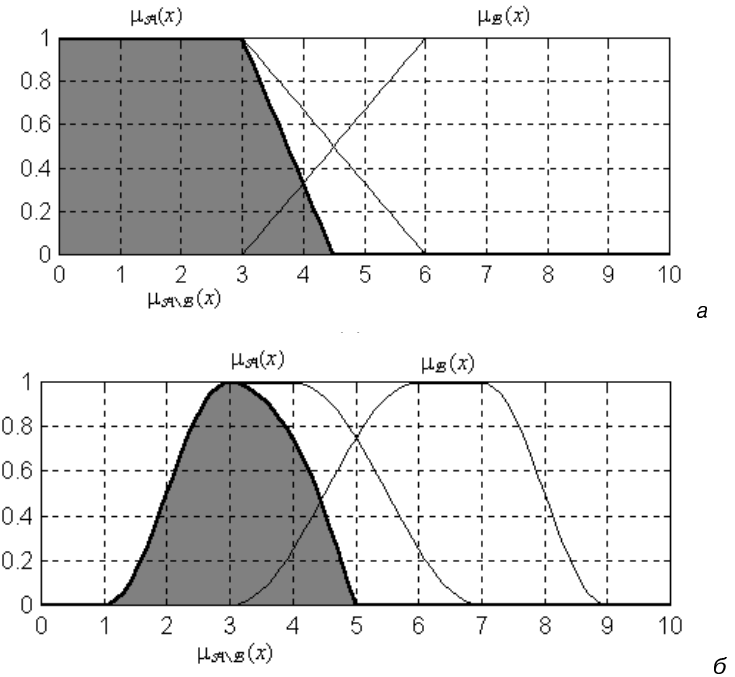


Рис. 3.4. Графическое представление операции разности двух нечетких множеств $\mathcal{A}\setminus\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Симметрическая разность. Следует заметить, что операция разности двух нечетких множеств в отличие от операций $\vee$-объединения и $\wedge$-пересечения не является коммутативной, т. е. в общем случае $\mathcal{A}\setminus\mathcal{B}\neq\mathcal{B}\setminus\mathcal{A}$. По аналогии с обычными множествами иногда оказывается полезной операция *симметрической разности* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ (будем обозначать ее через $\mathcal{A}\ominus\mathcal{B}$). По определению:

$$\mu_{\mathcal{A}\ominus\mathcal{B}}(x)=|\mu_{\mathcal{A}}(x)-\mu_{\mathcal{B}}(x)| \qquad (\forall x\in\boldsymbol{X}), \tag{3.6}$$

где в правой части выражения применяется операция модуля (или вычисления абсолютного значения) числа. При этом оказывается справедливым следующее утверждение: $\mathcal{A}\ominus\mathcal{B}=(\mathcal{A}\setminus\mathcal{B})\cup(\mathcal{B}\setminus\mathcal{A})$, т. е. симметрическая разность двух нечетких множеств представляет собой объединение двух разностей нечетких множеств $\mathcal{A}$ и $\mathcal{B}$.

Определенные выше операции разности и симметрической разности двух нечетких множеств корректны в том смысле, что они остаются справедливыми для случая обычных множеств. А именно, если в качестве нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ взять обычные множества A и B как их частный случай, то определения опера-

ций разности (3.5) и симметрической разности (3.6) превратятся в соответствующие определения (П1.6) и (П1.7) для характеристических функций последних.

Если, как и выше, рассмотреть нечеткое множество $\mathcal{A}$, равное: $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}, и нечеткое множество $\mathcal{B}$, равное: $\mathcal{B}$ ={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}, то их симметрическая разность $\mathcal{A}\ominus\mathcal{B}$ будет равна: $\mathcal{A}\ominus\mathcal{B}$={<1, 0.5>, <2, 0>, <3, 0.3>, <4, 0.4>, <5, 0.4>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}. В данном случае результат совпадает с обычной разностью $\mathcal{A}\backslash\mathcal{B}$.

Операция симметрической разности двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных на одном и том же универсуме X различными функциями принадлежности, может быть проиллюстрирована графически (рис. 3.5). При этом результату операции симметрической разности нечетких множеств также соответствует более темная область на графике.

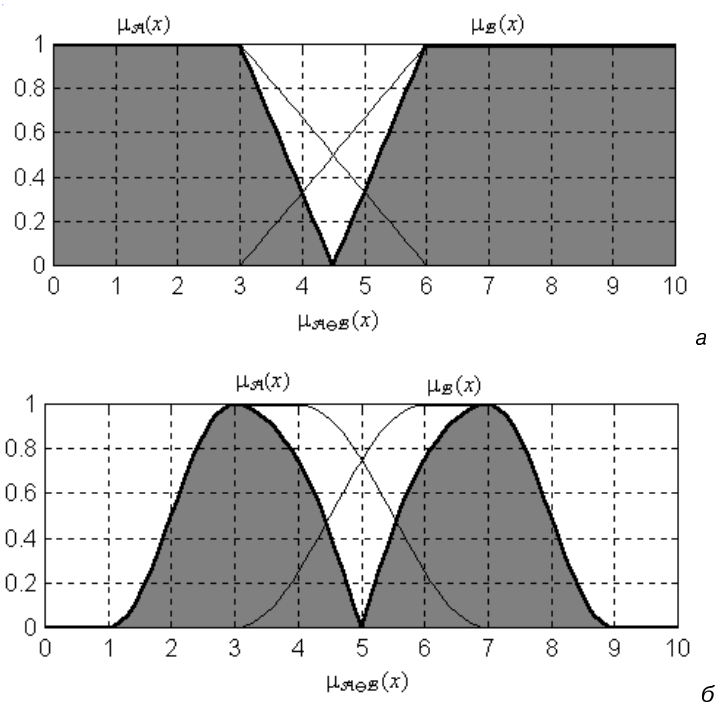


Рис. 3.5. Графическое представление операции симметрической разности двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Д о п о л н е н и е. Специально следует остановиться на унарной операции *дополнения* нечеткого множества. Дополнение нечеткого множества $\mathcal{A}$ обозначается

через $\overline{\mathcal{A}}$ и определяется как нечеткое множество $\overline{\mathcal{A}}=\{x|\ \mu_{\overline{\mathcal{A}}}(x)\}$, функция принадлежности которого $\mu_{\overline{\mathcal{A}}}(x)$ определяется по следующей формуле:

$$\mu_{\overline{\mathcal{A}}}(x)=1-\mu_{\mathcal{A}}(x) \qquad (\forall x\in \boldsymbol{X}). \tag{3.7}$$

Если, как и выше, рассмотреть нечеткое множество $\mathcal{A}$, равное: $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}, и нечеткое множество $\mathcal{B}$, равное: $\mathcal{B}$ ={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}, то их дополнения будут равны: $\overline{\mathcal{A}}$={<1, 0>, <2, 0>, <3, 0.1>, <4, 0.2>, <5, 0.4>, <6, 0.5>, <7, 0.6>, <8, 0.8>, <9, 0.9>} и $\overline{\mathcal{B}}$={<1, 0.5>, <2, 0>, <3, 0.4>, <4, 0.6>, <5, 0.8>, <6, 1.0>, <7, 1.0>, <8, 1.0>, <9, 1.0>}. Содержательно нечеткое множество $\overline{\mathcal{A}}$ может представлять в рассматриваемом контексте "*натуральное число, не являющееся небольшим*", а нечеткое множество $\overline{\mathcal{B}}$ — "*натуральное число, не равное приближенно двум*".

Примечание

Следует обратить внимание, что в рассмотренном выше примере в качестве универсума фактически использовалось множество из первых 9 натуральных чисел. Если взять в качестве универсума все множество натуральных чисел, то результат дополнения нечеткого множества $\mathcal{A}$ будет иным. Соответственно при интерпретации дополнений более строго следует говорить: "*натуральное число в пределах между 1 и 9, не являющееся небольшим*", а нечеткое множество $\overline{\mathcal{B}}$ — "*натуральное число между 1 и 9, не равное приближенно двум*".

Операция дополнения нечеткого множества $\mathcal{A}$ может быть проиллюстрирована графически (рис. 3.6). При этом результату операции дополнения $\overline{\mathcal{A}}$ также соответствует более темная область на графике. Как нетрудно видеть, график функции принадлежности дополнения нечеткого множества симметричен графику функции принадлежности исходного нечеткого множества относительно линии: y=0.5.

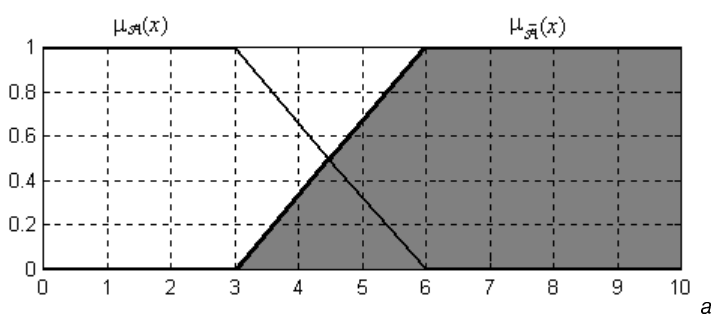


Рис. 3.6. Графическое представление операции дополнения нечеткого множества $\mathcal{A}$, которое задано линейной Z-образной (*а*) функцией принадлежности

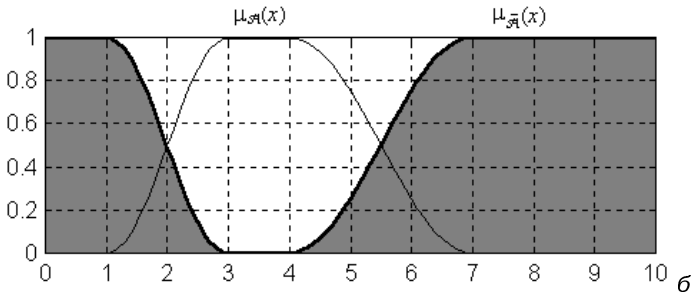


Рис. 3.6. Графическое представление операции дополнения нечеткого множества A, которое задано *П*-образной (*б*) функцией принадлежности

Для рассмотренных операций над нечеткими множествами имеют место следующие фундаментальные свойства, аналогичные свойствам обычных теоретико-множественных операций.

Пусть A, B и C — произвольные (конечные или бесконечные) нечеткие множества, заданные на одном и том же универсуме X. Справедливы следующие утверждения.

- *Коммутативность* операций объединения и пересечения нечетких множеств:

$$A \cup B = B \cup A; \qquad A \cap B = B \cap A. \tag{3.8}$$

- *Ассоциативность* операций объединения и пересечения нечетких множеств:

$$A \cup (B \cup C) = (A \cup B) \cup C; \qquad A \cap (B \cap C) = (A \cap B) \cap C. \tag{3.9}$$

- *Дистрибутивность* операций объединения и пересечения нечетких множеств относительно друг друга:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C); \; A \cap (B \cup C) = (A \cap B) \cup (A \cap C). \tag{3.10}$$

- *Идемпотентность* операций объединения и пересечения нечетких множеств:

$$A \cup A = A; \qquad A \cap A = A. \tag{3.11}$$

- *Поглощение* одного из нечетких множеств при операциях объединения и пересечения:

$$A \cup (A \cap B) = A \cap (A \cup B) = A. \tag{3.12}$$

- *Универсальные верхняя и нижняя границы* (единичные элементы) операций пересечения и объединения нечетких множеств:

$$A \cup \varnothing = A; \qquad A \cup X = X; \tag{3.13}$$

$$A \cap X = A; \qquad A \cap \varnothing = \varnothing. \tag{3.14}$$

- *Инволюция* (двойное дополнение) нечеткого множества:

$$\overline{\overline{A}} = A. \tag{3.15}$$

- Законы *де Моргана* (1806—1871):

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}; \qquad \overline{(A \cap B)} = \overline{A} \cup \overline{B}. \tag{3.16}$$

Особенность рассматриваемых операций над нечеткими множествами состоит в том, что для них *не* выполняются закон *исключенного третьего* и закон *тождества* (свойства дополняемости операций пересечения и объединения). А именно, в общем случае оказываются справедливыми неравенства:

$$\mathcal{A} \cap \bar{\mathcal{A}} \neq \varnothing; \tag{3.17}$$

$$\mathcal{A} \cup \bar{\mathcal{A}} \neq \boldsymbol{X}. \tag{3.18}$$

Доказательство отмеченных свойств непосредственно следует из свойств операций минимума и максимума, используемых в определениях соответствующих нечетких операций.

Примечание

Важность перечисленных свойств обусловливается тем обстоятельством, что они представляют собой аксиомы дистрибутивной решетки (структуры) с единственными единичными элементами относительно аддитивной и мультипликативной операций. Следует заметить, что аксиомы булевой алгебры включают в себя аксиомы дистрибутивной решетки с дополнениями. Если рассмотреть алгебру нечетких множеств (нечеткую алгебраическую систему): $\mathcal{Z}=<\mathcal{A}, \cup, \cap, \bar{\ }, \varnothing, \boldsymbol{X}>$ с операциями, которые были определены выше, то она не будет являться булевой алгеброй, поскольку для нее имеют место неравенства (3.17)—(3.18). Таким образом, каждая булева алгебра является нечеткой алгеброй, но не наоборот.

Введенные в рассмотрение операции над нечеткими множествами, основанные на использовании операций max(•) и min(•), получили наибольшее распространение при решении практических задач нечеткого моделирования. Эти операции обладают двумя основными достоинствами. Во-первых, они наиболее естественны для интуитивного представления неопределенности, связанной с использованием соответствующих им логических связок "И", "ИЛИ", "НЕ". Во-вторых, они удовлетворяют свойствам (3.8)—(3.18), что в максимальной степени приближает структуру нечетких множеств к булевой алгебре.

Тем не менее, операции min-пересечения и max-объединения нечетких множеств было бы неверно считать единственными, поскольку в общем случае, как будет видно из последующего изложения, возможны и другие альтернативные способы их определения. При этом большинство из подобных альтернативных операций также оказываются корректными в смысле соответствия обычным теоретико-множественным операциям.

3.3. Альтернативные операции пересечения и объединения нечетких множеств

Целесообразность применения альтернативных операций может быть обусловлена специфическими особенностями конкретных практических задач и желанием повысить адекватность интерпретации используемых нечетких моделей на

основе учета разнообразных смысловых оттенков соответствующих им логических связок "И" и "ИЛИ".

Пусть $\mathcal{A}$ и $\mathcal{B}$ — произвольные (конечные или бесконечные) нечеткие множества, заданные на одном и том же универсуме $\boldsymbol{X}$.

Алгебраическое пересечение. *Алгебраическим пересечением* (или *алгебраическим произведением*) двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ называется некоторое третье нечеткое множество C, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_C(x)= \mu_{\mathcal{A}}(x)\cdot\mu_{\mathcal{B}}(x) \qquad (\forall x\in\boldsymbol{X}), \tag{3.19}$$

т. е. как результат обычного арифметического произведения соответствующих значений функций принадлежности.

Для альтернативных операций над нечеткими множествами также предложены специальные обозначения. В частности, алгебраическое пересечение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ обозначается через $C=\mathcal{A}\bullet\mathcal{B}$, где $C=\{x|\mu_C(x)\}$ — результат этой операции, функция принадлежности $\mu_C(x)$ которого определяется по формуле (3.19).

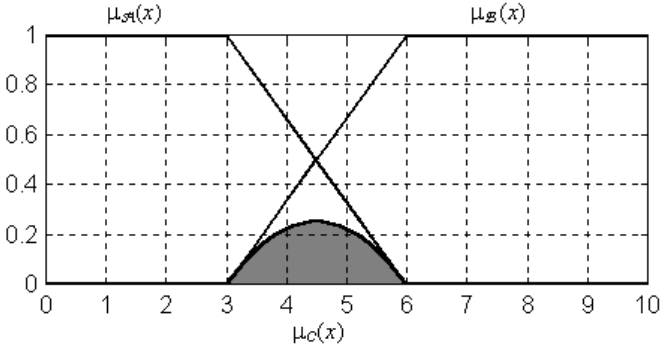
Все рассматриваемые в данном подразделе альтернативные операции будем иллюстрировать на следующем простом примере. Как и ранее, рассмотрим конечное нечеткое множество $\mathcal{A}$, которое представляет в некотором контексте "*небольшое натуральное число*", и равно: $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}, и конечное нечеткое множество $\mathcal{B}$, которое представляет "*натуральное число, приближенно равное двум*", и равно: $\mathcal{B}$ ={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}. Тогда нечеткое множество C, как результат операции алгебраического пересечения $C=\mathcal{A}\bullet\mathcal{B}$, будет равно: C={<1, 0.5>, <2, 1.0>, <3, 0.54>, <4, 0.32>, <5, 0.12>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}.

Операцию алгебраического пересечения двух бесконечных нечетких множеств, заданных на одном и том же универсуме $\boldsymbol{X}$, можно проиллюстрировать графически в декартовой системе координат на плоскости. В этом случае, как и выше, результат операции алгебраического пересечения двух бесконечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных различными функциями принадлежности, изображен затемненным на рис. 3.7, *а*, *б*.

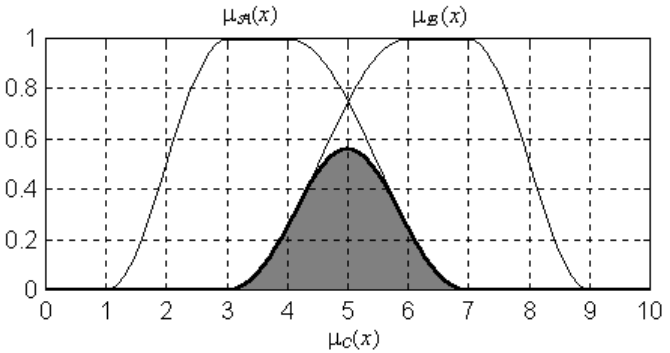
Алгебраическое объединение. *Алгебраическим объединением* (или *алгебраическим суммой*) двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ называется нечеткое множество $\mathcal{D}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{D}}(x)= \mu_{\mathcal{A}}(x)+\mu_{\mathcal{B}}(x)-\mu_{\mathcal{A}}(x)\cdot\mu_{\mathcal{B}}(x) \qquad (\forall x\in\boldsymbol{X}), \tag{3.20}$$

где справа от знака равенства использованы обычные арифметические операции. Алгебраическое объединение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ обозначается через $\mathcal{D}=\mathcal{A}+\mathcal{B}$, где $\mathcal{D}=\{x|\mu_{\mathcal{D}}(x)\}$ — результат этой операции с функцией принадлежности $\mu_{\mathcal{D}}(x)$, которая определяется по формуле (3.20).



а



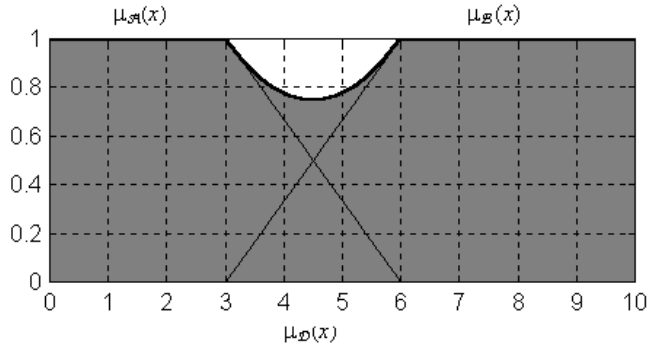
б

Рис. 3.7. Графическое представление операции алгебраического пересечения двух нечетких множеств $\mathcal{A}\bullet\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

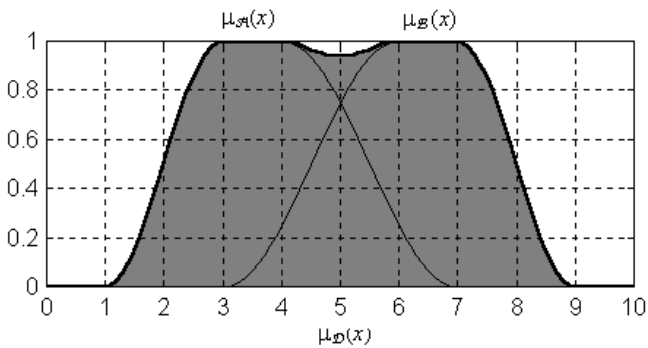
Для рассматриваемого примера результат алгебраического объединения двух конечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ будет равен: $\mathcal{D}=\mathcal{A}+\mathcal{B}=\{<1, 1>, <2, 1>, <3, 0.96>, <4, 0.88>, <5, 0.68>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>\}$.

Операцию алгебраического объединения двух бесконечных нечетких множеств, заданных на одном и том же универсуме $\boldsymbol{X}$, можно проиллюстрировать графически в декартовой системе координат на плоскости. В этом случае, как и выше, результат операции алгебраического объединения двух бесконечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных различными функциями принадлежности, изображен затемненным на рис. 3.8.

Как можно убедиться, для операций алгебраического пересечения и алгебраического объединения нечетких множеств имеют место лишь некоторые из свойств, аналогичные свойствам обычных теоретико-множественных операций.



а



б

Рис. 3.8. Графическое представление операции алгебраического объединения двух нечетких множеств $A+B$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

А именно, пусть A, B и C — произвольные (конечные или бесконечные) нечеткие множества, заданные на одном и том же универсуме X, а операция нечеткого дополнения определена по формуле (3.7). Тогда справедливы следующие утверждения.

- *Коммутативность* операций алгебраического объединения и пересечения нечетких множеств:

$$A+B=B+A; \qquad A \bullet B=B \bullet A. \tag{3.21}$$

- *Ассоциативность* операций алгебраического объединения и пересечения нечетких множеств:

$$A+(B+C)=(A+B)+C; \quad A \bullet (B \bullet C)=(A \bullet B) \bullet C. \tag{3.22}$$

- *Универсальные верхняя и нижняя границы* (единичные элементы) операций алгебраического пересечения и объединения нечетких множеств:

$$A+\varnothing=A; \qquad A+X=X; \tag{3.23}$$

$$A \bullet X=A; \qquad A \bullet \varnothing=\varnothing. \tag{3.24}$$

- Законы де Моргана:

$$\overline{(\mathcal{A}+\mathcal{B})}=\overline{\mathcal{A}}\bullet\overline{\mathcal{B}}\,; \qquad \overline{(\mathcal{A}\bullet\mathcal{B})}=\overline{\mathcal{A}}+\overline{\mathcal{B}}\,. \tag{3.25}$$

Однако в общем случае остальные свойства *не* выполняются.

- *Дистрибутивность* операций алгебраического объединения и пересечения нечетких множеств относительно друг друга:

$$\mathcal{A}+(\mathcal{B}\bullet\mathcal{C})\neq(\mathcal{A}+\mathcal{B})\bullet(\mathcal{A}+\mathcal{C}); \qquad \mathcal{A}\bullet(\mathcal{B}+\mathcal{C})\neq(\mathcal{A}\bullet\mathcal{B})+(\mathcal{A}\bullet\mathcal{C}). \tag{3.26}$$

- *Идемпотентность* операций алгебраического объединения и пересечения нечетких множеств:

$$\mathcal{A}+\mathcal{A}\neq\mathcal{A}; \qquad \mathcal{A}\bullet\mathcal{A}\neq\mathcal{A}. \tag{3.27}$$

- *Поглощение* одного из нечетких множеств при операциях алгебраического объединения и пересечения:

$$\mathcal{A}+(\mathcal{A}\bullet\mathcal{B})\neq\mathcal{A}; \qquad \mathcal{A}\bullet(\mathcal{A}+\mathcal{B})\neq\mathcal{A}. \tag{3.28}$$

- Закон *исключенного третьего* и закон *тождества* (свойства дополняемости операций алгебраического пересечения и объединения):

$$\mathcal{A}\bullet\overline{\mathcal{A}}\neq\varnothing; \tag{3.29}$$

$$\mathcal{A}+\overline{\mathcal{A}}\neq\boldsymbol{X}. \tag{3.30}$$

Доказательство выполнения свойств непосредственно следует из определения соответствующих операций и может служить в качестве упражнения. При этом следует помнить, что доказательством невыполнения свойств (3.26)—(3.30) может служить произвольный отрицательный пример.

Докажем, например, ассоциативность операции алгебраического объединения нечетких множеств. Для этого рассмотрим три произвольных нечетких множества: $\mathcal{A}$, $\mathcal{B}$ и $\mathcal{C}$. Пусть для произвольного $\forall x\in\mathbf{X}$ значения функций принадлежности этих нечетких множеств равны соответственно: $a=\mu_{\mathcal{A}}(x)$, $b=\mu_{\mathcal{B}}(x)$, $c=\mu_{\mathcal{C}}(x)$. Тогда по определению $\mathcal{A}+(\mathcal{B}+\mathcal{C})$ результирующее значение функции принадлежности будет равно: $a+(b+c-bc)-a(b+c-bc)$. Раскрывая скобки, получим: $a+b+c-bc-ab-ac+abc$. Выполнив простейшие преобразования, можно получить следующий результат: $a+b+c-bc-ab-ac+abc=a+b-ab+c-bc-ac+abc=(a+b-ab)+c-c(a+b-ab)=(a+b-ab)+c-(a+b-ab)c$. Последнее выражение как раз и означает $(\mathcal{A}+\mathcal{B})+\mathcal{C}$, что и доказывает свойство ассоциативности данной операции.

Аналогично можно доказать невыполнение свойства дистрибутивности операций алгебраического объединения и пересечения нечетких множеств относительно друг друга. Например, докажем, что $\mathcal{A}\bullet(\mathcal{B}+\mathcal{C})\neq(\mathcal{A}\bullet\mathcal{B})+(\mathcal{A}\bullet\mathcal{C})$. В этом случае левая часть выражения с учетом введенных выше обозначений равна: $a(b+c-bc)=ab+ac-abc$. Правая часть этого же выражения равна: $ab+ac-(ab)(ac)=ab+ac-a^2bc$. Различие этих простейших выражений и доказывает данное неравенство.

Примечание

Выполнение только части свойств, которые в совокупности определяют аксиомы дистрибутивной решетки, позволяют утверждать, что нечеткая алгебраическая система: $\mathcal{Z}_1=<\mathcal{A}, +, \bullet, \bar{}, \varnothing, \boldsymbol{X}>$ не будет являться ни булевой алгеброй, ни алгебраической решеткой.

Граничное пересечение. *Граничное пересечение* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ определяется как нечеткое множество C, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_C(x)= \max\{\mu_{\mathcal{A}}(x)+\mu_{\mathcal{B}}(x) -1, 0\} \qquad (\forall x \in \boldsymbol{X}), \qquad (3.31)$$

где справа от знака равенства использованы обычные арифметические операции. Граничное пересечение нечетких множеств обозначается через $C=\mathcal{A}\odot\mathcal{B}$, где $C=\{x|\mu_C(x)\}$ — результат данной операции, функция принадлежности $\mu_C(x)$ которого определяется по формуле (3.31).

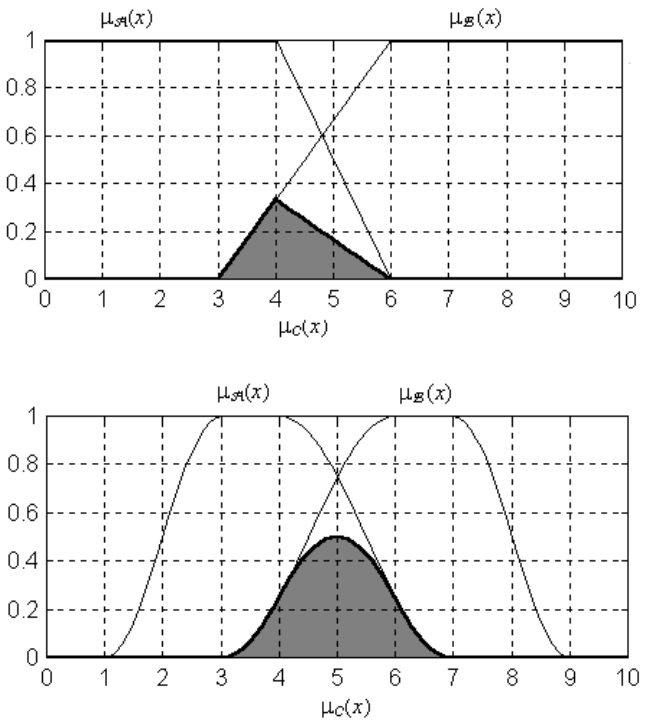


Рис. 3.9. Графическое представление операции граничного пересечения двух нечетких множеств $\mathcal{A}\odot\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Для рассматриваемого примера результат граничного пересечения нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ будет равен: $C=\mathcal{A}\odot\mathcal{B}=\{<1, 0.5>, <2, 1>, <3, 0.5>, <4, 0.2>, <5, 0>, <6, 0>, <7, 0>, <8, 0>, <9, 0>\}$.

Результат операции граничного пересечения двух бесконечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных различными функциями принадлежности, изображен затемненным на рис. 3.9, *а*, *б*.

Граничное объединение. *Граничное объединение* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ определяется как нечеткое множество $\mathcal{D}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{D}}(x)= \min\{\mu_{\mathcal{A}}(x)+\mu_{\mathcal{B}}(x), 1\} \qquad (\forall x \in \boldsymbol{X}), \tag{3.32}$$

где справа от знака равенства использованы обычные арифметические операции. Граничное объединение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ обозначается через $\mathcal{D}=\mathcal{A}\oplus\mathcal{B}$, где $\mathcal{D}=\{x|\mu_{\mathcal{D}}(x)\}$ — результат этой операции с функцией принадлежности $\mu_{\mathcal{D}}(x)$, которая определяется по формуле (3.32).

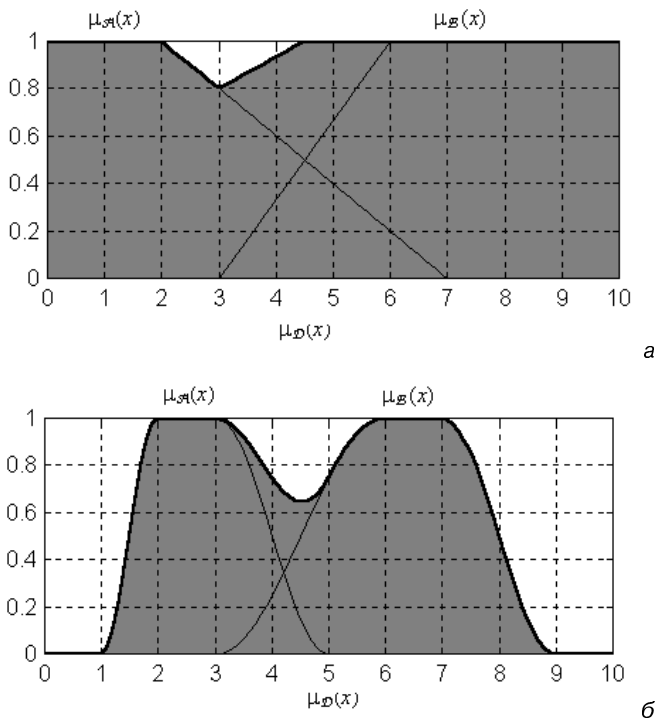


Рис. 3.10. Графическое представление операции граничного объединения двух нечетких множеств $\mathcal{A}\oplus\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Для рассматриваемого примера результат граничного объединения нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ будет равен: $\mathcal{D}=\mathcal{A}\oplus\mathcal{B}=\{<1, 1>, <2, 1>, <3, 1>, <4, 1>, <5, 0.8>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>\}$.

Результат операции граничного объединения двух бесконечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных различными функциями принадлежности, изображен затемненным на рис. 3.10, *а*, *б*.

Драстическое пересечение. *Драстическое пересечение* (от англ. *drastic* — решительный, радикальный) двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ определяется как нечеткое множество C, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_C(x)=\begin{cases}\mu_{\mathcal{B}}(x), & \text{если } \mu_{\mathcal{A}}(x)=1,\\ \mu_{\mathcal{A}}(x), & \text{если } \mu_{\mathcal{B}}(x)=1,\\ 0, & \text{в остальных случаях.}\end{cases} \qquad (\forall x\in\boldsymbol{X}). \qquad (3.33)$$

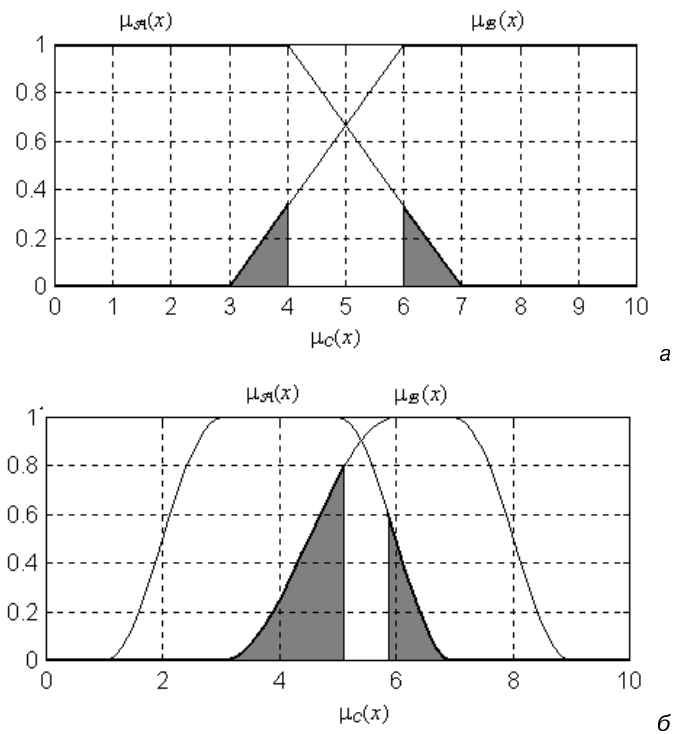


Рис. 3.11. Графическое представление операции драстического пересечения двух нечетких множеств $\mathcal{A}\Delta\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Драстическое пересечение нечетких множеств обозначается через $C=\mathcal{A}\Delta\mathcal{B}$, где $C=\{x|\mu_C(x)\}$ — результат данной операции, функция принадлежности $\mu_C(x)$ которого определяется по формуле (3.33).

Для рассматриваемого примера результат драстического пересечения нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ будет равен: $C=\mathcal{A}\,\Delta\mathcal{B}$={<1, 0.5>, <2, 1>, <3, 0>, <4, 0>, <5, 0>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}.

Результат операции драстического пересечения двух бесконечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных различными функциями принадлежности, изображен затемненным на рис. 3.11, *а*, *б*.

Драстическое объединение. *Драстическое объединение* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ определяется как нечеткое множество $\mathcal{D}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{D}}(x)=\begin{cases}\mu_{\mathcal{B}}(x), & \text{если}\,\mu_{\mathcal{A}}(x)=0,\\ \mu_{\mathcal{A}}(x), & \text{если}\,\mu_{\mathcal{B}}(x)=0,\\ 1, & \text{в остальных случаях.}\end{cases}\qquad(\forall x\in\boldsymbol{X}). \tag{3.34}$$

Драстическое объединение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ обозначается через $\mathcal{D}=\mathcal{A}\nabla\mathcal{B}$, где $\mathcal{D}=\{x|\mu_{\mathcal{D}}(x)\}$ — результат этой операции с функцией принадлежности $\mu_{\mathcal{D}}(x)$, которая определяется по формуле (3.34).

Для рассматриваемого примера результат драстического объединения нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ будет равен: $\mathcal{D}=\mathcal{A}\nabla\mathcal{B}$={<1, 1>, <2, 1>, <3, 1>, <4, 1>, <5, 1>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}.

Результат операции драстического объединения двух бесконечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных различными функциями принадлежности, изображен затемненным на рис. 3.12, *а*, *б*.

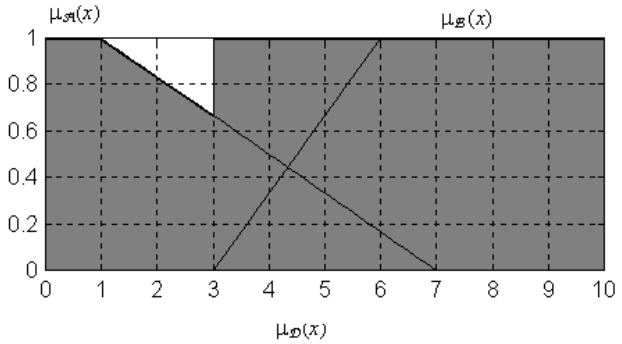
Примечание

Для рассмотренных операций над нечеткими множествами имеет место выполнение только части свойств, аналогичных свойствам обычных теоретико-множественных операций. Поэтому соответствующие нечеткие алгебраические системы не будут являться алгебраическими решетками. Доказательство этих свойств оставляем заинтересованным читателям в качестве упражнения.

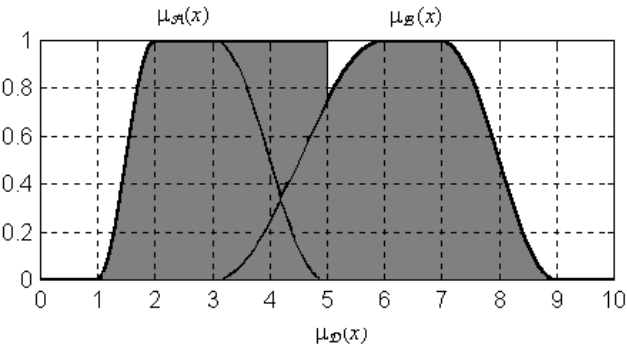
Операция λ-суммы нечетких множеств. Для полноты изложения следует отметить еще операцию λ-*суммы* для двух нечетких множеств. Эта операция для нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ обозначается через $\mathcal{D}=\mathcal{A}+_{\lambda}\mathcal{B}$, где $\mathcal{D}=\{x|\mu_{\mathcal{D}}(x)\}$. Функция принадлежности $\mu_{\mathcal{D}}(x)$ результата этой операции определяется по формуле:

$$\mu_{\mathcal{D}}(x)=\lambda\cdot\mu_{\mathcal{A}}(x)+(1-\lambda)\cdot\mu_{\mathcal{B}}(x)\qquad(\forall x\in\boldsymbol{X}), \tag{3.35}$$

где параметр $\lambda\in[0, 1]$.



а



б

Рис. 3.12. Графическое представление операции драстического объединения двух нечетких множеств $\mathcal{A}\nabla\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Это среднее между двумя функциями принадлежности с весами λ и $1 - \lambda$ соответственно. При этом следует заметить, что операция λ-суммы по своему определению оказывается некорректной с точки зрения обычных множеств, поскольку значения характеристической функции ее результата применительно к обычным множествам могут не принадлежать двухэлементному множеству $\{0, 1\}$.

Однако все остальные альтернативные операции над нечеткими множествами, кроме λ-суммы, являются корректными в том смысле, что они сохраняют свое определение для случая обычных множеств. При этом оказывается справедливой следующая цепочка неравенств:

$$\begin{aligned} \varnothing \subseteq \mathcal{A}\Delta\mathcal{B} \subseteq \mathcal{A}\odot\mathcal{B} \subseteq \mathcal{A}\bullet\mathcal{B} \subseteq \mathcal{A}\cap\mathcal{B} \subseteq \mathcal{A} +_{\lambda} \mathcal{B} \subseteq \\ \subseteq \mathcal{A}\cup\mathcal{B} \subseteq \mathcal{A}+\mathcal{B} \subseteq \mathcal{A}\oplus\mathcal{B} \subseteq \mathcal{A}\nabla\mathcal{B} \subseteq \boldsymbol{X}. \end{aligned} \tag{3.36}$$

Эти неравенства следует понимать в том смысле, что для произвольных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ результат более левой операции всегда будет являться нечетким подмножеством результата более правой операции. Границами этих неравенств служат пустое множество и универсум.

Чтобы убедиться в справедливости этих неравенств, следует обратиться к более общему подходу к определению операций с нечеткими множествами, основанному на так называемых нечетких операторах.

Нечеткие операторы

Введенные в рассмотрение нечеткие теоретико-множественные операции не исчерпывают все возможные способы их задания, которые потенциально можно предложить в контексте общей теории нечетких множеств. Большой класс подобных операций, включая и уже рассмотренные операции пересечения и объединения нечетких множеств, допускает обобщенное представление на основе так называемых *нечетких операторов*. Эти операторы действуют на множествах значений функций принадлежности (в нашем случае — на интервалах [0, 1]) и поэтому могут быть непосредственно применены к функциям принадлежности произвольных нечетких множеств. Из многообразия нечетких операторов наибольший интерес представляют треугольные норма и конорма.

Треугольная норма (**T**-норма, t-норма). Произвольная действительная функция от 2-х переменных $\mathbf{T} : [0, 1]\times[0, 1]\rightarrow[0, 1]$ называется *треугольной нормой*, если она удовлетворяет следующим свойствам, называемым *аксиомами треугольной нормы*:

$$\mathbf{T}(x, 0) = 0; \qquad \mathbf{T}(x, 1) = x \qquad \text{(ограниченность);} \qquad (3.37)$$

$$\mathbf{T}(x, y) = \mathbf{T}(y, x) \qquad \text{(коммутативность);} \qquad (3.38)$$

$$\mathbf{T}(x, \mathbf{T}(y, z)) = \mathbf{T}(\mathbf{T}(x, y), z) \qquad \text{(ассоциативность);} \qquad (3.39)$$

$$\mathbf{T}(x, y) \le \mathbf{T}(z_1, z_2), \qquad \text{(монотонность),} \qquad (3.40)$$

если одновременно $x \le z_1$ и $y \le z_2$.

Аксиома ограниченности обеспечивает выполнение граничных условий, которые должны выполняться для всех операций пересечения нечетких множеств, включая и обычные множества. Аксиомы коммутативности и ассоциативности обеспечивают выполнение соответствующих свойств у всех операций пересечения нечетких множеств. Аксиома монотонности гарантирует неизменность порядка величин значений функций принадлежности от каких бы то ни было значений других функций принадлежности. Типичной треугольной нормой является операция $\wedge$-пересечения нечетких множеств.

Рассмотренные выше альтернативные операции пересечения нечетких множеств, которые определены по формулам (3.19), (3.31) и (3.33), также являются треугольными нормами. А именно, для алгебраического пересечения двух нечетких множеств $\mathcal{A}\bullet\mathcal{B}$, граничного пересечения $\mathcal{A}\odot\mathcal{B}$ и драстического пересечения $\mathcal{A}\Delta\mathcal{B}$ выполняются все аксиомы треугольной нормы (3.37)—(3.40), в чем можно убедиться непосредственной проверкой.

Треугольная конорма (**T**-конорма, s-норма). Произвольная действительная функция от 2-х переменных $\mathbf{S} : [0, 1]\times[0, 1]\rightarrow[0, 1]$ называется *треугольной*

конормой, если она удовлетворяет следующим свойствам, называемым *аксиомами треугольной конормы*:

$$\mathbf{S}(x, 0)= x; \qquad \mathbf{S}(x, 1)=1 \qquad \text{(ограниченность);} \qquad (3.41)$$

$$\mathbf{S}(x, y) = \mathbf{S}(y, x) \qquad \text{(коммутативность);} \qquad (3.42)$$

$$\mathbf{S}(x, \mathbf{S}(y, z)) = \mathbf{S}(\mathbf{S}(x, y), z) \qquad \text{(ассоциативность);} \qquad (3.43)$$

$$\mathbf{S}(x, y)\le \mathbf{S}(z_1, z_2), \qquad \text{(монотонность),} \qquad (3.44)$$

если одновременно $x\le z_1$ и $y\le z_2$.

Как можно видеть, аксиоматика этих двух норм практически одинакова, кроме первой аксиомы ограниченности конормы. Эта аксиома обеспечивает выполнение граничных условий, которые должны выполняться для всех операций объединения нечетких множеств, включая и обычные множества. Типичной треугольной конормой является операция max-объединения нечетких множеств.

Альтернативные операции объединения нечетких множеств, которые определяются по формулам (3.20), (3.32) и (3.34), также являются треугольными конормами. А именно, для алгебраического объединения двух нечетких множеств $\mathcal{A}+\mathcal{B}$, граничного объединения $\mathcal{A}\oplus\mathcal{B}$ и драстического объединения $\mathcal{A}\nabla\mathcal{B}$ выполняются все аксиомы треугольной конормы (3.41)—(3.44), в чем можно убедиться непосредственной проверкой.

Поскольку областью определения и областью значений треугольных норм и конорм является интервал [0, 1], то все рассмотренные ранее операции над нечеткими множествами могут быть проиллюстрированы графически с использованием так называемого трехмерного единичного куба. При этом результатам операций $\wedge$-пересечения (*а*), алгебраического пересечения (*б*), граничного пересечения (*в*) и драстического пересечения (*г*) нечетких множеств соответствует более темная область на графике (рис. 3.13). Аналогично на рис. 3.14 представлены операции $\vee$-объединения (*а*), алгебраического объединения (*б*), граничного объединения (*в*) и драстического объединения (*г*) нечетких множеств в трехмерном пространстве.

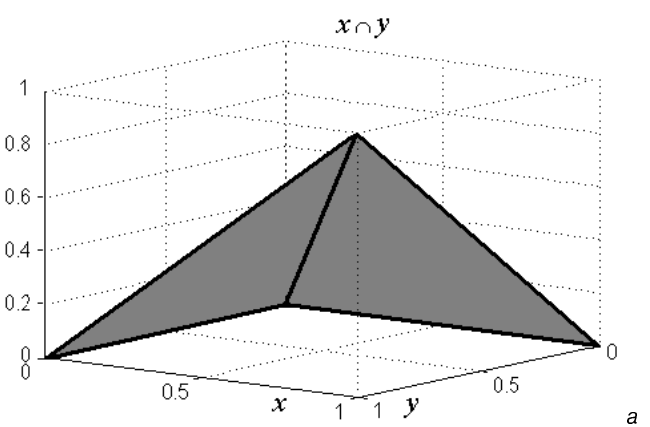


Рис. 3.13. Графическое представление операций $\wedge$-пересечения (*а*) нечетких множеств в трехмерном пространстве

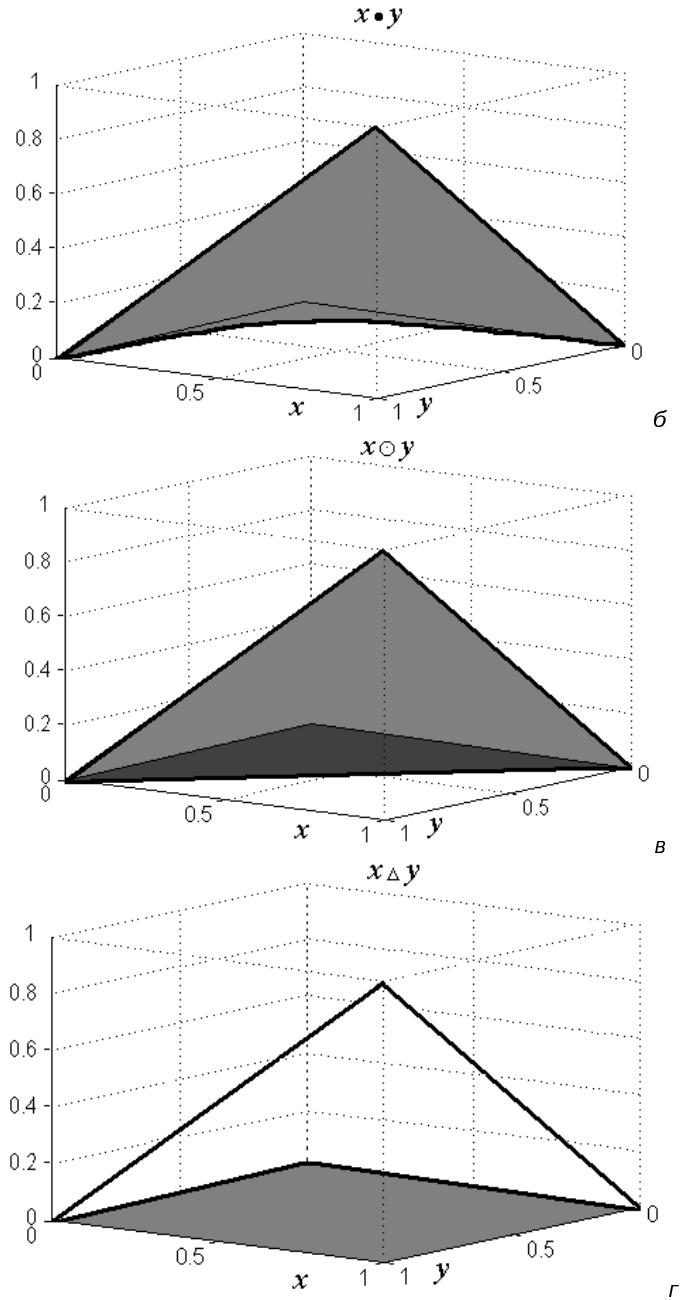


Рис. 3.13. Графическое представление алгебраического пересечения (*б*), граничного пересечения (*в*) и драстического пересечения (*г*) нечетких множеств в трехмерном пространстве

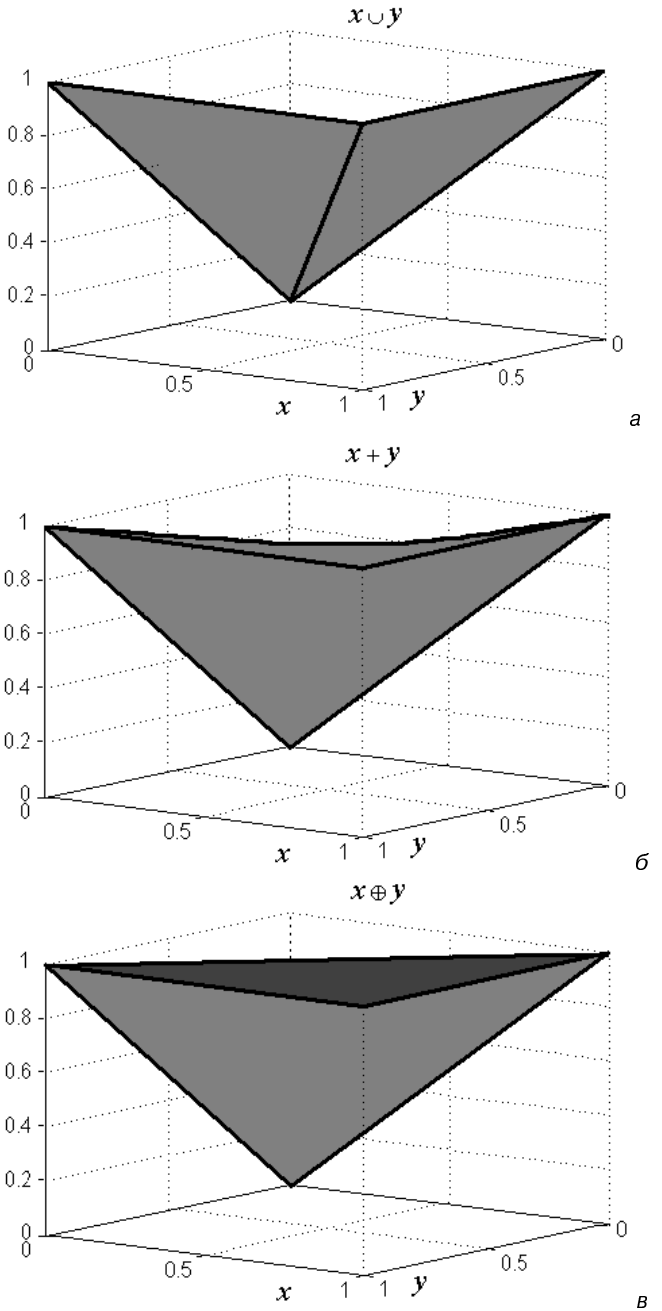


Рис. 3.14. Графическое представление операций $\vee$-объединения (*а*), алгебраического объединения (*б*), граничного объединения (*в*) нечетких множеств в трехмерном пространстве

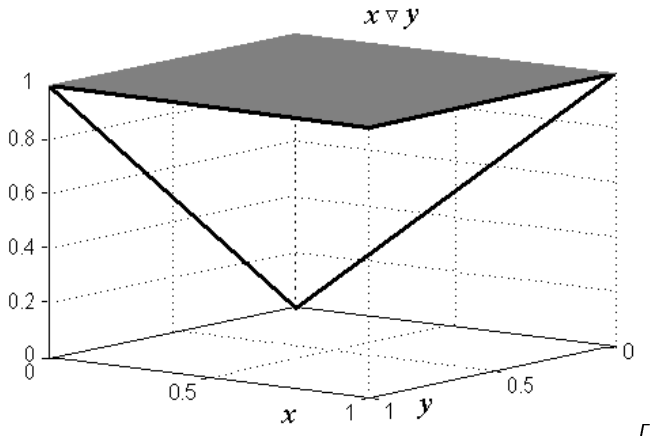


Рис. 3.14. Графическое представление драстического объединения (*г*) нечетких множеств в трехмерном пространстве

3.4. Некоторые дополнительные операции над нечеткими множествами

Среди дополнительных операций, которые находят применение при построении нечетких моделей сложных систем, следует отметить унарные операции умножения нечеткого множества на число и возведение нечеткого множества в степень.

Умножение нечеткого множества на число. Пусть $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$ — произвольное нечеткое множество, заданное на универсуме $\boldsymbol{X}$; a — положительное действительное число ($a \in \boldsymbol{R}_+$), такое, что $a \cdot h_{\mathcal{A}} \le 1$ (напомним, что $h_{\mathcal{A}}$ — высота нечеткого множества $\mathcal{A}$). Результат операции *умножения нечеткого множества* $\mathcal{A}$ *на число* a определяется как нечеткое множество $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по формуле:

$$\mu_{\mathcal{B}}(x)= a \cdot \mu_{\mathcal{A}}(x) \quad (\forall x \in \boldsymbol{X}). \tag{3.45}$$

Эту операцию в дальнейшем будем обозначать через $a \cdot \mathcal{A}$.

Например, для конечного нечеткого множества $\mathcal{A}=\{<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>\}$ и числа a=0.7 нечеткое множество $a \cdot \mathcal{A}$ равно: $a \cdot \mathcal{A} = \{<1, 0.7>, <2, 0.7>, <3, 0.63>, <4, 0.56>, <5, 0.42>, <6, 0.35>, <7, 0.28>, <8, 0.14>, <9, 0.07>\}$.

Возведение в степень. Пусть $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$ — произвольное нечеткое множество, заданное на универсуме $\boldsymbol{X}$; k — положительное действительное число ($a \in \boldsymbol{R}_+$). В этом случае чисто формально можно определить операцию *возведения*

нечеткого множества $\mathcal{A}$ *в степень* k как нечеткое множество $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по формуле:

$$\mu_{\mathcal{B}}(x) = \mu_{\mathcal{A}}(x)^k \quad (\forall x \in \boldsymbol{X}). \tag{3.46}$$

Эту операцию иногда обозначают через $\mathcal{A}^k$. Например, для конечного нечеткого множества $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>} и числа a=3 нечеткое множество $\mathcal{A}^3$ равно: $\mathcal{A}^3$ ={<1, 1.0>, <2, 1.0>, <3, 0.729>, <4, 0.512>, <5, 0.216>, <6, 0.125>, <7, 0.064>, <8, 0.008>, <9, 0.001>}.

Операции умножения бесконечного нечеткого множества на число и возведения бесконечного нечеткого множества в степень могут быть проиллюстрированы графически следующим образом (рис. 3.15).

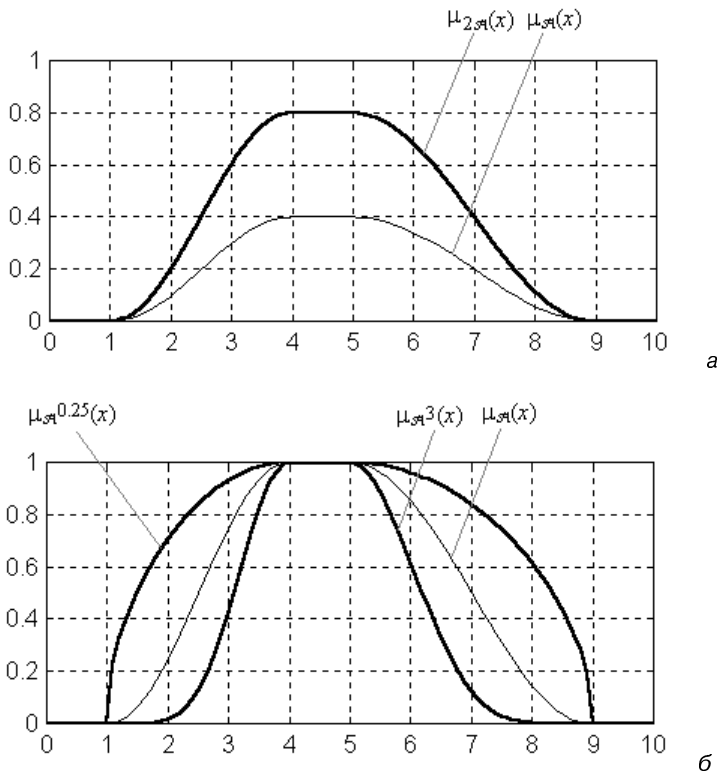


Рис. 3.15. Графическое представление операций умножения нечеткого множества $\mathcal{A}$ на число 2 (*а*) и возведения нечеткого множества $\mathcal{A}$ в степени 0.25 и 3 (*б*) для *П*-образных функций принадлежности, первая из которых не является нормальной

На основе операции возведения в степень определяются две специальные операции над нечеткими множествами: операция концентрирования и операция растяжения нечеткого множества.

К о н ц е н т р и р о в а н и е. Пусть на универсуме X задано произвольное нечеткое множество $A=\{x, \mu_A(x)\}$. Операция *концентрирования*, обозначаемая через CON(A), дает в результате нечеткое множество $C=\{x, \mu_C(x)\}$, функция принадлежности которого равна значениям функции принадлежности исходного нечеткого множества, возведенным в квадрат, т. е.

$$\mu_C(x) = \mu_A(x)^2 \qquad (\forall x \in X). \tag{3.47}$$

Очевидно, в этом случае CON(A)=A^2. Например, для конечного нечеткого множества A={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>} его концентрирование равно: CON(A)=A^2 ={<1, 1.0>, <2, 1.0>, <3, 0.81>, <4, 0.64>, <5, 0.36>, <6, 0.25>, <7, 0.16>, <8, 0.04>, <9, 0.01>}.

Р а с т я ж е н и е. Операция *растяжения*, обозначаемая через DIL(A), дает в результате нечеткое множество $D=\{x, \mu_D(x)\}$, функция принадлежности которого равна значениям функции принадлежности исходного нечеткого множества, возведенным в степень 0.5, т. е.

$$\mu_B(x) = \mu_A(x)^{0.5} \qquad (\forall x \in X). \tag{3.48}$$

Очевидно, в этом случае CON(A)=$A^{0.5}$. Например, для конечного нечеткого множества A={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>} его растяжение равно: DIL(A)=$A^{0.5}$={<1, 1.0>, <2, 1.0>, <3, 0.949>, <4, 0.894>, <5, 0.775>, <6, 0.707>, <7, 0.632>, <8, 0.447>, <9, 0.316>}.

Операции концентрирования и растяжения нечеткого множества могут быть проиллюстрированы графически (см. рис. 3.16). Результатам этих операций соответствует более темная область на графике.

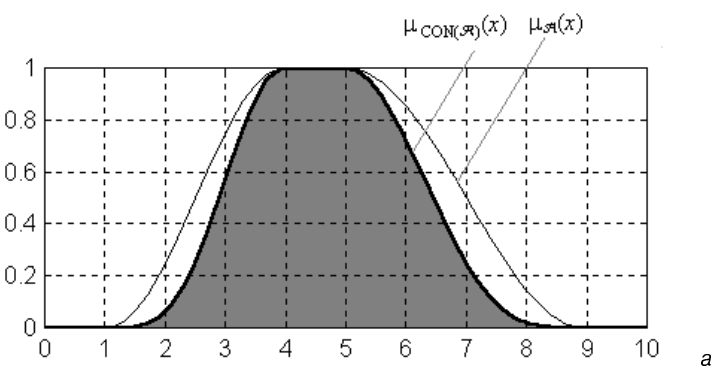


Рис. 3.16. Графическое представление операций концентрирования нечеткого множества (*а*) для *П*-образных функций принадлежности

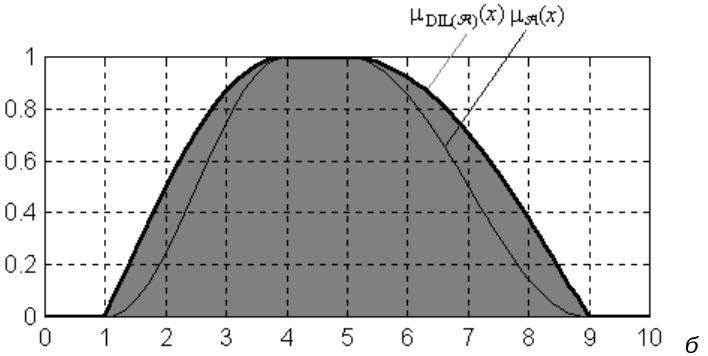


Рис. 3.16. Графическое представление операций растяжения нечеткого множества (*б*) для *П*-образных функций принадлежности

Примечание

Применение операции концентрирования к нечеткому множеству означает уменьшение нечеткости или неопределенности в задании этого множества. Зачастую это может быть следствием поступления дополнительной информации, которая уточняет некоторые аспекты соответствующей предметной области. Напротив, применение операции растяжения означает усиление неопределенности в задании нечеткого множества, что может быть следствием либо потери части информации, либо поступления информации о дополнительных факторах, не учитываемых в исходной нечеткой модели. Эти операции находят применение при оперировании лингвистическими переменными, которые рассматриваются далее в *главе 5*.

Обобщением операции λ-*суммы* двух нечетких множеств является операция определения выпуклой комбинации произвольного конечного числа нечетких множеств. Пусть $\mathcal{A}_1$, $\mathcal{A}_2$,..., $\mathcal{A}_n$ — нечеткие множества, заданные на универсуме $\boldsymbol{X}$, а λ_1, λ_2,..., λ_n — неотрицательные действительные числа, сумма которых равна 1.

Выпуклая комбинация нечетких множеств. *Выпуклой комбинацией* нечетких множеств $\mathcal{A}_1$, $\mathcal{A}_2$,..., $\mathcal{A}_n$ называется нечеткое множество $\mathcal{D}=\{x|\mu_{\mathcal{D}}(x)\}$, функция принадлежности которого определяется по формуле:

$$\mu_{\mathcal{D}}(x)= \lambda_1\cdot\mu_{\mathcal{A}1}(x)+\lambda_2\cdot\mu_{\mathcal{A}2}(x)+\ldots+\lambda_n\cdot\mu_{\mathcal{A}n}(x) \quad (\forall x\in\boldsymbol{X}), \tag{3.49}$$

при этом параметры $\lambda_i\in[0, 1]$ для всех $i\in\{1, 2,\ldots, n\}$ и $\lambda_1+\lambda_2+\ldots+\lambda_n=1$.

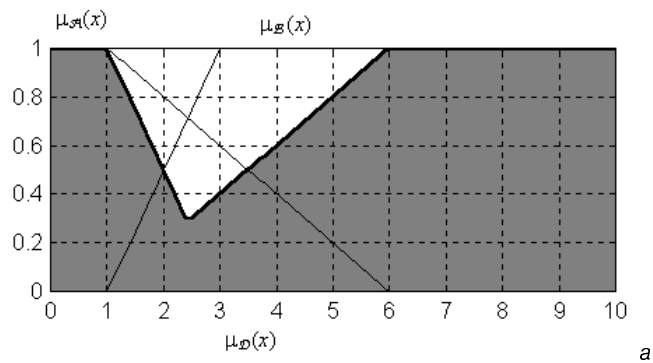
Дизъюнктивная сумма. В завершение рассмотрения операций с нечеткими множествами определим так называемую *дизъюнктивную сумму* двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, которая хотя и редко, но все же используется в практике нечеткого моделирования. Результатом этой операции называется некоторое

третье нечеткое множество $\mathcal{D}$, заданное на этом же универсуме $\boldsymbol{X}$, функция принадлежности которого определяется по следующей формуле:

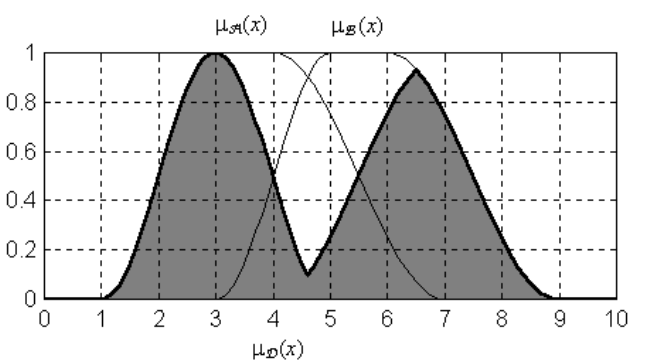
$$\mu_{\mathcal{D}}(x)= \max\{\min\{\mu_{\mathcal{A}}(x),\ 1-\mu_{\mathcal{B}}(x)\},\ \min\{1-\mu_{\mathcal{A}}(x),\ \mu_{\mathcal{B}}(x)\}\} \qquad (3.50)$$

для любого $x\in\boldsymbol{X}$, где операции max() и min() справа от знака равенства выполняются над парами соответствующих значений функций принадлежности исходных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$. Эквивалентная запись для определения операции дизъюнктивной суммы: $\mathcal{D}=(\mathcal{A}\cap\overline{\mathcal{B}})\cup(\mathcal{A}\cap\mathcal{B})$.

Для конечного нечеткого множества $\mathcal{A}$={<1, 1.0>, <2, 1.0>, <3, 0.9>, <4, 0.8>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>} и нечеткого множества: $\mathcal{B}$ ={<1, 0.5>, <2, 1.0>, <3, 0.6>, <4, 0.4>, <5, 0.2>, <6, 0>, <7, 0>, <8, 0>, <9, 0>}, результат их дизъюнктивной суммы будет равен: $\mathcal{D}=(\mathcal{A}\cap\mathcal{B})\cup(\overline{\mathcal{A}}\cap\mathcal{B})$={<1, 0.5>, <2, 0>, <3, 0.4>, <4, 0.6>, <5, 0.6>, <6, 0.5>, <7, 0.4>, <8, 0.2>, <9, 0.1>}.



а



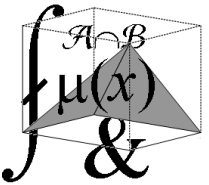
б

Рис. 3.17. Графическое представление операции дизъюнктивной суммы двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных линейными *Z*-образной и *S*-образной функциями принадлежности (*а*) и *П*-образными (*б*) функциями принадлежности

Операция дизъюнктивной суммы двух бесконечных нечетких множеств $\mathcal{A}$ и $\mathcal{B}$, заданных на одном и том же универсуме $\boldsymbol{X}$ различными функциями принадлежности, может быть проиллюстрирована графически (рис. 3.17). При этом результату операции дизъюнктивной суммы нечетких множеств соответствует более темная область на графике.

В завершение этой главы следует отметить то обстоятельство, что все введенные в рассмотрение операции над нечеткими множествами образуют необходимый концептуальный базис, который будет использоваться на всем протяжении книги. При этом в процессе решения конкретных задач нечеткого моделирования будут отмечены дополнительные семантические особенности нечетких теоретико-множественных операций в том или ином контексте.

Глава 4



Нечеткие отношения

Понятие нечеткого отношения наряду с понятием самого нечеткого множества следует отнести к фундаментальным основам всей теории нечетких множеств. На основе нечетких отношений определяется целый ряд дополнительных понятий, используемых для построения нечетких моделей сложных систем. Нечеткое отношение обобщает понятие обычного отношения и часто заменяется терминами нечеткая связь, ассоциация, взаимосвязь или соотношение.

4.1. Нечеткое отношение и способы его задания

Содержательно нечеткое отношение определяется как любое нечеткое подмножество упорядоченных кортежей, построенных из элементов тех или иных базисных множеств, в качестве которых в данном случае используются универсумы. При этом под *кортежем*, так же как и в случае обычных множеств, понимается произвольный набор или список упорядоченных элементов.

Нечеткое отношение. В общем случае *нечетким отношением* или, более точно, *нечетким k-арным отношением*, заданным на множествах (универсумах) $\boldsymbol{X}_1$, $\boldsymbol{X}_2$,..., $\boldsymbol{X}_k$, называется некоторое фиксированное нечеткое подмножество декартова произведения этих универсумов. Другими словами, если обозначить произвольное нечеткое отношение через Q, то по определению $Q=\{<x_1, x_2,..., x_k>, \mu_Q(<x_1, x_2,..., x_k>)\}$, где $\mu_Q(<x_1, x_2,..., x_k>)$ — функция принадлежности данного нечеткого отношения, которая определяется как отображение $\mu_Q : \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times ... \times \boldsymbol{X}_k \to [0, 1]$. Здесь через $<x_1, x_2,..., x_k>$ обозначен кортеж из k элементов, каждый из которых выбирается из своего универсума: $x_1 \in \boldsymbol{X}_1$, $x_2 \in \boldsymbol{X}_2$,..., $x_k \in \boldsymbol{X}_k$ (*см.* Приложение 1).

Так же как и в случае обычных множеств с целью характеризовать количество универсальных множеств, на основе которых строится то или иное нечеткое отношение, принято называть нечеткое отношение между элементами из двух универсальных множеств — *бинарным*, между элементами трех множеств — *тернарным*, а в общем случае — *k-арным* отношением. При этом на форму и вид функции принадлежности нечеткого отношения предварительно не накладывается никаких ограничений.

Примечание

Следует отметить, что существуют и другие способы определения нечетких отношений. Так, например, можно предварительно задать нечеткие базисные множества $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ и определить нечеткое декартово произведение этих нечетких множеств в форме: $\mathcal{P}=\mathcal{A}_1\times\mathcal{A}_2\times\ldots\times\mathcal{A}_k$, где $\mu_{\mathcal{P}}(<x_1, x_2, \ldots, x_k>)= =\min\{\mu_{\mathcal{A}1}(x_1), \mu_{\mathcal{A}2}(x_2), \ldots, \mu_{\mathcal{A}k}(x_k)\}$ и $x_1\in\mathcal{A}_1, x_2\in\mathcal{A}_2, \ldots, x_k\in\mathcal{A}_k$. После чего определить нечеткое отношение Q как некоторое нечеткое подмножество этого нечеткого декартова произведения. Поскольку этот способ основан на использовании так называемого *принципа обобщения*, он будет рассмотрен ниже в этой главе.

Пустое нечеткое отношение. В теории нечетких отношений *пустое* нечеткое отношение определяется как отношение, которое не содержит ни одного кортежа. Это отношение по-прежнему обозначается через $\varnothing$ и формально определяется как такое нечеткое отношение, функция принадлежности которого тождественно равна 0 на всем декартовом произведении его универсумов. Из этого определения также следует, что пустое нечеткое отношение совпадает с обычным пустым отношением.

Полное нечеткое отношение. Что касается другого крайнего случая, то так называемое *полное нечеткое отношение* по своей сути совпадает с обычным полным отношением, которое, в свою очередь, равно по определению декартову произведению соответствующих универсумов $\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$. Как его обозначать — из соображений удобства можно просто через $\boldsymbol{X}$. Важно представлять себе, что функция принадлежности полного нечеткого отношения тождественно равна единице для всех без исключения кортежей, т. е. $\mu_{\boldsymbol{X}}(<x_1, x_2, \ldots, x_k>)=1$.

Особое значение в нечетком моделировании имеют бинарные нечеткие отношения, для задания которых используется одно или разные базисные множества (универсумы). В связи с этим приведем их формальное определение.

Бинарное нечеткое отношение. В общем случае *бинарное нечеткое отношение* задается на базисных множествах $\boldsymbol{X}_1$, $\boldsymbol{X}_2$ и определяется как нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$. Здесь $\mu_Q(<x_i, x_j>)$ — функция принадлежности бинарного нечеткого отношения, которая определяется как отображение $\mu_Q: \boldsymbol{X}_1\times\boldsymbol{X}_2\rightarrow[0, 1]$, а через $<x_i, x_j>$ обозначен кортеж из двух элементов, при этом $x_i\in\boldsymbol{X}_1$ и $x_j\in\boldsymbol{X}_2$.

Обратное нечеткое отношение. Применительно к бинарным нечетким отношениям определяется так называемое *обратное* нечеткое отношение. А именно, если задано бинарное нечеткое отношение Q на декартовом произведении $\boldsymbol{X}_1\times\boldsymbol{X}_2$, то обратным к нему нечетким отношением (обозначается через Q^{-1}) называется такое бинарное нечеткое отношение, которое заданно на декартовом произведении $\boldsymbol{X}_2\times\boldsymbol{X}_1$, а функция принадлежности которого определяется по следующей формуле:

$$\mu_{Q^{-1}}(x_i, x_j)=\mu_Q(x_j, x_i) \quad \text{для любых } x_i\in\boldsymbol{X}_2 \text{ и } x_j\in\boldsymbol{X}_1. \qquad (4.1)$$

Бинарное нечеткое отношение, заданное на одном универсуме. *Бинарное нечеткое отношение, заданное на одном базисном множестве* (универсуме) $\boldsymbol{X}$, определяется как нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, где $\mu_Q(<x_i, x_j>)$ — функция принадлежности бинарного нечеткого отношения, которая определяется как отображение $\mu_Q : \boldsymbol{X}\times\boldsymbol{X}\rightarrow[0, 1]$. Здесь через $<x_i, x_j>$ обозначен кортеж из двух элементов, при этом как $x_i\in\boldsymbol{X}$, так и $x_j\in\boldsymbol{X}$.

Способы задания нечетких отношений

Существуют различные способы, которыми в общем случае могут быть формально заданы те или иные нечеткие отношения. Наибольшее распространение из них получили следующие:

- В форме *списка с явным перечислением* всех кортежей нечеткого отношения и соответствующих им значений функции принадлежности: $Q=\{(\boldsymbol{w}_1, \mu_Q(\boldsymbol{w}_1)), \boldsymbol{w}_2, \mu_Q(\boldsymbol{w}_2)),\ldots, (\boldsymbol{w}_q, \mu_Q(\boldsymbol{w}_q))\}$, где $\boldsymbol{w}_i$ — i-й кортеж $<x_1, x_2,\ldots, x_k>$ элементов этого отношения, а q — рассматриваемое число кортежей нечеткого отношения Q. При этом для сокращения подобной записи кортежи с нулевыми значениями функции принадлежности не указываются в данном списке. Как нетрудно заметить, этот способ подходит только для задания нечетких отношений с конечным и небольшим числом кортежей q.
- *Аналитически* в форме некоторого математического выражения для соответствующей функции принадлежности этого нечеткого отношения. Этот способ может быть использован для задания произвольных нечетких отношений как с конечным, так и с бесконечным числом кортежей. В этом случае нечеткое отношение записывается в виде: $Q=\{<x_1, x_2,\ldots, x_k>, \mu_Q(<x_1, x_2,\ldots, x_k>)\}$ или сокращенно: $Q=\{\boldsymbol{w}, \mu_Q(\boldsymbol{w})\}$, понимая под $\boldsymbol{w}$ общее обозначение кортежа длины k. Если в качестве универсумов используются числовые множества, то в этом случае удобно представить функцию принадлежности μ_Q аналитически в форме некоторой функции $f(x_1, x_2,\ldots, x_k)$ от k переменных, которая конкретизирует отображение $f(\boldsymbol{x}) : \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k\rightarrow[0, 1]$. Некоторые примеры использования этого способа задания нечетких отношений будут рассмотрены ниже в этой главе.

В дополнение к этим способам бинарные нечеткие отношения также могут быть заданы следующим образом.

- *Графически* в форме некоторой поверхности или совокупности отдельных точек в трехмерном пространстве. При этом две координаты (независимые переменные) будут соответствовать значениям универсумов $\boldsymbol{X}_1$ и $\boldsymbol{X}_2$, а третья координата — интервалу [0, 1]. Например, график математической функции: $z=x^2+y^2$ для x, $y\in[-0.5, 0.5]$ может служить примером графического способа формального задания некоторого нечеткого отношения. Здесь функция z является представлением функции принадлежности соответствующего нечеткого отношения. Этот способ зачастую используется в дополнение к аналитиче-

скому для визуализации бинарных нечетких отношений с бесконечным числом кортежей.

- В форме *матрицы* нечеткого отношения. Этот способ основан на представлении нечеткого бинарного отношения с конечным числом кортежей в форме матрицы $\mathbf{M}_Q$, строки которой соответствуют первым элементам кортежей, а столбцы — вторым элементам кортежей рассматриваемого нечеткого отношения. При этом элементами матрицы являются соответствующие значения функции принадлежности μ_Q данного отношения. Если бинарное нечеткое отношение задается на одном универсуме, то матрица такого отношения $\mathbf{M}_Q$ является квадратной. Определенную таким образом матрицу называют *матрицей бинарного нечеткого отношения* и обозначают $\mathbf{M}_Q$. В этом контексте *табличный* способ может рассматриваться как разновидность матричного, поскольку конечная матрица всегда может быть представлена в форме таблицы. В случае счетного универсума можно использовать бесконечные матрицы, однако эти математические конструкции лежат за пределами тематики книги.

- В форме так называемого *нечеткого графа*, который формально может быть задан в виде двух обычных конечных множеств и некоторой функции принадлежности. А именно, *нечеткий граф*, а точнее, *ориентированный нечеткий граф*, есть $\mathcal{G}=(V, E, \mu_{\mathcal{G}})$, где $V=\{v_1, v_2,\ldots, v_n\}$ — множество вершин нечеткого графа, $E=\{e_1, e_2,\ldots, e_m\}$ — множество дуг нечеткого графа, $\mu_{\mathcal{G}}$ — функция принадлежности дуг данному нечеткому графу, т. е. $\mu_{\mathcal{G}}: E\rightarrow[0, 1]$. При этом вершины нечеткого графа, как и в случае обычных графов, изображаются точками, дуги — отрезками прямых линий со стрелкой на одном из концов. Рядом с вершинами записываются условные обозначения соответствующих вершин, а рядом с каждой дугой — значение функции принадлежности для соответствующей дуги. Натуральное число n определяет общее количество вершин конкретного нечеткого графа, а натуральное число m — общее количество дуг нечеткого графа. При этом дуги с нулевой функцией принадлежности в нечетком графе обычно не изображаются.

  Как нетрудно заметить, каждому ориентированному нечеткому графу $\mathcal{G}$ соответствует некоторое бинарное нечеткое отношение $Q_{\mathcal{G}}$, состоящее из всех пар вида $<v_i, v_j>$, где $v_i, v_j\in V$. При этом для каждой пары $<v_i, v_j>$ определено некоторое действительное число из интервала [0, 1], которое равно значению функции принадлежности $\mu_{\mathcal{G}}(e_k)$ для дуги $e_k\in E$, которая соответствует этой паре вершин. И обратно, если вершины v_i и v_j соединяются в нечетком графе $\mathcal{G}$ некоторой дугой $e_k\in E$ со значением функции принадлежности $\mu_{\mathcal{G}}(e_k)$ и направленной из вершины v_i в вершину v_j, то тем самым задается некоторое бинарное нечеткое отношение $Q_{\mathcal{G}}$. При задании нечеткого отношения Q с помощью ориентированного нечеткого графа $\mathcal{G}$ каждому элементу универсума $x_i\in X$ будет соответствовать отдельная вершина $v_i\in\mathcal{G}$ этого нечеткого графа, а каждому кортежу нечеткого отношения $<x_i, x_j>\in Q$ будет соответствовать дуга графа $e_k=<v_i, v_j>$ с началом в вершине v_i, концом в вершине v_j и значением функции принадлежности $\mu_Q(<x_i, x_j>)$.

Как и в случае обычных графов, можно определить различные типы нечетких графов. Кроме рассмотренных выше ориентированных нечетких графов, можно ввести в рассмотрение *неориентированные* нечеткие графы, т. е. такие графы, у которых соединяющие вершины ребра не имеют направления или ориентации. Неориентированные нечеткие графы также удобно считать частным случаем ориентированных, у которых каждая дуга одновременно имеет дугу противоположной ориентации. В случае счетного универсума можно использовать *бесконечные* графы, т. е. графы с бесконечным числом вершин. Бесконечные графы, хотя и рассматриваются в современной теории графов, но выходят за рамки тематики настоящей книги. Именно поэтому мы будем использовать нечеткие графы для задания бинарных нечетких отношений только в случае их конечного универсума.

Примечание

При рассмотрении обычных отношений иногда отмечают способ их задания на основе явного указания некоторого *свойства*, которым должны обладать все элементы данного отношения. Применительно к нечетким отношениям этот способ во многом аналогичен описанным выше, поскольку в этом случае необходимо ввести в рассмотрение некоторое характеристическое свойство, которое может быть записано в виде *многоместного нечеткого предиката* $\mathrm{P}(<x_1, x_2,\ldots, x_k>)$. Данный нечеткий предикат $\mathrm{P}(<x_1, x_2,..., x_k>)$ определяется на декартовом произведении универсумов $\boldsymbol{X}_1, \boldsymbol{X}_2,\ldots, \boldsymbol{X}_k$ и может принимать значения истинности из некоторого вполне упорядоченного множества, в частности, из интервала [0, 1]. При этом в отличие от случая обычных множеств возникает проблема задания самого нечеткого предиката или его функции принадлежности *(см. приложение 2)*. Тем не менее, принято считать, что в общем случае нечеткое отношение Q может быть также задано с помощью нечеткого предиката следующим образом: $Q=\{<x_1, x_2,\ldots, x_k>|\mathrm{P}_Q(<x_1, x_2,...,x_k>), x_1\in\boldsymbol{X}_1, x_2\in\boldsymbol{X}_2,\ldots, x_k\in\boldsymbol{X}_k\}$.

В зависимости от количества кортежей нечеткое отношение может быть конечным или бесконечным. Нечеткое отношение называется *конечным*, если его носитель является конечным отношением. При этом вполне уместно говорить, что такое нечеткое отношение имеет конечную *мощность*, которая численно равна количеству кортежей его носителя, рассматриваемого как обычное множество *(см. приложение 1)*. В этом случае для обозначения мощности произвольного нечеткого отношения Q можно использовать общепринятое обозначение $card(Q)$. Аналогично *счетным* нечетким отношением будем называть нечеткое отношение со счетным носителем, т. е. носитель которого имеет счетную мощность $\aleph_0$ в обычном смысле. *Несчетным* нечетким отношением называется нечеткое отношение с несчетным носителем, т. е. носитель которого имеет несчетную мощность или мощность континуума $\boldsymbol{c}$ (или $\aleph$) в обычном смысле.

Для иллюстрации описанных способов задания отношений рассмотрим следующие примеры конкретных нечетких отношений.

П р и м е р 4.1. В качестве первого примера рассмотрим конечное бинарное нечеткое отношение Q_1, заданное на одном универсуме $\boldsymbol{X}$, в качестве которого возьмем подмножество первых 10 натуральных чисел: $\boldsymbol{X}=\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Пусть отношение Q_1 описывает свойство: "*натуральное число x_i приближенно равно натуральному числу x_j*".

Конкретное бинарное нечеткое отношение Q_1 может быть задано в форме списка следующим образом: $Q_1 = \{(<1, 1>, 1.0), (<1, 2>, 0.8), (<1, 3>, 0.5), (<1, 4>, 0.2), (<2, 1>, 0.8), (<2, 2>, 1), (<2, 3>, 0.8), (<2, 4>, 0.5), (<2, 5>, 0.2), (<3, 1>, 0.5), (<3, 2>, 0.8), (<3, 3>, 1), (<3, 4>, 0.8), (<3, 5>, 0.5), (<3, 6>, 0.2), (<4, 1>, 0.2), (<4, 2>, 0.5), (<4, 3>, 0.8), (<4, 4>, 1), (<4, 5>, 0.8), (<4, 6>, 0.5), (<4, 7>, 0.2), (<5, 2>, 0.2), (<5, 3>, 0.5), (<5, 4>, 0.8), (<5, 5>, 1), (<5, 6>, 0.8), (<5, 7>, 0.5), (<5, 8>, 0.2), (<6, 3>, 0.2), (<6, 4>, 0.5), (<6, 5>, 0.8), (<6, 6>, 1), (<6, 7>, 0.8), (<6, 8>, 0.5), (<6, 9>, 0.2), (<7, 4>, 0.2), (<7, 5>, 0.5), (<7, 6>, 0.8), (<7, 7>, 1), (<7, 8>, 0.8), (<7, 9>, 0.5), (<7, 10>, 0.2), (<8, 5>, 0.2), (<8, 6>, 0.5), (<8, 7>, 0.8), (<8, 8>, 1), (<8, 9>, 0.8), (<8, 10>, 0.5), (<9, 6>, 0.2), (<9, 7>, 0.5), (<9, 8>, 0.8), (<9, 9>, 1), (<9, 10>, 0.8), (<10, 7>, 0.2), (<10, 8>, 0.5), (<10, 9>, 0.8), (<10, 10>, 1)\}$.

В этом списке отсутствуют кортежи с нулевым значением функции принадлежности.

Это же бинарное нечеткое отношение может быть задано матрицей $\mathbf{M}_{Q1}$:

$$\mathbf{M}_{Q1} = \begin{bmatrix} 1 & 0.8 & 0.5 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.8 & 1 & 0.8 & 0.5 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.8 & 1 & 0.8 & 0.5 & 0.2 & 0 & 0 & 0 & 0 \\ 0.2 & 0.5 & 0.8 & 1 & 0.8 & 0.5 & 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0.5 & 0.8 & 1 & 0.8 & 0.5 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0.5 & 0.8 & 1 & 0.8 & 0.5 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 & 0.5 & 0.8 & 1 & 0.8 & 0.5 & 0.2 \\ 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.8 & 1 & 0.8 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.8 & 1 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.5 & 0.8 & 1 \end{bmatrix}.$$

Данное нечеткое отношение можно также представить графически в форме совокупности точек в трехмерном пространстве и в форме нечеткого графа. Однако эти представления не совсем удобны для визуализации рассматриваемого нечеткого отношения и поэтому здесь не приводятся.

Пример 4.2. В качестве бесконечного бинарного нечеткого отношения рассмотрим нечеткое отношение Q_2, которое задается на одном универсуме X — множестве неотрицательных действительных чисел $\mathbb{R}_+$. Содержательно отношение Q_2 описывает свойство: "*действительное число x_i значительно больше действительного числа x_j*". Это нечеткое отношение удобно задать аналитически, например, в виде следующей функции принадлежности:

$$\mu_{Q2}(<x_i, x_j>) = 0 \qquad \text{для } x_i \le x_j$$

и

$$\mu_{Q2}(<x_i, x_j>) = 1 - \frac{1}{x_i - x_j} \qquad \text{для } x_i > x_j \ (\forall x_i, x_j \in \mathbb{R}_+). \tag{4.2}$$

Фрагмент данного нечеткого отношения может быть изображен в форме графика этой функции в трехмерном пространстве (рис. 4.1).

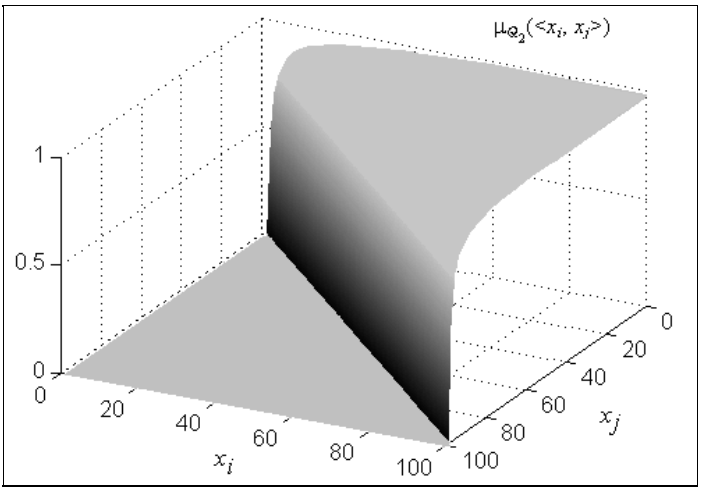


Рис. 4.1. Графическое представление нечеткого отношения Q_2 в форме графика его функции принадлежности

Очевидно, данное бесконечное нечеткое отношение нельзя представить в матричной форме и в форме нечеткого графа.

П р и м е р 4.3. Предположим, необходимо построить нечеткое отношение, которое содержательно описывает упрощенную ситуацию поиска неисправности в автомобиле. С этой целью в качестве первого универсума рассмотрим множество предпосылок или причин неисправности $\boldsymbol{X}=\{x_1, x_2, x_3, x_4\}$, в котором x_1 — "*неисправность аккумулятора*", x_2 — "*неисправность карбюратора*", x_3 — "*низкое качество бензина*", x_4 — "*неисправность системы зажигания*". В качестве второго универсума рассмотрим множество заключений или проявлений неисправности $\boldsymbol{Y}=\{y_1, y_2, y_3\}$, где y_1 — "*двигатель не запускается*", y_2 — "*двигатель работает неустойчиво*", y_3 — "*двигатель не развивает полной мощности*". При этом между каждым элементом множества предпосылок и каждым элементом множества следствий существует некоторая причинная взаимосвязь.

Особенность построения нечеткой модели для описываемой ситуации заключается в том, что рассматриваемая причинная взаимосвязь не является однозначной. Более того, исходя из субъективного опыта конкретного механика, марки автомобиля, условий его эксплуатации и учета других факторов эта причинная взаимосвязь наиболее адекватно может быть представлена в виде бинарного нечеткого отношения $\mathcal{P}=\{<x_i, y_j>, \mu_{\mathcal{P}}(<x_i, y_j>)\}$, заданного на базисных множествах $\boldsymbol{X}$ и $\boldsymbol{Y}$. В этом случае функция принадлежности $\mu_{\mathcal{P}}(<x_i, y_j>)$ этого бинарного нечеткого отношения количественно описывает степень уверенности в том, что та или иная причина неисправности может привести к тому или иному следствию.

Применительно к нашему примеру конкретное нечеткое отношение $\mathcal{P}$ может быть записано в форме списка следующим образом: $\mathcal{P}=\{(<x_1, y_1>, 1), (<x_1, y_2>, 0.1), (<x_1, y_3>, 0.2), (<x_2, y_1>, 0.8), (<x_2, y_2>, 0.9), (<x_2, y_3>, 1), (<x_3, y_1>, 0.7), (<x_3, y_2>, 0.8), (<x_3, y_3>, 0.5), (<x_4, y_1>, 1), (<x_4, y_2>, 0.5), (<x_4, y_3>, 0.2)\}$.

Поскольку нечеткое отношение $\mathcal{P}$ бинарное и конечное, оно может быть представлено в форме табл. 4.1, представленной ниже.

Таблица 4.1. *Нечеткое отношение диагностики неисправности в автомобиле*

	y_1	y_2	y_3
x_1	1	0.1	0.2
x_2	0.8	0.9	1
x_3	0.7	0.8	0.5
x_4	1	0.5	0.2

Эта таблица может быть легко преобразована в матрицу $\mathbf{M}_{\mathcal{P}}$ нечеткого отношения, которая в данном конкретном случае имеет следующий вид:

$$\mathbf{M}_{\mathcal{P}} = \begin{bmatrix} 1 & 0.1 & 0.2 \\ 0.8 & 0.9 & 1 \\ 0.7 & 0.8 & 0.5 \\ 1 & 0.5 & 0.2 \end{bmatrix}.$$

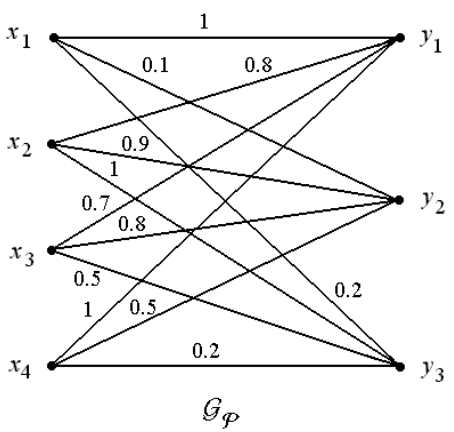


Рис. 4.2. Нечеткий граф отношения $\mathcal{P}$ (стрелки дуг, направленных от вершин x_i к вершинам y_j для удобства не указаны)

Для того чтобы представить это нечеткое отношение в форме нечеткого графа, изобразим на плоскости его вершины, в качестве которых выступают элементы множеств $\boldsymbol{X}$ и $\boldsymbol{Y}$. Соединим эти вершины дугами, направленными от вершин, соответствующих элементам множества $\boldsymbol{X}$, к вершинам, соответствующим элементам множества $\boldsymbol{Y}$. Рядом с каждой из дуг запишем значение ее функции принадлежности. Тем самым получим нечеткий граф $\mathcal{G}_{\mathcal{P}}$ рассматриваемого отношения $\mathcal{P}$ (рис. 4.2).

Что касается аналитического способа представления данного нечеткого отношения, то поскольку отсутствует математическое выражение для записи соответствующей функции принадлежности, использовать этот способ в данном случае не представляется возможным.

Пример 4.4. Модель "Продукция / Рынок", используемая в стратегическом бизнес-планировании. Эта модель, известная также под названием *матрица "продукция/рынок"* или "продукция/рыночная определенность", является классической моделью для разработки корпоративной стратегии.

Данная модель представляет собой практический инструмент для планирования выпускаемой продукции и рынков ее сбыта в зависимости от степени неопределенности перспектив продажи продукции или возможностей проникновения конкретной продукции на тот или иной рынок. Эта матрица строится исходя из субъективных оценок менеджеров с учетом того обстоятельства, что гораздо проще продать имеющимся покупателям уже известную продукцию, чем совершенно новую или мало известную. При этом под продукцией понимаются как товары, так и оказываемые услуги.

Исходя из практического опыта также известно, что продавать существующий ассортимент товаров или услуг категориям потребителей, близким к тем, которые уже приобретали их ранее, проще, чем осваивать совершенно новые рынки. Рассмотренные обстоятельства могут служить основой для задания бинарного нечеткого отношения $\mathcal{R}=\{<x_i, y_j>, \mu_{\mathcal{R}}(<x_i, y_j>)\}$, заданного на базисных множествах $\boldsymbol{X}=\{x_1, x_2, x_3\}$ и $\boldsymbol{Y}=\{y_1, y_2, y_3\}$. При этом элементы базисных множеств имеют следующий содержательный смысл: x_1 — "*имеющийся известный рынок*", x_2 — "*новый рынок, связанный с имеющимся*", x_3 — "*совершенно новый рынок*", y_1 — "*продукция, выпускаемая в настоящее время*", y_2 — "*новая продукция, связанная с выпускаемой*", y_3 — "*совершенно новая продукция*".

Функция принадлежности $\mu_{\mathcal{R}}(<x_i, y_j>)$ рассматриваемого бинарного нечеткого отношения количественно описывает степень уверенности в успешной продаже различного типа продукции на том или ином рынке. При этом в стратегическом бизнес-планировании используется следующее конкретное нечеткое отношение $\mathcal{R}$, записанное в форме списка: $\mathcal{R}=\{(<x_1, y_1>, 0.9), (<x_1, y_2>, 0.6), (<x_1, y_3>, 0.3), (<x_2, y_1>, 0.6), (<x_2, y_2>, 0.4), (<x_2, y_3>, 0.2), (<x_3, y_1>, 0.3), (<x_3, y_2>, 0.2), (<x_3, y_3>, 0.1)\}$.

Наиболее часто данная модель представляется в форме таблицы (табл. 4.2).

Таблица 4.2. *Нечеткое отношение модели "Продукция/Рынок"*

	Продукция, выпускаемая в настоящее время	**Новая продукция, связанная с выпускаемой**	**Совершенно новая продукция**
Имеющийся известный рынок	0.9	0.6	0.3
Новый рынок, связанный с имеющимся	0.6	0.4	0.2
Совершенно новый рынок	0.3	0.2	0.1

Эта таблица легко преобразуется в матрицу нечеткого отношения

$$\mathbf{M}_{\mathcal{R}} = \begin{bmatrix} 0.9 & 0.6 & 0.3 \\ 0.6 & 0.4 & 0.2 \\ 0.3 & 0.6 & 0.1 \end{bmatrix},$$

откуда и произошло название рассматриваемой модели.

Аналитический способ представления данного нечеткого отношения отсутствует, поскольку отсутствует компактное математическое выражение для записи соответствующей функции принадлежности. Очевидно, рассматриваемое нечеткое отношение можно также представить в форме нечеткого графа, что предлагается выполнить читателям в качестве упражнения.

Примечание

В качестве иллюстрации модели "Продукция/Рынок" рассмотрим практическую ситуацию, связанную с деятельностью менеджера по продаже престижных марок автомобилей бизнесменам. Задача менеджера состоит в том, чтобы обеспечить максимальный уровень продаж автомобилей, учитывая индивидуальные предпочтения клиентов. Эта ситуация соответствует левой верхней клетке в табл. 4.2. Если этот бизнес процветает, менеджер может принять решение о его расширении. Один из вариантов расширения может быть основан на решении организовать сеть станций технического обслуживания престижных марок автомобилей, проданных бизнесменам. Этот вариант будет соответствовать левой средней клетке таблицы. Другой вариант — начать продавать бизнесменам бытовую электронику (левая нижняя клетка в таблице).

Иная ситуация расширения может быть связана с решением продавать престижные автомобили не только бизнесменам, но и другим категориям покупателей, например, женам бизнесменов (второй столбец таблицы) или широким слоям населения (третий столбец таблицы). В этой ситуации модель "Продукция/Рынок" количественно характеризует успешность того или иного варианта расширения рассматриваемого бизнеса.

П р и м е р 4.5. Рассмотрение примеров нечетких отношений завершим моделью изучения профилей бизнес-систем, которые используются для комплексного анализа текущего состояния последних. С этой целью в качестве первого универсума введем в рассмотрение множество качественных признаков: $\boldsymbol{X}=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$, элементы которого имеют следующий содержательный смысл: x_1 — "*качество выпускаемой продукции*", x_2 — "*производственные мощности*", x_3 — "*финансовые возможности*", x_4 — "*конкурентоспособность*", x_5 — "*общий уровень себестоимости продукции*", x_6 — "*компетенция руководителей*", x_7 — "*наличие стабильных рынков сбыта*", x_8 — "*наличие налоговых льгот*", x_9 — "*возможности выхода на международные рынки*", x_{10} — "*наличие таможенных льгот*". Очевидно, перечень признаков можно продолжить.

В качестве элементов второго базисного множества выступают бизнес-системы, которые подлежат комплексному анализу. Например, пусть это множество состоит из 3-х бизнес-систем: $\boldsymbol{Y}=\{y_1, y_2, y_3\}$. Тогда задача профилирования бизнес-систем заключается в формировании бинарного нечеткого отношения $\mathcal{F}=\{<x_i, y_j>, \mu_{\mathcal{F}}(<x_i, y_j>)\}$. Один из вариантов решения этой задачи может быть представлен в форме таблицы профилей бизнес-систем (табл. 4.3) и соответствующей матрицы нечеткого отношения

$$\mathbf{M}_{\mathcal{F}} = \begin{bmatrix} 0.6 & 0.5 & 0.7 \\ 0.5 & 0.8 & 0.8 \\ 0.7 & 0.4 & 0.9 \\ 0.2 & 0.6 & 0.6 \\ 0.5 & 0.6 & 0.8 \\ 0.7 & 0.5 & 0.7 \\ 0.6 & 0.8 & 0.5 \\ 0.6 & 0.7 & 0.5 \\ 0.3 & 0.5 & 0.8 \\ 0.1 & 0.3 & 0.4 \end{bmatrix}.$$

Таблица 4.3. *Нечеткое отношение результата профилирования 3-х бизнес-систем*

	Бизнес-система y_1	**Бизнес-система y_2**	**Бизнес-система y_3**
Качество выпускаемой продукции	0.6	0.5	0.7
Производственные мощности	0.5	0.8	0.8
Финансовые возможности	0.7	0.4	0.9

Таблица 4.3 *(окончание)*

	Бизнес-система y_1	**Бизнес-система y_2**	**Бизнес-система y_3**
Конкурентоспособность	0.2	0.6	0.6
Общий уровень себестоимости продукции	0.5	0.6	0.8
Компетенция руководителей	0.7	0.5	0.7
Наличие стабильных рынков сбыта	0.6	0.8	0.5
Наличие налоговых льгот	0.6	0.7	0.5
Возможности выхода на международные рынки	0.3	0.5	0.8
Наличие таможенных льгот	0.1	0.3	0.4

Примечание

Следует заметить, что дальнейшее использование матрицы профилей бизнес-систем является предметом теории принятия решений, в рамках которой разработаны различные модели многокритериальной оценки альтернатив. В этом контексте описанная нечеткая модель может рассматриваться как одна из наиболее конструктивных для многокритериального выбора наиболее предпочтительной бизнес-системы, например, с целью приобретения ее акций или инвестирования капитала.

4.2. Основные характеристики нечетких отношений

Пусть $Q = \{<x_1, x_2,\ldots, x_k>, \mu_Q(<x_1, x_2,\ldots, x_k>)\}$ — произвольное нечеткое k-арное отношение с кортежами из декартова произведения соответствующих универсумов $\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$ и функцией принадлежности $\mu_Q(<x_1, x_2,\ldots, x_k>)$.

Носитель нечеткого отношения. *Носителем* нечеткого отношения Q называется обычное отношение Q_S, которое формально определяется следующим образом:

$$Q_S = \{<x_1, x_2,\ldots, x_k> \mid \mu_Q(<x_1, x_2,\ldots, x_k>)>0\}$$
$$(\forall <x_1, x_2,\ldots, x_k> \in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k). \qquad (4.3)$$

Другими словами, носитель нечеткого отношения содержит те и только те кортежи, для которых значение соответствующей функции принадлежности отлично от 0. Очевидно, данное определение корректно, поскольку как для конечных, так и для бесконечных нечетких отношений выражение (4.3) имеет смысл. При этом пустое нечеткое отношение имеет пустой носитель, поскольку $\mu_{\varnothing}=0$ для любого его кортежа, а носитель полного нечеткого отношения совпадает с этим полным отношением.

Например, носителем нечеткого отношения Q_1, рассмотренного в примере 4.1, является отношение: Q_{S^1} ={<1, 1>, <1, 2>, <1, 3>, <1, 4>, <2, 1>, <2, 2>, <2, 3>, <2, 4>, <2, 5>, <3, 1>, <3, 2>, <3, 3>, <3, 4>, <3, 5>, <3, 6>, <4, 1>, <4, 2>, <4, 3>, <4, 4>, <4, 5>, <4, 6>, <4, 7>, <5, 2>, <5, 3>, <5, 4>, <5, 5>, <5, 6>, <5, 7>, <5, 8>, <6, 3>, <6, 4>, <6, 5>, <6, 6>, <6, 7>, <6, 8>, <6, 9>, <7, 4>, <7, 5>, <7, 6>, <7, 7>, <7, 8>, <7, 9>, <7, 10>, <8, 5>, <8, 6>, <8, 7>, <8, 8>, <8, 9>, <8, 10>, <9, 6>, <9, 7>, <9, 8>, <9, 9>, <9, 10>, <10, 7>, <10, 8>, <10, 9>, <10, 10>}.

О т н о ш е н и е α-у р о в н я. Обобщением носителя нечеткого отношения является понятие *отношения α-уровня*, под которым понимается обычное отношение Q_α, которое формально определяется следующим образом:

$$Q_\alpha = \{<x_1, x_2,\ldots, x_k> \mid \mu_Q<x_1, x_2,\ldots, x_k>) \geq \alpha\}$$
$$(\forall <x_1, x_2,\ldots, x_k> \in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k), \qquad (4.4)$$

где α — некоторое действительное число из интервала [0, 1], т. е. $\alpha\in[0, 1]$.

Примером отношений α-уровня для нечеткого отношения Q_1, рассмотренного в примере 4.1, могут служить отношения: $Q_{0.8}$ ={<1, 1>, <1, 2>, <2, 1>, <2, 2>, <2, 3>, <3, 2>, <3, 3>, <3, 4>, <4, 3>, <4, 4>, <4, 5>, <5, 4>, <5, 5>, <5, 6>, <6, 5>, <6, 6>, <6, 7>, <7, 6>, <7, 7>, <7, 8>, <8, 7>, <8, 8>, <8, 9>, <9, 8>, <9, 9>, <9, 10>, <10, 9>, <10, 10>} и $Q_{0.5}$ ={<1, 1>, <1, 2>, <1, 3>, <2, 1>, <2, 2>, <2, 3>, <2, 4>, <3, 1>, <3, 2>, <3, 3>, <3, 4>, <3, 5>, <4, 2>, <4, 3>, <4, 4>, <4, 5>, <4, 6>, <5, 3>, <5, 4>, <5, 5>, <5, 6>, <5, 7>, <6, 4>, <6, 5>, <6, 6>, <6, 7>, <6, 8>, <7, 5>, <7, 6>, <7, 7>, <7, 8>, <7, 9>, <8, 6>, <8, 7>, <8, 8>, <8, 9>, <8, 10>, <9, 7>, <9, 8>, <9, 9>, <9, 10>, <10, 8>, <10, 9>, <10, 10>}.

В ы с о т а н е ч е т к о г о о т н о ш е н и я. Величина $h_Q = \sup\{\mu_Q(<x_1, x_2,\ldots, x_k>)\}$, где супремум берется по всем значениям функции принадлежности для кортежей $<x_1, x_2,\ldots, x_k>\in\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$, называется *высотой* нечеткого отношения Q.

Например, высота конечного нечеткого отношения Q_1 (см. пример 4.1) равна 1 и соответствует элементам главной диагонали матрицы $\mathbf{M}_{Q1}$ этого отношения. Высота нечеткого отношения Q_2, описывающего свойство "*действительное число x_i значительно больше действительного числа x_j*" (см. пример 4.2), также равна 1. Однако среди элементов универсума $\mathbb{R}_+\times\mathbb{R}_+$ отсутствуют числа, для которых бы $\mu_{Q2}(<x_i, x_j>) = 1$ (см. рис. 4.1). Действительно, какие бы числа мы не рассмотрели, соответствующее значение функции принадлежности всегда будет строго меньше 1.

Нормальное нечеткое отношение. Нечеткое отношение Q называется *нормальным*, если максимальное значение его функции принадлежности равно 1. Формально это означает, что для нормального нечеткого отношения необходимо выполнение следующего условия:

$$\mu_Q(<x_1, x_2,\ldots, x_k>)=1 \qquad (\exists <x_1, x_2,\ldots, x_k>\in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k). \tag{4.5}$$

Нечеткое отношение Q_1 в примере 4.1 является нормальным, поскольку его высота равна 1 и соответствует элементам главной диагонали матрицы этого отношения. Напротив, нечеткое отношение $\mathcal{R}$, представляющее модель "продукция/рынок" (см. пример 4.4), и нечеткое отношение $\mathcal{F}$, представляющее модель профилирования бизнес-систем (см. пример 4.5), не являются нормальными.

Субнормальное нечеткое отношение. Если высота нечеткого отношения равна единице ($h_Q=1$), но условие (4.5) не выполняется, то такое нечеткое отношение будем называть *субнормальным*.

Произвольное непустое нечеткое отношение Q можно сделать субнормальным, используя следующее преобразование:

$$\mu_{Q'}(<x_1, x_2,\ldots, x_k>)=\frac{\mu_Q(<x_1, x_2,\ldots, x_k>)}{h_Q}. \tag{4.6}$$

Очевидно, нечеткое отношение Q_2 (см. пример 4.2) является субнормальным.

Мода нечеткого отношения. Некоторый кортеж $\boldsymbol{w}_m\in\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$ нечеткого отношения Q называется *модой*, если этот кортеж является точкой локального максимума соответствующей функции принадлежности $\mu_Q(<x_1, x_2,\ldots, x_k>)$, т. е. выполняется условие:

$$\boldsymbol{w}_m=\arg\max\{\mu_Q(<x_1, x_2,\ldots, x_k>)\}, \tag{4.7}$$

где максимум рассматривается в некоторой локальной окрестности кортежа $\boldsymbol{w}_m$ из области определения функции принадлежности.

Если произвольное нечеткое отношение имеет моду, совпадающую с его высотой, то преобразование (4.7) дает в результате нормальное нечеткое отношение.

Например, нечеткое отношение Q_1 (см. пример 4.1) имеет 10 мод, соответствующих элементам главной диагонали матрицы этого нечеткого отношения. Напротив, нечеткое отношение Q_2 (см. пример 4.2) не имеет ни одной моды.

Ядро нечеткого отношения. *Ядром* нечеткого отношения Q называется обычное отношение Q_1, которое определяется следующим образом:

$$\begin{aligned} Q_1 =\{&<x_1, x_2,\ldots, x_k>|\ \mu_Q(<x_1, x_2,\ldots, x_k>)=1\} \\ &(<x_1, x_2,\ldots, x_k>\in\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k). \end{aligned} \tag{4.8}$$

Например, ядро нечеткого отношения $\mathcal{P}$ (см. пример 4.3) равно $P_1=\{<x_1, y_1>, <x_2, y_3>, <x_4, y_1>\}$.

Ближайшее четкое отношение. Часто оказывается полезным понятие четкого отношения Q, *ближайшего* к нечеткому отношению Q. Характери-

стическая функция такого отношения может быть определена следующим выражением:

$$\chi_Q(< x_1, x_2,..., x_k >) = \begin{cases} 0, & \text{если } \mu_Q(< x_1, x_2,..., x_k >) < 0.5 \\ 1, & \text{если } \mu_Q(< x_1, x_2,..., x_k >) > 0.5 \\ 0 \text{ или } 1, & \text{если } \mu_Q(< x_1, x_2,..., x_k >) = 0.5 \end{cases}. \qquad (4.9)$$

Например, ближайшее к нечеткому отношению $\mathcal{P}$ (см. пример 4.3) есть отношение: $P=\{<x_1, y_1>, <x_2, y_1>, <x_2, y_2>, <x_2, y_3>, <x_3, y_1>, <x_3, y_2>, <x_4, y_1>\}$.

Границы, *точки перехода*, а также свойство выпуклости нечеткого отношения определяются аналогично нечетким множествам *(см. главу 3)*.

Прежде чем приступить к определению операций, рассмотрим два простейших отношения между двумя нечеткими отношениями. Первое из них — равенство двух нечетких отношений.

Равенство нечетких отношений. Два нечетких отношения считаются *равными*, если они заданы на одних и тех же универсумах $\boldsymbol{X}_1, \boldsymbol{X}_2,..., \boldsymbol{X}_k$, имеют одинаковую арность и их функции принадлежности принимают равные значения на всем декартовом произведении соответствующих универсумов.

Формально равенство двух нечетких множеств можно записать следующим образом. А именно, нечеткое отношение $Q=\{<x_1, x_2,..., x_k>, \mu_Q(<x_1, x_2,..., x_k>)\}$ равно нечеткому отношению $\mathcal{R}=\{<x_1, x_2,..., x_k>, \mu_{\mathcal{R}}(<x_1, x_2,..., x_k>)\}$ (записывается как $Q=\mathcal{R}$) тогда и только тогда, когда значения функций принадлежности этих отношений равны на всем декартовом произведении их универсумов, т. е. выполняется следующее условие:

$$\mu_Q(<x_1, x_2,..., x_k>) = \mu_{\mathcal{R}}(<x_1, x_2,..., x_k>) \qquad (4.10)$$

для любых кортежей $<x_1, x_2,..., x_k> \in \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times ... \times \boldsymbol{X}_k$.

При этом следует отметить, что применительно к бинарным нечетким отношениям матрицы равных отношений и соответствующие им нечеткие графы равны, как это определено для соответствующих математических объектов.

Нечеткое доминирование. Говорят, что нечеткое отношение Q *строго включает в себя (строго доминирует)* нечеткое отношение $\mathcal{R}$ (записывается как $\mathcal{R} \subset Q$), если значения функции принадлежности первого строго больше соответствующих значений функции принадлежности второго, т. е. выполняется следующее формальное условие:

$$\mu_Q(<x_1, x_2,..., x_k>) > \mu_{\mathcal{R}}(<x_1, x_2,..., x_k>) \qquad (4.11)$$

для любых кортежей $<x_1, x_2,..., x_k> \in \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times ... \times \boldsymbol{X}_k$.

Здесь по аналогии с обычными множествами для обозначения строгого доминирования нечетких отношений используется символ "$\subset$". Если в данном определении в условии (4.11) вместо знака строгого неравенства записать знак нестрогого неравенства "$\geq$", то получим определение *нестрогого включения* нечетких отношений или *нестрогого доминирования*, которое обозначается как: $\mathcal{R} \subseteq Q$. При этом

в случае $\mathcal{R} \subseteq Q$ просто говорят, что нечеткое отношение Q *доминирует* нечеткое отношение $\mathcal{R}$, а нечеткое отношение $\mathcal{R}$ *содержится* в нечетком отношении Q.

Если для двух нечетких отношений Q и $\mathcal{R}$, заданных на одних и тех же базисных множествах, не выполняется ни отношение $\mathcal{R} \subseteq Q$, ни отношение $Q \subseteq \mathcal{R}$, то в том случае говорят, что нечеткие отношения Q и $\mathcal{R}$ *несравнимые*.

Как нетрудно заметить, любое нечеткое отношение, не являющееся пустым, строго включает в себя пустое отношение. Другими словами, для любого нечеткого отношения Q всегда справедливо утверждение: $\varnothing \subset Q$, где знак "$\subset$ " понимается в нечетком смысле, поскольку справа от него стоит нечеткое отношение. Из определения нечеткого доминирования также следует, что полное нечеткое отношение строго включает в себя любое нечеткое отношение этой же арности, не являющееся в свою очередь полным нечетким отношением. Другими словами, для любого нечеткого отношения Q всегда справедливо утверждение: $Q \subset \boldsymbol{X}$.

4.3. Операции над нечеткими отношениями

Поскольку каждое нечеткое отношение представляет собой нечеткое множество, то применительно к нечетким отношениям оказываются справедливыми все операции, которые были определены выше в *главе 3*. В то же время при использовании нечетких отношений имеет место целый ряд дополнительных особенностей, которые следует учитывать при оперировании соответствующими понятиями.

Пусть Q и $\mathcal{R}$ — произвольные (конечные или бесконечные) k-арные нечеткие отношения, заданные на одном и том же декартовом произведении универсумов: $\boldsymbol{X}_1 \times \boldsymbol{X}_2 \times \ldots \times \boldsymbol{X}_k$.

Пересечение. *Пересечением* двух нечетких отношений $Q = \{\langle x_1, x_2, \ldots, x_k \rangle | \mu_Q(\langle x_1, x_2, \ldots, x_k \rangle)\}$ и $\mathcal{R} = \{\langle x_1, x_2, \ldots, x_k \rangle | \mu_{\mathcal{R}}(\langle x_1, x_2, \ldots, x_k \rangle)\}$ называется некоторое третье нечеткое отношение $\mathcal{S}$, заданное на этом же декартовом произведении универсумов $\boldsymbol{X}_1 \times \boldsymbol{X}_2 \times \ldots \times \boldsymbol{X}_k$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{S}}(\langle x_1, x_2, \ldots, x_k \rangle) = \min\{\mu_Q(\langle x_1, x_2, \ldots, x_k \rangle), \mu_{\mathcal{R}}(\langle x_1, x_2, \ldots, x_k \rangle)\} \quad (4.12)$$

$$(\forall \langle x_1, x_2, \ldots, x_k \rangle \in \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times \ldots \times \boldsymbol{X}_k).$$

В этом случае результат операции пересечения двух отношений записывается в виде: $\mathcal{S} = Q \cap \mathcal{R}$, где $\mathcal{S} = \{\langle x_1, x_2, \ldots, x_k \rangle | \mu_{\mathcal{S}}(\langle x_1, x_2, \ldots, x_k \rangle)\}$ с функцией принадлежности $\mu_{\mathcal{S}}(\langle x_1, x_2, \ldots, x_k \rangle)$, которая определяется по формуле (4.12). Операцию пересечения нечетких отношений в смысле (4.12) также называют min-пересечением или $\wedge$-пересечением. Поэтому функция принадлежности пересечения двух нечет-

ких отношений, обозначаемая для краткости через $\mu_{\mathcal{S}}(\boldsymbol{w})$, иногда записывается в виде: $\mu_{\mathcal{S}}(\boldsymbol{w})=\mu_{Q}(\boldsymbol{w})\wedge\mu_{\mathcal{R}}(\boldsymbol{w})$ $(\forall \boldsymbol{w}\in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k)$.

П р и м е р 4.6. Для иллюстрации операции пересечения нечетких отношений рассмотрим два бинарных нечетких отношения Q и $\mathcal{R}$, заданных на одном универсуме — числовом множестве $\boldsymbol{X}=\{1, 2, 3, 4, 5\}$. Первое нечеткое отношение Q содержательно описывает условие: "*натуральное число x_i приближенно равно натуральному числу x_j*", аналогично рассмотренному в примере 4.1. Второе нечеткое отношение $\mathcal{R}$ содержательно описывает условие: "*натуральное число x_i значительно превосходит натуральное число x_j*". Пусть эти нечеткие отношения заданы следующими матрицами:

$$\mathbf{M}_{Q}=\begin{bmatrix} 1 & 0.8 & 0.5 & 0.2 & 0 \\ 0.8 & 1 & 0.8 & 0.5 & 0.2 \\ 0.5 & 0.8 & 1 & 0.8 & 0.5 \\ 0.2 & 0.5 & 0.8 & 1 & 0.8 \\ 0 & 0.2 & 0.5 & 0.8 & 1 \end{bmatrix} \text{ и } \mathbf{M}_{\mathcal{R}}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & 0 \\ 0.5 & 0.2 & 0 & 0 & 0 \\ 0.7 & 0.5 & 0.2 & 0 & 0 \\ 0.9 & 0.7 & 0.5 & 0.2 & 0 \end{bmatrix}.$$

Тогда результат пересечения этих нечетких отношений может быть представлен в виде матрицы:

$$\mathbf{M}_{Q\cap\mathcal{R}}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & 0 \\ 0.5 & 0.2 & 0 & 0 & 0 \\ 0.2 & 0.5 & 0.2 & 0 & 0 \\ 0 & 0.2 & 0.5 & 0.2 & 0 \end{bmatrix},$$

которая содержательно соответствует одновременному выполнению двух условий: "*натуральное число x_i приближенно равно натуральному числу x_j*" и "*натуральное число x_i значительно превосходит натуральное число x_j*".

О б ъ е д и н е н и е. *Объединением* двух нечетких отношений Q и $\mathcal{R}$ называется некоторое третье нечеткое отношение $\mathcal{U}$, заданное на этом же декартовом произведении универсумов $\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{U}}(<x_1, x_2,\ldots, x_k>)= \max\{\mu_{Q}(<x_1, x_2,\ldots, x_k>), \mu_{\mathcal{R}}(<x_1, x_2,\ldots, x_k>)\} \quad (4.13)$$

$$(\forall <x_1, x_2,\ldots, x_k>\in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k).$$

В этом случае результат операции объединения двух отношений можно записать в виде: $\mathcal{U}=Q\cup\mathcal{B}$, где $\mathcal{U}=\{<x_1, x_2,\ldots, x_k>|\mu_{\mathcal{U}}(<x_1, x_2,\ldots, x_k>)\}$ с функцией принадлежности $\mu_{\mathcal{U}}(<x_1, x_2,\ldots, x_k>)$, которая определяется по формуле (4.13). Операцию объединения нечетких отношений в смысле (4.13) также называют max-объединением или $\vee$- объединением. Поэтому функция принадлежности объединения двух нечетких отношений, обозначаемая для краткости через $\mu_{\mathcal{U}}(\boldsymbol{w})$, иногда записывается в виде: $\mu_{\mathcal{U}}(\boldsymbol{w})=\mu_{Q}(\boldsymbol{w})\vee\mu_{\mathcal{R}}(\boldsymbol{w})$ $(\forall \boldsymbol{w}\in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k)$.

Для рассмотренного выше примера 4.6 результат объединения соответствующих нечетких отношений может быть представлен в виде матрицы:

$$\mathbf{M}_{\mathcal{Q}\cup\mathcal{R}} = \begin{bmatrix} 1 & 0.8 & 0.5 & 0.2 & 0 \\ 0.8 & 1 & 0.8 & 0.5 & 0.2 \\ 0.5 & 0.8 & 1 & 0.8 & 0.5 \\ 0.7 & 0.5 & 0.8 & 1 & 0.8 \\ 0.9 & 0.7 & 0.5 & 0.8 & 1 \end{bmatrix},$$

которая содержательно соответствует выполнению двух условий: "*натуральное число x_i приближенно равно натуральному числу x_j*" или "*натуральное число x_i значительно превосходит натуральное число x_j*".

Р а з н о с т ь. *Разностью* двух нечетких отношений $\mathcal{Q}$ и $\mathcal{R}$ называется такое нечеткое отношение $\mathcal{T}$, заданное на этом же декартовом произведении универсумов $\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{T}}(<x_1, x_2,\ldots, x_k>) = \max\{\mu_{\mathcal{Q}}(<x_1, x_2,\ldots, x_k>)-\mu_{\mathcal{R}}(<x_1, x_2,\ldots, x_k>), 0\} \qquad (4.14)$$

$$(\forall <x_1, x_2,\ldots, x_k>\in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k).$$

Здесь под знаком максимума применяется обычная операция арифметической разности. Операция разности двух нечетких отношений в смысле (4.14) по аналогии с обычными отношениями также обозначается знаком "\". В этом случае результат операции разности двух отношений можно записать в виде: $\mathcal{T}=\mathcal{Q}\setminus\mathcal{R}$, где $\mathcal{T}=\{<x_1, x_2,\ldots, x_k>|\mu_{\mathcal{T}}(<x_1, x_2,\ldots, x_k>)\}$ с функцией принадлежности $\mu_{\mathcal{T}}(<x_1, x_2,\ldots, x_k>)$, которая определяется по формуле (4.14).

Для рассмотренного выше примера 4.6 результат разности $\mathcal{Q}\setminus\mathcal{R}$ соответствующих нечетких отношений может быть представлен в виде матрицы:

$$\mathbf{M}_{\mathcal{Q}\setminus\mathcal{R}} = \begin{bmatrix} 1 & 0.8 & 0.5 & 0.2 & 0 \\ 0.6 & 1 & 0.8 & 0.5 & 0.2 \\ 0 & 0.6 & 1 & 0.8 & 0.5 \\ 0 & 0 & 0.6 & 1 & 0.8 \\ 0 & 0 & 0 & 0.6 & 1 \end{bmatrix},$$

которая содержательно соответствует выполнению двух условий: "*натуральное число x_i приближенно равно натуральному числу x_j*" и одновременно "*натуральное число x_i не превосходит значительно натуральное число x_j*".

С и м м е т р и ч е с к а я р а з н о с т ь. Операция *симметрической разности* двух нечетких отношений $\mathcal{Q}$ и $\mathcal{R}$ (здесь мы будем обозначать ее через Θ) по определению есть такое нечеткое отношение $\mathcal{Q}\Theta\mathcal{R}$, функция принадлежности которого равна:

$$\mu_{\mathcal{Q}\square\mathcal{R}}(<x_1, x_2,\ldots, x_k>)= |\mu_{\mathcal{Q}}(<x_1, x_2,\ldots, x_k>) — \mu_{\mathcal{R}}(<x_1, x_2,\ldots, x_k>)| \qquad (4.15)$$

$$(\forall <x_1, x_2,\ldots, x_k>\in \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k).$$

Здесь в правой части выражения применяется операция модуля (или вычисления абсолютного значения) числа. При этом оказывается справедливым следующее

утверждение: $Q \square \mathcal{R} = (Q \setminus \mathcal{R}) \cup (\mathcal{R} \setminus Q)$, т. е. симметрическая разность двух нечетких отношений представляет собой объединение двух разностей нечетких отношений Q и $\mathcal{R}$.

Операции $\wedge$-пересечения и $\vee$-объединения, а также операции разности и симметрической разности нечетких отношений сохраняют свои определения для случая обычных отношений. А именно, если в качестве нечетких отношений Q и $\mathcal{R}$ взять обычные отношения как их частный случай, то все определения нечетких операций будут справедливы и для характеристических функций этих обычных отношений.

Для рассмотренного выше примера 4.6 результат симметрической разности $Q \ominus \mathcal{R}$ соответствующих нечетких отношений может быть представлен в виде матрицы:

$$\mathbf{M}_{Q \square \mathcal{R}} = \begin{bmatrix} 1 & 0.8 & 0.5 & 0.2 & 0 \\ 0.6 & 1 & 0.8 & 0.5 & 0.2 \\ 0 & 0.6 & 1 & 0.8 & 0.5 \\ 0.5 & 0 & 0.6 & 1 & 0.8 \\ 0.9 & 0.5 & 0 & 0.6 & 1 \end{bmatrix}.$$

Дополнение. Унарная операция *дополнения* нечеткого отношения Q обозначается через $\bar{Q}$ и определяется аналогично операции дополнения нечеткого множества. А именно, $\bar{Q} = \{\langle x_1, x_2, \ldots, x_k \rangle | \mu_{\bar{Q}}(\langle x_1, x_2, \ldots, x_k \rangle)\}$, где функция принадлежности $\mu_{\bar{Q}}(\langle x_1, x_2, \ldots, x_k \rangle)$ определяется по следующей формуле:

$$\mu_{\bar{Q}}(\langle x_1, x_2, \ldots, x_k \rangle) = 1 - \mu_Q(\langle x_1, x_2, \ldots, x_k \rangle) \quad (4.16)$$
$$(\forall \langle x_1, x_2, \ldots, x_k \rangle \in \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times \ldots \times \boldsymbol{X}_k).$$

Для рассмотренного выше примера 4.6 дополнения соответствующих нечетких отношений могут быть представлены в виде матриц:

$$\mathbf{M}_{\bar{Q}} = \begin{bmatrix} 0 & 0.2 & 0.5 & 0.8 & 1 \\ 0.2 & 0 & 0.2 & 0.5 & 0.8 \\ 0.5 & 0.2 & 0 & 0.2 & 0.5 \\ 0.8 & 0.5 & 0.2 & 0 & 0.2 \\ 1 & 0.8 & 0.5 & 0.2 & 0 \end{bmatrix} \text{ и } \mathbf{M}_{\bar{\mathcal{R}}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0.8 & 1 & 1 & 1 & 1 \\ 0.5 & 0.8 & 1 & 1 & 1 \\ 0.3 & 0.5 & 0.8 & 1 & 1 \\ 0.1 & 0.3 & 0.5 & 0.8 & 1 \end{bmatrix}.$$ Для сравнения приведем матрицы обратных нечетких отношений:

$$\mathbf{M}_{Q^{-1}} = \begin{bmatrix} 1 & 0.8 & 0.5 & 0.2 & 0 \\ 0.8 & 1 & 0.8 & 0.5 & 0.2 \\ 0.5 & 0.8 & 1 & 0.8 & 0.5 \\ 0.2 & 0.5 & 0.8 & 1 & 0.8 \\ 0 & 0.2 & 0.5 & 0.8 & 1 \end{bmatrix} \text{ и } \mathbf{M}_{\mathcal{R}^{-1}} = \begin{bmatrix} 0 & 0.2 & 0.5 & 0.7 & 0.9 \\ 0 & 0 & 0.2 & 0.5 & 0.7 \\ 0 & 0 & 0 & 0.2 & 0.5 \\ 0 & 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$ Как можно заметить, $\mathbf{M}_Q = \mathbf{M}_{Q^{-1}}$, но $\mathbf{M}_{\mathcal{R}} \neq \mathbf{M}_{\mathcal{R}^{-1}}$.

Для рассмотренных операций над нечеткими отношениями имеют место фундаментальные свойства, аналогичные свойствам нечетких теоретико-множест-

венных операций (3.8)—(3.18). Поэтому, если рассмотреть алгебру нечетких отношений (нечеткую алгебраическую систему): $\mathcal{L}=<Q, \cup, \cap, \bar{}, \varnothing, X>$ с операциями, которые были определены выше, то она будет являться дистрибутивной решеткой (структурой) с единственными единичными элементами: $\varnothing$ относительно операции нечеткого объединения и X относительно операции нечеткого пересечения.

Наряду с этим можно также рассматривать алгебру (алгебраическую систему) матриц нечетких отношений: $\mathcal{M}=<\mathbf{M}_Q, \wedge, \vee, \bar{}, \mathbf{M}_0, \mathbf{M}_1>$ с операциями $\wedge$-пересечения и $\vee$-объединения матриц одинакового размера. Эти матричные операции могут быть определены поэлементно по формулам (4.12) и (4.13) соответственно. Операция дополнения для матрицы может быть определена согласно формуле (4.16). Что касается специальных матриц $\mathbf{M}_0$ и $\mathbf{M}_1$, то элементы первой из них тождественно равны 0, а элементы второй — тождественно равны 1. Определенная таким способом алгебра матриц $\mathcal{M}$ также будет являться дистрибутивной решеткой (структурой) с единственными единичными элементами: $\mathbf{M}_0$ относительно операции нечеткого $\vee$-объединения матриц и $\mathbf{M}_1$ относительно операции нечеткого $\wedge$-пересечения.

Примечание

Для нечетких отношений сохраняют свой смысл альтернативные и дополнительные операции над нечеткими множествами. А именно, применительно к нечетким отношениям могут быть аналогично определены операции: алгебраическое пересечение (3.19), алгебраическое объединение (3.20), граничное пересечение (3.31), граничное объединение (3.32), драстическое пересечение (3.33), драстическое объединение (3.34), умножение нечеткого отношения на число (3.45), возведение нечеткого отношения в степень (3.46), концентрирование (3.47), растяжение (3.48), выпуклая комбинация (3.49) и дизъюнктивная сумма (3.50). Формальную запись соответствующих определений и проверку свойств альтернативных операций читателям предлагается выполнить самостоятельно в качестве упражнения.

Композиция бинарных нечетких отношений

Пусть Q и $\mathcal{R}$ — конечные или бесконечные бинарные нечеткие отношения. Причем нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$ задано на декартовом произведении универсумов $X_1 \times X_2$, а нечеткое отношение $\mathcal{R}=\{<x_j, x_k>, \mu_{\mathcal{R}}(<x_j, x_k>)\}$ — на декартовом произведении универсумов $X_2 \times X_3$.

Композиция двух бинарных нечетких отношений. Нечеткое бинарное отношение, заданное на декартовом произведении $X_1 \times X_3$ и обозначаемое через $Q \otimes \mathcal{R}$, называется *композицией* бинарных нечетких отношений Q и $\mathcal{R}$, а его функция принадлежности определяется следующим выражением:

$$\mu_{Q \otimes \mathcal{R}}(<x_i, x_k>) = \max_{x_j \in X_2} \{\min\{\mu_Q(<x_i, x_j>), \mu_{\mathcal{R}}(<x_j, x_k>)\}\} \qquad (4.17)$$

$$(\forall <x_i, x_k> \in X_1 \times X_3).$$

Определенную таким образом композицию бинарных нечетких отношений называют иногда *(max-min)-композицией* или *максиминной* сверткой нечетких отношений.

Из определения данной операции композиции следует, что она *ассоциативна*, *дистрибутивна* относительно нечеткого объединения, но *не дистрибутивна* относительно нечеткого пересечения. Другими словами, для произвольных бинарных нечетких отношений $\mathcal{P}$, $\mathcal{Q}$ и $\mathcal{R}$, заданных на декартовых произведениях $X_1 \times X_2$, $X_2 \times X_3$ и $X_3 \times X_4$ соответственно, имеет место следующее свойство:

$$\mathcal{P} \otimes (\mathcal{Q} \otimes \mathcal{R}) = (\mathcal{P} \otimes \mathcal{Q}) \otimes \mathcal{R}. \tag{4.18}$$

Для бинарных нечетких отношений $\mathcal{P}$, $\mathcal{Q}$ и $\mathcal{R}$, заданных на декартовых произведениях $X_1 \times X_2$, $X_2 \times X_3$ и $X_2 \times X_3$ соответственно, имеют место следующие свойства:

$$\mathcal{P} \otimes (\mathcal{Q} \cup \mathcal{R}) = (\mathcal{P} \otimes \mathcal{Q}) \cup (\mathcal{P} \otimes \mathcal{R}); \tag{4.19}$$

$$\mathcal{P} \otimes (\mathcal{Q} \cap \mathcal{R}) \neq (\mathcal{P} \otimes \mathcal{Q}) \cap (\mathcal{P} \otimes \mathcal{R}). \tag{4.20}$$

Кроме того, для (max-min)-композиции произвольных бинарных нечетких отношений $\mathcal{P}$, $\mathcal{Q}$ и $\mathcal{R}$, заданных на декартовых произведениях $X_1 \times X_2$, $X_2 \times X_3$ и $X_2 \times X_3$ соответственно, выполняется следующее свойство *монотонности*: если $\mathcal{Q} \subseteq \mathcal{R}$, то $(\mathcal{P} \otimes \mathcal{Q}) \subseteq (\mathcal{P} \otimes \mathcal{R})$.

П р и м е р 4.8. Рассмотрим типичную ситуацию, связанную с консалтингом в области выбора профессии для последующего обучения и получения соответствующей специальности. С этой целью построим нечеткую модель, основанную на двух бинарных нечетких отношениях $\mathcal{S}$ и $\mathcal{T}$. Первое из этих нечетких отношений строится на двух базисных множествах X и Y, а второе — на двух базисных множествах Y и Z. Здесь X описывает множество специальностей, по которым проводится набор на обучение, Y — множество психо-физиологических характеристик, а Z — множество кандидатов на обучение. В интересуемом нас контексте нечеткое отношение $\mathcal{S}$ содержательно описывает психо-физиологическое профилирование специальностей, а $\mathcal{T}$ — психо-физиологическое профилирование кандидатов на обучение.

Для конкретности, пусть $X=\{x_1, x_2, x_3, x_4, x_5\}$, $Y=\{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}\}$ и $Z=\{z_1, z_2, z_3, z_4\}$. Элементы универсумов имеют следующий содержательный смысл:

- x_1 — "менеджер", x_2 — "программист", x_3 — "водитель", x_4 — "секретарь-референт", x_5 — "переводчик";
- y_1 — "быстрота и гибкость мышления", y_2 — "умение быстро принимать решения", y_3 — "устойчивость и концентрация внимания", y_4 — "зрительная память", y_5 — "быстрота реакции", y_6 — "двигательная память", y_7 — "физическая выносливость", y_8 — "координация движений", y_9 — "эмоционально-волевая устойчивость", y_{10} — "ответственность";
- z_1 — "Петров", z_2 — "Иванов", z_3 — "Сидоров", z_4 — "Васильева", z_5 — "Григорьева".

Конкретные значения функций принадлежности $\mu_{\mathcal{S}}(<x_i, y_j>)$ и $\mu_{\mathcal{T}}(<y_j, z_k>)$ рассматриваемых нечетких отношений представлены следующими таблицами (табл. 4.4 и 4.5).

Таблица 4.4. *Нечеткое отношение $\mathcal{S}$ профилирования специальностей обучения*

	Быстрота и гибкость мышления	**Умение быстро принимать решения**	**Устойчивость и концентрация внимания**	**Зрительная память**	**Быстрота реакции**
Менеджер	0.9	0.9	0.8	0.4	0.5
Программист	0.8	0.5	0.9	0.3	0.1
Водитель	0.3	0.9	0.6	0.5	0.9
Секретарь	0.5	0.4	0.5	0.5	0.2
Переводчик	0.7	0.8	0.8	0.2	0.6
	Двигательная память	**Физическая выносливость**	**Координация движений**	**Эмоционально-волевая устойчивость**	**Ответственность**
Менеджер	0.3	0.6	0.2	0.9	0.8
Программист	0.2	0.2	0.2	0.5	0.5
Водитель	0.8	0.9	0.8	0.6	0.3
Секретарь	0.2	0.3	0.3	0.9	0.8
Переводчик	0.2	0.2	0.3	0.3	0.2

Таблица 4.5. *Нечеткое отношение $\mathcal{T}$ профилирования кандидатов на обучение*

	Петров	**Иванов**	**Сидоров**	**Васильева**	**Григорьева**
Быстрота и гибкость мышления	0.9	0.8	0.7	0.9	1
Умение быстро принимать решения	0.6	0.4	0.8	0.5	0.6
Устойчивость и концентрация внимания	0.5	0.2	0.3	0.8	0.7
Зрительная память	0.5	0.9	0.5	0.8	0.4

Таблица 4.5 *(окончание)*

	Петров	Иванов	Сидоров	Васильева	Григорьева
Быстрота реакции	1	0.6	0.5	0.7	0.4
Двигательная память	0.4	0.5	1	0.7	0.8
Физическая выносливость	0.5	0.8	0.9	0.5	0.4
Координация движений	0.5	0.6	0.7	0.6	0.5
Эмоционально-волевая устойчивость	0.8	1	0.2	0.5	0.6
Ответственность	0.3	0.5	0.9	0.6	0.8

Матрицы этих нечетких отношений имеют следующий вид:

$$\mathbf{M}_S = \begin{bmatrix} 0.9 & 0.9 & 0.8 & 0.4 & 0.5 & 0.3 & 0.6 & 0.2 & 0.9 & 0.8 \\ 0.8 & 0.5 & 0.9 & 0.3 & 0.1 & 0.2 & 0.2 & 0.2 & 0.5 & 0.5 \\ 0.3 & 0.9 & 0.6 & 0.5 & 0.9 & 0.8 & 0.9 & 0.8 & 0.6 & 0.3 \\ 0.5 & 0.4 & 0.5 & 0.5 & 0.2 & 0.2 & 0.3 & 0.3 & 0.9 & 0.8 \\ 0.7 & 0.8 & 0.8 & 0.2 & 0.6 & 0.2 & 0.2 & 0.3 & 0.3 & 0.2 \end{bmatrix};$$

$$\mathbf{M}_T = \begin{bmatrix} 0.9 & 0.8 & 0.7 & 0.9 & 1 \\ 0.6 & 0.4 & 0.8 & 0.5 & 0.6 \\ 0.5 & 0.2 & 0.3 & 0.8 & 0.7 \\ 0.5 & 0.9 & 0.5 & 0.8 & 0.4 \\ 1 & 0.6 & 0.5 & 0.7 & 0.4 \\ 0.4 & 0.5 & 1 & 0.7 & 0.8 \\ 0.5 & 0.8 & 0.9 & 0.5 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.6 & 0.5 \\ 0.8 & 1 & 0.2 & 0.5 & 0.6 \\ 0.3 & 0.5 & 0.9 & 0.6 & 0.8 \end{bmatrix}.$$

Поскольку рассматриваемые нечеткие отношения удовлетворяют формальным требованиям, необходимым для выполнения их нечеткой композиции согласно

(4.17), результат операции нечеткой композиции этих отношений может быть представлен в виде матрицы результирующего нечеткого отношения:

$$\mathbf{M}_{S \otimes T} = \begin{bmatrix} 0.9 & 0.9 & 0.8 & 0.9 & 0.9 \\ 0.8 & 0.8 & 0.7 & 0.8 & 0.8 \\ 0.9 & 0.8 & 0.9 & 0.7 & 0.8 \\ 0.8 & 0.9 & 0.8 & 0.6 & 0.8 \\ 0.7 & 0.7 & 0.8 & 0.8 & 0.7 \end{bmatrix}.$$

Для наглядности преобразуем эту матрицу к табличной форме (табл. 4.6).

Таблица 4.6. *Нечеткая композиция двух исходных отношений*

	Петров	**Иванов**	**Сидоров**	**Васильева**	**Григорьева**
Менеджер	0.9	0.9	0.8	0.9	0.9
Программист	0.8	0.8	0.7	0.8	0.8
Водитель	0.9	0.8	0.9	0.7	0.8
Секретарь	0.8	0.9	0.8	0.6	0.8
Переводчик	0.7	0.7	0.8	0.8	0.7

Рассмотрим, каким образом получается одно из значений функции принадлежности композиции, например, значение $\mu_{Q \otimes R}(<x_1, y_1>) = 0.9$. Вначале найдем минимальные значения функции принадлежности всех пар элементов первой строки табл. 4.4 и первого столбца табл. 4.5. А именно: $\min\{0.9, 0.9\} = 0.9$, $\min\{0.9, 0.8\} = 0.8$, $\min\{0.8, 0.5\} = 0.5$, $\min\{0.4, 0.5\} = 0.4$, $\min\{0.5, 1\} = 0.5$, $\min\{0.3, 0.4\} = 0.3$, $\min\{0.6, 0.5\} = 0.5$, $\min\{0.2, 0.5\} = 0.2$, $\min\{0.9, 0.8\} = 0.8$, $\min\{0.8, 0.3\} = 0.3$. После этого найдем максимальное из 10 полученных значений, которое и будет являться искомым значением функции принадлежности: $\mu_{Q \otimes R}(<x_1, y_1>) = \max\{0.9, 0.8, 0.5, 0.4, 0.5, 0.3, 0.5, 0.2, 0.8, 0.3\} = 0.9$. Остальные значения функции принадлежности находятся аналогично.

Примечание

Операцию композиции нечетких отношений можно распространить на матрицы соответствующих нечетких отношений. В этом случае результатом композиции матрицы $\mathbf{M_1}$ размерности $(n \times m)$ и матрицы $\mathbf{M_2}$ размерности $(m \times k)$ будет матрица $\mathbf{M_3}$ размерности $(n \times k)$, элементы которой получаются согласно формуле (4.17). Тем самым оказывается корректным следующее обозначение: $\mathbf{M_3} = \mathbf{M_1} \otimes \mathbf{M_2}$, которое будет нами использоваться в дальнейшем.

Анализ табл. 4.6 показывает, что имеющимся кандидатам можно порекомендовать обучение по следующим специальностям (на основе максимальных значений функции принадлежности композиции рассматриваемых нечетких отноше-

ний): Петров — *менеджер*, *водитель*; Иванов — *менеджер*, *секретарь*; Сидоров — *водитель*; Васильева — *менеджер*, Григорьева — *менеджер*. С точки зрения подготовки рассматриваемых специалистов для обучения по специальности *менеджер* наиболее подходят кандидаты: Петров, Иванов, Васильева и Григорьева; по специальности *программист* — те же кандидаты; по специальности *водитель* — Сидоров; по специальности *секретарь* — Иванов; по специальности *переводчик* — Сидоров и Васильева.

Альтернативные операции композиции двух бинарных нечетких отношений. Нечеткое бинарное отношение, заданное на декартовом произведении $X_1 \times X_3$ и обозначаемое через $Q * \mathcal{R}$, называется *(max-*)-композицией* бинарных нечетких отношений Q и $\mathcal{R}$, если его функция принадлежности определяется следующим выражением:

$$\mu_{Q*\mathcal{R}}(<x_i, x_k>) = \max_{x_j \in X_2} \{\mu_Q(<x_i, x_j>) * \mu_{\mathcal{R}}(<x_j, x_k>)\} \quad (\forall <x_i, x_k> \in X_1 \times X_3) \qquad (4.21)$$

В частности, если в выражении (4.21) вместо операции "*" использовать операцию алгебраического умножения, то получим определение *(max-prod)-композиции.*

Проиллюстрируем результат (max-prod)-композиции нечетких отношений из примера 4.8. Эти нечеткие отношения удовлетворяют формальным требованиям, необходимым для выполнения их нечеткой (max-prod)-композиции согласно (4.21). Результат операции нечеткой композиции может быть представлен в виде следующей таблицы (табл. 4.7).

Таблица 4.7. *Нечеткая (max-prod)-композиция двух исходных отношений*

	Петров	**Иванов**	**Сидоров**	**Васильева**	**Григорьева**
Менеджер	0.81	0.90	0.72	0.81	0.90
Программист	0.72	0.64	0.56	0.72	0.80
Водитель	0.90	0.72	0.81	0.63	0.64
Секретарь	0.72	0.90	0.72	0.48	0.64
Переводчик	0.63	0.56	0.64	0.64	0.70

Анализ табл. 4.7 показывает, что имеющимся кандидатам можно порекомендовать обучение по следующим специальностям (на основе максимальных значений функции принадлежности композиции рассматриваемых нечетких отношений): Петров — *водитель*; Иванов — *менеджер*, *секретарь*; Сидоров — *водитель*; Васильева — *менеджер*, Григорьева — *менеджер*. С точки зрения подготовки рассматриваемых специалистов для обучения по специальности *менеджер* наиболее подходят кандидаты: Иванов и Григорьева; по специальности *программист* — Григорьева; по специальности *водитель* — Петров; по специальности *секретарь* — Иванов; по специальности *переводчик* — Сидоров и Васильева.

Примечание

Следуя общим рекомендациям прикладного системного анализа относительно принципа многомодельности, можно сделать следующий вывод. Если при использовании различных моделей получены одинаковые результаты, то этот факт может свидетельствовать о наличии устойчивой связи или закономерности между отдельными элементами моделей. Применительно к рассматриваемым нечетким моделям совпадение результатов, полученных на основе операций (max-min)-композиции и (max-prod)-композиции, дает основание для более уверенных выводов относительно выбора тех или иных специальностей для обучения кандидатов.

4.4. Нечеткое отображение

Давая определение нечеткого отображения, следует иметь в виду, что, с одной стороны, оно является обобщением обычного теоретико-множественного отображения, а с другой стороны, частным случаем бинарного нечеткого отношения.

Нечеткое отображение. Бинарное нечеткое отношение $\mathcal{F}=\{<x_i, x_j>, \mu_{\mathcal{F}}(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}_1\times\boldsymbol{X}_2$, называется *нечетким отображением*, если для любого $a_i\in\boldsymbol{X}_1$ существует не более одного элемента $a_j\in\boldsymbol{X}_2$ с отличным от нуля значением функции принадлежности $\mu_{\mathcal{F}}(<x_i, x_j>)$. Другими словами, каждому из элементов a_i универсума $\boldsymbol{X}_1$ нечеткое отображение $\mathcal{F}$ ставит в соответствие не более одного элемента a_j из универсума $\boldsymbol{X}_2$, такого что $\mu_{\mathcal{F}}(<x_i, x_j>)>0$. В этом случае говорят, что отображение $\mathcal{F}$ действует *из* универсума $\boldsymbol{X}_1$ *в* универсум $\boldsymbol{X}_2$. Для формальной записи нечеткого отображения используется обозначение, аналогичное обозначению обычного отображения: $\mathcal{F}: \boldsymbol{X}_1\to\boldsymbol{X}_2$, при этом не исключается случай, когда $\boldsymbol{X}_1=\boldsymbol{X}_2$.

Нечеткая функция. Если в качестве универсумов $\boldsymbol{X}_1$ и $\boldsymbol{X}_2$ рассматривать числовые множества, то соответствующее нечеткое отображение естественно назвать *нечеткой функцией*. В этом случае можно использовать общепринятый способ обозначения функциональной зависимости малыми латинскими буквами в форме $f: \boldsymbol{X}_1\to\boldsymbol{X}_2$.

Понятие нечеткого отображения допускает обобщение на декартово произведение произвольного конечного числа универсумов слева от стрелки. Поэтому в общем случае нечеткое отображение может быть записано в виде $\mathcal{F}: \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k\to\boldsymbol{X}$ и ставит в соответствие каждому кортежу $<x_1, x_2,\ldots, x_k>\in \in\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$ не более одного элемента x из универсума $\boldsymbol{X}$, для которого выполняется условие:

$$\mu_{\mathcal{F}}(<x_1, x_2,\ldots, x_k, x>)>0. \qquad (4.22)$$

Нечеткая алгебраическая операция. Аналогичным образом можно ввести понятие *нечеткой алгебраической операции*, которая является частным случаем нечеткого отображения, когда все универсумы $\boldsymbol{X}_1, \boldsymbol{X}_2,\ldots, \boldsymbol{X}_k$ тожде-

ственно равны X. В этом случае нечеткая операция, точнее, *нечеткая k-местная операция*, может быть записана в форме $\mathcal{F}: X \times X \times \ldots \times X \to X$.

Примечание

Для дальнейшего анализа нечетких отображений можно ввести в рассмотрение специальные нечеткие множества, которые по аналогии с обычными отображениями характеризуют особенности структурного строения нечетких отображений. Речь идет об *области определения* нечеткого отображения *и области значений* нечеткого отображения $\mathcal{F}$. В этом случае эти области естественно определить как обычные подмножества соответствующих универсумов, которые являются носителями нечеткого отношения, индуцируемого рассматриваемым нечетким отображением. Поскольку эти понятия довольно редко используются на практике, их формальные определения оставим читателям в качестве упражнения.

Принцип обобщения в теории нечетких множеств

Пусть задано обычное отображение $f: X_1 \times X_2 \times \ldots \times X_k \to X$, где $X_1, X_2, \ldots, X_k, X$ — обычные конечные или бесконечные множества. Предположим, что на основе каждого из множеств $X_1, X_2, \ldots, X_k$, используемых в качестве универсумов, заданы некоторые нечеткие множества $\mathcal{A}_1 = \{x, \mu_{\mathcal{A}1}(x)\}, \mathcal{A}_2 = \{x, \mu_{\mathcal{A}2}(x)\}, \ldots, \mathcal{A}_k = \{x, \mu_{\mathcal{A}k}(x)\}$.

Принцип обобщения утверждает, что отображение f и совокупность нечетких множеств $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ однозначным образом порождают нечеткое отображение $\mathcal{F}: X_1 \times X_2 \times \ldots \times X_k \to X$, функция принадлежности которого определяется по следующей формуле:

$$\mu_{\mathcal{F}}(<x_1, x_2, \ldots, x_k, x>) = \min\{\mu_{\mathcal{A}1}(x_1), \mu_{\mathcal{A}2}(x_2), \ldots, \mu_{\mathcal{A}k}(x_k)\} \quad (4.23)$$

для всех кортежей $<x_1, x_2, \ldots, x_k, x> \in X_1 \times X_2 \times \ldots \times X_k \times X$, таких что $x = f(<x_1, x_2, \ldots, x_k>)$.

Действительно, согласно определению обычного отображения f каждому кортежу $<x_1, x_2, \ldots, x_k> \in X_1 \times X_2 \times \ldots \times X_k$ соответствует единственный элемент $x \in X$, который становится $(k+1)$-м элементом кортежа $<x_1, x_2, \ldots, x_k, x>$, используемого для определения нечеткого отношения $\mathcal{F}$. При этом для всех остальных элементов $y \in X$, таких что $y \neq x$, очевидно $\mu_{\mathcal{F}}(<x_1, x_2, \ldots, x_k, y>) = 0$. Последнее условие является достаточным для того, чтобы нечеткое отношение $\mathcal{F}$ удовлетворяло определению нечеткого отображения (4.22).

Принцип обобщения может быть использован не только для задания нечетких отображений, но и, что более важно, — для формального определения различных нечетких конструкций, обобщающих известные теоретико-множественные понятия. Так, например, на его основе можно дать определение нечеткого декартова произведения нечетких множеств $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$. А именно, *нечетким декартовым произведением* нечетких множеств $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$, заданных на универсумах

$\boldsymbol{X}_1$, $\boldsymbol{X}_2$,..., $\boldsymbol{X}_k$ соответственно, называется такое нечеткое отношение $\mathcal{P}$, которое обозначается через $\mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_k$, а функция принадлежности которого определяется по формуле:

$$\mu_{\mathcal{P}}(<x_1, x_2,..., x_k>) = \min\{\mu_{\mathcal{A}1}(x_1), \mu_{\mathcal{A}2}(x_2),..., \mu_{\mathcal{A}k}(x_k)\} \quad (4.24)$$

$$(\forall <x_1, x_2,..., x_k> \in \boldsymbol{X}_1 \times \boldsymbol{X}_2 \times \ldots \times \boldsymbol{X}_k).$$

Принцип обобщения будет также использован для определения операций с нечеткими числами и интервалами далее в *главе 5*.

4.5. Свойства бинарных нечетких отношений, заданных на одном универсуме

В контексте нечеткого моделирования наибольший интерес представляют такие свойства бинарных нечетких отношений, которые обобщают известные свойства обычных отношений. Интересуемыми нас свойствами являются рефлексивность, симметричность и транзитивность, поскольку эти свойства используются в дальнейшем при определении некоторых специальных типов бинарных нечетких отношений.

Р е ф л е к с и в н о с т ь. Бинарное нечеткое отношение $Q = \{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X} \times \boldsymbol{X}$, называется *рефлексивным*, если для любого из кортежей $<x_i, x_i>$ выполняется равенство:

$$\mu_Q(<x_i, x_i>) = 1 \qquad (\forall x_i \in \boldsymbol{X}). \quad (4.25)$$

Как нетрудно заметить, все элементы главной диагонали матрицы рефлексивного бинарного нечеткого отношения с конечным универсумом равны 1.

Возвращаясь к рассмотренным выше примерам, можно утверждать, что нечеткое отношение Q_1 из примера 4.1 является рефлексивным.

А н т и р е ф л е к с и в н о с т ь. Бинарное нечеткое отношение $Q = \{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X} \times \boldsymbol{X}$, называется *антирефлексивным*, если для любого из кортежей $<x_i, x_i>$ выполняется равенство:

$$\mu_Q(<x_i, x_i>) = 0 \qquad (\forall x_i \in \boldsymbol{X}). \quad (4.26)$$

Как нетрудно заметить, все элементы главной диагонали матрицы антирефлексивного бинарного нечеткого отношения с конечным универсумом равны 0. Поэтому нечеткое отношение Q_2 из примера 4.2 является антирефлексивным.

С и м м е т р и ч н о с т ь. Бинарное нечеткое отношение $Q = \{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X} \times \boldsymbol{X}$, называется *симметричным*, если для любого из кортежей $<x_i, x_j>$ выполняется равенство:

$$\mu_Q(<x_i, x_j>) = \mu_Q(<x_j, x_i>) \qquad (\forall <x_i, x_j> \in \boldsymbol{X} \times \boldsymbol{X}). \quad (4.27)$$

Следует заметить, что матрица симметричного бинарного нечеткого отношения с конечным универсумом симметрична относительно главной диагонали. Это подтверждает нечеткое отношение Q_1 из примера 4.1, которое является симметричным.

А с и м м е т р и ч н о с т ь. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется *асимметричным*, если выполняется следующее условие:

$$\min\{\mu_Q(<x_i, x_j>), \mu_Q(<x_j, x_i>)\} = 0 \qquad (\forall <x_i, x_j>\in \boldsymbol{X}\times\boldsymbol{X}). \tag{4.28}$$

Следует заметить, что все элементы главной диагонали матрицы асимметричного бинарного нечеткого отношения с конечным универсумом равны 0. В дополнение к этому один из двух (а может быть и оба) симметричных относительно главной диагонали элементов должен быть равен 0. Нечеткое отношение Q_2 из примера 4.2 является асимметричным, что непосредственно следует из определения его функции принадлежности.

А н т и с и м м е т р и ч н о с т ь. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется *антисимметричным*, если выполняется следующее условие:

$$\min\{\mu_Q(<x_i, x_j>), \mu_Q(<x_j, x_i>)\} = 0 \tag{4.29}$$

$$(\forall <x_i, x_j>\in \boldsymbol{X}\times\boldsymbol{X}, \text{ причем } x_i\neq x_j).$$

Как не трудно заметить, антисимметричность является более слабым свойством, чем асимметричность. Для выполнения этого свойства требуется лишь, чтобы один из двух (а может быть и оба) симметричных относительно главной диагонали элементов матрицы соответствующего бинарного нечеткого отношения был равен 0. Нечеткое отношение Q_2 из примера 4.2 также является и антисимметричным, что непосредственно следует из определения его функции принадлежности.

Т р а н з и т и в н о с т ь. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется *транзитивным*, если выполняется следующее условие:

$$\mu_Q(<x_i, x_k>)\geq \max_{x_j\in X}\{\min\{\mu_Q(<x_i, x_j>), \mu_Q(<x_j, x_k>)\}\} \tag{4.30}$$

$$(\forall x_i, x_j, x_k\in \boldsymbol{X}).$$

Нечеткое отношение Q_2 из примера 4.2 также является транзитивным, поскольку его функция принадлежности монотонно возрастает относительно разности $x_i - x_j$.

К о т р а н з и т и в н о с т ь. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется *котранзитивным*, если выполняется следующее условие:

$$\mu_Q(<x_i, x_k>)\leq \min_{x_j\in X}\{\max\{\mu_Q(<x_i, x_j>), \mu_Q(<x_j, x_k>)\}\} \tag{4.31}$$

$$(\forall x_i, x_j, x_k\in \boldsymbol{X}).$$

Примечание

Непосредственная проверка свойств транзитивности и котранзитивности для конкретных нечетких отношений представляется довольно трудоемкой процедурой,

поскольку требует большого числа попарных сравнений вида (4.30) и (4.31). Более конструктивным представляется способ эмпирического установления данных свойств на основе выполнения операции нечеткого транзитивного замыкания соответствующего нечеткого отношения, о котором пойдет речь в завершение этого раздела.

Сильная полнота. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется *сильно полным*, если выполняется следующее условие:

$$\max\{\mu_Q(<x_i, x_j>), \mu_Q(<x_j, x_i>)\} = 1 \qquad (4.32)$$

$$(\forall <x_i, x_j> \in \boldsymbol{X}\times\boldsymbol{X}).$$

Нечеткие отношения Q_1 из примера 4.1 и Q_2 из примера 4.2 не являются сильно полными, поскольку для первого из них только элементы главной диагонали матрицы этого отношения равны 1, а второе вообще является субнормальным.

Слабая полнота. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется *слабо полным* (*линейным* или *связным*), если выполняется следующее условие:

$$\max\{\mu_Q(<x_i, x_j>), \mu_Q(<x_j, x_i>)\} > 0 \qquad (4.33)$$

$$(\forall <x_i, x_j> \in \boldsymbol{X}\times\boldsymbol{X}, \text{ причем } x_i \neq x_j).$$

Нечеткое отношение Q_2 из примера 4.2 является слабо полным, поскольку для всех кортежей $<x_i, x_j>\in\boldsymbol{X}\times\boldsymbol{X}$, таких что $x_i>x_j$, по определению функции принадлежности (4.2) $\mu_{Q1}(<x_j, x_i>)>0$. Нечеткое отношение Q_1 из примера 4.1 не является слабо полным.

Операция транзитивного замыкания бинарного нечеткого отношения

Рассмотрим произвольное конечное бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на одном базисном множестве $\boldsymbol{X}$. В основе операции транзитивного замыкания лежит определенная выше операция (max-min)-композиции (4.17) произвольных бинарных нечетких отношений.

Транзитивное замыкание бинарного нечеткого отношения. *Транзитивным замыканием бинарного нечеткого отношения* Q, заданного на конечном универсуме $\boldsymbol{X}$, называется такое бинарное нечеткое отношение $Q^{\mathcal{T}}=\{<x_i, x_j>, \mu_Q^{\mathcal{T}}(<x_i, x_j>)\}$, которое задано на том же универсуме, а его функция принадлежности определяется следующим выражением:

$$\mu_Q^{\mathcal{T}}(<x_i, x_k>) = \max_{x_{j1}, x_{j2}, \ldots, x_{j\,k-1}\in X} \{\min\{\mu_Q(<x_i, x_{j1}>), \mu_Q(<x_{j1}, x_{j2}>), \ldots, \mu_Q(<x_{j,k-1}, x_k>)\}\}$$

$$(\forall <x_i, x_k> \in \boldsymbol{X}\times\boldsymbol{X}, \forall k\in\{1, 2, \ldots, n\}, \text{ где } n=\text{card}(\boldsymbol{X})). \qquad (4.34)$$

Определенную таким образом операцию транзитивного замыкания бинарного нечеткого отношения также называют *(max-min)-транзитивным замыканием* (максиминным транзитивным замыканием).

Можно показать, что транзитивное замыкание произвольного конечного бинарного нечеткого отношения, заданного на одном универсуме, всегда обладает свойством транзитивности.

Для практического выполнения операции транзитивного замыкания бинарного нечеткого отношения удобно использовать представление данного отношения в форме матрицы $\mathbf{M}_Q$. В этом случае результат операции транзитивного замыкания бинарного нечеткого отношения также представляется в форме соответствующей матрицы $M_Q{}^{\mathcal{T}}$, которая может быть получена по следующей формуле:

$$\mathbf{M}_Q{}^{\mathcal{T}} = \mathbf{M}_Q \vee \mathbf{M}_Q{}^2 \vee \mathbf{M}_Q{}^3 \vee \ldots \mathbf{M}_Q{}^k \ldots \qquad (4.35)$$

где через $\mathbf{M}_Q{}^k$ обозначена *k-степень* композиции матрицы $\mathbf{M}_Q$ данного нечеткого отношения. При этом k-степень матрицы бинарного нечеткого отношения определяется рекуррентно следующим выражением:

$$\mathbf{M}_Q{}^k = \mathbf{M}_Q \otimes \mathbf{M}_Q{}^{k-1} \qquad (4.36)$$

для любого натурального $k>1$.

При этом имеет место замечательное свойство, которое существенно упрощает численные расчеты, связанные с выполнением операций (4.35) и (4.36). А именно, для получения матрицы транзитивного замыкания бинарного нечеткого отношения $\mathbf{M}_Q{}^{\mathcal{T}}$ достаточно ограничиться выполнением одного из следующих условий:

- если для некоторого натурального k при условии $1 < k < n$, где $n = \mathrm{card}(\boldsymbol{X})$, выполнено равенство $\mathbf{M}_Q{}^k = \mathbf{M}_Q{}^{k-1}$, то дальнейшие расчеты степеней композиции матрицы исходного нечеткого отношения можно прекратить, а матрица транзитивного замыкания рассматриваемого нечеткого отношения будет равна:
$$\mathbf{M}_Q{}^{\mathcal{T}} = \mathbf{M}_Q \vee \mathbf{M}_Q{}^2 \vee \mathbf{M}_Q{}^3 \vee \ldots \mathbf{M}_Q{}^k \qquad (4.37)$$
- выражение (4.37) всегда имеет место при $k=n$, где $n=\mathrm{card}(\boldsymbol{X})$.

П р и м е р 4.9. В качестве примера использования операции транзитивного замыкания нечеткого отношения рассмотрим так называемую задачу анализа эффективности коммуникации, известную также как задача распространения слухов среди хорошо знакомых между собой людей.

С этой целью рассмотрим в качестве исходного универсума $\boldsymbol{X}=\{x_1, x_2, x_3, \ldots, x_n\}$ некоторую совокупность людей. Определим на этом универсуме бинарное нечеткое отношение $\mathcal{H}$, которое содержательно описывает условие: "*человек x_i хорошо знаком с человеком x_j*". По определению есть все основания считать это нечеткое отношение рефлексивным и симметричным. Однако в общем случае данное нечеткое отношение не является транзитивным, поскольку факт знакомства имеет место в основном между парами людей. Предположим, нас интересует потенциальная возможность передачи информации или распространения слухов среди рассматриваемой совокупности людей. Оказывается, эта задача может быть решена применением операции нечеткого транзитивного замыкания данного нечеткого отношения.

Для простоты, пусть $\boldsymbol{X}=\{x_1, x_2, x_3, x_4, x_5\}$ и нечеткое отношение $\mathcal{H}$ задано следующей матрицей $\mathbf{M}_{\mathcal{H}} = \mathbf{M} = \begin{bmatrix} 1 & 0.8 & 0.4 & 0.2 & 0 \\ 0.8 & 1 & 0.1 & 0.7 & 0.2 \\ 0.4 & 0.1 & 1 & 0.6 & 0.5 \\ 0.2 & 0.7 & 0.6 & 1 & 0 \\ 0 & 0.2 & 0.5 & 0 & 1 \end{bmatrix}$.

Для получения матрицы транзитивного замыкания этого нечеткого отношения последовательно найдем матрицы:

$$\mathbf{M}^2 = \mathbf{M} \otimes \mathbf{M} = \begin{bmatrix} 1 & 0.8 & 0.4 & 0.7 & 0.4 \\ 0.8 & 1 & 0.6 & 0.7 & 0.2 \\ 0.4 & 0.6 & 1 & 0.6 & 0.5 \\ 0.7 & 0.7 & 0.6 & 1 & 0.5 \\ 0.4 & 0.2 & 0.5 & 0.5 & 1 \end{bmatrix},$$

$$\mathbf{M}^3 = \mathbf{M} \otimes \mathbf{M}^2 = \begin{bmatrix} 1 & 0.8 & 0.6 & 0.7 & 0.4 \\ 0.8 & 1 & 0.6 & 0.7 & 0.5 \\ 0.6 & 0.6 & 1 & 0.6 & 0.5 \\ 0.7 & 0.7 & 0.6 & 1 & 0.5 \\ 0.4 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix},$$

$$\mathbf{M}^4 = \mathbf{M} \otimes \mathbf{M}^3 = \begin{bmatrix} 1 & 0.8 & 0.6 & 0.7 & 0.5 \\ 0.8 & 1 & 0.6 & 0.7 & 0.5 \\ 0.6 & 0.6 & 1 & 0.6 & 0.5 \\ 0.7 & 0.7 & 0.6 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix},$$

$$\mathbf{M}^5 = \mathbf{M} \otimes \mathbf{M}^4 = \begin{bmatrix} 1 & 0.8 & 0.6 & 0.7 & 0.5 \\ 0.8 & 1 & 0.6 & 0.7 & 0.5 \\ 0.6 & 0.6 & 1 & 0.6 & 0.5 \\ 0.7 & 0.7 & 0.6 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}.$$

Наконец, можно записать матрицу транзитивного замыкания нечеткого отношения $\mathcal{H}$, которую получим с использованием выражения: $\mathbf{M}^T = \mathbf{M} \vee \mathbf{M}^2 \vee \mathbf{M}^3 \vee \mathbf{M}^4$ (заметим, что $\mathbf{M}^4 = \mathbf{M}^5$).

$$\mathbf{M}^{T} = \begin{bmatrix} 1 & 0.8 & 0.6 & 0.7 & 0.5 \\ 0.8 & 1 & 0.6 & 0.7 & 0.5 \\ 0.6 & 0.6 & 1 & 0.6 & 0.5 \\ 0.7 & 0.7 & 0.6 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}.$$

Анализ этой матрицы показывает, что любой слух достаточно "*быстро*" распространится среди всех без исключения людей рассматриваемой совокупности. Следует отметить, что при этом некоторые из них могут быть даже не знакомы между собой (например, x_1 и x_5).

Примечание

Наряду с максиминным транзитивным замыканием бинарного нечеткого отношения можно определить также (max-prod)-транзитивное замыкание. В этом случае, как не трудно предположить, в качестве композиции используется (max-prod)-композиция нечетких отношений, определяемая выражением (4.21). Проверку свойств (max-prod)-транзитивного замыкания бинарного нечеткого отношения, заданного на одном конечном универсуме, предоставляем заинтересованным читателям в качестве упражнения.

4.6. Некоторые специальные виды нечетких бинарных отношений, заданных на одном базисном множестве

Как и в случае обычных отношений, совместное наличие нескольких свойств может характеризовать общий вид того или иного бинарного нечеткого отношения.

Нечеткое отношение частичного строгого порядка. Бинарное нечеткое отношение $Q = \{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется нечетким отношением *частичного строгого порядка*, если оно одновременно является антирефлексивным, асимметричным и транзитивным.

Нечеткое отношение линейного строгого порядка. Нечеткое отношение частичного строгого порядка, которое дополнительно удовлетворяет условию слабой полноты, называется нечетким отношением *линейного строгого порядка*.

Например, нечеткое отношение Q_2 из примера 4.2 является отношением линейного строгого порядка, поскольку, как было отмечено выше, оно удовлетворяет условиям антирефлексивности, асимметричности, транзитивности и слабой полноты. Что касается нечеткого отношения Q_1 из примера 4.1, то оно не является нечетким отношением строгого порядка.

Нечеткое отношение толерантности. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется отношением *толерантности*, если оно является рефлексивным и симметричным. Нечеткое отношение толерантности также называют отношением *нечеткого сходства*, поскольку оно используется для содержательного представления попарного подобия или похожести различных объектов между собой.

Иллюстрацией нечеткого отношения толерантности может служить нечеткое отношение $\mathcal{H}$ из примера 4.9, представляющее хорошо знакомых между собой людей.

Нечеткое отношение толерантности тесно связано с так называемым нечетким покрытием нечеткого множества.

Нечеткое покрытие нечеткого множества. Система нечетких подмножеств $\mathfrak{I}(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ нечеткого множества $\mathcal{A}$ называется *нечетким покрытием*, если выполняется следующее условие:

$$\bigcup_k \mathcal{A}_k = \mathcal{A}, \qquad (\mathcal{A}_k \in \mathfrak{I}), \tag{4.38}$$

т. е. объединение всех (или части) подмножеств из $\mathfrak{I}(\mathcal{A})$ совпадает (или "*покрывает*") с исходным нечетким множеством $\mathcal{A}$.

Примечание

Используя принцип обобщения теории нечетких множеств, можно показать, что нечеткое отношение толерантности порождает некоторое нечеткое покрытие $\mathfrak{I}(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ нечеткого множества $\mathcal{A}$, если в качестве нечетких подмножеств $\mathcal{A}_k \subseteq \mathcal{A}$ взять двухэлементные нечеткие множества $\mathcal{A}_k=\{x_i, x_j\}$, совокупность всех кортежей которых удовлетворяет условиям нечеткой рефлексивности и симметричности.

Нечеткое отношение эквивалентности. Бинарное нечеткое отношение $Q=\{<x_i, x_j>, \mu_Q(<x_i, x_j>)\}$, заданное на декартовом произведении $\boldsymbol{X}\times\boldsymbol{X}$, называется нечетким отношением *эквивалентности*, если оно одновременно является рефлексивным, симметричным и транзитивным.

Иллюстрацией нечеткого отношения эквивалентности может служить транзитивное замыкание нечеткого отношения $\mathcal{H}$ из примера 4.9, представляющее хорошо знакомых между собой людей.

Нечеткое отношение эквивалентности тесно связано с так называемым нечетким разбиением нечеткого множества.

Нечеткое разбиение нечеткого множества. Система нечетких подмножеств $\mathfrak{R}(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ нечеткого множества $\mathcal{A}$ называется *нечетким разбиением*, если выполняются следующие условия:

$$\bigcup_k \mathcal{A}_k = \mathcal{A}, \qquad (\mathcal{A}_k \in \mathfrak{R}); \tag{4.39}$$

$$h_C < 1, \qquad \text{где} \quad C=\mathcal{A}_l \cap \mathcal{A}_m, \ (\forall \mathcal{A}_l, \mathcal{A}_m \in \mathfrak{R}). \tag{4.40}$$

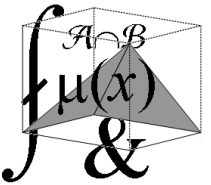
Другими словами, объединение всех (или части) нечетких подмножеств из $\Re(\mathcal{A})$ совпадает (или "покрывает") с исходным нечетким множеством $\mathcal{A}$, при этом высота попарного пересечения любых нечетких подмножеств нечеткого разбиения строго меньше единицы. Нетрудно заметить, что нечеткое разбиение является нечетким покрытием, которое дополнительно удовлетворяет свойству нечеткой транзитивности.

Примечание

Используя принцип обобщения теории нечетких множеств, можно показать, что нечеткое отношение эквивалентности порождает некоторое нечеткое разбиение $\Re(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ нечеткого множества $\mathcal{A}$, если в качестве нечетких подмножеств $\mathcal{A}_k \subseteq \mathcal{A}$ взять нечеткие множества $\mathcal{A}_k$, кортежи всех попарных элементов которых удовлетворяют условиям нечеткой рефлексивности, симметричности и транзитивности.

Дальнейшее рассмотрение концепций теории нечетких множеств связано с изучением специальных случаев нечетких множеств, которые представляют собой лингвистические переменные, а также нечеткие числа и интервалы. Эти понятия и их основные свойства будут рассмотрены в следующей главе.

Глава 5



Нечеткая и лингвистическая переменные. Нечеткие величины, числа и интервалы

Рассмотренное выше понятие нечеткого множества допускает различные уточнения, которые целесообразно использовать для более адекватного отражения семантики неопределенности при построении нечетких моделей сложных систем. Одним из таких уточнений является понятие лингвистической переменной, которое широко используется в нечетком управлении для представления входных и выходных переменных управляемой системы. В этой главе также будут рассмотрены нечеткие аналоги обычных чисел и интервалов, которые оказываются весьма удобным средством для численных расчетов значений соответствующих функции принадлежности при выполнении арифметических операций.

5.1. Определения нечеткой и лингвистической переменных

Нечеткая переменная. *Нечеткая переменная* определяется как кортеж: $<\alpha, \boldsymbol{X}, \mathcal{A}>$, где α — наименование или название нечеткой переменной; $\boldsymbol{X}$ — область ее определения (универсум); $\mathcal{A}=\{x, \mu_{\mathcal{A}}(x)\}$ — нечеткое множество на $\boldsymbol{X}$, описывающее возможные значения, которые может принимать нечеткая переменная α. Таким образом, говоря о нечеткой переменной α, мы всегда будем иметь в виду некоторое нечеткое множество $\mathcal{A}$, которое определяет ее возможные значения.

В качестве примера нечеткой переменной можно привести рассмотренное в *главе 2* нечеткое множество $\mathcal{B}$, которое характеризует "*горячий кофе*" (см. пример 2.2). В этом случае соответствующая нечеткая переменная может быть представлена следующим образом: $<$*Горячий кофе*, $\{x \mid 0\ ^\circ\text{C} < x < 100\ ^\circ\text{C}\}, \mathcal{B}>$, где $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$ — нечеткое множество с функцией принадлежности $\mu_{\mathcal{B}}(x)$, которая может быть задана, в частности, графически (рис. 2.4, *а* или рис. 2.4, *б*).

Обобщением нечеткой переменной является так называемая лингвистическая переменная.

Лингвистическая переменная. *Лингвистическая переменная* также определяется как кортеж: <β, *T*, ***X***, *G*, *M*>, где:

- β — наименование или название лингвистической переменной;
- *T* — базовое *терм-множество* лингвистической переменной или множество ее значений *(термов)*, каждое из которых представляет собой наименование отдельной нечеткой переменной α;
- ***X*** — область определения (универсум) нечетких переменных, которые входят в определение лингвистической переменной β;
- *G* — некоторая синтаксическая процедура, которая описывает процесс образования или генерирования из множества *T* новых, осмысленных в рассматриваемом контексте значений для данной лингвистической переменной;
- *M* — семантическая процедура, которая позволяет поставить в соответствие каждому новому значению данной лингвистической переменной, получаемому с помощью процедуры *G*, некоторое осмысленное содержание посредством формирования соответствующего нечеткого множества.

Пример 5.1. В качестве примера рассмотрим ситуацию со скоростью движения автомобильного транспорта в пределах городской черты. Хотя правила дорожного движения регламентируют величину этой скорости, однако многие автолюбители предпочитают давать собственную субъективную оценку своей скорости движения. При этом используются такие определения, как "малая скорость", "средняя скорость" и "высокая скорость" движения. Очевидно, что подобная практическая оценка скорости может относиться к диапазону скоростей в пределах интервала от 0 км/ч до некоторой величины, определяемой личными предпочтениями того или иного водителя. Пусть в нашем примере из соображений удобства это будет величина 100 км/ч.

Формализация субъективной оценки скорости движения может быть выполнена с помощью следующей лингвистической переменной <β_1, *T*, ***X***, *G*, *M* >, где

- β_1 — скорость движения автомобиля;
- *T* = {"*малая скорость*", "*средняя скорость*", "*высокая скорость*"};
- ***X*** = [0, 100];
- *G* — процедура образования новых термов с помощью связок логических связок "И", "ИЛИ" и модификаторов типа "очень", "НЕ", "слегка" и др. Например: "малая или средняя скорость", "очень высокая скорость" и др.;
- *M* — процедура задания на ***X***=[0, 100] нечетких переменных α_1= "*малая скорость*", α_2 = "*средняя скорость*", α_3 = "*высокая скорость*", а также соответствующих нечетких множеств для термов из *G*(*T*) в соответствии с правилами трансляции нечетких связок и модификаторов "И", "ИЛИ", "НЕ", "очень", "слегка".

Примечание

Конкретные процедуры *G* и *M* будут рассмотрены нами далее в *главе 6*, посвященной изложению основ нечеткой логики. Применительно к данному конкретному примеру можно ограничиться предположением об их тривиальном характере, т. е. никаких логических связок и модификаторов мы не будем использовать.

Для рассматриваемого примера нечеткие множества $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, соответствующие нечетким переменным: α_1 = "*малая скорость*", α_2 = "*средняя скорость*", α_3 = "*высокая скорость*", удобно задать графически с помощью кусочно-линейных функций принадлежности. Один из возможных конкретных вариантов этих нечетких множеств изображен на рис. 5.1.

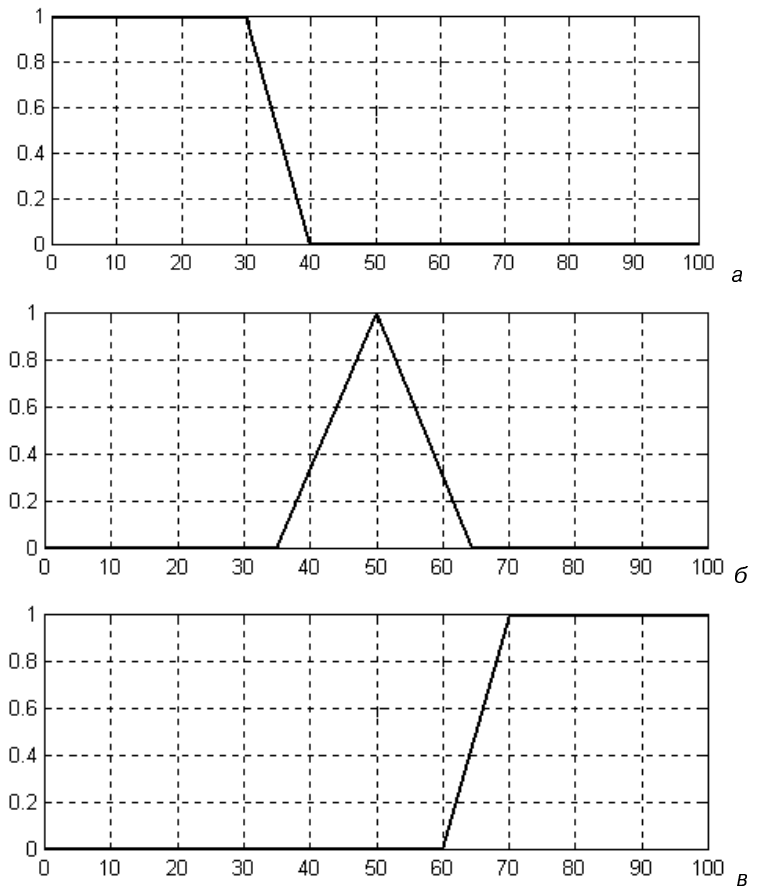


Рис. 5.1. Графики функций принадлежности нечетких множеств $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, соответствующих нечетким переменным α_1= "*малая скорость*" (*а*), α_2 = "*средняя скорость*" (*б*), α_3= "*высокая скорость*" (*в*) для лингвистической переменной β_1 (скорость движения автомобиля)

Иногда для наглядности графики функций принадлежности нескольких нечетких переменных, используемых для задания одной лингвистической переменной, изображают на одном рисунке. Применительно к примеру 5.1 все три графика представлены на рис. 5.2, что позволяет сравнивать значения функций принадлежности соответствующих нечетких переменных для различных значений универсума.

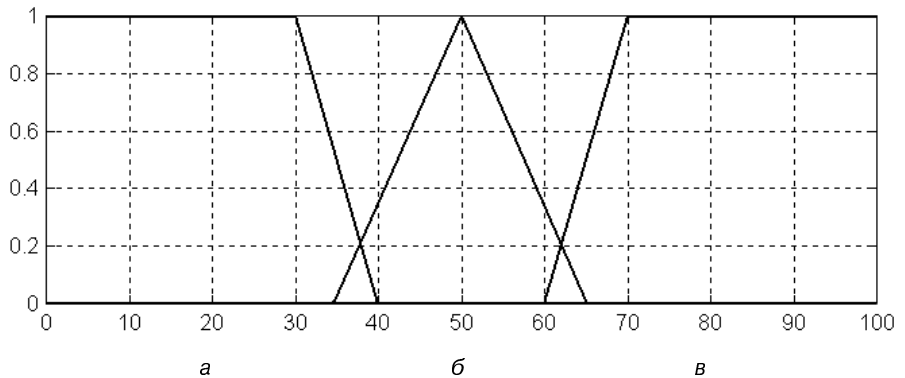


Рис. 5.2. Графики функций принадлежности нечетких множеств $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, изображенные на одном рисунке

Примечание

Наряду с рассмотренными выше базовыми значениями лингвистической переменной "скорость движения автомобиля" (T ={"*малая скорость*", "*средняя скорость*", "*высокая скорость*"}) возможны и другие значения этой же лингвистической переменной, зависящие от конкретной величины скорости движения. Например, могут быть определены такие дополнительные значения лингвистической переменной "*скорость движения автомобиля*", как "около 30 км/ч", "около 50 км/ч", "около 70 км/ч". Как будет видно из дальнейшего изложения, эти значения лингвистической переменной удобно моделировать с помощью нечетких чисел.

5.2. Нечеткие величины, числа и интервалы

Процесс нечеткого моделирования основывается на количественном представлении входных и выходных переменных системы в форме нечетких множеств. Такое представление связано с рассмотрением специальных нечетких множеств, которые задаются на множестве действительных чисел и обладают некоторыми дополнительными свойствами. Наиболее общим понятием в этом контексте является понятие нечеткой величины.

Нечеткая величина. *Нечеткой величиной* называется произвольное нечеткое множество $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$, заданное на множестве действительных чисел $\mathbb{R}$, т. е. для которого универсумом X служит все множество $\mathbb{R}$. Другими словами, функция принадлежности нечеткой величины есть отображение $\mu_{\mathcal{B}}(x)$: $\mathbb{R}\rightarrow[0, 1]$. Если в качестве универсума взять подмножество неотрицательных действительных чисел $\mathbb{R}+$, то получим определение *неотрицательной нечеткой величины* $\mathcal{B}+$.

Примерами нечетких величин являются нечеткие множества, функции принадлежности которых изображены на рис. 2.11—2.18. Более того, все эти нечеткие величины являются неотрицательными. С другой стороны, рассмотренные в примерах 2.1 и 2.3 нечеткие множества не являются нечеткими величинами.

Наибольший интерес для нечеткого моделирования представляет конкретизация нечеткой величины в форме нечетких чисел и интервалов.

Нечеткий интервал. В общем случае *нечетким интервалом* называется нечеткая величина с выпуклой функцией принадлежности.

Примерами нечетких интервалов могут служить нечеткие множества с функциями принадлежности, изображенными на рис. 2.9, *а*, 2.10, *а* и 2.11, *б*, а также на рис. 2.12—2.16. С другой стороны, нечеткое множество с функцией принадлежности, изображенной на рис. 2.10, *б*, не является нечетким интервалом.

Примечание

В литературе нечеткий интервал иногда называют также *толерантным нечетким числом*.

Нечеткое число. В общем случае *нечетким числом* называется такая нечеткая величина, функция принадлежности которой является выпуклой и унимодальной.

Примерами нечетких чисел могут служить нечеткие множества с функциями принадлежности, изображенными на рис. 2.10, *а*, 2.11, *а* и 2.18, *б*. С другой стороны, нечеткое множество с функцией принадлежности, изображенной на рис. 2.10, *б*, не является нечетким интервалом. Как видно из этих примеров, нечеткое число в общем случае является частным случаем нечеткого интервала, что полностью согласуется с обычными числами и интервалами на множестве действительных чисел.

Примечание

При общем определении нечеткого интервала и нечеткого числа не делается никаких предположений относительно нормальности соответствующих нечетких множеств. С другой стороны, функции принадлежности нечетких чисел и интервалов, вообще говоря, могут и не иметь аналитического представления. Все это затрудняет практическое использование этих общих понятий для решения конкретных задач нечеткого моделирования. По этой причине в дальнейшем рассматриваются некоторые способы уточнения данных понятий на основе использования типовых функций принадлежности.

Поскольку нечеткие числа и интервалы представляют собой нечеткие множества, то для них оказываются справедливыми все свойства и операции, определенные ранее для нечетких множеств. Это в полной мере относится к определению нормального нечеткого числа и нормального нечеткого интервала, носителя и ядра, а также свойств выпуклости и унимодальности нечетких чисел и нечетких интервалов, которые были использованы при их определении *(см. главы 2 и 3)*.

Дополнительно нечеткие числа могут характеризоваться следующими свойствами.

Нечеткий нуль. Нечеткое число называется *нечетким нулем*, если его модальное значение (мода) равно 0.

Положительное (отрицательное) нечеткое число. Нечеткое число называется *положительным* (или *отрицательным*) нечетким числом, если оно имеет строго положительный (соответственно, строго отрицательный) носитель.

Операции над нечеткими числами и интервалами

Для нечетких чисел и интервалов в общем случае с использованием принципа обобщения (4.23) могут быть определены аналоги обычных арифметических операций. В этом случае расширенные бинарные арифметические операции (сложение, вычитание, умножение и деление) для нечетких чисел и интервалов определяются через соответствующие операции для обычных действительных чисел.

Пусть $\mathcal{A}$ и $\mathcal{B}$ — произвольные нечеткие числа (нечеткие интервалы) с функциями принадлежности $\mu_{\mathcal{A}}(x)$ и $\mu_{\mathcal{B}}(y)$ соответственно.

Сложение. Операция *сложения* нечетких чисел (интервалов) обозначается через $\mathcal{A}+\mathcal{B} = C = \{z, \mu_C(z)\}$, где функция принадлежности результата $\mu_C(z)$ определяется по формуле:

$$\mu_C(z) = \sup_{z=x+y} \{\min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)\}\}. \qquad (5.1)$$

Вычитание. Операция *вычитания* нечетких чисел (интервалов) обозначается через $\mathcal{A}-\mathcal{B} = C = \{z, \mu_C(z)\}$, где функция принадлежности результата $\mu_C(z)$ определяется по формуле:

$$\mu_C(z) = \sup_{z=x-y} \{\min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)\}\}. \qquad (5.2)$$

Умножение. Операция *умножения* нечетких чисел (интервалов) обозначается через $\mathcal{A}\bullet\mathcal{B} = C = \{z, \mu_C(z)\}$, где функция принадлежности результата $\mu_C(z)$ определяется по формуле:

$$\mu_C(z) = \sup_{z=x\cdot y} \{\min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)\}\}. \qquad (5.3)$$

Д е л е н и е. Операция *деления* нечетких чисел (интервалов) обозначается через $\mathcal{A} \div \mathcal{B} = C = \{z, \mu_C(z)\}$, где функция принадлежности результата $\mu_C(z)$ определяется по формуле:

$$\mu_C(z) = \sup_{z=x \div y} \{\min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)\}\}. \qquad (5.4)$$

В выражениях (5.1)—(5.4) справа от знака равенства супремум берется по каждому из совокупности значений элементов универсума, которые в свою очередь являются результатом соответствующей обычной арифметической операции над численными значениями элементов универсума исходных нечетких чисел (интервалов).

Например, пусть задано нечеткое число — "*нечеткая единица*", которое описывается следующим конечным нечетким множеством: $\mathcal{I} = \{<0, 0.2>, <1, 1.0>, <2, 0.2>\}$.

Рассмотрим выполнение нечеткой операции сложения — "*нечеткая единица*" плюс "*нечеткая единица*" с использованием формулы (5.1). Последовательно получим: $\mathcal{I} + \mathcal{I} = \{<0, 0.2>, <1, 1.0>, <2, 0.2>\} + \{<0, 0.2>, <1, 1.0>, <2, 0.2>\} = \{<0, \min\{0.2, 0.2\}>, <1, \max\{\min\{0.2, 1.0\}, \min\{1.0, 0.2\}\}>, <2, \max\{\min\{0.2, 0.2\}, \min\{1.0, 1.0\}, \min\{0.2, 0.2\}\}>, <3, \max\{\min\{1.0, 0.2\}, \min\{0.2, 1.0\}\}>, <4, \min\{0.2, 0.2\}>\} = \{<0, 0.2>, <1, 0.2>, <2, 1.0>, <3, 0.2>, <4, 0.2>\}$.

Возможно, операция сложения нечетких чисел станет более понятной, если принять во внимание, что значения результата получаются как различные комбинации слагаемых обычной арифметической операции сложения: $0=0+0$, $1=0+1=1+0$, $2=0+2=1+1=2+0$, $3=1+2=2+1$, $4=2+2$. Очевидно, что для конечных множеств вместо операции *супремум* можно использовать операцию *максимум*. Полученное в результате нечеткое число можно назвать "*нечеткая двойка*".

Аналогичным образом можно получить другое нечеткое число — "*нечеткий нуль*", как результат выполнения операции разности с использованием формулы (5.2). В этом случае получим: "*нечеткий нуль*" равен "*нечеткая единица*" минус "*нечеткая единица*" или $\mathcal{I} - \mathcal{I} = \{<0, 0.2>, <1, 1.0>, <2, 0.2>\} - \{<0, 0.2>, <1, 1.0>, <2, 0.2>\} = \{<-2, \min\{0.2, 0.2\}>, <-1, \max\{\min\{0.2, 1.0\}, \min\{1.0, 0.2\}\}>, <0, \max\{\min\{0.2, 0.2\}, \min\{1.0, 1.0\}, \min\{0.2, 0.2\}\}>, <1, \max\{\min\{1.0, 0.2\}, \min\{0.2, 1.0\}\}>, <2, \min\{0.2, 0.2\}>\} = \{<-2, 0.2>, <-1, 0.2>, <0, 1.0>, <1, 0.2>, <2, 0.2>\}$.

Иногда могут представлять интерес операции расширенного максимума и расширенного минимума нечетких чисел (интервалов), которые определяются следующим образом.

Р а с ш и р е н н ы й м а к с и м у м. Операция *расширенного максимума* нечетких чисел (интервалов) обозначается через $\max\{\mathcal{A}, \mathcal{B}\} = C = \{z, \mu_C(z)\}$, где функция принадлежности результата $\mu_C(z)$ определяется по формуле:

$$\mu_C(z) = \sup_{z=\max\{x,\ y\}} \{\min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)\}\}. \qquad (5.5)$$

Расширенный минимум. Операция *расширенного минимума* нечетких чисел (интервалов) обозначается через $\min\{\mathcal{A}, \mathcal{B}\} = C = \{z, \mu_C(z)\}$, где функция принадлежности результата $\mu_C(z)$ определяется по формуле:

$$\mu_C(z) = \sup_{z=\min\{x, y\}} \{\min\{\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(y)\}\}. \qquad (5.6)$$

Например, пусть задано два нечетких числа — "*нечеткая единица*" и "*нечеткий нуль*", которые описываются следующими конечными нечеткими множествами: $\mathcal{I} = \{<0, 0.2>, <1, 1.0>, <2, 0.2>\}$ и $O = \{<-1, 0.1>, <0, 1.0>, <1, 0.1>\}$. Рассмотрим выполнение нечеткой операции расширенного максимума с использованием формулы (5.5). Последовательно получим: $\max\{\mathcal{I}, O\} = \{<0, \max\{\min\{0.2, 0.1\}, \min\{0.2, 1.0\}\}>, <1, \max\{\min\{1.0, 0.1\}, \min\{1.0, 1.0\}, \min\{1.0, 0.1\}, \min\{0.2, 1.0\}\}>, <2, \max\{\min\{0.2, 0.1\}, \min\{0.2, 1.0\}, \min\{0.2, 0.1\}\}>\} = \{<0, 0.2>, <1, 1.0>, <2, 0.2>\}$, т.е. результат равен "*нечеткой единице*".

При этом значения результата получаются как различные комбинации операции обычного максимума над парами значений исходных нечетких множеств: $0 = \max\{0, -1\} = \max\{0, 0\}$, $1 = \max\{1, -1\} = \max\{1, 0\} = \max\{1, 1\} = \max\{0, 1\}$, $2 = \max\{2, -1\} = \max\{2, 0\} = \max\{2, 1\}$.

Аналогичным образом для этого примера можно выполнить нечеткую операцию расширенного минимума с использованием формулы (5.6). Последовательно получим: $min\{\mathcal{I}, O\} = \{<-1, \max\{\min\{0.2, 0.1\}, \min\{1.0, 0.1\}, \min\{0.2, 0.1\}\}>, <0, \max\{\min\{0.2, 1.0\}, \min\{0.2, 0.1\}, \min\{0.2, 1.0\}, \min\{1.0, 1.0\}\}>, <1, \max\{\min\{1.0, 0.1\}, \min\{0.2, 0.1\}\}>\} = \{<-1, 0.1>, <0, 1.0>, <1, 0.1>\}$, т. е. результат равен "*нечеткому нулю*".

5.3. Нечеткие числа и интервалы в форме (*L-R*)-функций

Нечеткие числа и интервалы, которые наиболее часто используются для представления нечетких множеств в нечетком моделировании, являются нормальными. Однако данные выше определения нечеткого числа и нечеткого интервала слишком общие, что затрудняет их практическое использование. С вычислительной точки зрения удобно использовать более конкретные определения нечетких чисел и интервалов в форме аналитической аппроксимации с помощью так называемых *(L-R)-функций.* Получаемые в результате нечеткие числа и интервалы в форме *(L-R)*-функций позволяют охватить достаточно широкий класс конкретных функций принадлежности.

Функции *L*-типа и *R*-типа. *Функция L-типа* (а также и *R-типа*), в общем случае определяется как произвольная функция $L: \mathbb{R} \to [0, 1]$ и $R: \mathbb{R} \to [0, 1]$, заданная на множестве действительных чисел, невозрастающая на подмножестве неотрицательных чисел $\mathbb{R}_+$ и удовлетворяющая следующим дополнительным условиям:

$$L(-x) = L(x), \qquad R(-x) = R(x) \text{ — } \textit{условие четности}; \qquad (5.7)$$

$$L(0) = R(0) = 1 \text{ — } \textit{условие нормирования}. \qquad (5.8)$$

Примечание

Иногда в литературе можно встретить еще одно условие, которому должны, по мнению некоторых авторов, удовлетворять функции (L-R)-типа: $L(1) = R(1) = 0$. Поскольку с одной стороны это условие существенно ограничивает класс функций (L-R)-типа, а с другой стороны, рассматриваемые ниже треугольные нечеткие числа и трапециевидные нечеткие интервалы согласуются с выполнением этого свойства, мы не будем его включать в определение функций (L-R)-типа.

Как нетрудно заметить, рассмотренные ранее в *главе 2* треугольная функция принадлежности $f_{\Delta}(x; a, b, c)$ при $b = 0$ и $a = -c$ (2.8), трапециевидная функция принадлежности $f_{T}(x; a, b, c, d)$ при $a = -d$ и $c = -b$ (2.9), а также Π-образные функции принадлежности (2.19)—(2.22), симметричные относительно оси ординат, являются функциями (L-R)-типа, поскольку удовлетворяют условиям определения (5.7)—(5.8).

Примерами L-функций и, соответственно, R-функций являются также следующие функции, которые в общем случае могут быть заданы аналитически в виде:

$$f(x) = e^{-|x|^p}; \tag{5.9}$$

$$f(x) = \frac{1}{1+|x|^p}, \tag{5.10}$$

где p — некоторый параметр, который удовлетворяет условию: $p \geq 0$. Графики функций этого вида для конкретного значения параметра $p=2$ изображены на рис. 5.3.

Нечеткое число (L-R)-типа. *Нечетким числом (L-R)-типа* будем называть нечеткую величину $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$, функция принадлежности которой может быть представлена в форме композиции некоторой L-функции и некоторой R-функции в следующем виде:

$$\mu_{\mathcal{B}}(x) = \begin{cases} L\left(\dfrac{a-x}{\alpha}\right), & \text{если } x \leq a; \\ R\left(\dfrac{x-a}{\beta}\right), & \text{если } x \geq a, \end{cases} \tag{5.11}$$

где $\alpha>0$ и $\beta>0$. При этом параметр a является *модой* или модальным значением нечеткого числа, а параметры α и β являются *левым и правым коэффициентами нечеткости* соответственно. Как видно из этого определения, при задании нечетких чисел (L-R)-типа могут использоваться, вообще говоря, две различные функции указанного вида, что существенно расширяет диапазон их возможных представлений.

Из данного определения следует, что нечеткое число (L-R)-типа с функцией принадлежности $\mu_{\mathcal{B}}(x)$ при фиксированных L и R функциях вполне определяется тройкой своих параметров $<a, \alpha, \beta>$, что оказывается весьма удобным для выполнения операций с подобными числами. Чтобы отметить тот факт, что нечет-

кое число является (L-R)-типа, будем его обозначать специальным образом: $\mathcal{B}_{LR}=<a, \alpha, \beta>_{LR}$. Расширением понятия нечеткого числа (L-R)-типа является понятие нечеткого интервала (L-R)-типа.

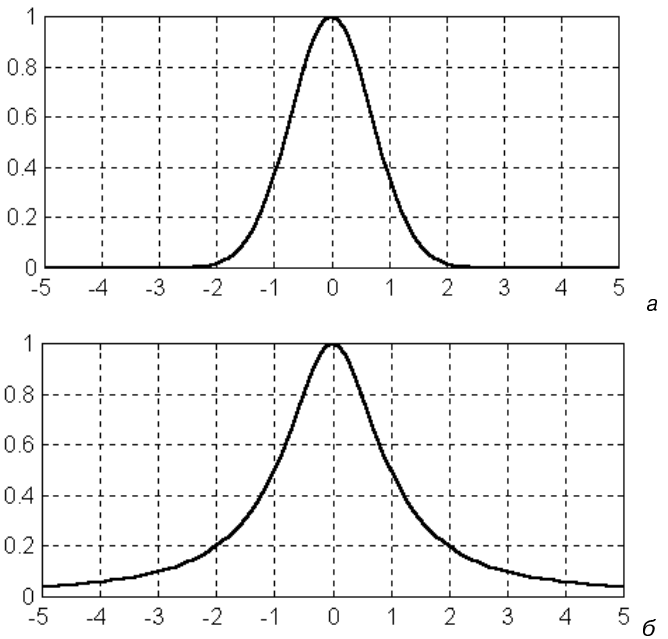


Рис. 5.3. Графики L-функций и R-функций, заданных формулами (5.9) (*а*) и (5.10) (*б*) соответственно, для значения параметра p=2

Нечеткий интервал (L-R)-типа. *Нечетким интервалом* (L-R)-*типа* будем называть нечеткую величину $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$, функция принадлежности которой может быть представлена в форме композиции некоторой L-функции и некоторой R-функции в следующем виде:

$$\mu_{\mathcal{B}}(x)=\begin{cases} L\left(\dfrac{a-x}{\alpha}\right), & \text{если } x \le a; \\ 1, & \text{если } a < x < b; \\ R\left(\dfrac{x-b}{\beta}\right), & \text{если } x \ge b, \end{cases} \qquad (5.12)$$

где $\alpha>0$ и $\beta>0$. При этом параметры a и b определяют ядро нечеткого интервала ($a < b$) и называются соответственно *нижним* и *верхним модальными значениями* нечеткого интервала. Параметры α и β по-прежнему называются *левым и правым коэффициентами нечеткости* соответственно. Следует отметить, что нечеткий интервал (L-R)-типа часто называют *толерантным* нечетким числом (L-R)-типа.

Функция принадлежности $\mu_{\mathcal{B}}(x)$ нечеткого интервала (L-R)-типа при фиксированных L и R функциях вполне определяется четверкой своих параметров $<a, b, \alpha, \beta>$, что оказывается весьма удобным для выполнения операций с подобными интервалами. Чтобы отметить тот факт, что нечеткий интервал является (L-R)-типа, будем его обозначать специальным образом: $\mathcal{B}_{LR} = <a, b, \alpha, \beta>_{LR}$.

Из определений (5.11) и (5.12) видно, что при задании нечетких чисел и интервалов (L-R)-типа могут использоваться две различные функции указанного вида. При этом в случае равенства параметров $a=b$ нечеткий интервал (L-R)-типа превращается в нечеткое число (L-R)-типа.

В качестве примеров введенных в рассмотрение понятий можно привести конкретное нечеткое число (L-R)-типа $\mathcal{B}_{LR} = <2, 1, 2>_{LR}$ и нечеткий интервал (L-R)-типа $\mathcal{B}_{LR} = <1, 3, 2, 1>_{LR}$, где в качестве функции L-типа использована функция (5.9) со значением параметра $p=2$, а в качестве функции R-типа использована функция (5.9) со значением параметра $p=3$. Графическое изображение этих нечеткого числа и нечеткого интервала (L-R)-типов представлено на рис. 5.4.

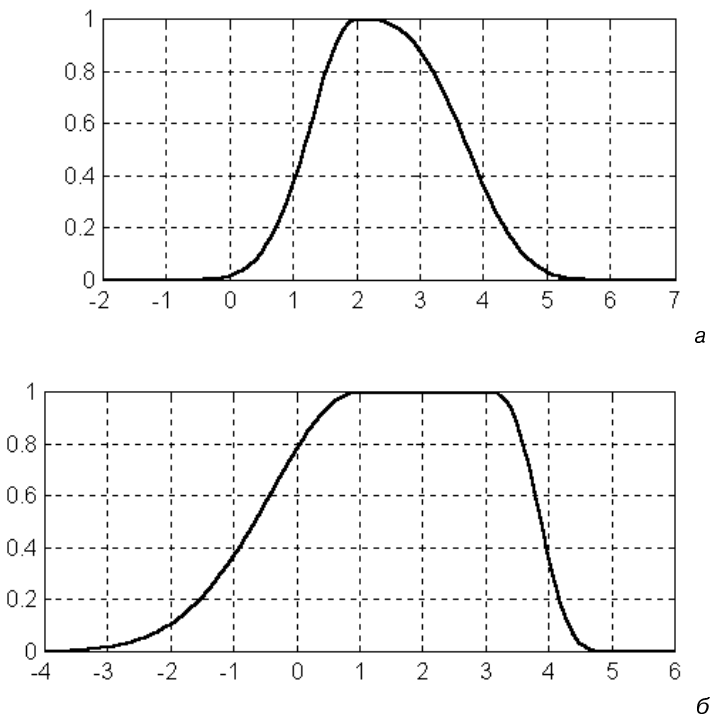


Рис. 5.4. Графики нечеткого числа (L-R)-типа $\mathcal{B}_{LR} = <2, 1, 2>_{LR}$ (*а*) и нечеткого интервала (L-R)-типа $\mathcal{B}_{LR} = <1, 3, 2, 1>_{LR}$ (*б*)

Операции над нечеткими числами и интервалами (*L-R*)-типа

При определении операций над нечеткими числами и интервалами (*L-R*)-типа следует исходить из следующих соображений. Результат арифметических операций сложения, вычитания, деления и умножения должен быть точно или приблизительно равен некоторому нечеткому числу или интервалу с теми же функциями *L*-типа и *R*-типа, а параметры α и β результата должны некоторым однозначным образом зависеть от аналогичных параметров исходных нечетких чисел и интервалов (*L-R*)-типа.

С этой целью для определения аналогов обычных арифметических операций над нечеткими числами и нечеткими интервалами (*L-R*)-типа целесообразно использовать принцип обобщения (4.23). Замечательным свойством определенных таким способом арифметических операций (сложение, вычитание, умножение и деление) является то, что они определяются на основе значений соответствующих параметров их (*L-R*)-представлений.

Пусть $\mathcal{A}_{LR}$ и $\mathcal{B}_{LR}$ — произвольные нечеткие числа (*L-R*)-типа, заданные параметрически в виде: $\mathcal{A}_{LR} = <a_1, \alpha_1, \beta_1>_{LR}$ и $\mathcal{B}_{LR} = <a_2, \alpha_2, \beta_2>_{LR}$.

С л о ж е н и е. Операция *сложения* нечетких чисел (*L-R*)-типа обозначается через $\mathcal{A}_{LR}+\mathcal{B}_{LR} = C_{LR} = <a, \alpha, \beta>_{LR}$, где параметры a, α и β результата определяются следующим образом:

$$a = a_1+a_2, \qquad \alpha=\alpha_1+\alpha_2, \qquad \beta=\beta_1+\beta_2. \tag{5.13}$$

В ы ч и т а н и е. Операция *вычитания* нечетких чисел (*L-R*)-типа обозначается через $\mathcal{A}_{LR} - \mathcal{B}_{LR} = C_{LR} = <a, \alpha, \beta>_{LR}$, где параметры a, α и β результата определяются следующим образом:

$$a = a_1 - a_2, \qquad \alpha=\alpha_1+\beta_2, \qquad \beta=\beta_1+\alpha_2. \tag{5.14}$$

Операции умножения и деления нечетких чисел (*L-R*)-типа могут быть определены при выполнении некоторых дополнительных условий.

У м н о ж е н и е положительных нечетких чисел (*L-R*)-типа $\mathcal{A}_{LR}$ и $\mathcal{B}_{LR}$, т. е. носители которых являются подмножествами $\boldsymbol{R}_+$, а модальные значения $a_1>0$ и $a_2>0$. Операция *умножения* таких нечетких чисел (*L-R*)-типа обозначается через $\mathcal{A}_{LR}\bullet\mathcal{B}_{LR} = C_{LR} = <a, \alpha, \beta>_{LR}$, где параметры a, α и β результата определяются следующим образом:

$$a = a_1a_2, \qquad \alpha= a_1\alpha_2+a_2\alpha_1, \qquad \beta= a_1\beta_2+ a_2\beta_1. \tag{5.15}$$

У м н о ж е н и е нечетких чисел (*L-R*)-типа $\mathcal{A}_{LR}$ и $\mathcal{B}_{LR}$, для которых модальные значения разных знаков: $a_1<0$ и $a_2>0$. Операция *умножения* таких нечетких чисел (*L-R*)-типа также обозначается через $\mathcal{A}_{LR}\bullet\mathcal{B}_{LR} = C_{LR} = <a, \alpha, \beta>_{LR}$, где параметры a, α и β результата определяются следующим образом:

$$a = a_1a_2, \qquad \alpha= a_2\alpha_1 - a_1\beta_2, \qquad \beta= a_2\beta_1 - a_1\alpha_2. \tag{5.16}$$

У м н о ж е н и е нечетких чисел (*L-R*)-типа $\mathcal{A}_{LR}$ и $\mathcal{B}_{LR}$, для которых модальные значения отрицательные: $a_1<0$ и $a_2<0$. Операция *умножения* таких нечетких чисел

(L-R)-типа также обозначается через $\mathcal{A}_{LR} \bullet \mathcal{B}_{LR} = C_{LR} = <a, \alpha, \beta>_{LR}$, где параметры a, α и β результата определяются следующим образом:

$$a = a_1 a_2, \qquad \alpha = -a_2\beta_1 - a_1\beta_2, \qquad \beta = -a_2\alpha_1 - a_1\alpha_2. \tag{5.17}$$

Деление положительных нечетких чисел (L-R)-типа $\mathcal{A}_{LR}$ и $\mathcal{B}_{LR}$, т. е. носители которых являются подмножествами $\mathbb{R}_+$, а модальные значения $a_1>0$ и $a_2>0$. Операция *деления* таких нечетких чисел (L-R)-типа обозначается через $\mathcal{A}_{LR} \div \mathcal{B}_{LR} = C_{LR} = <a, \alpha, \beta>_{LR}$, где параметры a, α и β результата определяются следующим образом:

$$a = a_1/a_2, \qquad \alpha = (a_1\beta_2 + a_2\alpha_1)/a_2^2, \qquad \beta = (a_1\alpha_2 + a_2\beta_1)/a_2^2. \tag{5.18}$$

Обратное нечеткое число для положительного нечеткого числа (L-R)-типа $\mathcal{A}_{LR}$, т. е. носитель которого является подмножеством $\mathbb{R}_+$, а модальное значение $a_1>0$. В этом случае *обратное нечеткое число* обозначается через $C_{LR}^{-1} = <a, \alpha, \beta>_{LR}$, параметры которого a, α и β определяются следующим образом:

$$a = 1/a_1, \qquad \alpha = \beta_1/a_1^2, \qquad \beta = a_1/a_1^2. \tag{5.19}$$

В качестве примера выполнения операций с нечеткими числами (L-R)-типа рассмотрим два конкретных нечетких числа: "*нечеткая тройка*" и "*нечеткая двойка*". Для удобства предположим, что эти нечеткие числа заданы с использованием одинаковой L-функции и R-функции, в качестве которой возьмем уже известную нам функцию (5.9) со значением параметра p=2. Конкретные значения функций принадлежности этих нечетких чисел изображены на рис. 5.5, при этом значения параметров нечетких чисел следующие: a_1=3, $\alpha_1=\beta_1$=2, a_2=2, $\alpha_2=\beta_2$=1.

Результаты выполнения операций сложения и вычитания этих нечетких чисел с использованием формул (5.13) и (5.14) изображены на рис. 5.6, *а* и рис. 5.6, *б*. Эти результаты можно назвать "*нечеткая пятерка*" и "*нечеткая единица*" соответственно. Результаты выполнения операций умножения и деления этих нечетких чисел с использованием формул (5.15) и (5.18) изображены на рис. 5.6, *в* и рис. 5.6, *г*. Эти результаты можно назвать "*нечеткая шестерка*" и "*нечеткая дробь 3/2*" соответственно. При этом следует заметить, что для удобства изображены лишь фрагменты графиков результирующих функций принадлежности вблизи их модальных значений.

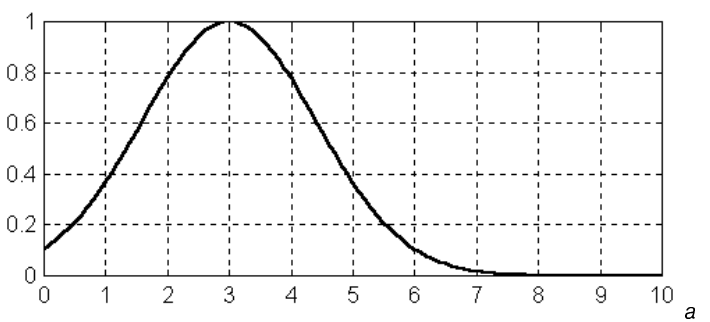


Рис. 5.5. Графики двух нечетких чисел (L-R)-типа: "*нечеткая тройка*" (*а*)

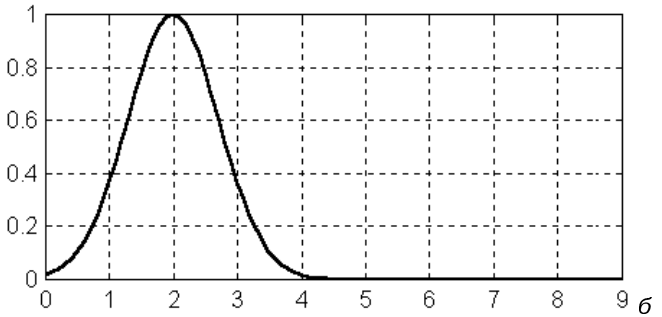


Рис. 5.5. Графики двух нечетких чисел (*L-R*)-типа: "*нечеткая двойка*" (*б*)

Аналогичным образом можно определить операции над нечеткими интервалами (*L-R*)-типа, а также операции расширенного максимума и расширенного минимума нечетких чисел и интервалов (*L-R*)-типа. Поскольку подобные операции характеризуются повышенной сложностью соответствующих численных расчетов, они не нашли широкого применения в практике нечеткого моделирования.

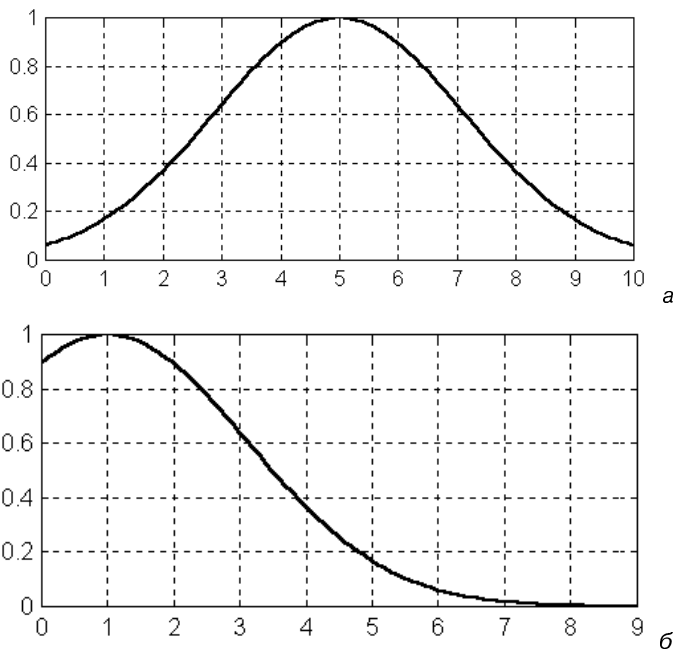


Рис. 5.6. Графики нечетких чисел (*L-R*)-типа: "*нечеткая пятерка*" (*а*) и "*нечеткая единица*" (*б*), которые являются результатами выполнения операций сложения, вычитания, умножения и деления нечетких чисел (*L-R*)-типа: "*нечеткая тройка*" и "*нечеткая двойка*" соответственно

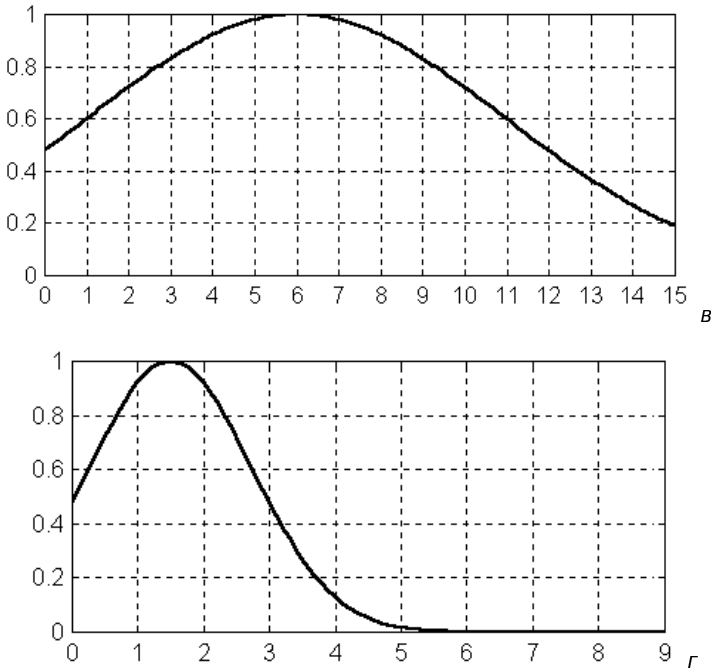


Рис. 5.6. Графики нечетких чисел (*L*-*R*)-типа: "*нечеткая шестерка*" (*в*) и "*нечеткая дробь 3/2*" (*г*), которые являются результатами выполнения операций сложения, вычитания, умножения и деления нечетких чисел (*L*-*R*)-типа: "*нечеткая тройка*" и "*нечеткая двойка*" соответственно

Наибольший интерес с практической точки зрения представляют аналоги арифметических операций, определенные для треугольных нечетких чисел и трапециевидных нечетких интервалов, которые отличаются наглядностью и простотой интерпретации получаемых результатов. Эти понятия являются темой следующего раздела.

5.4. Треугольные нечеткие числа и трапециевидные нечеткие интервалы

При решении практических задач нечеткого моделирования наибольшее применение нашли простейшие частные случаи нечетких чисел и интервалов, получившие свое название по виду их функций принадлежности. Эти нечеткие числа и интервалы можно рассматривать как частный случай нечетких чисел и интервалов (*L*-*R*)-типа, если в качестве соответствующих функций *L*-типа и *R*-типа использовать их предельные случаи, а именно — линейные функции. При этом целесообразность использования трапециевидных нечетких интервалов и тре-

угольных нечетких чисел обусловливается не только простотой выполнения операций над ними, но и их наглядной графической интерпретацией.

Треугольное нечеткое число. *Треугольным нечетким числом* (сокращенно — ТНЧ) будем называть такое нормальное нечеткое число, функция принадлежности которого может быть задана треугольной функцией f_Δ.

В этом случае ТНЧ удобно представить в виде кортежа из трех чисел: $\mathcal{A}_\Delta = <a, \alpha, \beta>_\Delta$, где a — *модальное значение* ТНЧ; α и β — *левый и правый коэффициенты нечеткости* ТНЧ. Поскольку, как было отмечено в *главе 2*, каждая треугольная функция принадлежности порождает нормальное унимодальное выпуклое нечеткое множество с непустым носителем — открытым интервалом $(a-\alpha, a+\beta)$, то ТНЧ является частным случаем нечеткого числа $(L\text{-}R)$-типа.

Примечание

Напомним, что треугольная функция принадлежности f_Δ характеризуется тремя параметрами и в общем случае с использованием выражения (2.8) может быть записана в виде $f_\Delta(x; a, b, c)$. При этом параметры ТНЧ $\mathcal{A}_\Delta = <a, \alpha, \beta>_\Delta$ однозначным образом связаны с параметрами треугольной функции принадлежности $f_\Delta(x; a, b, c)$. А именно, модальное значение ТНЧ тождественно равно параметру b функции принадлежности $f_\Delta(x; a, b, c)$, т. е. $a=b$, а левый и правый коэффициенты нечеткости ТНЧ соответственно равны: $\alpha=b-a$, $\beta=c-b$.

Пример конкретного ТНЧ $<3, 1, 2>_\Delta$, которое соответствует "*нечеткой тройке*", изображен на рис. 5.7, *а*. Очевидно, примерами ТНЧ также могут служить нечеткие множества, функции принадлежности которых изображены на рис. 2.7, *а*, 2.11, *а*, а также на рис. 5.1, *б*.

Трапециевидный нечеткий интервал. *Трапециевидным нечетким интервалом* (сокращенно — ТНИ) будем называть нормальный нечеткий интервал, функция принадлежности которого может быть задана трапециевидной функцией $f_\text{Т}$.

В этом случае ТНИ удобно представить в виде кортежа из четырех чисел: $\mathcal{A}_\text{Т} = <a, b, \alpha, \beta>_\text{Т}$, где a и b — соответственно *нижнее и верхнее модальные значения* ТНИ; α и β — *левый и правый коэффициенты нечеткости* ТНИ. Поскольку каждая трапециевидная функция принадлежности порождает нормальное выпуклое нечеткое множество с непустым носителем — открытым интервалом $(a-\alpha, b+\beta)$, то ТНИ является частным случаем нечеткого интервала $(L\text{-}R)$-типа. Как нетрудно заметить, треугольное нечеткое число $\mathcal{A}_\Delta$ является частным случаем трапециевидного нечеткого интервала $\mathcal{A}_\text{Т} = <a, b, \alpha, \beta>_\text{Т}$ при $a = b$.

Примечание

Трапециевидная функция принадлежности $f_\text{Т}$ характеризуется четырьмя параметрами и в общем случае с использованием выражения (2.9) может быть записана в виде $f_\text{Т}(x; a, b, c, d)$. При этом параметры ТНИ $\mathcal{A}_\text{Т} = <a, b, \alpha, \beta>_\text{Т}$ однозначным образом связаны с параметрами трапециевидной функции принадлежности $f_\text{Т}(x; a, b, c, d)$. А именно, нижнее модальное значение ТНИ то-

ждественно равно параметру b функции принадлежности $f_T(x; a, b, c, d)$, верхнее модальное значение ТНИ тождественно равно параметру c функции принадлежности $f_T(x; a, b, c, d)$, т. е. $b=c$, а левый и правый коэффициенты нечеткости ТНЧ соответственно равны: $\alpha=b-a$, $\beta=d-c$.

Пример конкретного ТНИ $<4, 6, 2, 1>_T$, которое соответствует "*нечеткому интервалу от 4 до 6*", изображен на рис. 5.7, *б*. Примерами ТНИ могут служить также нечеткие множества, функции принадлежности которых изображены на рис. 2.11, *б*, 2.15, 5.1, *а*, *в*.

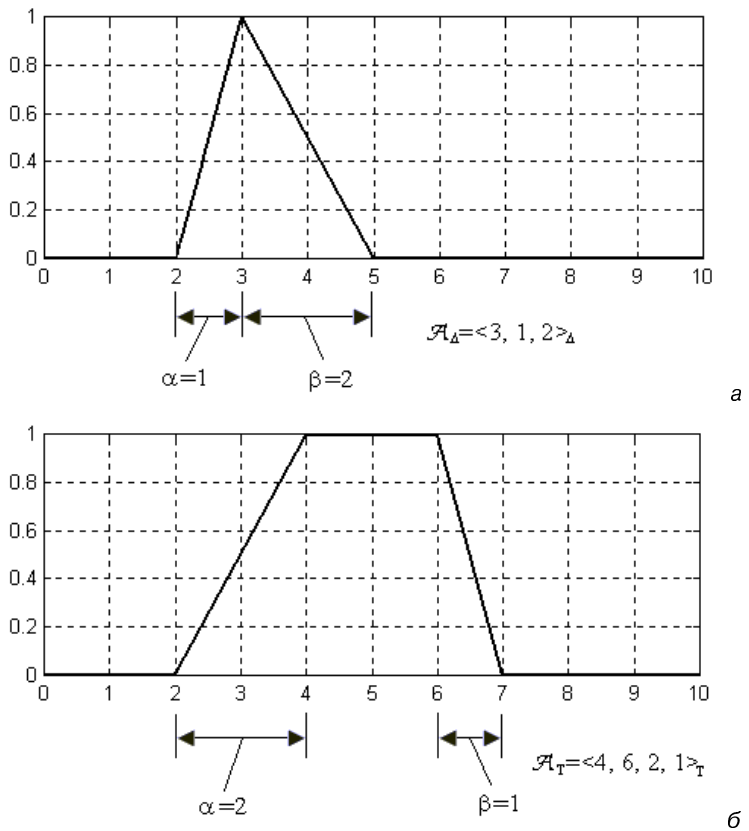


Рис. 5.7. Графическое представление ТНЧ $\mathcal{A}_\Delta=<3, 1, 2>_\Delta$ (*а*) и ТНИ $\mathcal{A}_T=<4, 6, 2, 1>_T$ (*б*)

Примечание

Иногда могут оказаться полезными и другие альтернативные определения ТНЧ и ТНИ, в которых не требуется нормальность соответствующих нечетких мно-

жеств. Так, например, некоторые авторы *треугольным нечетким числом* называют нечеткую величину, которую с использованием двух линейных функций можно представить в виде кортежа из четырех чисел: $\mathcal{V}_\Delta$= <*a*, α, β, *h*>, где *a* — модальное значение нечеткого числа; α и β — левый и правый коэффициенты нечеткости; *h* — высота нечеткого числа. Соответственно, *трапециевидным нечетким интервалом* называют нечеткую величину, которую с использованием двух линейных функций можно представить в виде кортежа из пяти чисел: $\mathcal{V}_\mathrm{T}$= <*a, b,* α, β, *h*>, где *a* и *b* — соответственно нижнее и верхнее модальные значения нечеткого интервала; α и β — левый и правый коэффициенты нечеткости; *h* — высота нечеткого интервала. В этом случае треугольное нечеткое число $\mathcal{V}_\Delta$ также является частным случаем трапециевидного нечеткого интервала $\mathcal{V}_\mathrm{T}$ при *a* = *b*.

Как не трудно заметить, введенные ранее определения ТНЧ и ТНИ становятся частными случаями этих более общих понятий треугольного нечеткого числа и трапециевидного нечеткого интервала. А именно, ТНЧ есть не что иное как *нормальное* треугольное нечеткое число $\mathcal{V}_\Delta^0$ = <*a*, α, β, 1>, а ТНИ есть *нормальный* трапециевидный нечеткий интервал $\mathcal{V}_\mathrm{T}^0$ = <*a*, *b*, α, β, 1>. Рассмотренные понятия можно проиллюстрировать графически, где на рис. 5.8, *а* изображено треугольное нечеткое число $\mathcal{V}_\Delta$= <3, 2, 2, 0.8>, а на рис. 5.8, *б* изображен трапециевидный нечеткий интервал $\mathcal{V}_\mathrm{T}$= <3, 6, 1, 3, 0.6>. В дальнейшем наше рассмотрение будет ограничено только нормальными ТНЧ $\mathcal{A}_\Delta$ и нормальными ТНИ $\mathcal{A}_\mathrm{T}$.

Для решения задач нечеткого моделирования необходимо определить некоторые простейшие операции над ТНИ и ТНЧ, аналогичные обычным арифметическим операциям над обычными числами и интервалами.

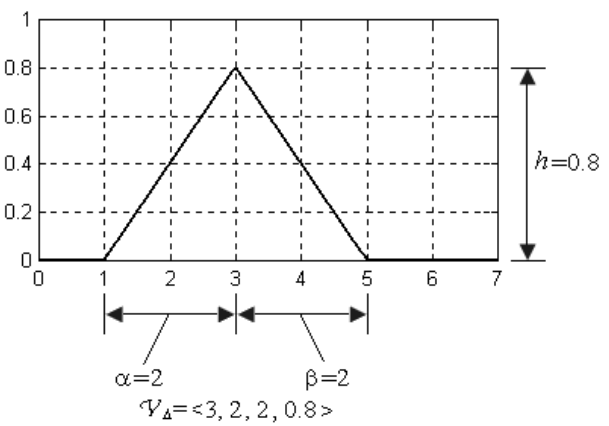


Рис. 5.8. Графическое представление треугольного нечеткого числа $\mathcal{V}_\Delta$= <3, 2, 2, 0.8>, не являющегося нормальным (*а*)

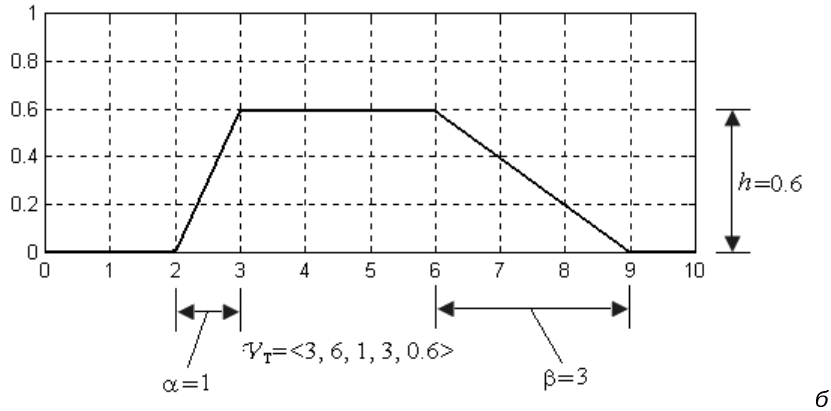


Рис. 5.8. Графическое представление трапециевидного нечеткого интервала $\mathcal{V}_T$= <3, 6, 1, 3, 0.6>, не являющегося нормальным (*б*)

Операции над треугольными нечеткими числами и трапециевидными нечеткими интервалами

Пусть $\mathcal{A}_\Delta$ и $\mathcal{B}_\Delta$ — два произвольных треугольных нечетких числа, которые заданы параметрически в виде: $\mathcal{A}_\Delta = \langle a_1, \alpha_1, \beta_1 \rangle_\Delta$ и $\mathcal{B}_\Delta = \langle a_2, \alpha_2, \beta_2 \rangle_\Delta$. Для этих ТНЧ оказываются справедливыми аналоги обычных арифметических операций, введенных в рассмотрение выше в *разд. 5.3* для треугольных нечетких чисел (*L-R*)-типа. А именно, операция сложения ТНЧ определяется выражением (5.13), операция вычитания ТНЧ — выражением (5.14), операция умножения ТНЧ — выражениями (5.15)—(5.17), операция деления — выражением (5.18), и, наконец, обратное ТНЧ — выражением (5.19).

Например, для конкретных ТНЧ $\mathcal{A}_\Delta = \langle 3, 1, 2 \rangle_\Delta$ и $\mathcal{B}_\Delta = \langle 2, 2, 1 \rangle_\Delta$ результаты арифметических операций равны: $\mathcal{A}_\Delta + \mathcal{B}_\Delta = \langle 5, 3, 3 \rangle_\Delta$, $\mathcal{A}_\Delta - \mathcal{B}_\Delta = \langle 1, 2, 4 \rangle_\Delta$, $\mathcal{A}_\Delta \bullet \mathcal{B}_\Delta = \langle 6, 8, 7 \rangle_\Delta$, $\mathcal{A}_\Delta \div \mathcal{B}_\Delta = \langle 1.5, 1.25, 2.5 \rangle_\Delta$. Графики результатов операций с этими ТНЧ изображены на рис. 5.9 (*а—г*) соответственно.

Перейдем к рассмотрению операций с ТНИ. Пусть $\mathcal{A}$ и $\mathcal{B}$ — два произвольных трапециевидных нечетких интервала, которые заданы параметрически в виде: $\mathcal{A}_T = \langle a_1, b_1, \alpha_1, \beta_1 \rangle_T$ и $\mathcal{B}_T = \langle a_2, b_2, \alpha_2, \beta_2 \rangle_T$.

С л о ж е н и е. Операция *сложения* ТНИ обозначается через $\mathcal{A}_T + \mathcal{B}_T = C_T = \langle a, b, \alpha, \beta \rangle_T$, где параметры a, b, α и β результата определяются следующим образом:

$$a = a_1 + a_2, \qquad b = b_1 + b_2, \qquad \alpha = \alpha_1 + \alpha_2, \qquad \beta = \beta_1 + \beta_2. \tag{5.20}$$

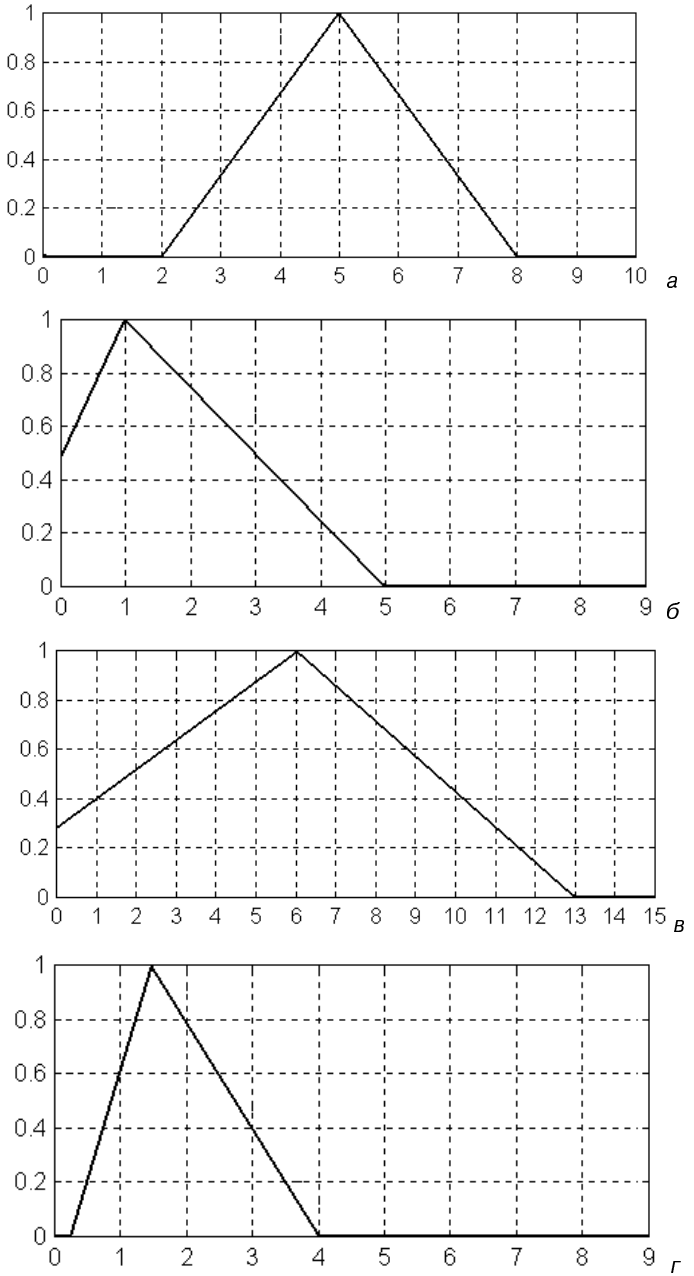


Рис. 5.9. Графики ТНЧ: "*нечеткая пятерка*" (*а*) и "*нечеткая единица*" (*б*), "*нечеткая шестерка*" (*в*) и "*нечеткая дробь 3/2*" (*г*), которые являются результатами выполнения операций сложения, вычитания, умножения и деления ТНЧ: "*нечеткая тройка*" и "*нечеткая двойка*" соответственно

В ы ч и т а н и е. Операция *вычитания* ТНИ обозначается через $\mathcal{A}_T - \mathcal{B}_T = \mathcal{C}_T = <a, b, \alpha, \beta>_T$, где параметры a, b, α и β результата определяются следующим образом:

$$a = a_1 - a_2, \quad b = b_1 - b_2, \quad \alpha = \alpha_1 + \beta_2, \quad \beta = \beta_1 + \alpha_2. \tag{5.21}$$

У м н о ж е н и е положительных ТНИ $\mathcal{A}_T$ и $\mathcal{B}_T$, т. е. носители которых являются подмножествами $\mathbb{R}_+$, а все модальные значения положительные. Операция *умножения* таких ТНИ обозначается через $\mathcal{A}_T \bullet \mathcal{B}_T = \mathcal{C}_T = <a, b, \alpha, \beta>_T$, где параметры a, b, α и β результата определяются следующим образом:

$$a = a_1 a_2, \quad b = b_1 b_2, \quad \alpha = a_1\alpha_2 + a_2\alpha_1, \quad \beta = b_1\beta_2 + b_2\beta_1. \tag{5.22}$$

Д е л е н и е положительных ТНИ $\mathcal{A}_T$ и $\mathcal{B}_T$, т. е. носители которых являются подмножествами $\mathbb{R}_+$, а все модальные значения положительные. Операция *деления* таких ТНИ обозначается через $\mathcal{A}_T \div \mathcal{B}_T = \mathcal{C}_T = <a, b, \alpha, \beta>_T$, где параметры a, b, α и β результата определяются следующим образом:

$$a = a_1/b_2, \quad a = b_1/a_2, \quad \alpha = (a_1\beta_2 + b_2\alpha_1)/b_2^2, \quad \beta = (b_1\alpha_2 + a_2\beta_1)/a_2^2. \tag{5.23}$$

Например, рассмотрим два конкретных ТНИ: $\mathcal{A}_T = <3, 5, 1, 2>_T$ и $\mathcal{B}_T = <1, 2, 1, 1>_T$. Первый из них соответствует "*нечеткому интервалу от трех до пяти*", а второй — "*нечеткому интервалу от единицы до двух*". Тогда результат их сложения с использованием формул (5.20) равен ТНИ $\mathcal{A}_T + \mathcal{B}_T = <4, 7, 2, 3>_T$ и соответствует "*нечеткому интервалу от четырех до семи*". Результат вычитания из первого ТНИ второго ТНИ с использованием формул (5.21) равен ТНИ: $\mathcal{A}_T - \mathcal{B}_T = <2, 3, 2, 3>_T$ и соответствует "*нечеткому интервалу от двух до трех*". Графики соответствующих ТНИ представлены на рис. 5.10.

Примечание

Что касается операций с альтернативными определениями ТНЧ и ТНИ, которые не являются нормальными, то предложенные в этом случае способы определения результатов соответствующих операций уже не являются столь очевидными и наглядными. В частности, для двух трапециевидных нечетких интервалов (не являющихся нормальными): $\mathcal{V}_T = <a_1, b_1, \alpha_1, \beta_1, h_1>$, $\mathcal{W}_T = <a_2, b_2, \alpha_2, \beta_2, h_2>$ параметры результата их сложения $\mathcal{U}_T = \mathcal{V}_T + \mathcal{W}_T = <a, b, \alpha, \beta, h>$ рассчитываются последовательно по следующим формулам:

$$h = min(h_1, h_1); \tag{5.24}$$

$$\alpha = h \cdot (\alpha_1/h_1 + \alpha_2/h_2), \quad \beta = h \cdot (\beta_1/h_1 + \beta_2/h_2); \tag{5.25}$$

$$a = a_1 + a_2 - \alpha_1 - \alpha_2 + \alpha, \quad b = b_1 + b_2 + \beta_1 + \beta_2 - \beta. \tag{5.26}$$

Данная операция сложения может быть проиллюстрирована графически (рис. 5.11), где на рис. 5.11, *а* изображен трапециевидный нечеткий интервал: $\mathcal{V}_T = <3, 5, 1, 2, 0.8>$, на рис. 5.11, *б* изображен трапециевидный нечеткий интервал: $\mathcal{W}_T = <1, 2, 1, 1, 0.6>$, а на рис. 5.11, *в* изображен результат их сложения $\mathcal{U}_T = <3, 6, 1, 3, 0.6>$, полученный с использованием формул (5.24)—(5.26).

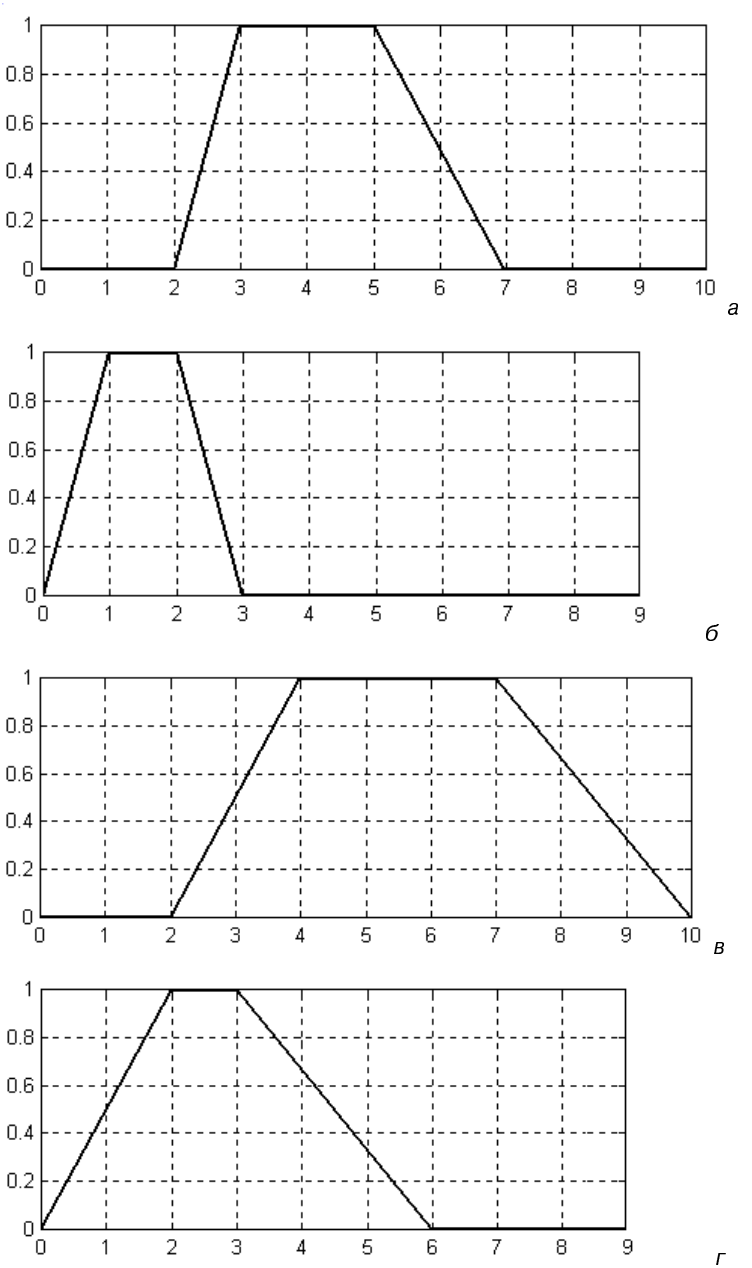


Рис. 5.10. Графики ТНИ: "*нечеткий интервал от трех до пяти*" (*а*), "*нечеткий интервал от единицы до двух*" (*б*), а также результат их сложения (*в*) и вычитания (*г*)

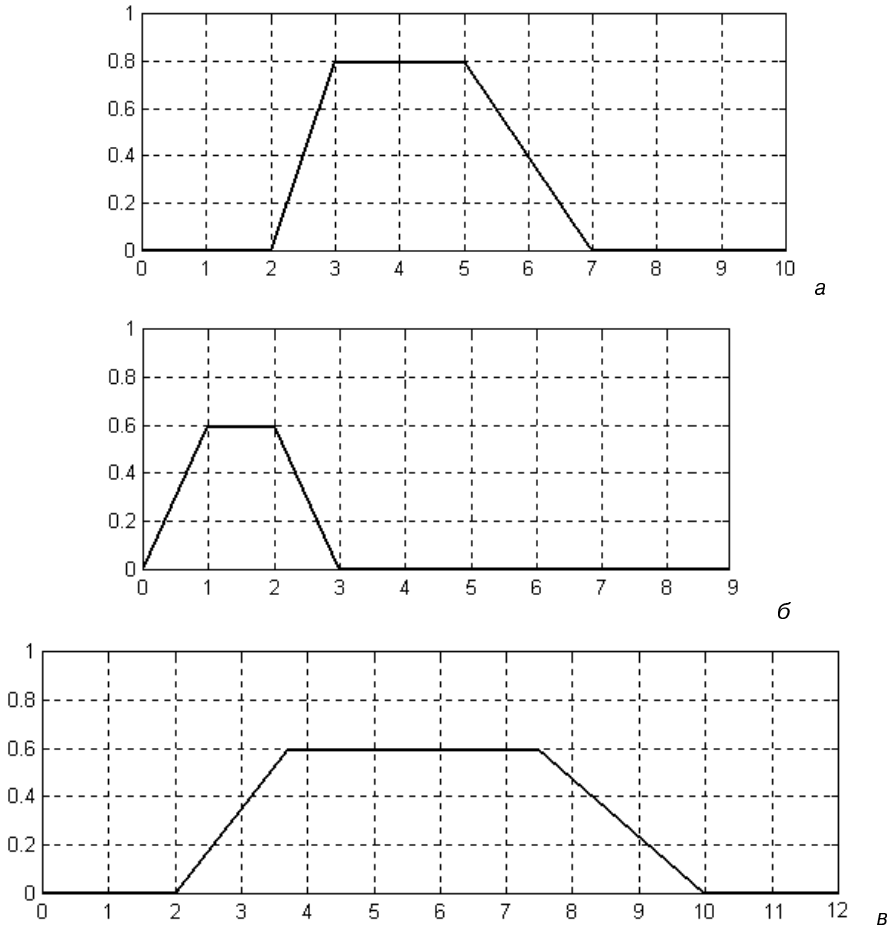


Рис. 5.11. Графики двух трапециевидных нечетких интервалов, не являющихся нормальными (*а*, *б*), и результата их сложения (*в*)

Для нас представляют интерес операции расширенного максимума и расширенного минимума ТНИ $\mathcal{A}_T = <a_1, b_1, \alpha_1, \beta_1>_T$ и $\mathcal{B}_T = <a_2, b_2, \alpha_2, \beta_2>_T$ (соответственно — ТНЧ), которые определяются следующим образом.

Р а с ш и р е н н ы й м а к с и м у м. Операция *расширенного максимума* ТНИ обозначается через $\max\{\mathcal{A}_T, \mathcal{B}_T\} = C_T = <a, b, \alpha, \beta>_T$, где параметры a, b, α и β результата определяются следующим образом:

$$a = \max\{a_1, a_2\}, \qquad b = \max\{b_1, b_2\},$$
$$\alpha = a - \max\{a_1 - \alpha_1, a_2 - \alpha_2\}, \qquad \beta = \max\{b_1 + \beta_1, b_2 + \beta_2\} - b. \tag{5.27}$$

Р а с ш и р е н н ы й м и н и м у м. Операция *расширенного минимума* ТНИ обозначается через $\min\{\mathcal{A}_T, \mathcal{B}_T\} = C_T = <a, b, \alpha, \beta>_T$, где параметры a, b, α и β результата определяются следующим образом:

$$a = \min\{a_1, a_2\}, \qquad b = \min\{b_1, b_2\},$$
$$\alpha = a - \min\{a_1 - \alpha_1, a_2 - \alpha_2\}, \qquad \beta = \min\{b_1 + \beta_1, b_2 + \beta_2\} - b. \tag{5.28}$$

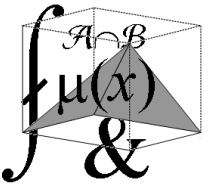
В заключение этой главы приведем некоторые рекомендации, которые целесообразно использовать в процессе представления различных термов тех или иных лингвистических переменных нечеткими числами и интервалами (L-R)-типа, а также их более простыми частными случаями — ТНЧ и ТНИ (табл. 5.1).

Таблица 5.1. *Рекомендации по представлению термов лингвистических переменных*

Терм лингвистической переменной	**(L-R)-представление**	**Представление в форме ТНЧ и ТНИ**
Средний, около, приблизительно	$<a, \alpha, \beta>_{LR}$, где $\alpha<\infty$, $\beta<\infty$	$<a, \alpha, \beta>_{\Delta}$, где $\alpha<\infty$, $\beta<\infty$
Малый, низкий	$<a, \infty, \beta>_{LR}$, где $\alpha=\infty$, $\beta<\infty$	$<a, \infty, \beta>_{\Delta}$, где $\alpha=\infty$, $\beta<\infty$
Большой, высокий	$<a, \alpha, \infty>_{LR}$, где $\alpha<\infty$, $\beta=\infty$	$<a, \alpha, \infty>_{\Delta}$, где $\alpha<\infty$, $\beta=\infty$
Приблизительно в диапазоне, в интервале (a, b), где $a<\infty$ и $b<\infty$	$<a, b, \alpha, \beta>_{LR}$, где $\alpha<\infty$, $\beta<\infty$	$<a, b, \alpha, \beta>_T$, где $\alpha<\infty$, $\beta<\infty$
Точно в диапазоне, в интервале $[a, b]$, где $a<\infty$ и $b<\infty$		$<a, b, \alpha, \beta>_T$, где $\alpha=0$, $\beta=0$
Точно равен числу a, где $a<\infty$		$<a, \alpha, \beta>_{\Delta}$, где $\alpha=0$, $\beta=0$
Не превышает значения b, где $b<\infty$	$<a, b, \alpha, \beta>_{LR}$, где $a=\infty$, $\alpha=\infty$, $\beta<\infty$	$<a, b, \alpha, \beta>_T$, где $a=\infty$, $\alpha=\infty$, $\beta<\infty$
Не меньше, чем значение a, где $a<\infty$	$<a, b, \alpha, \beta>_{LR}$, где $b=\infty$, $\alpha<\infty$, $\beta=\infty$	$<a, b, \alpha, \beta>_T$, где $b=\infty$, $\alpha<\infty$, $\beta=\infty$

Подводя итог этой главы, заметим, что при решении практических задач нечеткого моделирования наиболее удобными оказываются ТНЧ и ТНИ. Именно они наиболее часто используются для представления входных и выходных переменных нечетких моделей систем управления, о которых пойдет речь в последующих главах.

Глава 6



Основы нечеткой логики

Нечеткая логика предназначена для формализации человеческих способностей к неточным или приближенным рассуждениям, которые позволяют более адекватно описывать ситуации с неопределенностью. Классическая логика по своей сути игнорирует проблему неопределенности, поскольку все высказывания и рассуждения в формальных логических системах могут иметь только значение "*истина*" (**И**, 1) или значение "*ложь*" (**Л**, 0). В отличие от этого в нечеткой логике истинность рассуждений оценивается в некоторой степени, которая может принимать и другие отличные {**И**, **Л**} значения.

Чтобы иметь возможность выражать неопределенные знания, необходима такая логическая система, которая позволяет некоторому предложению иметь истинностное значение, отличающееся от бинарного **И** или **Л**. Один из подходов — расширить множество истинностных значений {**И**, **Л**} и позволить предложениям принимать некоторые дополнительные значения истинности. Одним из первых логиков, предложивших в 1930 г. вариант многозначной логической системы, отличающийся от классической бинарной логики, был польский математик Ян Лукасевич (1878—1956). В трехзначной логике Лукасевича используется 3 истинностных значения: {0, 0.5, 1}, где значение 0 интерпретируется как "*ложь*", 1 — как "*истина*", а число 0.5 — как "*возможно*". В качестве высказываний с истинностным значением "*возможно*" могут выступать такие, которые относятся к некоторому моменту времени в будущем.

Так, например, высказывание "*Сборная России по футболу выйдет в 1/8 финала на предстоящем Чемпионате мира*" до начала Чемпионата не может быть оценено ни как истинное, ни как ложное. Именно по этой причине более адекватным ответом на вопрос об его истинности будет использование трехзначной логики с соответствующей интерпретацией истинности в форме значения "*возможно*".

Наряду с понятием нечеткого множества, Л. Заде предложил обобщение классической логики на основе рассмотрения бесконечного множества значений истинности. Далее в этой главе изложены основы нечеткой логики, которая использует основные понятия теории нечетких множеств для формализации неточных знаний и выполнения приближенных рассуждений в той или иной проблемной области.

6.1. Понятие нечеткого высказывания и нечеткого предиката

В предложенном Л. Заде варианте нечеткой логики множество истинностных значений высказываний обобщается до интервала действительных значений [0, 1], что позволяет высказыванию принимать любое значение истинности из этого интервала. Это численное значение является количественной оценкой степени истинности высказывания, относительно которого нельзя с полной уверенностью заключить о его истинности или ложности. Использование в качестве множества истинностных значений интервала [0, 1] позволяет построить логическую систему, в рамках которой оказалось возможным выполнять рассуждения с неопределенностью и оценивать истинность высказываний типа: "*Скорость автомобиля довольно высокая*", "*Давление в системе весьма значительное*", "*Высота полета самолета предельно низкая*" и др.

Исходным понятием нечеткой логики является понятие элементарного нечеткого высказывания.

Элементарное нечеткое высказывание. В общем случае *элементарным нечетким высказыванием* называется повествовательное предложение, выражающее законченную мысль, относительно которой мы можем судить об ее истинности или ложности только с некоторой степенью уверенности.

Элементарные нечеткие высказывания для удобства будем обозначать теми же буквами, что и нечеткие множества: $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$ (возможно, с индексами). Сами элементарные нечеткие высказывания иногда называют просто нечеткими высказываниями.

Главным отличием элементарного нечеткого высказывания от элементарного высказывания математической логики является следующий факт. Множество значений истинности элементарных высказываний классической математической логики состоит из двух элементов: {"*истина*", "*ложь*"} ({**И**, **Л**} или {0, 1}), при этом значению "*истина*" соответствует цифра 1 или буква **И**, а значению "*ложь*" — цифра 0 или буква **Л** *(см. приложение 2)*. В нечеткой логике *степень истинности* элементарного нечеткого высказывания принимает значение из замкнутого интервала [0, 1], причем 0 и 1 являются предельными значениями степени истинности и совпадают со значениями "*ложь*" и "*истина*" соответственно.

Примечание

Следует обратить внимание на то обстоятельство, что использование интервала [0, 1] в качестве множества значений истинности нечетких высказываний естественным образом порождает бинарное отношение нестрогого порядка на декартовом произведении произвольного множества нечетких высказываний. Хотя измерение степени истинности нечеткого высказывания выполняется в шкале интервалов, допустимые преобразования этой шкалы являются избыточными и могут оказаться неадекватными содержательным аспектам той или иной практической задачи. Это следует помнить при построении нечетких мо-

делей реальных систем. Детальный анализ семантических особенностей измерения степени истинности нечетких высказываний возможен на основе рассмотрения теории нечеткой меры, основы которой излагаются далее в *главе 9*.

Пример 6.1. Ниже приводится несколько примеров элементарных нечетких высказываний:

1. *О. Бендер имеет довольно высокий рост.*
2. *Завтра будет пасмурная погода.*
3. *3 — малое число.*
4. *ВАЗ 2110 является скоростным автомобилем.*
5. *Возможно, нам подадут горячий кофе.*

Содержательно неопределенность нечетких высказываний может иметь различную природу. Так, например, неопределенность оценки истинности в высказывании (1) связана с нечеткостью определения понятия "*высокий рост*", которое является нечеткой переменной *(см. главу 5)*. Аналогичный характер неопределенности имеют нечеткие высказывания (3) и (4), связанные с определением нечетких переменных "*малое число*" и "*скоростной автомобиль*" соответственно. Что касается высказываний (2) и (5), то здесь кроме определения нечетких переменных "*пасмурная погода*" и "*горячий кофе*" следует оценить их истинность относительного некоторого момента времени в будущем. Общим для всех этих высказываний является то обстоятельство, что относительно их истинности мы можем судить лишь с некоторой степенью, количественно оцениваемой действительным числом из интервала [0, 1].

Примечание

Конечно, наиболее строгие из читателей могут дополнить содержательный анализ истинности рассмотренных выше высказываний. В частности, можно заметить, что точное измерение роста О. Бендера как литературного персонажа вряд ли возможно и в свою очередь может быть основано лишь на приближенных визуальных оценках тех или иных экранизаций или субъективном анализе контекста соответствующего художественного произведения. Поскольку эти тонкие семантические нюансы лежат за пределами собственно нечеткой логики, мы оставляем их рассмотрение за указанной категорией читателей.

Для оценки степени истинности произвольного нечеткого высказывания удобно ввести в рассмотрение специальное отображение T, которое действует из множества рассматриваемых нечетких высказываний $\mathcal{U}$ в интервал [0, 1], т. е. $T : \mathcal{U} \rightarrow [0, 1]$. Это отображение будем называть *отображением истинности* нечетких высказываний. В этом случае значение истинности некоторого нечеткого высказывания $\mathcal{A} \in \mathcal{U}$ будем обозначать через $T(\mathcal{A})$. Так, если обозначить нечеткое высказывание (1) из примера 6.1 через $\mathcal{A}_1$, то его истинность формально может быть записана как $T(\mathcal{A}_1)$, а количественно равна, например, 0.7, т. е. $T(\mathcal{A}_1) = 0.7$.

Нечеткие предикаты

Как было отмечено ранее в *главе 5* при описании способов задания нечетких отношений, в общем случае можно ввести в рассмотрение некоторое характеристическое свойство, которое может быть записано в виде *многоместного нечеткого предиката* $\mathcal{P}(<x_1, x_2,\ldots, x_k>)$.

Нечеткий предикат. *Нечеткий предикат* $\mathcal{P}(<x_1, x_2,..., x_k>)$ или, более строго, *k-местный нечеткий предикат*, формально определяется как некоторое отображение из декартова произведения универсумов $\boldsymbol{X}_1, \boldsymbol{X}_2,\ldots, \boldsymbol{X}_k$ в некоторое вполне упорядоченное множество значений истинности, в частности, в интервал [0, 1], т. е. $\mathcal{P} : \boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k \rightarrow [0, 1]$. По аналогии с обычными предикатами, переменные $x_1, x_2,..., x_k$ называются *предметными* переменными нечеткого предиката $\mathcal{P}(<x_1, x_2,..., x_k>)$, а декартово произведение универсумов $\boldsymbol{X}_1\times\boldsymbol{X}_2\times\ldots\times\boldsymbol{X}_k$ — его *предметной областью*.

Примечание

Следует заметить, что это определение с точностью до обозначений соответствует определению нечеткого отношения $Q=\{<x_1, x_2,\ldots, x_k>| \mu_Q(<x_1, x_2,..., x_k>), x_1\in\boldsymbol{X}_1, x_2\in\boldsymbol{X}_2,\ldots, x_k\in\boldsymbol{X}_k\}$ (*см. главу* 4). Этот факт подчеркивает формальную и содержательную взаимосвязь функции принадлежности произвольного нечеткого отношения $\mu_Q(<x_1, x_2,\ldots, x_k>)\}$ с соответствующим нечетким предикатом $\mathcal{P}_Q(<x_1, x_2,..., x_k>)$, который, возможно неявно, выражает некоторое характеристическое свойство этого нечеткого отношения.

Таким образом, анализ семантических особенностей нечетких предикатов $\mathcal{P}(<x_1, x_2,..., x_k>)$ в общем случае тесно взаимосвязан с анализом соответствующих нечетких отношений $Q=\{<x_1, x_2,\ldots, x_k>| \mu_Q(<x_1, x_2,..., x_k>), x_1\in\boldsymbol{X}_1, x_2\in\boldsymbol{X}_2,\ldots, x_k\in\boldsymbol{X}_k\}$, которые были рассмотрены ранее в *главе 4*. Верно и обратное заключение. Эта связь функции принадлежности нечеткого отношения и соответствующего ему нечеткого предиката играет важную роль при установлении взаимосвязей между теорией нечетких множеств и нечеткой логикой, поскольку изучение нечетких предикатов может быть выполнено с помощью анализа рассмотренных выше свойств нечетких отношений.

В свою очередь взаимосвязь между нечеткими высказываниями и нечеткими предикатами устанавливается с помощью процесса так называемого *означивания* нечеткого предиката $\mathcal{P}(<x_1, x_2,..., x_k>)$, под которым понимается подстановка вместо предикатных переменных $x_1, x_2,\ldots, x_k$ конкретных значений из соответствующих универсумов: $a_1\in\boldsymbol{X}_1, a_2\in\boldsymbol{X}_2,\ldots, a_k\in\boldsymbol{X}_k$. В этом случае нечеткий предикат $\mathcal{P}(<x_1, x_2,..., x_k>)$ превращается в некоторое нечеткое высказывание $\mathcal{P}$, которое принимает конкретное значение истинности, равное числу из интервала [0, 1].

Нечеткое обобщение логики предикатов первого порядка, так же как и соответствующие ей нечеткие исчисления, не нашли широкого применения при решении прикладных задач. Наиболее конструктивным направлением в нечеткой логике оказалось нечеткое обобщение правил продукций, использующих нечеткие вы-

сказывания в форме означивания лингвистических переменных. В этом случае нечеткие высказывания могут комбинироваться с помощью нечетких логических операций или связок, которые и рассматриваются ниже.

6.2. Основные логические операции с нечеткими высказываниями

Пусть $\mathcal{U}$ — некоторое множество элементарных нечетких высказываний, а $T : \mathcal{U} \to [0, 1]$ — отображение истинности высказываний.

Логическое отрицание нечетких высказываний

Логическое отрицание. *Отрицанием* нечеткого высказывания $\mathcal{A}$ (записывается как: $\neg\mathcal{A}$ и читается — "*не* $\mathcal{A}$", "*неверно, что* $\mathcal{A}$") называется унарная логическая операция, результат которой является нечетким высказыванием, истинность которого по определению принимает значение:

$$T(\neg\mathcal{A}) = 1 - T(\mathcal{A}). \qquad (6.1)$$

Очевидно, что принятый в математической логике способ определять логические операции с помощью таблиц истинности не может быть использован в нечеткой логике. Причина этого заключается в континуальной мощности множества истинностных значений [0, 1]. Именно поэтому центральную роль в определении логических операций в нечеткой логике приобретает отображение истинности T, которое имеет вспомогательное значение в классической математической логике *(см. приложение 2)*.

Примером применения операции логического отрицания к нечеткому высказыванию "*О. Бендер имеет довольно высокий рост*" будет высказывание "*Неверно, что О. Бендер имеет довольно высокий рост*", степень истинности которого с учетом ранее определенного значения для $T(\mathcal{A}_1)$ равна 0.3. Отрицанием высказывания "*Завтра будет пасмурная погода*" будет высказывание "*Завтра будет не пасмурная погода*", степень истинности которого, если принять $T(\mathcal{A}_2) = 0.2$, принимает значение 0.8.

Примечание

Говоря об логических операциях с нечеткими высказываниями, нельзя не сказать о наличии большого числа альтернативных способов их определения. Особенно это относится к нечеткой импликации, которая занимает центральное место в системах нечеткого вывода.

Логическая конъюнкция нечетких высказываний

Логическая конъюнкция. *Конъюнкцией* нечетких высказываний $\mathcal{A}$ и $\mathcal{B}$ (записывается как: $\mathcal{A}\wedge\mathcal{B}$ и читается — "$\mathcal{A}$ и $\mathcal{B}$") называется бинарная логическая операция, результат которой является нечетким высказыванием, истинность которого определяется по формуле:

$$T(\mathcal{A}\wedge\mathcal{B}) = \min\{T(\mathcal{A}), T(\mathcal{B})\}. \quad (6.2)$$

Логическую конъюнкцию нечетких высказываний также называют *нечетким логическим* "И", *нечеткой конъюнкцией* или min-конъюнкцией и иногда записывают также в форме $\mathcal{A}$ AND $\mathcal{B}$. При этом исторически принято считать формулу (6.2) *основной* для определения степени истинности ее результата.

По аналогии с операциями над нечеткими множествами, рассмотренными в *главе 3*, для определения степени истинности конъюнкции нечетких высказываний могут быть использованы следующие альтернативные формулы.

- *Алгебраическое произведение* степеней истинности нечетких высказываний:

$$T(\mathcal{A}\wedge\mathcal{B}) = T(\mathcal{A})\cdot T(\mathcal{B}). \quad (6.3)$$

- *Граничное произведение* степеней истинности нечетких высказываний:

$$T(\mathcal{A}\wedge\mathcal{B}) = \max\{T(\mathcal{A})+T(\mathcal{B})-1, 0\}. \quad (6.4)$$

- *Драстическое произведение* степеней истинности нечетких высказываний:

$$T(\mathcal{A}\wedge\mathcal{B}) = \begin{cases} T(\mathcal{B}), & \text{если } T(\mathcal{A}) = 1; \\ T(\mathcal{A}), & \text{если } T(\mathcal{B}) = 1; \\ 0, & \text{в остальных случаях.} \end{cases} \quad (6.5)$$

Для этих альтернативных способов определения истинности логической конъюнкции нечетких высказываний могут быть использованы обозначения, аналогичные соответствующим обозначениям операций над нечеткими множествами. Так нечеткая конъюнкция, рассчитываемая по формуле (или методом алгебраического произведения, обозначается через $\mathcal{A}\bullet\mathcal{B}$, методом граничного произведения — через $\mathcal{A}\odot\mathcal{B}$, методом драстического произведения — через $\mathcal{A}\Delta\mathcal{B}$.

Пример 6.2. Рассмотрим составное нечеткое высказывание, состоящее из двух элементарных: "*О. Бендер имеет довольно высокий рост и завтра будет пасмурная погода*" и предположим, что истинность первого из них равна $T(\mathcal{A}_1) = 0.7$, а истинность второго — $T(\mathcal{A}_2) = 0.2$. Тогда истинность логической конъюнкции этих нечетких высказываний, вычисленная по основной формуле (6.2), равна: $T(\mathcal{A}_1\wedge\mathcal{A}_2) = 0.2$. Значения истинности этой же конъюнкции, рассчитанные по остальным формулам, равны: $T(\mathcal{A}_1\bullet\mathcal{A}_2) = 0.14$, $T(\mathcal{A}_1\odot\mathcal{A}_2) = 0$, $T(\mathcal{A}_1\Delta\mathcal{A}_2) = 0$.

Логическая дизъюнкция нечетких высказываний

Логическая дизъюнкция. *Дизъюнкцией* нечетких высказываний A и B (записывается как: $A \vee B$ и читается — "A или B") называется бинарная логическая операция, результат которой является нечетким высказыванием, истинность которого по определению принимает значение:

$$T(A \vee B) = \max\{T(A), T(B)\}. \qquad (6.6)$$

Логическую также называют логическим *неисключающим* "ИЛИ".

Логическую дизъюнкцию нечетких высказываний также называют *нечетким неисключающим логическим* "ИЛИ", *нечеткой дизъюнкцией* или max-дизъюнкцией и иногда записывают также в форме A OR B. При этом исторически принято считать формулу (6.6) *основной* для определения степени истинности ее результата.

По аналогии с операциями над нечеткими множествами, рассмотренными в главе 3, для определения степени истинности дизъюнкции нечетких высказываний могут быть использованы следующие альтернативные формулы.

- *Алгебраическая сумма* степеней истинности нечетких высказываний:

$$T(A \vee B) = T(A) + T(B) - T(A) \cdot T(B) \qquad (6.7)$$

- *Граничная сумма* степеней истинности нечетких высказываний:

$$T(A \vee B) = \min\{T(A) + T(B), 1\} \qquad (6.8)$$

- *Драстическая сумма* степеней истинности нечетких высказываний:

$$T(A \vee B) = \begin{cases} T(B), & \text{если } T(A) = 0; \\ T(A), & \text{если } T(B) = 0; \\ 1, & \text{в остальных случаях.} \end{cases} \qquad (6.9)$$

Для этих альтернативных способов определения истинности логической дизъюнкции нечетких высказываний могут быть использованы обозначения, аналогичные соответствующим обозначениям операций над нечеткими множествами. Так, нечеткая дизъюнкция, рассчитываемая по формуле (6.7) или методом алгебраической суммы, обозначается через $A+B$, методом граничной суммы — через $A \oplus B$, методом драстической суммы — через $A \nabla B$. При этом алгебраическую сумму (6.7) часто называют также *вероятностной суммой*.

Пример 6.3. Как и выше, рассмотрим составное нечеткое высказывание: "*О. Бендер имеет довольно высокий рост или завтра будет пасмурная погода*" и предположим, что истинность входящих в него элементарных нечетких высказываний по-прежнему равна $T(A_1) = 0.7$ и $T(A_2) = 0.2$. Тогда истинность логической дизъюнкции этих нечетких высказываний, вычисленная по основной формуле (6.6), равна: $T(A_1 \vee A_2) = 0.7$. Значения истинности этой же дизъюнкции, рассчитанные по остальным формулам, равны: $T(A_1 + A_2) = 0.76$, $T(A_1 \oplus A_2) = 0.9$, $T(A_1 \nabla A_2) = 1$.

Примечание

В общем случае для определения истинности результатов нечеткой конъюнкции и нечеткой дизъюнкции могут быть предложены и другие расчетные формулы, основанные на рассмотрении треугольных норм (3.37)—(3.40) и конорм (3.41)—(3.44). Подобное рассмотрение базируется на содержательной взаимосвязи между операциями над нечеткими множествами и логическими операциями с нечеткими высказываниями. Как будет видно далее, эта взаимосвязь становится еще более очевидной в случае нечетких лингвистических высказываний.

Нечеткая импликация

Нечеткая импликация. *Нечеткой импликацией* или просто — *импликацией* нечетких высказываний $\mathcal{A}$ и $\mathcal{B}$ (записывается как: $\mathcal{A}\supset\mathcal{B}$ и читается — "из $\mathcal{A}$ следует $\mathcal{B}$", "ЕСЛИ $\mathcal{A}$, ТО $\mathcal{B}$") называется бинарная логическая операция, результат которой является нечетким высказыванием, истинность которого может принимать значение, определяемое по одной из следующих формул.

- Классическая нечеткая импликация, предложенная Л. Заде:

$$T(\mathcal{A}\supset\mathcal{B}) = \max\{\min\{T(\mathcal{A}), T(\mathcal{B})\}, 1-T(\mathcal{A})\}. \quad (6.10)$$

  Эту форму нечеткой импликации называют также *нечеткой импликацией Заде.*

- Классическая нечеткая импликация для случая $T(\mathcal{A})\geq T(\mathcal{B})$:

$$T(\mathcal{A}\supset\mathcal{B}) = \max\{T(\neg\mathcal{A}), T(\mathcal{B})\} = \max\{1-T(\mathcal{A}), T(\mathcal{B})\}. \quad (6.11)$$

  Эту форму нечеткой импликации иногда называют *нечеткой импликацией Гёделя.*

- Нечеткая импликация, предложенная Э. Мамдани:

$$T(\mathcal{A}\supset\mathcal{B}) = \min\{T(\mathcal{A}), T(\mathcal{B})\}. \quad (6.12)$$

  Эту форму нечеткой импликации также называют *нечеткой импликацией Мамдани* или нечеткой импликацией *минимума корреляции.* Можно заметить, что в случае $T(\mathcal{A})\geq 0.5$ и $T(\mathcal{B})\geq 0.5$ классическая нечеткая импликация превращается в нечеткую импликацию Мамдани.

- Нечеткая импликация, предложенная Я. Лукасевичем:

$$T(\mathcal{A}\supset\mathcal{B}) = \min\{1, 1-T(\mathcal{A})+T(\mathcal{B})\}. \quad (6.13)$$

  Эту форму нечеткой импликации также называют *нечеткой импликацией Лукасевича.*

- Нечеткая импликация, предложенная Дж. Гогеном:

$$T(\mathcal{A}\supset\mathcal{B}) = \min\{1, T(\mathcal{B})/T(\mathcal{A})\}, \quad \text{где } T(\mathcal{A})>0. \quad (6.14)$$

  Эту форму нечеткой импликации также называют *нечеткой импликацией Гогена.*

- Нечеткая импликация по формуле граничной суммы:

$$T(\mathcal{A}\supset\mathcal{B}) = \min\{1, T(\mathcal{A})+T(\mathcal{B})\}. \quad (6.15)$$

- Нечеткая импликация по формуле произведения:

$$T(\mathcal{A}\supset\mathcal{B}) = T(\mathcal{A})\cdot T(\mathcal{B})\}. \qquad (6.16)$$

- Нечеткая импликация, предложенная Н. Вади:

$$T(\mathcal{A}\supset\mathcal{B}) = max\{T(\mathcal{A})\cdot T(\mathcal{B}),\ 1-T(\mathcal{A})\}. \qquad (6.17)$$

- Нечеткая импликация Брауэра:

$$T(\mathcal{A}\supset\mathcal{B}) = \begin{cases} 1, & \text{если} \quad T(\mathcal{A}) \le T(\mathcal{B}); \\ T(B), & \text{в противном случае.} \end{cases} \qquad (6.18)$$

Эта форма нечеткой импликации получила свое название в честь голландского логика Л. Брауэра (1881—1966), основоположника математического интуиционизма.

- Нечеткая импликация стандартной логики последовательностей (R-SEQ):

$$T(\mathcal{A}\supset\mathcal{B}) = \begin{cases} 1, & \text{если} \quad T(\mathcal{A}) \le T(\mathcal{B}); \\ 0, & \text{в противном случае.} \end{cases} \qquad (6.19)$$

Нечеткая импликация играет важную роль в процессе нечетких логических рассуждений. Так же, как и в математической логике первый ее операнд (нечеткое высказывание) называется *посылкой* или *антецедентом*, а второй — *заключением* или *консеквентом*.

Хотя классическая нечеткая импликация находит наибольшее применение при решении прикладных задач и она остается справедливой в случае обычных высказываний классической логики, остальные способы вычисления нечеткой импликации в отдельных ситуациях оказываются более эффективными с вычислительной точки зрения.

Примечание

Рассмотренные выше формулы для определения истинности результата нечеткой импликации можно считать основными и далеко не исчерпывающими все предложенные для этой цели способы. По некоторым оценкам число таких способов приближается к 100. Одно из наиболее известных исследований различных определений нечеткой импликации было выполнено японскими математиками Мидзумото, Танака и Фуками, а также в работах Кицки, Кочански и Сливински. Интерес к изучению различных вариантов нечеткой импликации основан на том обстоятельстве, что нечеткая импликация занимает центральное место в той или иной системе нечеткого вывода в качестве метода активации или композиции. Назначение и особенности различных этапов нечеткого вывода рассматриваются в следующей главе.

Пример 6.4. Рассмотрим составное нечеткое высказывание в форме нечеткой импликации: "*Если О. Бендер имеет довольно высокий рост, то завтра будет пасмурная погода*", при этом истинность входящих в него элементарных нечетких высказываний по-прежнему равна $T(\mathcal{A}_1) = 0.7$ и $T(\mathcal{A}_2) = 0.2$. Тогда истинность этой нечеткой импликации, вычисленная по основной формуле (6.10), равна $T(\mathcal{A}_1\supset\mathcal{A}_2) = 0.3$, по формуле (6.12) — $T(\mathcal{A}_1\supset\mathcal{A}_2) = 0.2$, по формуле (6.13) —

$T(A_1 \supset A_2) = 0.5$, по формуле (6.14) — $T(A_1 \supset A_2) \approx 0.29$, по формуле (6.15) — $T(A_1 \supset A_2) = 0.9$, по формуле (6.16) — $T(A_1 \supset A_2) = 0.14$, по формуле (6.17) — $T(A_1 \supset A_2) = 0.3$, по формуле (6.18) — $T(A_1 \supset A_2) = 0.2$, по формуле (6.19) — $T(A_1 \supset A_2) = 0$.

Нечеткая эквивалентность

Нечеткая эквивалентность. *Эквивалентностью* нечетких высказываний A и B или просто нечеткой эквивалентностью (записывается как: $A \equiv B$ и читается — "A эквивалентно B") называется бинарная логическая операция, результат которой является нечетким высказыванием, истинность которого определяется по следующей формуле:

$$T(A \equiv B) = \min\{\max\{T(\neg A), T(B)\}, \max\{T(A), T(\neg B)\}\}. \quad (6.20)$$

Примером логической эквивалентности может служить составное нечеткое высказывание: "*О. Бендер имеет довольно высокий рост эквивалентно тому, что завтра будет пасмурная погода*", истинность которого принимает значение 0.3.

Так же, как в классической математической логике, в нечеткой логике с помощью рассмотренных логических связок могут быть образованы достаточно сложные нечеткие высказывания. При этом для явного указания порядка их следования используются круглые скобки, а иногда — и приоритет соответствующих нечетких логических операций.

Так, например, составное высказывание: $(A_1 \supset A_2) \supset (\neg A_2 \supset \neg A_1)$ в случае $T(A_1) = 0.7$ и $T(A_2) = 0.2$ имеет значение истинности, равное: $\max\{1-\max\{1-T(A_1), T(A_2)\}\}, \max\{1-T(\neg A_2), T(\neg A_1)\}\} = \max\{1-\max\{1-0.7, 0.2\}\}, \max\{1-0.8, 0.3\}\} = \max\{0.7, 0.3\} = 0.7$.

Следует отметить, что обязательным условием корректности определения истинности составных нечетких высказываний является требование одновременной подстановки вместо одинаковых букв одних и тех же нечетких высказываний. Кроме рассмотренных логических операций могут быть определены и другие бинарные логические операции с нечеткими высказываниями. Более детально познакомиться с ними можно в специальной литературе.

6.3. Правила нечетких продукций

Продукционные системы были разработаны в рамках исследований по методам искусственного интеллекта и нашли широкое применение для представления знаний и вывода заключений в экспертных системах, основанных на правилах. Поскольку нечеткий вывод реализуется на основе нечетких продукционных правил, рассмотрение базового формализма нечетких продукционных моделей приобретает самостоятельное значение. При этом нечеткие правила продукций не только во многом близки к логическим моделям, но и, что наиболее важно, по-

зволяют адекватно представить практические знания экспертов в той или иной проблемной области.

Правило нечеткой продукции. В общем случае под *правилом нечеткой продукции* или просто — *нечеткой продукцией* понимается выражение следующего вида:

$$(i) : Q; P; \mathcal{A} \Rightarrow \mathcal{B}; S, F, N, \qquad (6.21)$$

где (i) — имя нечеткой продукции; Q — сфера применения нечеткой продукции; P — условие применимости ядра нечеткой продукции; $\mathcal{A}\Rightarrow\mathcal{B}$ — ядро нечеткой продукции, в котором $\mathcal{A}$ — условие ядра (или антецедент); $\mathcal{B}$ — заключение ядра (или консеквент); "$\Rightarrow$" — знак логической секвенции (или следования); S — метод или способ определения количественного значения степени истинности заключения ядра; F — коэффициент определенности или уверенности нечеткой продукции; N — постусловия продукции.

По аналогии с обычным правилом продукции, в качестве имени (i) нечеткой продукции может выступать та или иная совокупность букв или символов, позволяющая однозначным образом идентифицировать нечеткую продукцию в системе нечеткого вывода или базе нечетких правил. В качестве имени нечеткой продукции может использоваться ее номер в системе.

Сфера применения нечеткой продукции Q, условие применимости ядра нечеткой продукции P и постусловие нечеткой продукции N определяются аналогично обычной не нечеткой продукции *(см. приложение 2)*.

Аналогично обычным правилам продукций ядро $\mathcal{A}\Rightarrow\mathcal{B}$ также является центральным компонентом нечеткой продукции. Ядро продукции записывается в более привычной форме: "ЕСЛИ $\mathcal{A}$, ТО $\mathcal{B}$" или в наиболее распространенном виде: "IF $\mathcal{A}$, THEN $\mathcal{B}$", где $\mathcal{A}$ и $\mathcal{B}$ — некоторые выражения нечеткой логики, которые наиболее часто представляются в форме нечетких высказываний. При этом секвенция интерпретируется в обычном логическом смысле как знак логического следования заключения $\mathcal{B}$ из условия $\mathcal{A}$. В качестве выражений $\mathcal{A}$ и $\mathcal{B}$ могут использоваться составные логические нечеткие высказывания, т. е. элементарные нечеткие высказывания, соединенные нечеткими логические связками, такими как нечеткое отрицание, нечеткая конъюнкция и нечеткая дизъюнкция.

S — метод или способ определения количественного значения степени истинности заключения $\mathcal{B}$ на основе известного значения степени истинности условия $\mathcal{A}$. Данный способ в общем случае определяет так называемую *схему* или алгоритм нечеткого вывода в продукционных нечетких системах и называется также *методом композиции* или *методом активации* согласно Стандарту IEC 1131-7 *(см. главу 8)*. В настоящее время для этой цели предложено несколько способов, основные из которых рассматриваются ниже в настоящем разделе.

F — коэффициент определенности или уверенности выражает количественную оценку степени истинности или относительный вес нечеткой продукции. Коэффициент уверенности принимает свое значение из интервала [0, 1] и часто называется весовым коэффициентом нечеткого правила продукции.

Примечание

Наряду с обычной формой ядра нечеткого правила продукции рассматривается так называемая расширенная форма в виде: "ЕСЛИ $\mathcal{A}$, ТО $\mathcal{B}$, ИНАЧЕ $\mathcal{C}$" или в эквивалентной записи: "IF $\mathcal{A}$ THEN $\mathcal{B}$ ELSE $\mathcal{C}$". С точки зрения нечеткой логики эта форма нечеткого правила продукции по определению эквивалентна двум нечетким правилам продукции: "ЕСЛИ $\mathcal{A}$, ТО $\mathcal{B}$" и "ЕСЛИ НЕ $\mathcal{A}$, ТО $\mathcal{C}$", где связка "НЕ" используется в смысле операции нечеткого отрицания.

Продукционная нечеткая система. *Продукционная нечеткая система* или *система нечетких правил продукций* представляет собой некоторое согласованное множество отдельных *нечетких продукций* или правил нечетких продукций в форме "ЕСЛИ $\mathcal{A}$, ТО $\mathcal{B}$" (или в виде: "IF $\mathcal{A}$ THEN $\mathcal{B}$", как определено в Стандарте IEC 1131-7). Далее обе эти формы записи будут использоваться как эквивалентные в зависимости от удобства в том или ином контексте.

Основная проблема приближенных рассуждений с использованием нечетких правил продукций заключается в том, чтобы на основе некоторых нечетких высказываний с известной степенью истинности, которые являются условиями нечетких правил продукций, оценить степень истинности других нечетких высказываний, являющимися заключениями соответствующих нечетких правил продукций.

Чтобы иметь возможность решить эту проблему, необходимо ответить на более частный вопрос: Чему должна быть равна степень истинности заключения отдельного нечеткого правила продукции, если известна степень истинности условия этого правила? Таким образом, в системах нечетких продукций центральное место занимает способ или метод определения истинности заключений в нечетком правиле продукции.

Нетрудно заметить, что взаимосвязь между условием и заключением в нечетком правиле продукции в общем случае представляет собой некоторое бинарное нечеткое отношение на декартовом произведении универсумов соответствующих нечетких высказываний. Этот подход и будет использоваться в дальнейшем для определения различных схем или методов нечеткого вывода на основе продукционных нечетких систем.

В общем случае для формального определения различных методов нечеткого вывода применительно к нечеткому правилу продукции рассмотрим два нечетких множества $\mathcal{A}$ и $\mathcal{B}$, заданных соответственно на универсумах $\boldsymbol{X}$ и $\boldsymbol{Y}$. При этом нечеткое множество $\mathcal{A}$ интерпретируется как условие некоторого нечеткого правила продукции, а нечеткое множество $\mathcal{B}$ — как заключение этого же правила.

Основная идея заключается в том, что нечеткое множество $\mathcal{A}$ можно рассматривать как унарное отношение на универсуме $\boldsymbol{X}$, а нечеткое множество $\mathcal{B}$ можно рассматривать как унарное отношение на универсуме $\boldsymbol{Y}$. В этом случае первое отношение определяется функцией принадлежности $\mu_{\mathcal{A}}(x)$, а второе отношение — функцией принадлежности $\mu_{\mathcal{B}}(y)$.

Теперь предположим, что некоторым образом определено бинарное нечеткое отношение на декартовом произведении универсумов: $Q=\{<x, y>, \mu_Q(<x, y>)\}$,

где $x \in \boldsymbol{X}$ и $y \in \boldsymbol{Y}$. Если дополнительно известна функция принадлежности $\mu_{\mathcal{A}}(x)$ первого множества, то функция принадлежности $\mu_{\mathcal{B}}(y)$ второго множества может быть определена в результате нечеткой композиции соответствующих нечетких отношений с использованием, например, формулы (4.17) для максиминной нечеткой композиции.

Кроме максиминной нечеткой композиции, рассмотренной в *главе 4*, предложены и другие способы для определения результата композиции нечетких отношений. Таким образом, для определения функции принадлежности нечеткого множества $\mathcal{B}$ можно использовать следующие методы, основанные на различных расчетных формулах для определения функции принадлежности результата.

- Max-min-композиция или максиминная нечеткая свертка:

$$\mu_{\mathcal{B}}(y) = \max_{x \in X} \{\min\{\mu_{\mathcal{A}}(x), \mu_{Q}(<x, y>)\}\}. \tag{6.22}$$

- Max-prod-композиция:

$$\mu_{\mathcal{B}}(y) = \max_{x \in X} \{\mu_{\mathcal{A}}(x) \cdot \mu_{Q}(<x, y>)\}. \tag{6.23}$$

- Min-max-композиция:

$$\mu_{\mathcal{B}}(y) = \min_{x \in X} \{\max\{\mu_{\mathcal{A}}(x), \mu_{Q}(<x, y>)\}\}. \tag{6.24}$$

- Max-max-композиция:

$$\mu_{\mathcal{B}}(y) = \max_{x \in X} \{\max\{\mu_{\mathcal{A}}(x), \mu_{Q}(<x, y>)\}\}. \tag{6.25}$$

- Min-min-композиция:

$$\mu_{\mathcal{B}}(y) = \min_{x \in X} \{\min\{\mu_{\mathcal{A}}(x), \mu_{Q}(<x, y>)\}\}. \tag{6.26}$$

- Max-average-композиция:

$$\mu_{\mathcal{B}}(y) = 0.5 \cdot \max_{x \in X} \{\mu_{\mathcal{A}}(x) + \mu_{Q}(<x, y>)\}. \tag{6.27}$$

- Sum-prod-композиция:

$$\mu_{\mathcal{B}}(y) = f\left(\sum_{x \in X} (\mu_{\mathcal{A}}(x) \cdot \mu_{Q}(<x, y>))\right). \tag{6.28}$$

  где f — некоторая логистическая функция типа сигмоидной, которая ограничивает значения функции числом из интервала [0, 1]. Этот метод композиции применяется в приложениях искусственных нейронных сетей для установления взаимосвязей между параллельными слоями в многослойных сетях.

В системах нечеткого вывода, которые рассматриваются в следующей главе, наиболее часто применяются методы max-min-композиции (6.22) и max-prod-композиции (6.23). Первый из них был предложен Л. Заде в одной из его первых работ по приближенным рассуждениям с использованием естественного языка и правил продукций.

Прямой и обратный методы вывода заключений в системах нечетких продукций

По аналогии с обычными продукционными системами важным компонентом систем нечетких продукций является так называемый метод или схема вывода заключений на основе нечетких условий в базе правил нечетких продукций. Наиболее известными являются два таких метода вывода заключений: прямой и обратный, особенности которых рассматриваются ниже.

- *Прямой метод вывода* заключений в системах нечетких продукций, называемый также методом нечеткого восходящего вывода или методом прямой нечеткой цепочки рассуждений (fuzzy forward-chaining reasoning), основан на использовании нечеткого обобщения правила вывода модус поненс — FMP (fuzzy modus ponens, нечеткий модус поненс). Согласно Л. Заде, суть нечеткого модус поненс заключается в следующем. Классическая импликация $A \supset B$ в правиле вывода MP заменяется на правило нечеткой продукции: "ЕСЛИ x есть $\mathcal{A}$, ТО y есть $\mathcal{B}$", где $\mathcal{A}$ и $\mathcal{B}$ — нечеткие множества, а само правило нечеткой продукции представляет некоторое нечеткое отношение между переменными x и y, при этом $x \in \boldsymbol{X}$ и $y \in \boldsymbol{Y}$. Что касается посылки A правила MP, то она заменяется на нечеткое условие "x есть $\mathcal{A}'$", где $\mathcal{A}'$ — нечеткое множество, отражающее знания о реальном значении переменной x. Объединение правила нечеткой продукции и нечеткого условия позволяет получить новую информацию о значении переменной y в форме: "y есть $\mathcal{B}'$". При этом заключение по правилу FMP получается как функция принадлежности нечеткого множества $\mathcal{B}'$ на основе функции принадлежности условия $\mathcal{A}'$ и функции принадлежности нечеткой импликации как соответствующего нечеткого отношения с использованием одного из методов нечеткой композиции (6.22)—(6.27).

  Применительно к системам нечетких продукций прямой метод вывода реализуется посредством преобразования отдельных фактов проблемной области в конкретные значения функций принадлежности условий нечетких продукций. После этого преобразования по одному из методов нечеткой композиции находятся значения функций принадлежности заключений правых частей по каждому из правил нечетких продукций. Эти значения функций принадлежности либо являются искомым результатом вывода, либо могут быть использованы в качестве дополнительных условий в рассматриваемой базе правил нечетких продукций. При этом правила, которые могут быть использованы для выполнения нечеткой композиции, также называют *активными*.

  Процесс вывода прямым методом в системах нечетких продукций в общем случае может иметь рекурсивный (итеративный) характер. Он может быть остановлен либо в случае отсутствия активных правил нечетких продукций, либо в случае получения функции принадлежности заключения, которое является целевым в контексте решения исходной проблемы. В этом случае функция принадлежности заключения характеризует успех процесса вывода в системах нечетких продукций и решение поставленной проблемы.

- *Обратный метод вывода* в продукционных системах, называемый также методом нечеткого нисходящего вывода или методом обратной нечеткой цепочки рассуждений (fuzzy backward-chaining reasoning), основан на использовании нечеткого обобщения правила вывода модус толленс — FMT (fuzzy modus tollens, нечеткий модус толленс). Суть нечеткого модус толленс заключается в следующем. Классическая импликация $A \supset B$ в правиле вывода MT заменяется на правило нечеткой продукции: "ЕСЛИ x есть $\mathcal{A}$, ТО y есть $\mathcal{B}$", где $\mathcal{A}$ и $\mathcal{B}$ — нечеткие множества, а правило нечеткой продукции представляет некоторое нечеткое отношение между переменными x и y, при этом $x \in \boldsymbol{X}$ и $y \in \boldsymbol{Y}$, как и в методе FMP. Заключение B заменяется нечетким заключением в форме "является ли y $\mathcal{B}'$?" или "y есть $\mathcal{B}'$?". При этом нечеткое множество $\mathcal{B}'$ не равно нечеткому множеству $\mathcal{B}$, используемому в заключении правила нечеткой продукции. Целью вывода методом обратной нечеткой цепочки рассуждений является установление истинности условия правила нечеткой продукции в форме: "является ли x $\mathcal{A}'$?" или "x есть $\mathcal{A}'$?". В этом случае заключение по правилу FMT получается как функция принадлежности нечеткого множества $\mathcal{A}'$ на основе функции принадлежности заключения $\mathcal{B}'$ и функции принадлежности нечеткой импликации как соответствующего нечеткого отношения с использованием одного из методов нечеткой композиции (6.22)—(6.27).

Принципиальное различие между обратными методами вывода заключений в нечетких и обычных системах продукций заключается в том, что применительно к системам нечетких продукций функции принадлежности условий неизвестны и должны быть как-то заданы. Процесс обратного вывода в системах нечетких продукций начинается с подстановки отдельных интересующих нас значений функции принадлежности заключений в правые части соответствующих правил нечетких продукций, которые в этом случае становятся *активными*. После анализа каждого из активных правил находятся функции принадлежности условий, которые используются в этих правилах. Эти функции принадлежности условий принимаются в качестве подцелей, которые могут быть использованы в качестве функций принадлежности новых заключений в рассматриваемой базе правил нечетких продукций.

Процесс вывода обратным методом также имеет рекурсивный (итеративный) характер. Он может быть остановлен либо в случае отсутствия новых активных правил, либо в случае получения значений функций принадлежности условий, которые подтверждаются фактами проблемной области. Подобное подтверждение условий характеризует успех процесса вывода и справедливость значений функции принадлежности исходных заключений.

Примечание

Изложенные здесь особенности методов вывода заключений в системах нечетких продукций представлены в схематичном виде. Более детально эти методы рассматриваются в *главе 7*, где представлены конкретные алгоритмы нечеткого вывода в системах нечетких продукций.

П р и м е р 6.5. В качестве примера нахождения заключений прямым методом вывода в системах нечетких продукций рассмотрим использование модифицированной базы правил для проблемной области, связанной к проверкой граждан на таможенном пункте контроля с целью исключения возможности провоза наркотиков (см. пример П2.8).

В качестве системы нечетких продукций рассмотрим следующее множество правил нечетких продукций, которое более адекватно представляет ситуацию таможенного досмотра граждан при пересечении границы, чем в примере П2.8.

- ПРАВИЛО_1: ЕСЛИ "Гражданин не является высокопоставленным чиновником", ТО "он подвергается таможенному досмотру" (F_1=1.0).
- ПРАВИЛО_2: ЕСЛИ "Гражданин является высокопоставленным чиновником", ТО "он не подвергается таможенному досмотру" (F_2=0.9).
- ПРАВИЛО_3: ЕСЛИ "Гражданин не подвергается таможенному досмотру", ТО "не исключается возможность провоза наркотиков" (F_3=0.8).
- ПРАВИЛО_4: ЕСЛИ "Количество граждан, проходящих таможенный осмотр, велико", ТО "контролер испытывает чувство усталости" (F_4=0.6).
- ПРАВИЛО_5: ЕСЛИ "Контролер испытывает чувство усталости", ТО "не исключается возможность провоза наркотиков" (F_5=0.7).
- ПРАВИЛО_6: ЕСЛИ "Гражданин подвергается таможенному досмотру" И "в отношении этого гражданина имеется агентурная информация", ТО "исключается возможность провоза наркотиков" (F_6=0.95).
- ПРАВИЛО_7: ЕСЛИ "Гражданин подвергается таможенному досмотру" И "контролер использует новейшие технические средства", ТО "исключается возможность провоза наркотиков" (F_7=0.95).

При этом каждое из правил нечетких продукций имеет некоторый вес или коэффициент определенности F_i, который определяет значимость правила или уверенность в степени истинности заключения, получаемого по отдельному нечеткому правилу. В качестве условий и заключений каждого из правил используются нечеткие высказывания.

Предположим, что на таможенном пункте контроля сложилась следующая текущая ситуация. Среди граждан, въезжающих в страну, находятся высокопоставленные чиновники (Т=0.2). Количество граждан, проходящих таможенный осмотр, невелико (Т=0.1). Таможенный пункт контроля оснащен новейшими техническими средствами (Т=0.8). Какая-либо предварительная информация о наличии наркотиков у отдельных граждан отсутствует (Т=0.9). Здесь в скобках указаны степени истинности соответствующих нечетких высказываний.

Проблема заключается в оценке истинности нечеткого заключения об исключении возможности провоза наркотиков через данный пункт контроля.

Рассмотрим один из возможных способов решения данной проблемы с использованием прямого метода вывода, правила max-min-композиции и нечеткой операции max-дизъюнкции для оценки одинаковых заключений. С этой целью последовательно рассмотрим все правила для получения соответствующих нечетких заключений.

Правило 1 позволяет получить нечеткое заключение о том, что "граждане будут подвергнуты таможенному досмотру". Степень истинности этого нечеткого высказывания равна: $T=\min\{1-0.2, 1.0\}=0.8$.

Правило 2 позволяет получить нечеткое заключение о том, что "граждане не будут подвергнуты таможенному досмотру". Степень истинности этого нечеткого высказывания равна: $T=\min\{0.2, 0.9\}=0.2$.

Правило 3 позволяет получить нечеткое заключение о том, что "не исключается возможность провоза наркотиков". Степень истинности этого нечеткого высказывания равна: $T=\min\{0.2, 0.8\}=0.2$.

Правило 4 позволяет получить нечеткое заключение о том, что "контролер испытывает чувство усталости". Степень истинности этого нечеткого высказывания равна: $T=\min\{1-0.1, 0.6\}=0.6$.

Правило 5 позволяет получить нечеткое заключение о том, что "не исключается возможность провоза наркотиков". Степень истинности этого нечеткого высказывания равна: $T=\min\{0.6, 0.7\}=0.6$.

Правило 6 позволяет получить нечеткое заключение о том, что "исключается возможность провоза наркотиков". Степень истинности этого нечеткого высказывания равна: $T=\min\{\min\{0.8, 0.8\}, 0.95\}=0.1$.

Правило 7 позволяет получить нечеткое заключение о том, что "исключается возможность провоза наркотиков". Степень истинности этого нечеткого высказывания равна: $T=\min\{\min\{0.8, 0.8\}, 0.95\}=0.8$.

Интересующее нас нечеткое высказывание является нечетким следствием правил 6 и 7. Объединяя степени истинности нечетких заключений, полученных с помощью этих правил, получим искомое значение истинности нечеткого высказывания о том, что для рассматриваемой ситуации "исключается возможность провоза наркотиков". Это значение равно: $T=\max\{0.1, 0.8\}=0.8$. Дополнительно можно получить степень истинности нечеткого высказывания о том, что "не исключается возможность провоза наркотиков". Это значение равно: $T=\max\{0.6, 0.2\}=0.6$ (см. правила 3 и 5).

Анализ этих значений истинности показывает, что сложившаяся ситуация на таможенном пункте контроля характеризуется высоким уровнем неопределенности и может потребовать от руководства принятия дополнительных мер. Одно из таких решений может быть связано со своевременной заменой контролеров с целью предотвращения их усталости. Это решение основано на том обстоятельстве, что относительно высокая степень истинности заключения "не исключается возможность провоза наркотиков" получается при использовании правила 5, условием которого является "контролер испытывает чувство усталости".

Примечание

Рассмотренная система правил нечетких продукций не претендует на законченность и служит лишь иллюстрацией применения основных идей нечеткой логики. В этой связи следует отметить, что реальные экспертные системы, основанные на правилах нечетких продукций, могут содержать сотни отдельных правил. В этом случае собственно процесс получение нечетких заключе-

ний может превратиться в серьезную проблему, имеющую самостоятельное значение и требующую дополнительных средств для своего конструктивного решения.

Тем не менее, даже этот простой пример показывает, что системы правил нечетких продукций позволяют не только получить более адекватное решение проблемы, но и менее строго относиться к противоречивости и полноте исходных правил. Действительно, наличие совокупности правил, которая приводит в обычной ситуации к взаимно исключающим заключениям, в нечеткой продукционной системе еще не служит признаком ее противоречивости. Безусловно, это является одним из достоинств правил нечетких продукций.

Примечание

Можно предложить в качестве упражнения получить нечеткие заключения для системы правил нечетких продукций из примера 6.5 с использованием других методов нечеткой композиции (6.23)—(6.28), а полученные результаты сравнить.

П р и м е р 6.6. В заключение этой главы рассмотрим процесс получения заключений на основе непосредственного использования нечеткого отношения, которое было построено в примере 4.3. Этот пример характерен для задач технической диагностики и иллюстрирует два важных аспекта: особенности решения нечетких уравнений и неравенств, а также метод получения заключений с использованием посылок.

Напомним, что в примере 4.3 рассматривалось нечеткое отношение, которое содержательно описывает упрощенную ситуацию поиска неисправности в автомобиле. С этой целью в качестве предпосылок или причин неисправности рассматривалось множество $\boldsymbol{X}=\{x_1, x_2, x_3, x_4\}$, в котором x_1 — "*неисправность аккумулятора*", x_2 — "*неисправность карбюратора*", x_3 — "*низкое качество бензина*", x_4 — "*неисправность системы зажигания*". В качестве второго универсума рассмотрим множество заключений или проявлений неисправности $\boldsymbol{Y}=\{y_1, y_2, y_3\}$, где y_1 — "*двигатель не запускается*", y_2 — "*двигатель работает неустойчиво*", y_3 — "*двигатель не развивает полной мощности*".

Причинная взаимосвязь между множеством предпосылок и множеством следствий представлена в виде бинарного нечеткого отношения $\mathcal{P}=\{<x_i, y_j>, \mu_{\mathcal{P}}(<x_i, y_j>)\}$, заданного в форме матрицы $\mathbf{M}_{\mathcal{P}}$ этого нечеткого отношения: $\mathbf{M}_{\mathcal{P}} = \begin{bmatrix} 1 & 0.1 & 0.2 \\ 0.8 & 0.9 & 1 \\ 0.7 & 0.8 & 0.5 \\ 1 & 0.5 & 0.2 \end{bmatrix}$.

Если результаты осмотра конкретного автомобиля показывают, что двигатель не запускается, хотя и работает устойчиво и развивает полную мощность, то эта информации может быть представлена в форме нечеткого множества, например, $\mathcal{B}=\{<y_1, 0.9>, <y_2, 0.1>, <y_3, 0.2>\}$. Задача состоит в том, чтобы определить воз-

можные причины неустойчивой работы двигателя, используя для этого экспертную информацию в форме нечеткого отношения $\mathcal{P}$.

Другими словами, в этой задаче, характерной для задач технической диагностики, необходимо найти такое нечеткое множество $\mathcal{A}=\{<x_1, \mu_{\mathcal{A}}(x_1)>, <x_2, \mu_{\mathcal{A}}(x_2)>, <x_3, \mu_{\mathcal{A}}(x_3)>, <x_4, \mu_{\mathcal{A}}(x_4)>\}$, которое соответствовало бы нечеткому множеству $\mathcal{B}$.

С этой целью представим нечеткое множество $\mathcal{B}$ в форме вектора $\boldsymbol{b}$ = (0.9 0.1 0.2), компоненты которого равны значениям функции принадлежности соответствующих элементов. Нечеткое множество $\mathcal{A}$ также представим в форме вектора $\boldsymbol{a}$ = = (a_1, a_2, a_3, a_4), компоненты которого неизвестны и требуется определить. Базовой предпосылкой для решения этой задачи является предположение о том, что компоненты вектора $\boldsymbol{a}$ должны удовлетворять следующему условию:

$$\boldsymbol{a} \otimes \mathbf{M}_{\mathcal{P}} = \boldsymbol{b}, \tag{6.29}$$

где "⊗" — знак композиции, соответствующий одному из методов (6.22)—(6.28).

Если в качестве метода нечеткой композиции использовать max-min-композицию (6.22), то выражение (6.29) преобразуется в следующую систему так называемых *нечетких уравнений*:

$$(1\wedge a_1)\vee(0.8\wedge a_2)\vee(0.7\wedge a_3)\vee(1\wedge a_4) = 0.9; \tag{6.30}$$

$$(0.1\wedge a_1)\vee(0.9\wedge a_2)\vee(0.8\wedge a_3)\vee(0.5\wedge a_4) = 0.1; \tag{6.31}$$

$$(0.2\wedge a_1)\vee(1\wedge a_2)\vee(0.5\wedge a_3)\vee(0.2\wedge a_4) = 0.2. \tag{6.32}$$

Здесь связка "∨" используется для сокращения записи операции max, а связка "∧" — для сокращения записи операции min. Необходимо решить эту систему уравнений, т. е. найти такие значения a_1, a_2, a_3, a_4, которые бы удовлетворяли соотношениям (6.30)—(6.32).

Прежде всего, следует отметить, что в первом уравнении (6.30) второй и третий компоненты не оказывают влияния на результат правой части. Откуда следует, что $(1\wedge a_1) \vee(1\wedge a_4) = 0.9$ или $a_1\vee a_4 = 0.9$. Это приводит к необходимости рассмотреть два случая: a_1=0.9 и a_4 = 0.9. Оба эти случая удовлетворяют (6.32).

В то же время первый случай a_1=0.9 удовлетворяет (6.31), а второй a_4=0.9 — не удовлетворяет (6.31), откуда следует, что $a_4\le0.1$. Из (6.31) также следует, что: $a_2\le0.1$ и $a_3\le0.1$. Эти неравенства и являются решением поставленной задачи.

Таким образом, решением задачи диагностики является произвольный вектор значений функций принадлежности $\boldsymbol{a}$ = (a_1, a_2, a_3, a_4), компоненты которого удовлетворяют условиям: a_1=0.9, $a_2\le0.1$, $a_3\le0.1$, $a_4\le0.1$. Для рассматриваемой ситуации этого может оказаться и достаточно, поскольку первое значение $\mu_{\mathcal{A}}(x_1)$ = = 0.9 явно указывает на неисправность аккумулятора.

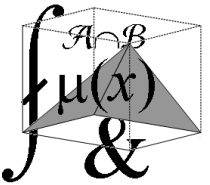
Примечание

В качестве упражнения предлагается решить аналогичную задачу, используя другие методы композиции (6.23)—(6.28) и другие начальные условия осмотра

автомобиля. Полученные результаты сравнить. При необходимости можно дополнить экспертные знания, расширив рассматриваемое нечеткое отношение на большее число элементов базисных универсумов ***X*** и ***Y***. В качестве еще одного упражнения можно предложить преобразовать нечеткое отношение из примера 6.6 в совокупность правил нечетких продукций и выполнить вывод заключений для аналогичных исходных данных.

Не исключая возможности практического использования рассмотренных здесь нечетких продукционных систем и нечетких отношений, следует отметить, что наибольшее практическое применение нашли так называемые системы нечеткого вывода, в которых сделаны некоторые дополнительные предположения о форме нечетких высказываний, используемых в качестве условий и заключений в правилах нечетких продукций. Именно эти системы нечеткого вывода, рассмотрению которых посвящена следующая глава, оказываются наиболее адекватным средством формализации экспертных знаний в самых различных проблемных областях и вывода соответствующих заключений.

Глава 7



Системы нечеткого вывода

Нечеткий вывод занимает центральное место в нечеткой логике и системах нечеткого управления. Процесс нечеткого вывода представляет собой некоторую процедуру или алгоритм получения нечетких заключений на основе нечетких условий или предпосылок с использованием рассмотренных выше понятий нечеткой логики. Этот процесс соединяет в себе все основные концепции теории нечетких множеств: функции принадлежности, лингвистические переменные, нечеткие логические операции, методы нечеткой импликации и нечеткой композиции.

Системы нечеткого вывода предназначены для реализации процесса нечеткого вывода и служат концептуальным базисом всей современной нечеткой логики. Достигнутые успехи в применении этих систем для решения широкого класса задач управления послужили основой становления нечеткой логики как прикладной науки с богатым спектром приложений. Системы нечеткого вывода позволяют решать задачи автоматического управления, классификации данных, распознавания образов, принятия решений, машинного обучения и многие другие.

Поскольку разработка и применение систем нечеткого вывода имеет междисциплинарный характер, данная проблематика исследований тесно взаимосвязана с целым рядом других научно-прикладных направлений, таких как: нечеткое моделирование, нечеткие экспертные системы, нечеткая ассоциативная память, нечеткие логические контроллеры, нечеткие регуляторы и просто нечеткие системы.

7.1. Базовая архитектура систем нечеткого вывода

Рассматриваемые в настоящей главе системы нечеткого вывода являются частным случаем продукционных нечетких систем или систем нечетких правил продукций, в которых условия и заключения отдельных правил формулируются в форме нечетких высказываний относительно значений тех или иных лингвистических переменных. Поскольку нечеткие лингвистические высказывания имеют фундаментальное значение в контексте современной нечеткой логики, изучение систем нечеткого вывода начнем именно с них.

Нечеткие лингвистические высказывания

Нечеткое лингвистическое высказывание. *Нечетким лингвистическим высказыванием* будем называть высказывания следующих видов.

1. Высказывание "β *есть* α", где β — наименование лингвистической переменной, α — ее значение, которому соответствует отдельный лингвистический терм из базового терм-множества T лингвистической переменной β.
2. Высказывание "β *есть* $\nabla\alpha$", где ∇ — *модификатор*, соответствующий таким словам, как: "ОЧЕНЬ", "БОЛЕЕ ИЛИ МЕНЕЕ", "МНОГО БОЛЬШЕ" и другим, которые могут быть получены с использованием процедур G и M данной лингвистической переменной.
3. Составные высказывания, образованные из высказываний видов 1 и 2 и нечетких логических операций в форме связок: "И", "ИЛИ", "ЕСЛИ-ТО", "НЕ".

Поскольку в системах нечеткого вывода нечеткие лингвистические высказывания занимают центральное место, далее будем их называть просто нечеткими высказываниями.

Пример 7.1. Рассмотрим некоторые примеры нечетких высказываний. Первое из них — "*скорость автомобиля высокая*" представляет собой нечеткое высказывание первого вида, в рамках которого лингвистической переменной "*скорость автомобиля*" присваивается значение "*высокая*". При этом предполагается, что на универсальном множестве X переменной "*скорость автомобиля*" определен соответствующий лингвистический терм "*высокая*", который задается в форме функции принадлежности некоторого нечеткого множества (например, рис. 5.2).

Нечеткое высказывание второго вида "*скорость автомобиля очень высокая*" означает, что лингвистической переменной "*скорость автомобиля*" присваивается значение "*высокая*" с модификатором "ОЧЕНЬ", который изменяет значение соответствующего лингвистического терма "*высокая*" на основе использования некоторой расчетной формулы, например (3.47) для операции концентрации $\mathrm{CON}(\mathcal{A})$ нечеткого множества $\mathcal{A}$ для терма "*высокая*".

Нечеткое высказывание второго вида "*скорость автомобиля более или менее высокая*" означает, что лингвистической переменной "*скорость автомобиля*" присваивается значение "*высокая*" с модификатором "БОЛЕЕ ИЛИ МЕНЕЕ", который изменяет значение соответствующего лингвистического терма "*высокая*" на основе использования некоторой расчетной формулы, например (3.48) для операции растяжения $\mathrm{DIL}(\mathcal{A})$ нечеткого множества $\mathcal{A}$ для терма "*высокая*".

Ниже на рис. 7.1. изображен пример функции принадлежности терм-множества "*средняя*" лингвистической переменной "*скорость автомобиля*" (*а*) и определение значений функций принадлежности этого же терм-множества для модификаторов "ОЧЕНЬ" (*б*) и "БОЛЕЕ МЕНЕЕ" (*в*).

Примечание

Поскольку в настоящее время отсутствует единая точка зрения на применение других модификаторов, соответствующие расчетные формулы в случае их ис-

пользования в конкретных системах нечеткого вывода принято указывать явно в рамках процедур *G* и *M* соответствующей лингвистической переменной. Несмотря на это, в рассматриваемых ниже правилах нечетких продукций стараются избегать употребления модификаторов даже на этапе их построения.

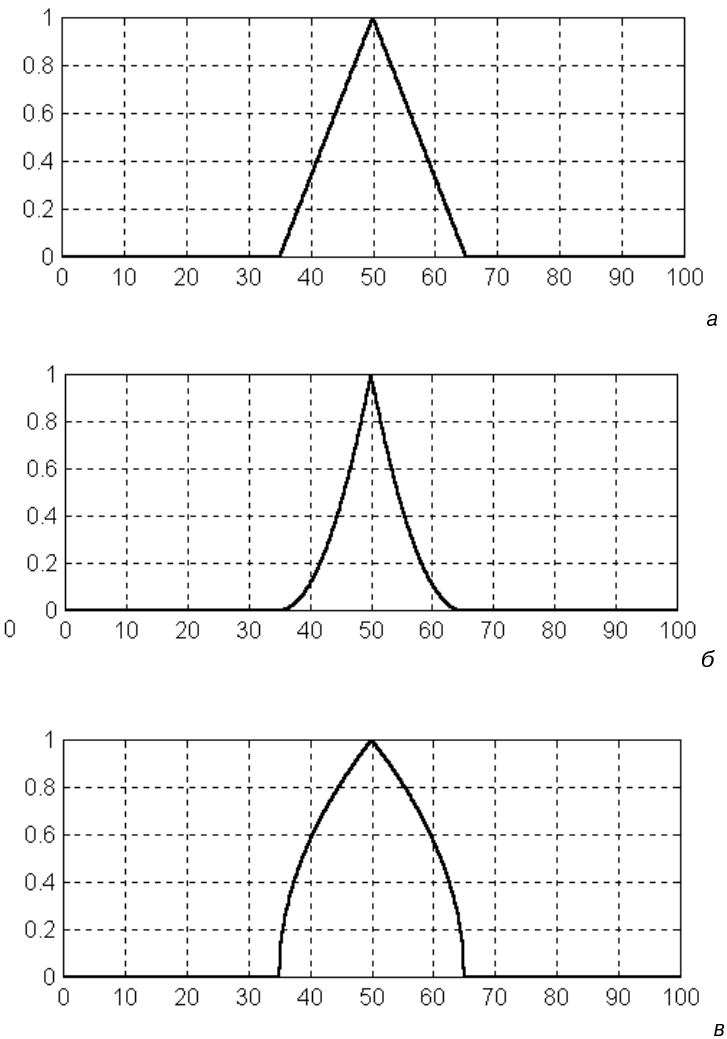


Рис. 7.1. Применение модификаторов "ОЧЕНЬ" (*б*) и "БОЛЕЕ МЕНЕЕ" (*в*) к терму "*средняя скорость*" (*а*)

Наконец, нечеткое высказывание третьего вида "*скорость автомобиля высокая и расстояние до перекрестка близкое*" означает, что одной лингвистической пере-

менной "*скорость автомобиля*" присваивается значение "*высокая*", а другой лингвистической переменной "*расстояние до перекрестка*" присваивается значение "*близкое*". Эти нечеткие высказывания первого вида соединены логической операцией нечеткая конъюнкция (операцией нечеткое "И").

Примечание

При записи нечетких высказываний в форме структурируемого текста на языке FCL используется специальная связка "IS". В этом случае нечеткое высказывание "*давление большое*" может быть представлено в виде "давление IS большое", что может оказаться более удобным для последующей интерпретации.

Правила нечетких продукций в системах нечеткого вывода

Как уже отмечалось в начале этой главы, рассматриваемые здесь системы нечеткого вывода являются частным случаем продукционных нечетких систем или систем нечетких правил продукций вида (6.21), определение которых было дано ранее в *главе 6*. Основная особенность нечетких правил, используемых в системах нечеткого вывода, — условия и заключения отдельных нечетких правил формулируются в форме нечетких высказываний вида 1—3 относительно значений тех или иных лингвистических переменных.

Таким образом, всюду далее под *правилом нечеткой продукции* или просто — *нечеткой продукцией* будем понимать выражение следующего вида: $(i) : Q; P; A \Rightarrow B; S, F, N$, в котором все компоненты определены согласно (6.21), за исключением того, что условие ядра (антецедент) $\mathcal{A}$ и заключение ядра (консеквент) $\mathcal{B}$ представляют собой нечеткие лингвистические высказывания вида 1—3.

Простейший вариант правила нечеткой продукции, который наиболее часто используется в системах нечеткого вывода, может быть записан в форме:

ПРАВИЛО <#>: ЕСЛИ "β_1 *есть* α'", ТО "β_2 *есть* α''". (7.1)

Здесь нечеткое высказывание "β_1 *есть* α'" представляет собой условие данного правила нечеткой продукции, а нечеткое высказывание "β_2 *есть* α''" — нечеткое заключение данного правила. При этом считается, что $\beta_1 \neq \beta_2$.

Примечание

Запись простейшего варианта правила нечеткой продукции в англоязычной транскрипции: RULE <#>: IF "β_1 IS α'", THEN "β_2 IS α''" считается эквивалентной записи (7.1), если не оговорено обратное. Использование той или иной записи определяется соображениями удобства или необходимости следования некоторым стандартным нотациям.

Система нечетких правил продукций. *Система нечетких правил продукций* или *продукционная нечеткая система* представляет собой некото-

рое согласованное множество отдельных *нечетких продукций* или правил нечетких продукций в форме "ЕСЛИ $\mathcal{A}$, ТО $\mathcal{B}$" (или в виде: "IF $\mathcal{A}$ THEN $\mathcal{B}$"), где $\mathcal{A}$ и $\mathcal{B}$ — нечеткие лингвистические высказывания вида 1, 2 или 3. Два последних случая нечетких высказываний требуют дополнительного пояснения.

Рассмотрим вариант использования в качестве условия или заключения в некотором правиле нечеткой продукции нечеткого высказывания вида 2, т. е. вида: "β *есть* $\nabla\alpha$", где ∇ — модификатор, определяемый процедурами G и M лингвистической переменной β. Пусть терму α соответствует нечеткое множество $\mathcal{A}$. В этом случае исходное нечеткое высказывание "β *есть* $\nabla\alpha$" можно преобразовать к виду 1 в форме нечеткого высказывание "β *есть* α'", где терм α' получается на основе применения определенной процедурами G и M операции к нечеткому множеству $\mathcal{A}$. Полученное в результате подобной операции нечеткое множество $\mathcal{A}'$ принимается за значение терм-множества α'.

Если в качестве условия или заключения используются составные нечеткие высказывания, т. е. образованные из высказываний видов 1 и 2 и нечетких логических операций в форме связок: "И", "ИЛИ", "ЕСЛИ-ТО", "НЕ", то ситуация несколько усложняется. Поскольку вариант использования нечетких высказываний вида 2 сводится к нечетким высказываниям вида 1, то достаточно рассмотреть сложные высказывания, в которых нечеткими логическими операциями соединены только нечеткие высказывания вида 1.

Эта ситуация может соответствовать простейшему случаю, когда нечеткими логическими операциями соединены нечеткие высказывания, относящиеся к одной и той же лингвистической переменной, т. е. в форме: "β *есть* α'" ОП "β *есть* α''", где ОП — некоторая из бинарных операций нечеткой конъюнкции "И" или нечеткой дизъюнкции "ИЛИ".

Примечание

Поскольку нечеткая импликация и нечеткая эквивалентность могут быть выражены через операции нечеткой конъюнкции и нечеткой дизъюнкции, а нечеткое отрицание в данном контексте является по сути модификатором, ограничимся рассмотрением только двух указанных выше нечетких операций.

Очевидно, в этом простейшем случае нечеткое высказывание "β *есть* α'" И "β *есть* α''" эквивалентно нечеткому высказыванию "β *есть* α^*", где терм-множеству α^* соответствует нечеткое множество $\mathcal{A}^*$, равное пересечению нечетких множеств $\mathcal{A}'$ и $\mathcal{A}''$, которые соответствуют термам α' и α''. При этом операция пересечения определяется одним из ранее рассмотренных способов (3.3), (3.19), (3.31), (3.33), (3.35).

Соответственно, нечеткое высказывание "β *есть* α'" ИЛИ "β *есть* α''" эквивалентно нечеткому высказыванию "β *есть* α^*", где терм-множеству α^* соответствует нечеткое множество $\mathcal{A}^*$, равное объединению нечетких множеств $\mathcal{A}'$ и $\mathcal{A}''$, которые соответствуют термам α' и α''. При этом операция объединения определяется одним из ранее рассмотренных способов (3.4), (3.20), (3.32), (3.34), (3.35).

П р и м е р 7.2. Рассмотрим составное нечеткое высказывание вида 3: "*скорость автомобиля средняя и скорость автомобиля высокая*". Ему соответствуют два нечетких высказывания первого вида, соединенные логической операцией нечеткой конъюнкции. Тогда исходное нечеткое высказывание эквивалентно нечеткому высказыванию первого вида: "*скорость автомобиля средняя и высокая*". Функция принадлежности терма "*средняя и высокая*" изображена на рис. 7.2, *б* более темным фоном, при этом результат нечеткой конъюнкции определялся по формуле (3.3).

Рассмотрим аналогичное составное нечеткое высказывание вида 3: "*скорость автомобиля средняя или скорость автомобиля высокая*". Ему также соответствуют два нечетких высказывания первого вида, соединенные логической операцией нечеткой дизъюнкции. Тогда исходное нечеткое высказывание эквивалентно нечеткому высказыванию первого вида: "*скорость автомобиля средняя или высокая*". Функция принадлежности терма "*средняя или высокая*" изображена на рис. 7.2, *в* более темным фоном, при этом результат нечеткой дизъюнкции определялся по формуле (3.4).

Во-вторых, ситуация может соответствовать более сложному случаю, когда нечеткими логическими операциями соединены нечеткие высказывания, относящиеся к разным лингвистическим переменным в условии правила нечеткой продукции, т. е. в форме: "β_1 *есть* α'" ОП "β_2 *есть* α''", где ОП — некоторая из бинарных операций нечеткой конъюнкции "И" или нечеткой дизъюнкции "ИЛИ", а β_1 и β_2 — различные лингвистические переменные.

Этот вариант правил нечетких продукций может быть записан в следующей общей форме:

ПРАВИЛО <#>: ЕСЛИ "β_1 *есть* α'" И "β_2 *есть* α''" ТО "β_3 *есть* ν"
или
ПРАВИЛО <#>: ЕСЛИ "β_1 *есть* α'" ИЛИ "β_2 *есть* α''" ТО "β_3 *есть* ν". (7.2)

Здесь нечеткие высказывания: "β_1 *есть* α'" И "β_2 *есть* α''", "β_1 *есть* α'" ИЛИ "β_2 *есть* α''"представляют собой условия правил нечетких продукций, а нечеткое высказывание "β_3 *есть* ν" — заключение правил. При этом считается, что $\beta_1 \neq \beta_2 \neq \beta_3$, а каждое из нечетких высказываний "β_1 *есть* α'", "β_2 *есть* α''" называют *подусловиями* данных правил нечетких продукций.

В случае правил нечетких продукций в форме (7.2) необходимо использовать один из методов агрегирования условий в левой части этих правил. Соответствующие методы агрегирования рассматриваются ниже при описании этапа агрегирования.

Наконец, нечеткими логическими операциями могут быть соединены нечеткие высказывания, относящиеся к разным лингвистическим переменным в заключении правила нечеткой продукции, т. е. в форме: "β_1 *есть* α'" ОП "β_2 *есть* α''", где ОП — некоторая из бинарных операций нечеткой конъюнкции "И" или нечеткой дизъюнкции "ИЛИ", а β_1 и β_2 — различные лингвистические переменные.

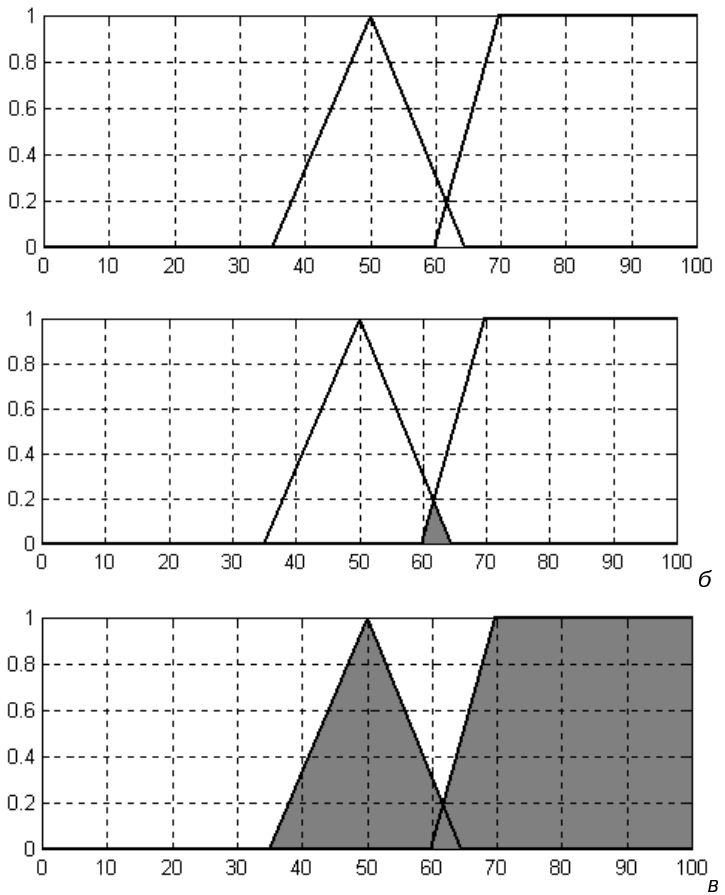


Рис. 7.2. Преобразование составных нечетких высказываний, относящихся к одной и той же лингвистической переменной

Этот вариант правил нечетких продукций может быть записан в следующей общей форме:

ПРАВИЛО <#>: ЕСЛИ "β_1 *есть* α'" ТО "β_2 *есть* α''" И "β_3 *есть* ν"
или
ПРАВИЛО <#>: ЕСЛИ "β_1 *есть* α'" ТО "β_2 *есть* α''" ИЛИ "β_3 *есть* ν". (7.3)

Здесь нечеткое высказывание "β_1 *есть* α'" представляет собой условие правил нечетких продукций, а нечеткие высказывания: "β_2 *есть* α''" И "β_3 *есть* ν", "β_2 *есть* α''" ИЛИ "β_3 *есть* ν" — заключения данных правил. При этом считается, что $\beta_1 \neq \beta_2 \neq \beta_3$, а каждое из нечетких высказываний "β_2 *есть* α''", "β_3 *есть* ν" называют *подзаключениями* данного правила нечеткой продукции.

В случае правил нечетких продукций в форме (7.3) необходимо использовать один из методов аккумуляции заключений в правилах нечетких продукций. Эти методы также рассматриваются ниже при описании этапа аккумуляции.

Примечание

Следует отметить, что если использование правил нечетких продукций в форме (7.2) является широко распространенным, то записи правил нечетких продукций в форме (7.2) стараются избегать, преобразуя их к форме (7.1) или (7.2) на этапе построения базы правил нечетких продукций. Примеры конкретных правил представлены в *разд. 7.4* настоящей главы.

Механизм или алгоритм вывода в системах нечеткого вывода

Механизм или алгоритм вывода является следующей важной частью базовой архитектуры систем нечеткого вывода. Применительно к системам нечеткого вывода механизм вывода представляет собой конкретизацию рассмотренных ранее методов прямого и обратного вывода заключений в системах нечетких продукций *(см. главу 6)*. В данном случае алгоритм вывода оперирует правилами нечетких продукций, в которых условия и заключения записаны в форме нечетких лингвистических переменных.

Для получения заключений в системах нечеткого вывода предложены несколько алгоритмов, характерные особенности и примеры применения которых изложены ниже в настоящей главе. Описание этих алгоритмов базируется на разделении процесса вывода на ряд последовательных этапов, которые рассматриваются в следующем разделе. Тем самым оказывается возможным не только достичь определенной систематизации понятий нечеткой логики, но и получить некоторую общую схему, которая позволяет формировать и другие алгоритмы нечеткого вывода.

7.2. Основные этапы нечеткого вывода

Говоря о нечеткой логике, чаще всего имеют в виду системы нечеткого вывода, которые широко используются для управления техническими устройствами и процессами. Разработка и применение систем нечеткого вывода включают в себя ряд этапов, реализация которых выполняется с помощью рассмотренных ранее основных положений нечеткой логики.

Информацией, которая поступает на вход системы нечеткого вывода, являются измеренные некоторым образом входные переменные. Эти переменные соответствуют реальным переменным процесса управления. Информация, которая формируется на выходе системы нечеткого вывода, соответствует выходным переменным, которыми являются управляющие переменные процесса управления.

Системы нечеткого вывода предназначены для преобразования значений входных переменных процесса управления в выходные переменные на основе использования нечетких правил продукций. Для этого системы нечеткого вывода должны содержать базу правил нечетких продукций и реализовывать нечеткий вывод заключений на основе посылок или условий, представленных в форме нечетких лингвистических высказываний.

Таким образом, основными этапами нечеткого вывода являются (рис. 7.3).

- Формирование базы правил систем нечеткого вывода.
- Фаззификация входных переменных.
- Агрегирование подусловий в нечетких правилах продукций.

Рис. 7.3. Диаграмма деятельности процесса нечеткого вывода в форме диаграммы деятельности языка UML

- Активизация или композиция подзаключений в нечетких правилах продукций.
- Аккумулирование заключений нечетких правил продукций.

Ниже рассматриваются основные особенности каждого из этих этапов и приводятся простые примеры их выполнения.

Формирование базы правил систем нечеткого вывода

База правил систем нечеткого вывода предназначена для формального представления эмпирических знаний или знаний экспертов в той или иной проблемной области. В системах нечеткого вывода используются правила нечетких продукций, в которых условия и заключения сформулированы в терминах нечетких лингвистических высказываний рассмотренных выше видов. Совокупность таких правил будем далее называть базами правил нечетких продукций.

Б а з а п р а в и л н е ч е т к и х п р о д у к ц и й. *База правил нечетких продукций* представляет собой конечное множество правил нечетких продукций, согласованных относительно используемых в них лингвистических переменных. Наиболее часто база правил представляется в форме структурированного текста:

ПРАВИЛО_1: ЕСЛИ "Условие_1" ТО "Заключение_1" (F_1)
ПРАВИЛО_2: ЕСЛИ "Условие_2" ТО "Заключение_2" (F_2)
…
ПРАВИЛО_n: ЕСЛИ "Условие_n" ТО "Заключение_n" (F_n) (7.4)

или в эквивалентной форме:

RULE_1: IF Condition_1 THEN Conclusion_1 (F_1)
RULE_2: IF Condition_2 THEN Conclusion_2 (F_2)
…
RULE_n: IF Condition_n THEN Conclusion_n (F_n) (7.5)

Здесь через F_i ($i \in \{1, 2, \ldots, n\}$) обозначены *коэффициенты определенности* или весовые коэффициенты соответствующих правил. Эти коэффициенты могут принимать значения из интервала [0, 1]. В случае, если эти весовые коэффициенты отсутствуют, удобно принять, что их значения равны 1.

Согласованность правил относительно используемых лингвистических переменных означает, что в качестве условий и заключений правил могут использоваться только нечеткие лингвистические высказывания вида (7.2) и (7.3), при этом в каждом из нечетких высказываний должны быть определены функции принадлежности значений терм-множества для каждой из лингвистических переменных.

В х о д н ы е и в ы х о д н ы е л и н г в и с т и ч е с к и е п е р е м е н н ы е. В системах нечеткого вывода лингвистические переменные, которые используются в нечетких высказываниях подусловий правил нечетких продукций, часто называют *входными лингвистическими переменными*, а переменные, которые ис-

пользуются в нечетких высказываниях подзаключений правил нечетких продукций, часто называют *выходными лингвистическими переменными*.

Таким образом, при задании или формировании базы правил нечетких продукций необходимо определить: множество правил нечетких продукций: $P=\{R_1, R_2,..., R_n\}$ в форме (7.5), множество входных лингвистических переменных: $V=\{\beta_1, \beta_2,..., \beta_m\}$ и множество выходных лингвистических переменных: $W=\{\omega_1, \omega_2,..., \omega_s\}$. Тем самым база правил нечетких продукций считается заданной, если заданы множества P, V, W.

Напомним, что входная $\beta_i \in V$ или выходная $\omega_j \in W$ лингвистическая переменная считается заданной или определенной, если для нее определено базовое терм-множество с соответствующими функциями принадлежности каждого терма, а также две процедуры G и M. Наиболее распространенным случаем является использование в качестве функций принадлежности термов треугольных или трапециевидных функций принадлежности, рассмотренных в *главе 2*. При этом для удобства записи применяют специальные сокращения для наименования отдельных термов входных и выходных лингвистических переменных (табл. 7.1).

Таблица 7.1. *Общепринятые сокращения для значений основных термов лингвистических переменных в системах нечеткого вывода*

Символическое обозначение	Англоязычная нотация	Русскоязычная нотация
NB	Negative Big	Отрицательное большое
NM	Negative Middle	Отрицательное среднее
NS	Negative Small	Отрицательное малое
ZN	Zero Negative	Отрицательное близкое к нулю
Z	Zero	Нуль, близкое к нулю
ZP	Zero Positive	Положительное близкое к нулю
PS	Positive Small	Положительное малое
PM	Positive Middle	Положительное среднее
PB	Positive Big	Положительное большое

На формирование базы правил систем нечеткого вывода часто оказывают влияние некоторые дополнительные факторы, которые определяются спецификой решаемой задачи или используемого алгоритма нечеткого вывода. Эти специфические особенности и соответствующие им требования, предъявляемые к базе правил, будут отмечены ниже при рассмотрении алгоритмов вывода и примеров систем нечеткого управления в *разд. 7.4*.

Примечание

Изложенные здесь понятия базы правил нечетких продукций в максимальной степени соответствуют алгоритму вывода Мамдани, который в настоящее время получил наибольшее практическое применение в задачах нечеткого моделирования.

Фаззификация (Fuzzification)

В контексте нечеткой логики под *фаззификацией* понимается не только отдельный этап выполнения нечеткого вывода, но и собственно процесс или процедура нахождения значений функций принадлежности нечетких множеств (термов) на основе обычных (не нечетких) исходных данных. Фаззификацию еще называют *введением нечеткости*.

Целью этапа фаззификации является установление соответствия между конкретным (обычно — численным) значением отдельной входной переменной системы нечеткого вывода и значением функции принадлежности соответствующего ей терма входной лингвистической переменной. После завершения этого этапа для всех входных переменных должны быть определены конкретные значения функций принадлежности по каждому из лингвистических термов, которые используются в подусловиях базы правил системы нечеткого вывода.

Формально процедура фаззификации выполняется следующим образом. До начала этого этапа предполагаются известными конкретные значения всех входных переменных системы нечеткого вывода, т. е. множество значений $V'=\{a_1, a_2,\ldots, a_m\}$. В общем случае каждое $a_i \in \boldsymbol{X}_i$, где $\boldsymbol{X}_i$ — универсум лингвистической переменной β_i. Эти значения могут быть получены либо от датчиков, либо некоторым другим, внешним по отношению к системе нечеткого вывода способом.

Далее рассматривается каждое из подусловий вида "β_i *есть* α'" правил системы нечеткого вывода, где α' — некоторый терм с известной функцией принадлежности $\mu(x)$. При этом значение a_i используется в качестве аргумента $\mu(x)$, тем самым находится количественное значение $b_i'=\mu(a_i)$. Это значение и является результатом фаззификации подусловия "β_i *есть* α'".

Этап фаззификации считается законченным, когда будут найдены все значения $b_i'=\mu(a_i)$ для каждого из подусловий всех правил, входящих в рассматриваемую базу правил системы нечеткого вывода. Это множество значений обозначим через $B=\{b_i'\}$. При этом если некоторый терм α'' лингвистической переменной β_i не присутствует ни в одном из нечетких высказываний, то соответствующее ему значение функции принадлежности не находится в процессе фаззификации.

Примечание

Если в некотором подусловии встречается терм с модификатором, то процедура фаззификации выполняется аналогичным образом применительно к функции принадлежности терма после выполнения операции, соответствующей данному модификатору.

П р и м е р 7.3. Для иллюстрации выполнения этого этапа рассмотрим пример процесса фаззификации трех нечетких высказываний: "*скорость автомобиля малая*", "*скорость автомобиля средняя*", "*скорость автомобиля высокая*" для входной лингвистической переменной β_1 — скорость движения автомобиля (см. пример 5.1). Им соответствуют нечеткие высказывания первого вида: "β_1 *есть* α_1", "β_1 *есть* α_2", "β_1 *есть* α_3". Предположим, что текущая скорость автомобиля равна 55 км/ч, т. е. $a_1 = 55$ км/ч.

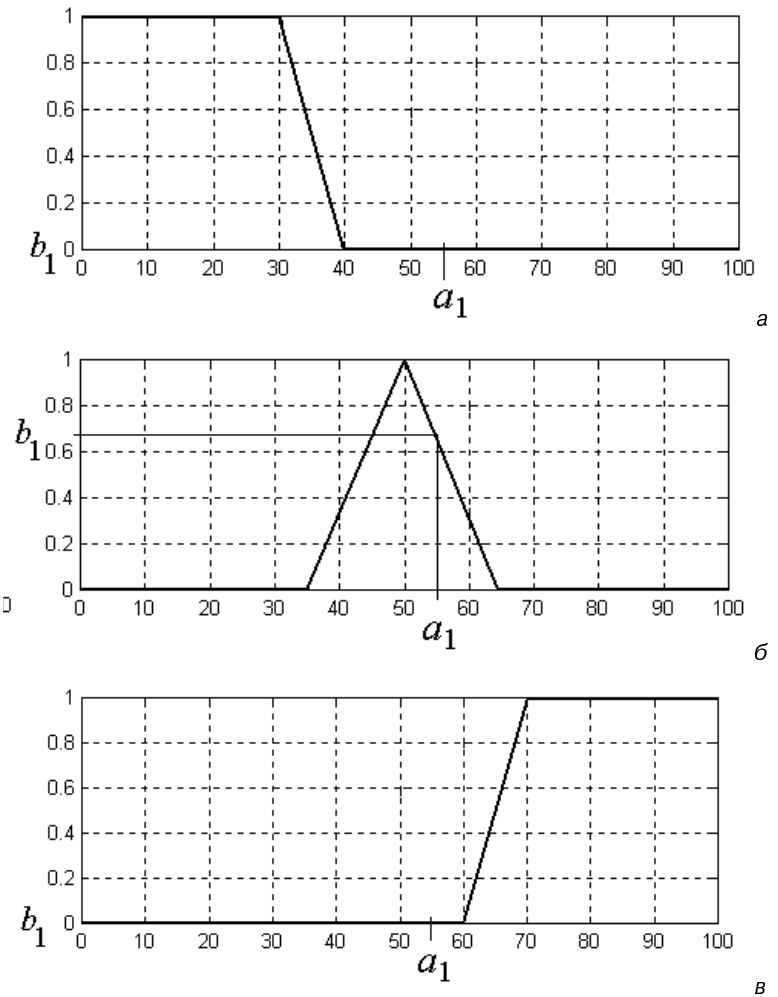


Рис. 7.4. Пример фаззификации входной лингвистической переменной "*скорость автомобиля*" для трех нечетких высказываний

Тогда фаззификация первого нечеткого высказывания дает в результате число 0, которое означает его степень истинности и получается подстановкой значения a_1= 55 км/ч в качестве аргумента функции принадлежности терма α_1 (рис. 7.4, *а*). Фаззификация второго нечеткого высказывания дает в результате число 0.67 (приближенное значение), которое означает его степень истинности и получается подстановкой значения a_1= 55 км/ч в качестве аргумента функции принадлежности терма α_2 (рис. 7.4, *б*). Фаззификация третьего нечеткого высказывания дает в результате число 0, которое означает его степень истинности и получается подстановкой значения a_1= 55 км/ч в качестве аргумента функции принадлежности терма α_3 (рис. 7.4, *в*).

Агрегирование (Aggregation)

Агрегирование представляет собой процедуру определения степени истинности условий по каждому из правил системы нечеткого вывода.

Формально процедура агрегирования выполняется следующим образом. До начала этого этапа предполагаются известными значения истинности всех подусловий системы нечеткого вывода, т. е. множество значений $B=\{b_i'\}$. Далее рассматривается каждое из условий правил системы нечеткого вывода. Если условие правила представляет собой нечеткое высказывание вида 1 или 2, то степень его истинности равна соответствующему значению b_i'.

Если же условие состоит из нескольких подусловий вида (7.2), причем лингвистические переменные в подусловиях попарно не равны друг другу, то определяется степень истинности сложного высказывания на основе известных значений истинности подусловий. При этом для определения результата нечеткой конъюнкции или связки "И" может быть использована одна из формул (6.2)—(6.5), а для определения результата нечеткой дизъюнкции или связки "ИЛИ" может быть использована одна из формул (6.6)—(6.9). При этом значения b_i' используются в качестве аргументов соответствующих логических операций. Тем самым находятся количественные значения истинности всех условий правил системы нечеткого вывода.

Этап агрегирования считается законченным, когда будут найдены все значения b_k'' для каждого из правил R_k, входящих в рассматриваемую базу правил P системы нечеткого вывода. Это множество значений обозначим через $B''=\{b_1'', b_2'',\ldots, b_n''\}$.

Примечание

Следует отметить, что при использовании расчетных формул для определения результатов нечеткой конъюнкции и нечеткой дизъюнкции целесообразно применять попарно согласованные методы расчета для всех правил системы нечетких продукций. Так, например, если в некоторой системе нечеткого вывода результат нечеткой конъюнкции определяется по формуле алгебраического произведения (6.3), то для определения результата нечеткой дизъюнкции предпочтительно использовать алгебраическую сумму (6.7).

П р и м е р 7.4. Для иллюстрации выполнения этого этапа рассмотрим пример процесса агрегирования двух нечетких высказываний: "*скорость автомобиля средняя*" И "*кофе горячий*" и "*скорость автомобиля средняя*" ИЛИ "*кофе горячий*" для входной лингвистической переменной β_1 — скорость движения автомобиля и β_2 — температура кофе. Предположим, что текущая скорость автомобиля равна 55 км/ч, т. е. a_1= 55 км/ч, а температура кофе равна a_2=70 °С.

Тогда агрегирование первого нечеткого высказывания с использованием операции нечеткой конъюнкции (6.2) дает в результате число b_1''= 0.67 (приближенное значение), которое означает его степень истинности и получается как минимальное из значений 0.67 и 0.8 (рис. 7.5, *а*). Агрегирование второго нечеткого высказывания с использованием операции нечеткой дизъюнкции (6.6) дает в результате число b_1''= 0.8, которое означает его степень истинности и получается как максимальное из значений 0.67 и 0.8 (рис. 7.5, *б*).

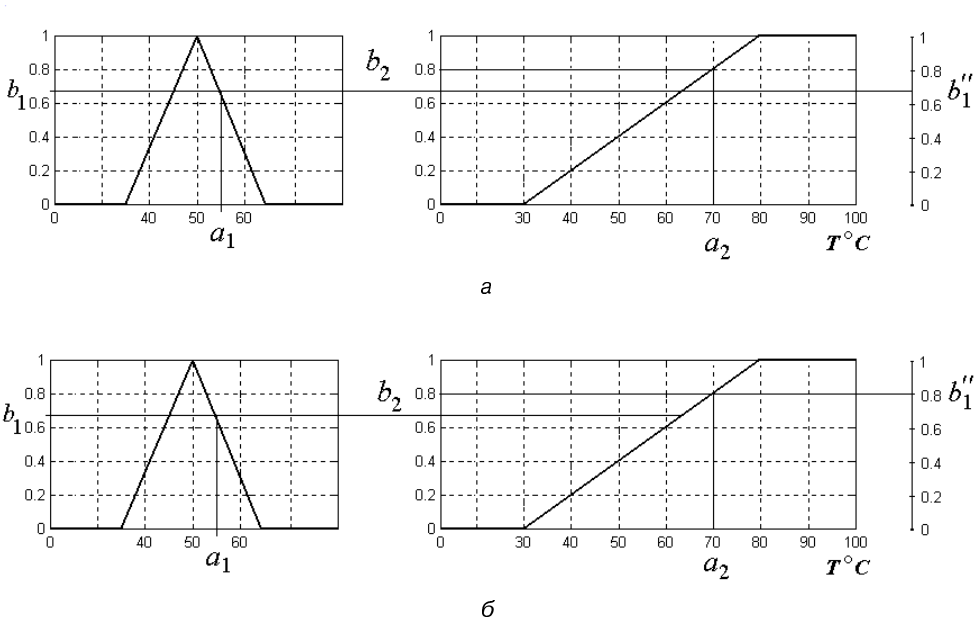


Рис. 7.5. Примеры агрегирования подусловий для двух нечетких высказываний "*скорость автомобиля средняя*" И "*температура кофе высокая*" (*а*) и "*скорость автомобиля средняя*" ИЛИ "*температура кофе высокая*" (*б*)

Активизация (Activation)

Активизация в системах нечеткого вывода представляет собой процедуру или процесс нахождения степени истинности каждого из подзаключений правил нечетких продукций. Активизация в общем случае во многом аналогична компо-

зиции нечетких отношений, но не тождественна ей. Поскольку в системах нечеткого вывода используются лингвистические переменные, то формулы (6.22)—(6.28) для нечеткой композиции теряют свое значение. В действительности при формировании базы правил системы нечеткого вывода задаются весовые коэффициенты F_i для каждого правила (по умолчанию предполагается, если весовой коэффициент не задан явно, то его значение равно 1).

Формально процедура активизации выполняется следующим образом. До начала этого этапа предполагаются известными значения истинности всех условий системы нечеткого вывода, т. е. множество значений $B''=\{b_1'', b_2'',\ldots, b_n''\}$ и значения весовых коэффициентов F_i для каждого правила. Далее рассматривается каждое из заключений правил системы нечеткого вывода. Если заключение правила представляет собой нечеткое высказывание вида 1 или 2, то степень его истинности равна алгебраическому произведению соответствующего значения b_i'' на весовой коэффициент F_i.

Если же заключение состоит из нескольких подзаключений вида (7.3), причем лингвистические переменные в подзаключениях попарно не равны друг другу, то степень истинности каждого из подзаключений равна алгебраическому произведению соответствующего значения b_i'' на весовой коэффициент F_i. Таким образом, находятся все значения c_k степеней истинности подзаключений для каждого из правил R_k, входящих в рассматриваемую базу правил P системы нечеткого вывода. Это множество значений обозначим через $C=\{c_1, c_2,\ldots, c_q\}$, где q — общее количество подзаключений в базе правил.

Примечание

При этом не исключается случай, когда весовой коэффициент F_i может быть задан индивидуально для отдельных подзаключений. При этом процедура активизации остается прежней.

После нахождения множества $C=\{c_1, c_2,\ldots, c_q\}$ определяются функции принадлежности каждого из подзаключений для рассматриваемых выходных лингвистических переменных. Для этой цели можно использовать один из методов, являющихся модификацией того или иного метода нечеткой композиции:

- min-активизация:

$$\mu'(y) = \min\{c_i, \mu(y)\}; \tag{7.6}$$

- prod-активизация:

$$\mu'(y) = c_i \cdot \mu(y); \tag{7.7}$$

- average-активизация:

$$\mu'(y) = 0.5\cdot(c_i+\mu(y)), \tag{7.8}$$

где $\mu(y)$ — функция принадлежности терма, который является значением некоторой выходной переменной ω_j, заданной на универсуме $\boldsymbol{Y}$.

Этап активизации считается законченным, когда для каждой из выходных лингвистических переменных, входящих в отдельные подзаключения правил нечет-

ких продукций, будут определены функции принадлежности нечетких множеств их значений, т. е. совокупность нечетких множеств: C_1, C_2,..., C_q, где q — общее количество подзаключений в базе правил системы нечеткого вывода.

Примечание

Следует отметить, что кроме методов (7.6)—(7.8) для выполнения активизации могут быть предложены и другие способы, основанные на модификации различных операций нечеткой композиции. Здесь приводятся лишь те из них, которые нашли наибольшее практическое применение в задачах нечеткого моделирования.

П р и м е р 7.5. Для иллюстрации выполнения этого этапа рассмотрим пример процесса активизации заключения в следующем правиле нечеткой продукции (это правило вряд ли имеет целевое применение и используется формальным образом):

ЕСЛИ "*скорость автомобиля средняя*" ТО "*кофе горячий*"

Входной лингвистической переменной в этом правиле является β_1 — скорость движения автомобиля, а выходной переменной является β_2 — температура кофе. Предположим, что текущая скорость автомобиля равна 55 км/ч, т. е. a_1= 55 км/ч.

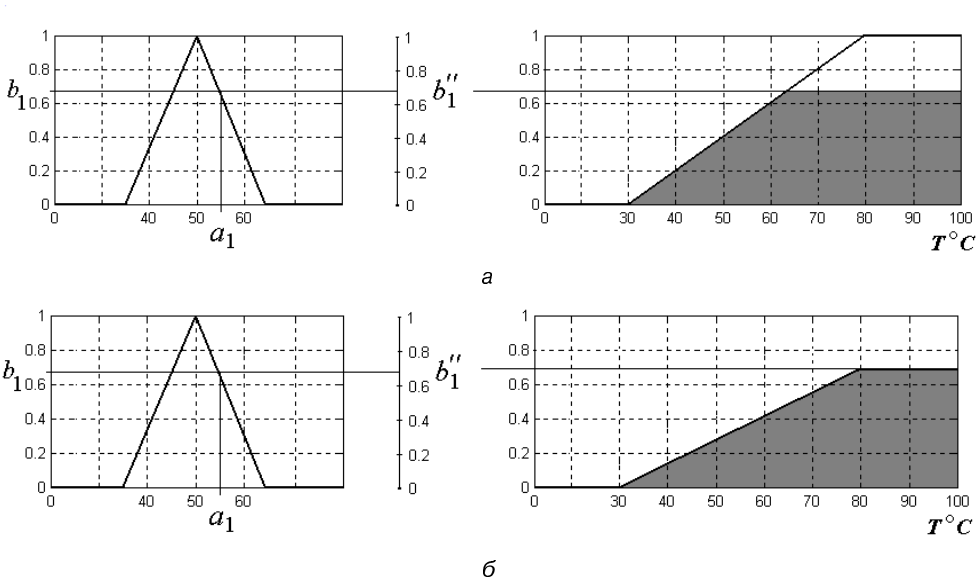


Рис. 7.6. Пример активизации заключения для правила нечеткой продукции

Поскольку агрегирование условия этого правила дает в результате b_1''= 0.67, а весовой коэффициент равен 1 (по умолчанию), то значение 0.67 будет использоваться в качестве c_1 для получения результата активизации. Результат, полученный методом min-активизации (7.6), изображен на рис. 7.6, *а* более темным цве-

том, а результат, полученный методом prod-активизации (7.7), изображен на рис. 7.6, *б* более темным цветом. Следует помнить, что в этом примере в отличие от предыдущего "*температура кофе*" — выходная лингвистическая переменная.

Аккумуляция (Accumulation)

Аккумуляция или аккумулирование в системах нечеткого вывода представляет собой процедуру или процесс нахождения функции принадлежности для каждой из выходных лингвистических переменных множества $W=\{\omega_1, \omega_2,..., \omega_s\}$.

Цель аккумуляции заключается в том, чтобы объединить или аккумулировать все степени истинности заключений (подзаключений) для получения функции принадлежности каждой из выходных переменных. Причина необходимости выполнения этого этапа состоит в том, что подзаключения, относящиеся к одной и той же выходной лингвистической переменной, принадлежат различным правилам системы нечеткого вывода.

Формально процедура аккумуляции выполняется следующим образом. До начала этого этапа предполагаются известными значения истинности всех подзаключений для каждого из правил R_k, входящих в рассматриваемую базу правил P системы нечеткого вывода, в форме совокупности нечетких множеств: $C_1, C_2,..., C_q$, где q — общее количество подзаключений в базе правил. Далее последовательно рассматривается каждая из выходных лингвистических переменных $\omega_j \in W$ и относящиеся к ней нечеткие множества: $C_{j1}, C_{j2},..., C_{jq}$. Результат аккумуляции для выходной лингвистической переменной ω_j определяется как объединение нечетких множеств $C_{j1}, C_{j2},..., C_{jq}$ по одной из формул (3.4), (3.20), (3.32), (3.34), (3.35).

Этап аккумуляции считается законченным, когда для каждой из выходных лингвистических переменных будут определены итоговые функции принадлежности нечетких множеств их значений, т. е. совокупность нечетких множеств: $C_1', C_2',..., C_s'$, где s — общее количество выходных лингвистических переменных в базе правил системы нечеткого вывода.

Примечание

Следует отметить, что кроме методов (3.4), (3.20), (3.32), (3.34), (3.35) для выполнения аккумуляции могут быть предложены и другие способы, основанные на модификации различных операций объединения нечетких множеств. Выбор того или иного метода определяется спецификой задачи нечеткого моделирования.

П р и м е р 7.6. Для иллюстрации выполнения этого этапа рассмотрим пример процесса аккумуляции заключений для трех нечетких множеств C_{11}, C_{12}, C_{13}, полученных в результате выполнения процедуры активизации для выходной лингвистической переменной "*скорость движения автомобиля*" в некоторой системе нечеткого вывода. Предположим, что функции принадлежности этих нечетких множеств изображены на рис. 7.7, *а*, *б*, *в* соответственно.

Аккумуляция этих функций принадлежности методом max-объединения нечетких множеств C_{11}, C_{12}, C_{13} по формуле (3.4) позволяет получить в результате функцию принадлежности выходной лингвистической переменной "*скорость*

движения автомобиля", которая представлена на рис. 7.7, *г*. Эта функция принадлежности соответствует нечеткому множеству C_1', если принять, что рассматриваемая выходная лингвистическая переменная есть ω_1.

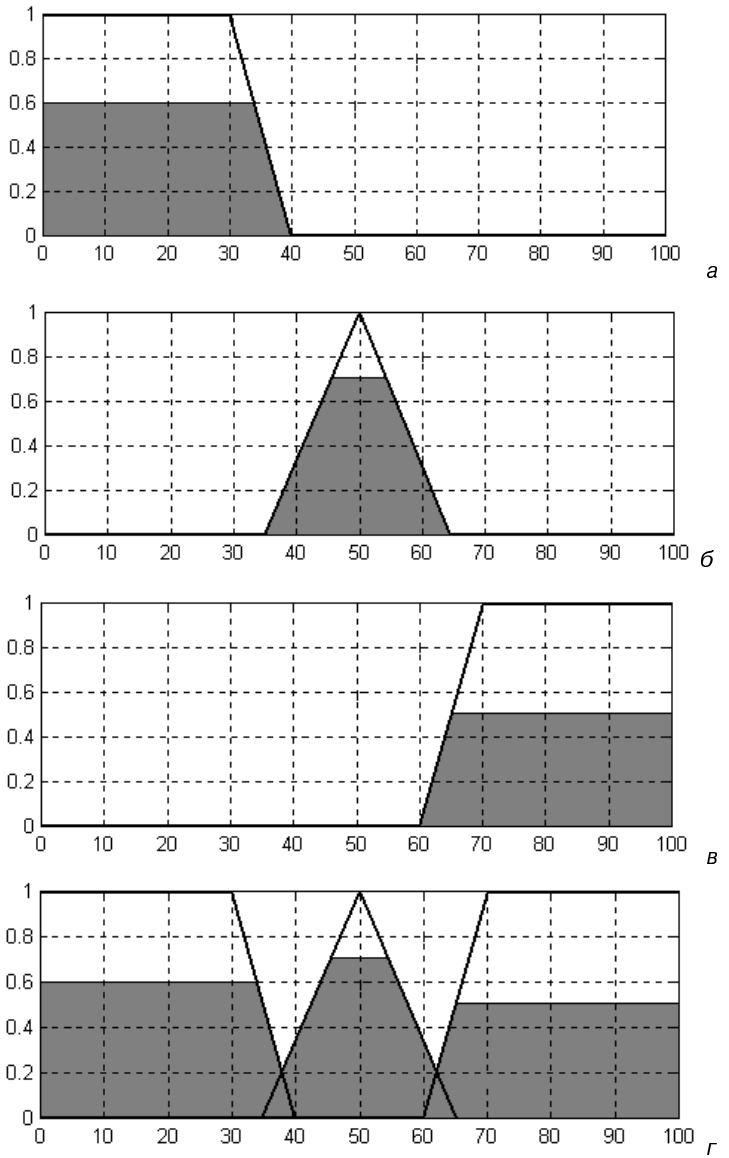


Рис. 7.7. Пример аккумуляции заключения
для выходной лингвистической переменной "*скорость движения автомобиля*"

Дефаззификация (Defuzzification)

Дефаззификация в системах нечеткого вывода представляет собой процедуру или процесс нахождения обычного (не нечеткого) значения для каждой из выходных лингвистических переменных множества $W=\{\omega_1, \omega_2,..., \omega_s\}$.

Цель дефаззификации заключается в том, чтобы, используя результаты аккумуляции всех выходных лингвистических переменных, получить обычное количественное значение (crisp value) каждой из выходных переменных, которое может быть использовано специальными устройствами, внешними по отношению к системе нечеткого вывода.

Действительно, применяемые в современных системах управления устройства и механизмы способны воспринимать традиционные команды в форме количественных значений соответствующих управляющих переменных. Именно по этой причине необходимо преобразовать нечеткие множества в некоторые конкретные значения переменных. Поэтому дефаззификацию называют также приведением к четкости.

Формально процедура дефаззификации выполняется следующим образом. До начала этого этапа предполагаются известными функции принадлежности всех выходных лингвистических переменных в форме нечетких множеств: C_1', C_2',..., C_s', где s — общее количество выходных лингвистических переменных в базе правил системы нечеткого вывода. Далее последовательно рассматривается каждая из выходных лингвистических переменных $\omega_j \in W$ и относящееся к ней нечеткое множество C_j'. Результат дефаззификации для выходной лингвистической переменной ω_j определяется в виде количественного значения $y_j \in \mathbb{R}$, получаемого по одной из рассматриваемых ниже формул.

Этап дефаззификации считается законченным, когда для каждой из выходных лингвистических переменных будут определены итоговые количественные значения в форме некоторого действительного числа, т. е. в виде y_1, y_2,..., y_s, где s — общее количество выходных лингвистических переменных в базе правил системы нечеткого вывода.

Для выполнения численных расчетов на этапе дефаззификации могут быть использованы следующие формулы, получившие название *методов дефаззификации*.

Метод центра тяжести

Центр тяжести (CoG, COG, Centre of Gravity) или центроид площади рассчитывается по формуле:

$$y = \frac{\int_{Min}^{Max} x \cdot \mu(x)\, dx}{\int_{Min}^{Max} \mu(x)\, dx}, \qquad (7.9)$$

В формуле используются следующие обозначения: y — результат дефаззификации; x — переменная, соответствующая выходной лингвистической переменной ω; $\mu(x)$ — функция принадлежности нечеткого множества, соответствующего выходной переменной ω после этапа аккумуляции; *Min* и *Max* — левая и правая точки интервала носителя нечеткого множества рассматриваемой выходной переменной ω.

При дефаззификации методом центра тяжести обычное (не нечеткое) значение выходной переменной равно абсциссе центра тяжести площади, ограниченной графиком кривой функции принадлежности соответствующей выходной переменной.

Пример дефаззификации методом центра тяжести функции принадлежности выходной лингвистической переменной "*скорость движения автомобиля*" изображен на рис. 7.8. В этом случае y_1=40 км/ч (приближенное значение).

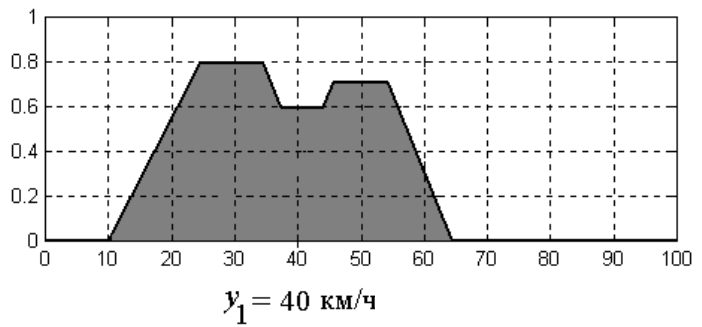


Рис. 7.8. Пример дефаззификации выходной лингвистической переменной "*скорость движения автомобиля*" методом центра тяжести

Метод центра тяжести для одноточечных множеств

Центр тяжести (COGS, Centre of Gravity for Singletons) для одноточечных множеств рассчитывается по формуле:

$$y = \frac{\sum_{i=1}^{n} x_i \cdot \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}, \qquad (7.10)$$

где n — число одноточечных (одноэлементных) нечетких множеств, каждое из которых характеризует единственное значение рассматриваемой выходной лингвистической переменной.

Пример дефаззификации методом центра тяжести для одноточечных множеств функции принадлежности выходной лингвистической переменной "*скорость движения автомобиля*" изображен на рис. 7.9. В этом случае y_1=41 км/ч (приближенное значение).

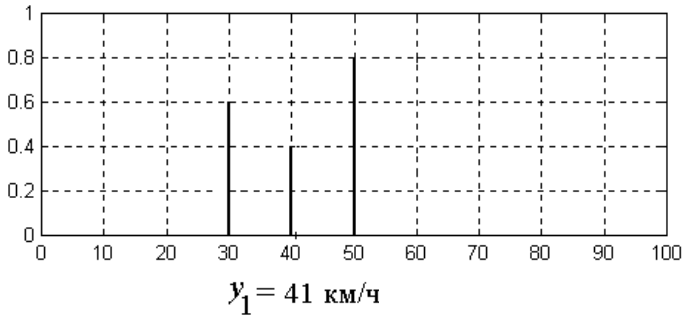


Рис. 7.9. Пример дефаззификации выходной лингвистической переменной "*скорость движения автомобиля*" методом центра тяжести для одноточечных множеств

Метод центра площади

Центр площади (CoA, COA, Centre of Area, Bisector of Area) равен $y = u$, где значение u определяется из уравнения:

$$\int_{Min}^{u} \mu(x)\, dx = \int_{u}^{Max} \mu(x)\, dx\,, \qquad (7.11)$$

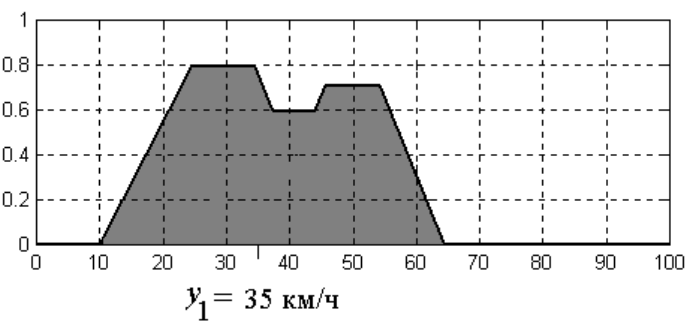


Рис. 7.10. Пример дефаззификации выходной лингвистической переменной "*скорость движения автомобиля*" методом центра площади

Другими словами, центр площади равен абсциссе, которая делит площадь, ограниченную графиком кривой функции принадлежности соответствующей выходной переменной, на две равные части. Иногда центр площади называют биссектрисой площади. Этот метод не может быть использован в случае одноточечных множеств.

Пример дефаззификации методом центра площади функции принадлежности выходной лингвистической переменной "*скорость движения автомобиля*" изображен на рис. 7.10. В этом случае y_1=35 км/ч (приближенное значение).

Метод левого модального значения

Левое модальное значение (LM, Left Most Maximum) рассчитывается по формуле:

$$y = \min\{x_m\}, \tag{7.12}$$

где x_m — модальное значение (мода) нечеткого множества, соответствующего выходной переменной ω после аккумуляции, рассчитываемое по формуле (2.4).

Другими словами, значение выходной переменной определяется как мода нечеткого множества для соответствующей выходной переменной или наименьшая из мод (самая левая), если нечеткое множество имеет несколько модальных значений.

Пример дефаззификации методом левого модального значения функции принадлежности выходной лингвистической переменной "*скорость движения автомобиля*" изображен на рис. 7.11. В этом случае y_1= 24 км/ч (приближенное значение).

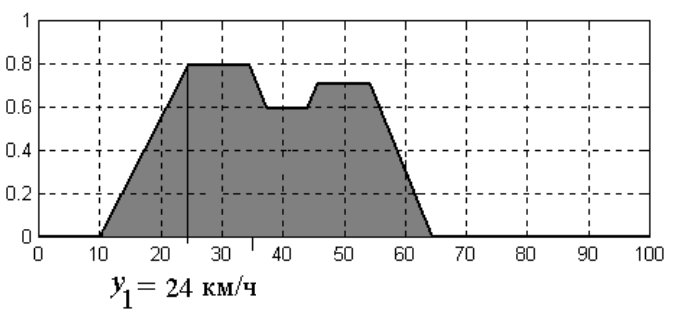


Рис. 7.11. Пример дефаззификации выходной лингвистической переменной "*скорость движения автомобиля*" методом левого модального значения

Метод правого модального значения

Правое модальное значение (RM, Right Most Maximum) рассчитывается по формуле:

$$y = \max\{x_m\}, \tag{7.13}$$

где x_m — модальное значение (мода) нечеткого множества для выходной переменной ω после аккумуляции, рассчитываемое по формуле (2.4).

В этом случае значение выходной переменной также определяется как мода нечеткого множества для соответствующей выходной переменной или наибольшая из мод (самая правая), если нечеткое множество имеет несколько модальных значений.

Примечание

Нетрудно заметить, что в случае строго унимодального нечеткого множества левое и правое модальные значения совпадают, а расчеты по формулам (7.12) и (7.13) приводят к одинаковому результату.

Пример дефаззификации функции принадлежности выходной лингвистической переменной "*скорость движения автомобиля*" изображен на рис. 7.12. В этом случае y_1=54 км/ч (приближенное значение) методом правого модального значения изображен на рис. 7.12.

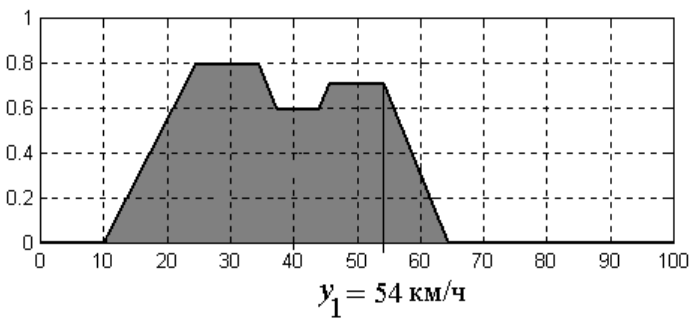


Рис. 7.12. Пример дефаззификации выходной лингвистической переменной "*скорость движения автомобиля*" методом правого модального значения

Примечание

Следует отметить, что кроме методов (7.9)—(7.13) для выполнения дефаззификации могут быть предложены и другие расчетные формулы. Здесь приводятся лишь те из них, которые нашли наибольшее практическое применение в задачах нечеткого моделирования и стали в некотором смысле традиционными для систем нечеткого вывода.

7.3. Основные алгоритмы нечеткого вывода

Рассмотренные выше этапы нечеткого вывода могут быть реализованы неоднозначным образом, поскольку включают в себя отдельные параметры, которые должны быть фиксированы или специфицированы. Тем самым выбор конкрет-

ных вариантов параметров каждого из этапов определяет некоторый алгоритм, который в полном объеме реализует нечеткий вывод в системах правил нечетких продукций. К настоящему времени предложено несколько алгоритмов нечеткого вывода. Те из них, которые получили наибольшее применение в системах нечеткого вывода, рассматриваются ниже.

Алгоритм Мамдани (Mamdani)

Алгоритм Мамдани является одним из первых, который нашел применение в системах нечеткого вывода. Он был предложен в 1975 г. английским математиком Е. Мамдани (Ebrahim Mamdani) в качестве метода для управления паровым двигателем. По своей сути этот алгоритм порождает рассмотренные выше этапы, поскольку в наибольшей степени соответствует их параметрам.

Формально *алгоритм Мамдани* может быть определен следующим образом.

- Формирование базы правил систем нечеткого вывода. Особенности формирования базы правил совпадают с рассмотренными выше при описании данного этапа.
- Фаззификация входных переменных. Особенности фаззификации совпадают с рассмотренными выше при описании данного этапа.
- Агрегирование подусловий в нечетких правилах продукций. Для нахождения степени истинности условий каждого из правил нечетких продукций используются парные нечеткие логические операции. Те правила, степень истинности условий которых отлична от нуля, считаются *активными* и используются для дальнейших расчетов.
- Активизация подзаключений в нечетких правилах продукций. Осуществляется по формуле (7.6), при этом для сокращения времени вывода учитываются только активные правила нечетких продукций.
- Аккумуляция заключений нечетких правил продукций. Осуществляется по формуле (3.4) для объединения нечетких множеств, соответствующих термам подзаключений, относящихся к одним и тем же выходным лингвистическим переменным.
- Дефаззификация выходных переменных. Традиционно используется метод центра тяжести в форме (7.9)—(7.10) или метод центра площади (7.11).

Алгоритм Цукамото (Tsukamoto)

Формально *алгоритм Цукамото* может быть определен следующим образом.

- Формирование базы правил систем нечеткого вывода. Особенности формирования базы правил совпадают с рассмотренными выше при описании данного этапа.
- Фаззификация входных переменных. Особенности фаззификации совпадают с рассмотренными выше при описании данного этапа.

- Агрегирование подусловий в нечетких правилах продукций. Для нахождения степени истинности условий всех правил нечетких продукций используются парные нечеткие логические операции. Те правила, степень истинности условий которых отлична от нуля, считаются *активными* и используются для дальнейших расчетов.
- Активизация подзаключений в нечетких правилах продукций. Осуществляется аналогично алгоритму Мамдани по формуле (7.6), после чего находятся обычные (не нечеткие) значения всех выходных лингвистических переменных в каждом из подзаключений активных правил нечетких продукций. В этом случае значение выходной лингвистической переменной w_j в каждом из подзаключений находится как решение уравнения:

$$c_i = \mu(w_j) \qquad (\forall i \in \{1, 2, \ldots, q\}), \tag{7.14}$$

  где q — общее количество подзаключений в базе правил.
- Аккумуляция заключений нечетких правил продукций. Фактически отсутствует, поскольку расчеты осуществляются с обычными действительными числами w_j.
- Дефаззификация выходных переменных. Используется модифицированный вариант в форме метода центра тяжести для одноточечных множеств:

$$y = \frac{\sum_{i=1}^{n} c_i \cdot w_i}{\sum_{i=1}^{n} c_i}, \tag{7.15}$$

  где n — общее количество активных правил нечетких продукций, в подзаключениях которых присутствует выходная лингвистическая переменная ω_j.

Алгоритм Ларсена (Larsen)

Формально *алгоритм Ларсена* может быть определен следующим образом.

- Формирование базы правил систем нечеткого вывода. Особенности формирования базы правил совпадают с рассмотренными выше при описании данного этапа.
- Фаззификация входных переменных. Особенности фаззификации также совпадают с рассмотренными выше при описании данного этапа.
- Агрегирование подусловий в нечетких правилах продукций. Используются парные нечеткие логические операции для нахождения степени истинности условий всех правил нечетких продукций (как правило, max-дизъюнкция и min-конъюнкция). Те правила, степень истинности условий которых отлична от нуля, считаются *активными* и используются для дальнейших расчетов.

- Активизация подзаключений в нечетких правилах продукций. Осуществляется использованием формулы (7.7), посредством чего находится совокупность нечетких множеств: $C_1, C_2,\ldots, C_q$, где q — общее количество подзаключений в базе правил.
- Аккумуляция заключений нечетких правил продукций. Осуществляется по формуле (3.4) для объединения нечетких множеств, соответствующих термам подзаключений, относящихся к одним и тем же выходным лингвистическим переменным.
- Дефаззификация выходных переменных. Может использоваться любой из рассмотренных выше методов дефаззификации.

Алгоритм Сугено (Sugeno)

Формально *алгоритм Сугено*, предложенный Сугено и Такаги, может быть определен следующим образом.

- Формирование базы правил систем нечеткого вывода. В базе правил используются только правила нечетких продукций в форме:

  ПРАВИЛО <#>: ЕСЛИ "β_1 *есть* α'" И "β_2 *есть* α''" ТО "$w=\varepsilon_1 \cdot a_1+\varepsilon_2 \cdot a_2$". (7.16)

  Здесь ε_1, ε_2 — некоторые весовые коэффициенты. При этом значение выходной переменной w в заключении определяется как некоторое действительное число.
- Фаззификация входных переменных. Особенности фаззификации совпадают с рассмотренными выше при описании данного этапа.
- Агрегирование подусловий в нечетких правилах продукций. Для нахождения степени истинности условий всех правил нечетких продукций, как правило, используется логическая операция min-конъюнкции. Те правила, степень истинности условий которых отлична от нуля, считаются *активными* и используются для дальнейших расчетов.
- Активизация подзаключений в нечетких правилах продукций. Во-первых, с использованием метода (7.6) находятся значения степеней истинности всех заключений правил нечетких продукций. Во-вторых, осуществляется расчет обычных (не нечетких) значений выходных переменных каждого правила. Это выполняется с использованием формулы для заключения (7.16), в которую вместо a_1 и a_2 подставляются значения входных переменных до этапа фаззификации. Тем самым определяются множество значений $C=\{c_1, c_2,\ldots, c_n\}$ и множество значений выходных переменных $W=\{w_1, w_2,\ldots, w_n\}$, где n — общее количество правил в базе правил.
- Аккумуляция заключений нечетких правил продукций. Фактически отсутствует, поскольку расчеты осуществляются с обычными действительными числами w_j.
- Дефаззификация выходных переменных. Используется модифицированный вариант в форме метода центра тяжести для одноточечных множеств (7.15).

Упрощенный алгоритм нечеткого вывода

Формально *упрощенный алгоритм* может быть определен следующим образом.

- Формирование базы правил систем нечеткого вывода. В базе правил используются только правила нечетких продукций в форме:

  ПРАВИЛО <#>: ЕСЛИ "β_1 *есть* α'" И "β_2 *есть* α''" ТО "$w=\varepsilon$". (7.17)

  Здесь ε — некоторое действительное число.

- Фаззификация входных переменных. Особенности фаззификации совпадают с рассмотренными выше при описании данного этапа.

- Агрегирование подусловий в нечетких правилах продукций. Для нахождения степени истинности условий всех правил нечетких продукций, как правило, используется логическая операция — min-конъюнкция. Те правила, степень истинности условий которых отлична от нуля, считаются *активными* и используются для дальнейших расчетов.

- Активизация подзаключений в нечетких правилах продукций. Осуществляется с использованием метода (7.6), посредством чего находятся значения степеней истинности всех заключений правил нечетких продукций $C=\{c_1, c_2,\ldots, c_n\}$, где n — общее количество правил в базе правил.

- Аккумуляция заключений нечетких правил продукций. Фактически отсутствует, поскольку расчеты осуществляются с обычными действительными числами c_j.

- Дефаззификация выходных переменных. Используется модифицированный вариант в форме метода центра тяжести для одноточечных множеств (7.15).

При решении практических задач нечеткого моделирования могут одновременно использоваться несколько алгоритмов нечеткого вывода с целью получения наиболее адекватных результатов. Ниже рассматриваются примеры применения некоторых из этих алгоритмов в задачах нечеткого управления.

7.4. Примеры использования систем нечеткого вывода в задачах управления

Одним из основных направлений практического использования систем нечеткого вывода является решение задач управления различными объектами или процессами. В этом случае построение нечеткой модели основывается на формальном представлении характеристик исследуемой системы в терминах лингвистических переменных. Поскольку кроме алгоритма управления, основными понятиями систем управления являются входные и выходные переменные, то именно они рассматриваются как лингвистические переменные при формировании базы правил в системах нечеткого вывода.

В общем случае цель управления заключается в том, чтобы на основе анализа текущего состояния объекта управления определить значения управляющих пе-

ременных, реализация которых позволяет обеспечить желаемое поведение или состояние объекта управления. В настоящее время для решения соответствующих задач используется общая теория управления, в рамках которой разработаны различные алгоритмы нахождения оптимальных законов управления объектами различной физической природы.

Не вдаваясь в детальное обсуждение концепций классической теории управления, рассмотрим лишь основные определения, необходимые для понимания особенностей и места систем нечеткого вывода при решении задач управления.

Базовая архитектура или модель классической теории управления основывается на представлении объекта и процесса управления в форме некоторых систем (рис. 7.13). При этом объект управления характеризуется некоторым конечным множеством входных параметров и конечным множеством выходных параметров. На вход системы управления поступают некоторые входные переменные, которые формируются с помощью конечного множества датчиков. На выходе системы управления с использованием некоторого алгоритма управления формируется множество значений выходных переменных, которые еще называют управляющими переменными или переменными процесса управления. Значения этих выходных переменных поступают на вход объекта управления и, комбинируясь со значениями входных параметров объекта управления, изменяют его поведение в желаемом направлении.

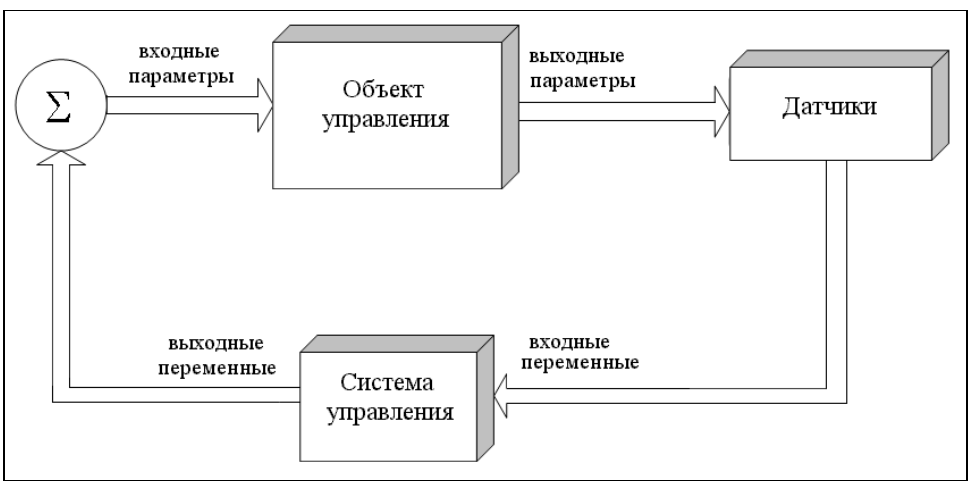


Рис. 7.13. Архитектура компонентов процесса управления с обратной связью

Рассмотренная архитектура называется *процессом управления с обратной связью*, а используемые для управления техническими объектами системы управления — *контроллерами*.

Наиболее типичным примером рассмотренной модели управления является так называемый *интегрально-дифференцирующий контроллер* или PID-контроллер

(proportional-integral-derivative controller). Алгоритм его управления основан на сравнении выходных параметров объекта управления с некоторыми заданными параметрами и определении величины расхождения между ними или ошибки. После этого рассчитываются величины выходных переменных в форме аддитивной суммы величины этой ошибки, значения интеграла и производной по времени в течение некоторого промежутка времени.

Один из недостатков PID-контроллеров заключается в предположении о линейном характере зависимости входных и выходных переменных процесса управления, что существенно снижает адекватность этой модели при решении отдельных практических задач. Другой недостаток модели связан со сложностью выполнения соответствующих расчетов, что может привести к недопустимым задержкам в реализации управляющих воздействий при оперативном управлении объектами с высокой динамикой изменения выходных параметров.

Архитектура или модель нечеткого управления основана на замене классической системы управления системой нечеткого управления, в качестве которой используются системы нечеткого вывода. В этом случае модель нечеткого управления (рис. 7.14) строится с учетом необходимости реализации всех этапов нечеткого вывода, а сам процесс вывода реализуется на основе одного из рассмотренных выше алгоритмов нечеткого вывода.

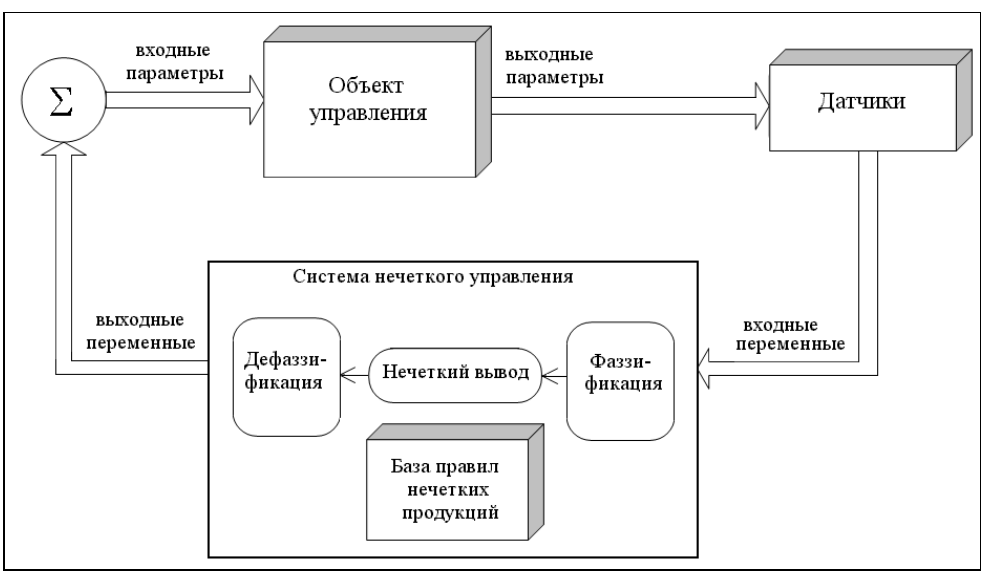


Рис. 7.14. Архитектура компонентов процесса нечеткого управления

Далее рассматриваются особенности построения некоторых моделей систем нечеткого управления с целью решения практических задач по управлению конкретными объектами.

Нечеткая модель управления смесителем воды при принятии душа

В качестве первого примера использования систем нечеткого вывода в задачах управления рассматривается задача управления смесителем воды при принятии душа. Эта задача является одной из наиболее простых, которая может быть решена методами нечеткого моделирования. Для определенности предположим, что в качестве алгоритма нечеткого вывода будет использоваться алгоритм Мамдани.

Содержательная постановка задачи

При принятии душа на вход смесителя подается холодная и горячая вода по соответствующим магистральным трубопроводам. Наиболее комфортные условия для душа создаются при наличии на выходе смесителя теплой воды постоянной температуры. Поскольку во время принятия душа может наблюдаться неравномерный расход воды, температура воды на выходе смесителя будет колебаться, приводя к необходимости ручного изменения подачи холодной или горячей воды. Задача состоит в том, чтобы сделать регулировку температуры воды автоматической, обеспечивая постоянную температуру воды на выходе смесителя (рис. 7.15).

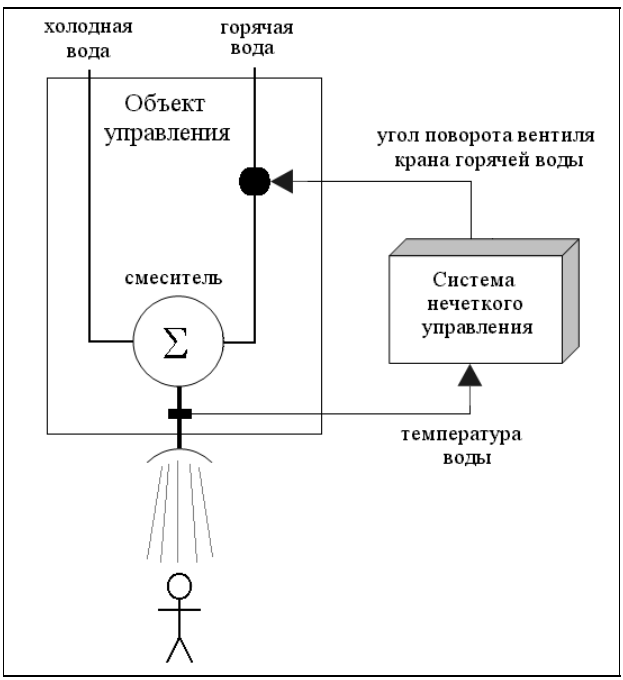


Рис. 7.15. Иллюстрация модели нечеткого управления смесителем воды при принятии душа

Опыт принятия душа позволяет сформулировать несколько эвристических правил, которые мы применяем в случае регулирования температуры воды на выходе смесителя:

1. Если вода горячая, то следует повернуть вентиль крана горячей воды на большой угол вправо.
2. Если вода не очень горячая, то следует повернуть вентиль крана горячей воды на небольшой угол вправо.
3. Если вода теплая, то оставить вентиль крана горячей воды без воздействия.
4. Если вода прохладная, то следует повернуть вентиль крана горячей воды на небольшой угол влево.
5. Если вода холодная, то следует повернуть вентиль крана горячей воды на большой угол влево.

Эта информация будет использоваться при построении базы правил системы нечеткого вывода, которая позволяет реализовать данную модель нечеткого управления.

Построение базы нечетких лингвистических правил

Для формирования базы правил систем нечеткого вывода необходимо предварительно определить входные и выходные лингвистические переменные. Очевидно, в качестве входной лингвистической переменной следует использовать температуру воды на выходе смесителя или формально: β_1 — "*температура воды*". В качестве выходной лингвистической переменной будем использовать угол поворота вентиля крана горячей воды или формально: β_2 — "*угол поворота*".

В этом случае система нечеткого вывода будет содержать 5 правил нечетких продукций следующего вида:

ПРАВИЛО_1: ЕСЛИ "вода горячая" ТО "повернуть вентиль крана горячей воды на большой угол вправо"

ПРАВИЛО_2: ЕСЛИ "вода не очень горячая" ТО "повернуть вентиль крана горячей воды на небольшой угол вправо"

ПРАВИЛО_3: ЕСЛИ "вода теплая" ТО "оставить угол поворота крана горячей воды без изменения"

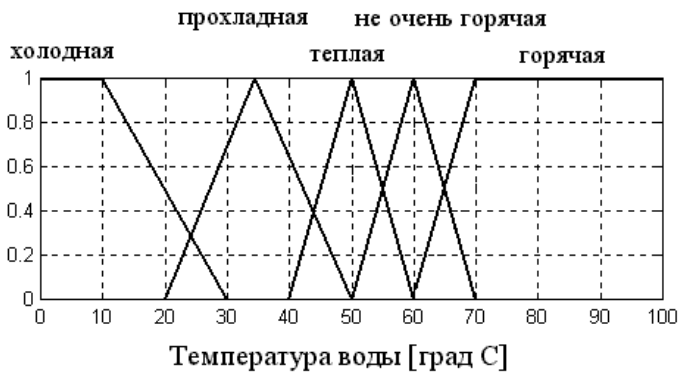
ПРАВИЛО_4: ЕСЛИ "вода прохладная" ТО "повернуть вентиль крана горячей воды на небольшой угол влево"

ПРАВИЛО_5: ЕСЛИ "вода холодная" ТО "повернуть вентиль крана горячей воды на большой угол влево"

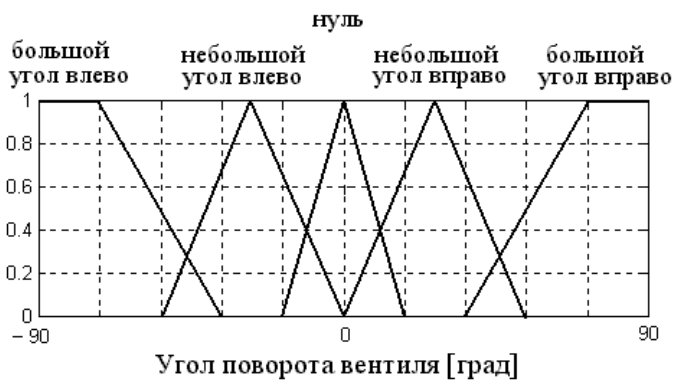
Фаззификация входных переменных

В качестве терм-множества первой лингвистической переменной будем использовать множество T_1={"*горячая*", "*не очень горячая*", "*теплая*", "*прохладная*", "*холодная*"} с функциями принадлежности, изображенными на рис. 7.16, *а*.

В качестве терм-множества второй лингвистической переменной будем использовать множество T_2={"*большой угол вправо*", "*небольшой угол вправо*", "*нуль*", "*небольшой угол влево*", "*большой угол влево*"} с кусочно-линейными функциями принадлежности, изображенными на рис. 7.16, *б*.



а



б

Рис. 7.16. Графики функций принадлежности для термов лингвистической переменной "*Температура воды*" (*а*) и лингвистической переменной "*Угол поворота вентиля крана*" (*б*)

При этом температура воды измеряется в градусах Цельсия, а угол поворота — в угловых градусах. В последнем случае поворот вправо означает положительное направление отсчета, а поворот влево — отрицательное.

Используя в качестве алгоритма вывода алгоритм Мамдани, рассмотрим пример его выполнения для случая, когда текущая температура воды на выходе смесителя равна 55 °С. В этом случае фаззификация входной лингвистической переменной приводит к значениям степеней истинности 0.5 для правил нечетких продукций с номерами 2 и 3. Эти правила считаются активными и используются в текущем процессе нечеткого вывода.

Поскольку все условия в правилах 1—5 заданы в форме нечетких лингвистических высказываний первого вида, этап их агрегирования тривиален и оставляет степени истинности 0.5 без изменения.

Следующим этапом нечеткого вывода является активизация заключений в нечетких правилах продукций. Поскольку все заключения правил 1—5 заданы в форме нечетких лингвистических высказываний первого вида, а весовые коэффициенты правил по умолчанию равны 1, то активизация правил 2 и 3 приводит к нечетким множествам, функции принадлежности которых изображены на рис. 7.17, *а*.

Аккумулирование заключений нечетких правил продукций с использованием операции max-дизъюнкции для правил 2 и 3 приводит в результате к нечеткому множеству, функция принадлежности которого изображена на рис. 7.17, *б*.

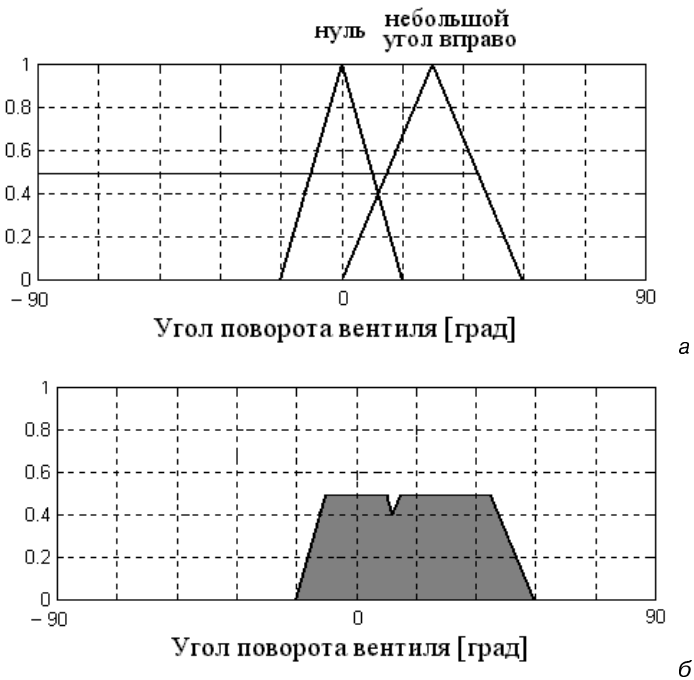


Рис. 7.17. Графики функции принадлежности для двух термов выходной лингвистической переменной *"Угол поворота вентиля крана"* (*а*) и функции принадлежности после аккумуляции (*б*)

Дефаззификация выходной лингвистической переменной "*Угол поворота вентиля крана*" методом центра тяжести для значений функции принадлежности, изображенной на рис. 7.17, приводит к значению управляющей переменной, равному повороту вентиля крана вправо на 16° (приближенное значение). Это значение и

является результатом решения задачи нечеткого вывода для текущего значения входной лингвистической переменной "*Температура воды*".

Для реализации этого алгоритма нечеткого управления необходимо организовать периодическое измерение температуры воды на выходе смесителя в некоторые дискретные моменты времени. При этом, чем меньше интервал измерения этой температуры, тем выше оказывается точность регулирования температуры воды.

Что касается реализации собственно процедуры нечеткого управления, то для этой цели необходимо использовать соответствующие программные или аппаратные средства, специально предназначенные для выполнения всех этапов нечеткого вывода. В частности, для этой цели могут быть применены специальные программируемые нечеткие контроллеры, которые обладают возможностью реализовывать программу нечеткого вывода, записанную, например, на языке нечеткого управления или языка FCL. Более подробно эти аспекты реализации алгоритмов нечеткого управления изложены в *главе 8.*

Нечеткая модель управления кондиционером воздуха в помещении

В качестве второго примера использования систем нечеткого вывода в задачах управления рассматривается задача управления кондиционером воздуха в помещении. Эта задача иллюстрирует процесс стабилизации температуры воздуха в помещении, в котором установлен бытовой кондиционер. Для определенности также предположим, что в качестве алгоритма нечеткого вывода будет использоваться алгоритм Мамдани.

Содержательная постановка задачи

В помещении установлен бытовой кондиционер, который позволяет охлаждать или нагревать воздух в этом помещении. Наиболее комфортные условия в помещении создаются при некоторой стабильной температуре воздуха. Поскольку температура окружающей среды вне помещения изменяется в течение суток и в большой степени зависит от внешних погодных условий, все это дестабилизирует температуру воздуха в помещении и приводит к необходимости ручной регулировки режима работы бытового кондиционера. Задача состоит в том, чтобы сделать регулировку кондиционера автоматической, обеспечивая постоянную температуру воздуха в помещении (рис. 7.18).

Опыт использования бытовых кондиционеров показывает, что процесс охлаждения или нагревания воздуха в помещении обладает некоторой инерционностью. А именно, после включения режима "холод" происходит нагнетание холодного воздуха, в связи с чем температура воздуха в помещении постепенно падает. При этом в момент отключения этого режима температура продолжает падать в течение небольшого, но конечного промежутка времени. Аналогичная картина наблюдается при включении и отключении режима "тепло". Предположим, что в рассматриваемой модели кондиционера включение режима "холод"

осуществляется поворотом регулятора влево, включение режима "тепло" осуществляется поворотом регулятора вправо относительно некоторой точки, в которой кондиционер выключен.

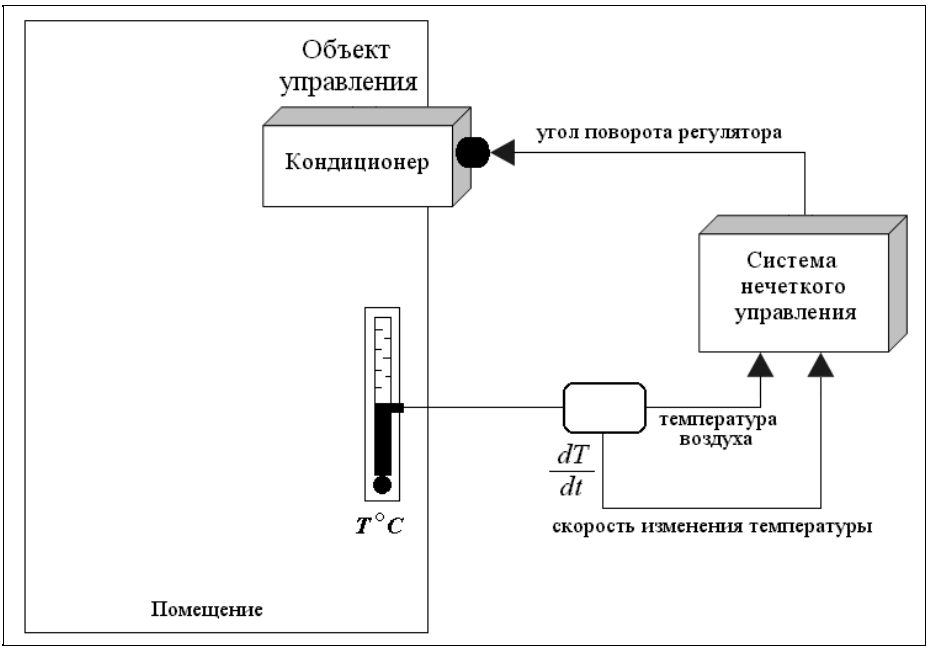


Рис. 7.18. Иллюстрация модели нечеткого управления кондиционером воздуха в помещении

Чтобы учесть эту особенность процесса управления кондиционером и исключить дополнительные затраты, связанные с частым включением и выключением указанных режимов, необходимо рассматривать в качестве выходного параметра не только температуру воздуха в помещении, но и скорость ее изменения. В этом случае эмпирические знания о рассматриваемой проблемной области могут быть представлены в форме эвристических правил, которые применяются в случае ручного регулирования температуры воздуха в помещении с кондиционером:

1. Если температура воздуха в помещении очень теплая, а скорость изменения температуры положительная, то следует включить режим "холод", повернув регулятор кондиционера на очень большой угол влево.
2. Если температура воздуха в помещении очень теплая, а скорость изменения температуры отрицательная, то следует включить режим "холод", повернув регулятор кондиционера на небольшой угол влево.
3. Если температура воздуха в помещении теплая, а скорость изменения температуры положительная, то следует включить режим "холод", повернув регулятор кондиционера на большой угол влево.

4. Если температура воздуха в помещении теплая, а скорость изменения температуры отрицательная, то кондиционер следует выключить.
5. Если температура воздуха в помещении очень холодная, а скорость изменения температуры отрицательная, то следует включить режим "тепло", повернув регулятор кондиционера на очень большой угол вправо.
6. Если температура воздуха в помещении очень холодная, а скорость изменения температуры положительная, то следует включить режим "тепло", повернув регулятор кондиционера на небольшой угол вправо.
7. Если температура воздуха в помещении холодная, а скорость изменения температуры отрицательная, то следует включить режим "тепло", повернув регулятор кондиционера на большой угол вправо.
8. Если температура воздуха в помещении холодная, а скорость изменения температуры положительная, то кондиционер следует выключить.
9. Если температура воздуха в помещении очень теплая, а скорость изменения температуры равна нулю, то следует включить режим "холод", повернув регулятор кондиционера на большой угол влево.
10. Если температура воздуха в помещении теплая, а скорость изменения температуры равна нулю, то следует включить режим "холод", повернув регулятор кондиционера на небольшой угол влево.
11. Если температура воздуха в помещении очень холодная, а скорость изменения температуры равна нулю, то следует включить режим "тепло", повернув регулятор кондиционера на большой угол вправо.
12. Если температура воздуха в помещении холодная, а скорость изменения температуры равна нулю, то следует включить режим "тепло", повернув регулятор кондиционера на небольшой угол вправо.
13. Если температура воздуха в помещении в пределах нормы, а скорость изменения температуры положительная, то следует включить режим "холод", повернув регулятор кондиционера на небольшой угол влево.
14. Если температура воздуха в помещении в пределах нормы, а скорость изменения температуры отрицательная, то следует включить режим "тепло", повернув регулятор кондиционера на небольшой угол вправо.
15. Если температура воздуха в помещении в пределах нормы, а скорость изменения температуры равна нулю, то кондиционер следует выключить.

Эта информация будет использоваться при построении базы правил системы нечеткого вывода, которая позволяет реализовать данную модель нечеткого управления.

Построение базы нечетких лингвистических правил

Для формирования базы правил систем нечеткого вывода необходимо предварительно определить входные и выходные лингвистические переменные. Очевидно, в качестве одной из входных лингвистических переменных следует использовать

температуру воздуха в помещении: β_1 — "*температура воздуха*", а в качестве второй входной лингвистической переменной β_2 — "*скорость изменения температура воздуха*". В качестве выходной лингвистической переменной будем использовать угол поворота регулятора включения режимов "холод" и "тепло" кондиционера: β_3 — "*угол поворота регулятора*". Для сокращения записи правил будем использовать рассмотренные символические обозначения (см. табл. 7.1), при этом модификатор ОЧЕНЬ преобразован к значению отдельного терма.

В этом случае система нечеткого вывода будет содержать 15 правил нечетких продукций следующего вида:

ПРАВИЛО_1: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* PS" ТО "β_3 *есть* NB"

ПРАВИЛО_2: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* NS" ТО "β_3 *есть* NS"

ПРАВИЛО_3: ЕСЛИ "β_1 *есть* PS" И "β_2 *есть* PS" ТО "β_3 *есть* NM"

ПРАВИЛО_4: ЕСЛИ "β_1 *есть* PS" И "β_2 *есть* NS" ТО "β_3 *есть* Z"

ПРАВИЛО_5: ЕСЛИ "β_1 *есть* NB" И "β_2 *есть* NS" ТО "β_3 *есть* PB"

ПРАВИЛО_6: ЕСЛИ "β_1 *есть* NB" И "β_2 *есть* PS" ТО "β_3 *есть* PS"

ПРАВИЛО_7: ЕСЛИ "β_1 *есть* NS" И "β_2 *есть* NS" ТО "β_3 *есть* PM"

ПРАВИЛО_8: ЕСЛИ "β_1 *есть* NS" И "β_2 *есть* PS" ТО "β_3 *есть* Z"

ПРАВИЛО_9: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* Z" ТО "β_3 *есть* NM"

ПРАВИЛО_10: ЕСЛИ "β_1 *есть* PS" И "β_2 *есть* Z" ТО "β_3 *есть* NS"

ПРАВИЛО_11: ЕСЛИ "β_1 *есть* NB" И "β_2 *есть* Z" ТО "β_3 *есть* PM"

ПРАВИЛО_12: ЕСЛИ "β_1 *есть* NS" И "β_2 *есть* Z" ТО "β_3 *есть* PS"

ПРАВИЛО_13: ЕСЛИ "β_1 *есть* Z" И "β_2 *есть* PS" ТО "β_3 *есть* NS"

ПРАВИЛО_14: ЕСЛИ "β_1 *есть* Z" И "β_2 *есть* NS" ТО "β_3 *есть* PS"

ПРАВИЛО_15: ЕСЛИ "β_1 *есть* Z" И "β_2 *есть* Z" ТО "β_3 *есть* Z"

Фаззификация входных переменных

В качестве терм-множества первой лингвистической переменной будем использовать множество T_1={"*очень холодная*", "*холодная*", "*в пределах нормы*", "*теплая*", "*очень теплая*"} или в символическом виде T_1={NB, NS, Z, PS, PB} с функциями принадлежности, изображенными на рис. 7.19. В качестве терм-множества второй лингвистической переменной будем использовать множество T_2={"*отрицательная*", "*равна нулю*", "*положительная*"} или в символическом виде T_2={NS, Z, PS} с функциями принадлежности, изображенными на рис. 7.20. В качестве терм-множества выходной лингвистической переменной будем использовать множество T_3={"*очень большой угол влево*", "*большой угол влево*", "*небольшой угол влево*", "*выключить кондиционер*", "*небольшой угол вправо*", "*большой угол вправо*", "*очень большой угол вправо*"} или в символическом виде T_3={NB, NM, NS, Z, PS, PM, PB} с функциями принадлежности, изображенными на рис. 7.21.

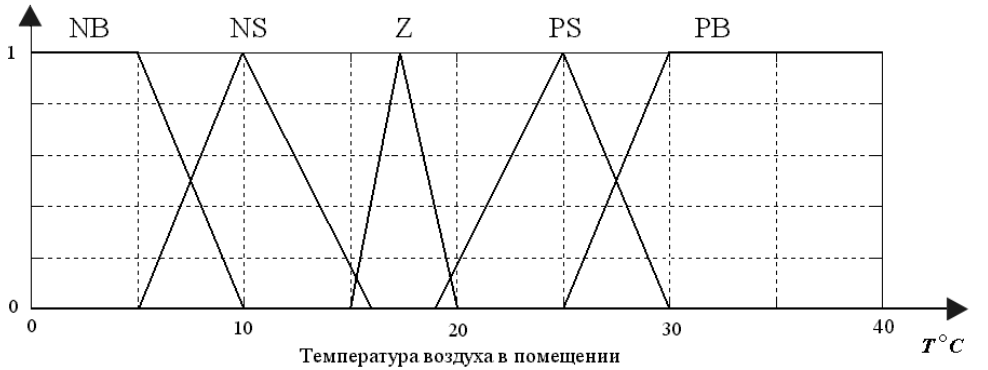


Рис. 7.19. Графики функций принадлежности для термов входной лингвистической переменной "*Температура воздуха*"

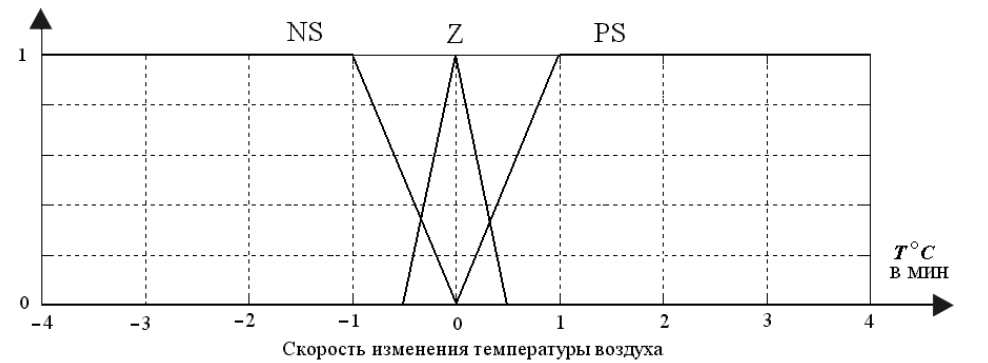


Рис. 7.20. Графики функций принадлежности для термов входной лингвистической переменной "*Скорость изменения температуры*"

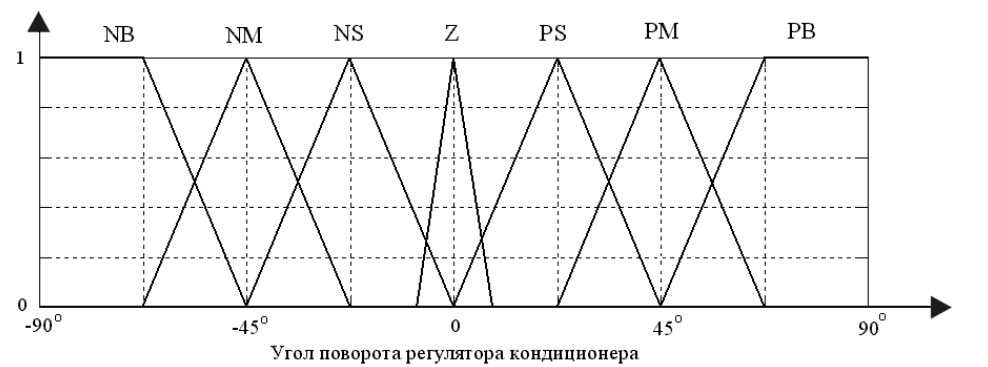


Рис. 7.21. Графики функций принадлежности для термов выходной лингвистической переменной "*Угол поворота регулятора*"

При этом температура воздуха измеряется в градусах Цельсия, скорость изменения температуры воздуха — в градусах Цельсия в минуту, а угол поворота — в угловых градусах. В последнем случае поворот регулятора вправо означает включение режима "тепло" и положительное направление отсчета, а поворот влево — включение режима "холод" и отрицательное направление отсчета.

Используя в качестве алгоритма вывода алгоритм Мамдани, рассмотрим пример его выполнения для случая, когда текущая температура воздуха равна 20 °С, а скорость ее изменения положительная и равна 0.2 °С/мин.

В этом случае фаззификация первой входной лингвистической переменной приводит к значению степени истинности 0.15 для терма PS, а фаззификация второй нечеткой переменной приводит к значению истинности 0.5 для терма Z и значению 0.2 для терма PS. Соответствующие подусловия используются в правилах нечетких продукций с номерами 3 и 10. Эти правила считаются активными и используются в текущем процессе нечеткого вывода.

Агрегирование подусловий правила 3 дает в результате число 0.15, агрегирование подусловий правила 10 — также число 0.15. Следующим этапом нечеткого вывода является активизация заключений в нечетких правилах продукций. Поскольку все заключения правил 1—5 заданы в форме нечетких лингвистических высказываний первого вида, а весовые коэффициенты правил по умолчанию равны 1, то активизация правил 3 и 10 приводит к двум нечетким множествам.

Аккумулирование заключений нечетких правил продукций с использованием операции max-дизъюнкции для правил 3 и 10 приводит в результате к нечеткому множеству, функция принадлежности которого изображена на рис. 7.22.

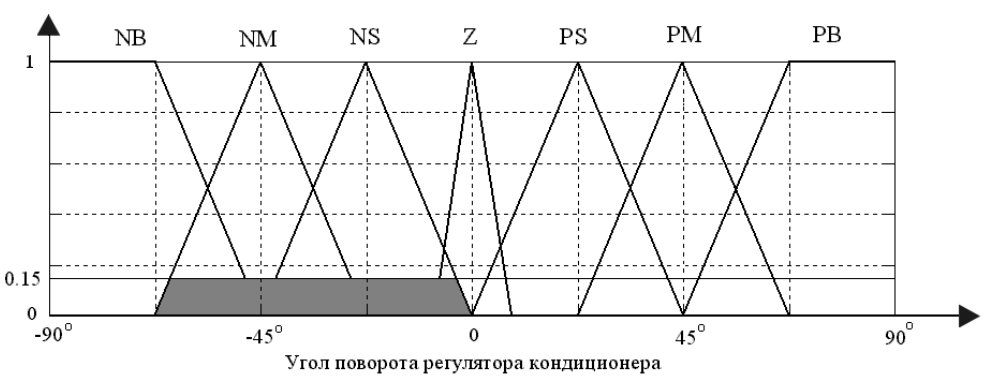


Рис. 7.22. График функции принадлежности двух термов выходной лингвистической переменной "*Угол поворота регулятора*" после аккумуляции

Дефаззификация выходной лингвистической переменной "*Угол поворота регулятора*" методом центра площади для значений функции принадлежности, изображенной на рис. 7.22, приводит к значению управляющей переменной, равному повороту регулятора кондиционера влево на угол 34° (приближенное

значение). Это значение соответствует включению режима "холод" на треть своей мощности и является результатом решения задачи нечеткого вывода.

Примечание

Следует отметить, что дефаззификация другим методом приводит к результатам, которые могут существенно отличаться от полученных. Это может потребовать дополнительных исследований по настройке используемых алгоритмов нечеткого вывода.

Нечеткая модель управления контейнерным краном

В заключение этой главы рассматривается пример разработки модели системы нечеткого управления контейнерным краном, который предназначен для транспортировки моноблочных контейнеров при выполнении разгрузочных работ морских судов.

Содержательная постановка задачи

Контейнерные краны используются при выполнении погрузочно-разгрузочных работ в портах. Они соединяются с моноблочным контейнером гибким тросом и поднимают контейнер к кабине крана. Кабина крана вместе с контейнером может перемещается в горизонтальном направлении по направляющим типа рельсов. Когда контейнер поднимается к кабине, а кран приходит в движение, контейнер начинает раскачиваться и отклоняться от строго вертикального положения под кабиной крана. Эта ситуация иллюстрируется на рис. 7.23.

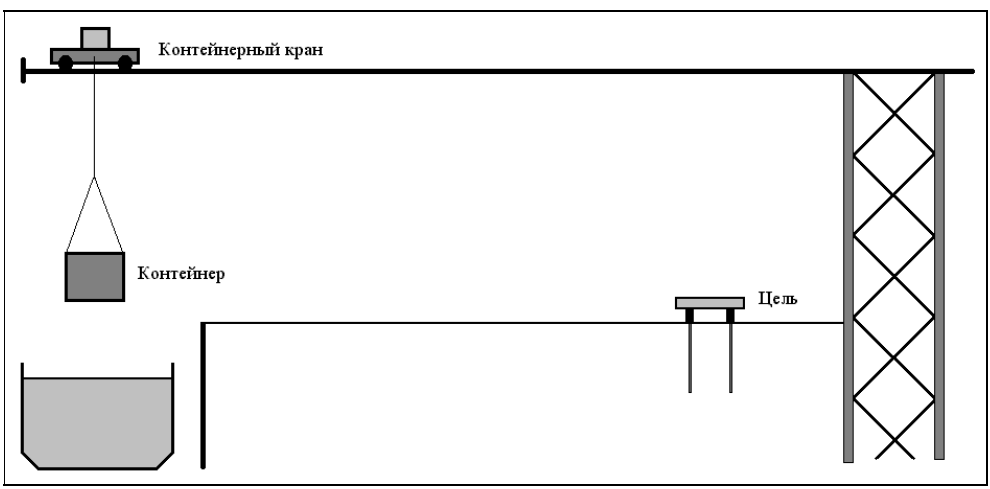


Рис. 7.23. Иллюстрация примера с контейнерным краном

Проблема заключается в том, что пока контейнер раскачивается в ходе своей транспортировки и отклоняется от вертикали, он не может быть опущен на основание цели перемещения, в качестве которой используются железнодорожные платформы или другие транспортные средства.

Анализ действий крановщиков-операторов, выполняющих управление краном, показывает, что они в своей работе применяют следующие эвристические правила:

1. Начинать движение следует со средней мощностью.
2. Если движение уже началось и кабина находится далеко от цели, отрегулировать мощность двигателя таким образом, чтобы контейнер оказался несколько впереди кабины крана.
3. Если кабина находится близко над целью, уменьшить скорость таким образом, чтобы контейнер находился несколько впереди кабины крана.
4. Когда контейнер находится очень близко от позиции цели, следует выключить мощность двигателя.
5. Когда контейнер находится прямо над позицией цели, следует остановить двигатель.

Формирование базы правил систем нечеткого вывода

Следующим этапом построения модели является построение базы правил. С этой целью преобразуем рассмотренные выше 5 эвристических правил в 6 правил нечетких продукций:

ПРАВИЛО_1: ЕСЛИ "расстояние далекое" И "угол равен нулю"
ТО "мощность положительная средняя"

ПРАВИЛО_2: ЕСЛИ "расстояние далекое" И "угол отрицательный малый"
ТО "мощность положительная большая"

ПРАВИЛО_3: ЕСЛИ "расстояние далекое" И "угол отрицательный большой"
ТО"мощность положительная средняя"

ПРАВИЛО_4: ЕСЛИ "расстояние среднее" И "угол отрицательный малый"
ТО "мощность отрицательная средняя"

ПРАВИЛО_5: ЕСЛИ "расстояние близкое" И "угол положительный малый"
ТО "мощность положительная средняя"

ПРАВИЛО_6: ЕСЛИ "расстояние ноль" И "угол равен нулю"
ТО "мощность равна нулю"

Примечание

Следует заметить, что правило 2 разделено на два отдельных правила, чтобы использовать простой формат ЕСЛИ-ТО. При этом расстояние становится отрицательным в том случае, когда кабина крана находится справа от положения цели.

Фаззификация входных переменных

Чтобы иметь возможность автоматически управлять таким краном, необходимо использовать датчики для измерения горизонтального положения кабины крана ("*Расстояние*") и измерения угла раскачивания контейнера ("*Угол*"). Выходом в этом случае является мощность мотора.

Для построения модели системы нечеткого управления контейнерным краном в первую очередь следует для всех переменных определить соответствующие лингвистические переменные. В нашем случае таких лингвистических переменных три — это расстояние, угол и мощность мотора. Каждая из них будет включать в себя 5 термов. При этом будем использовать функции принадлежности типа кусочно-линейных функций, а также функции принадлежности для одноточечных множеств. Ниже приводятся графики конкретных функций принадлежности для отдельных лингвистических термов соответствующих лингвистических переменных (рис. 7.24—7.26). Названия отдельных термов сокращены: "отр" — отрицательный, "пол" — положительный.

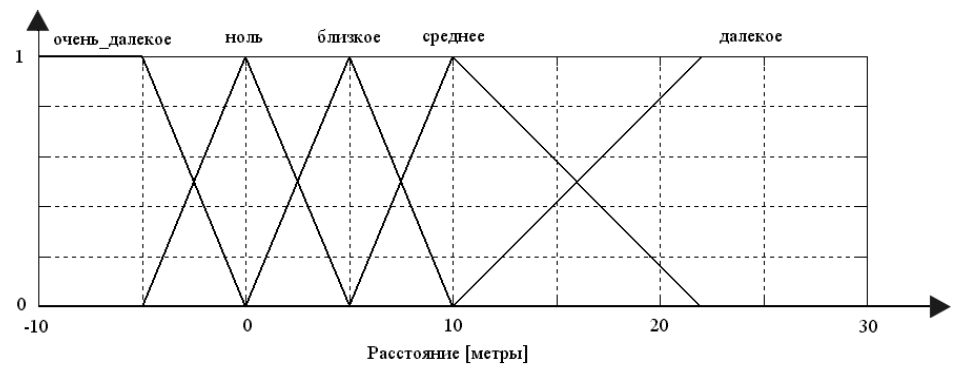


Рис. 7.24. Графики функций принадлежности для термов лингвистической переменной "*Расстояние*", измеряемое от кабины крана до положения цели

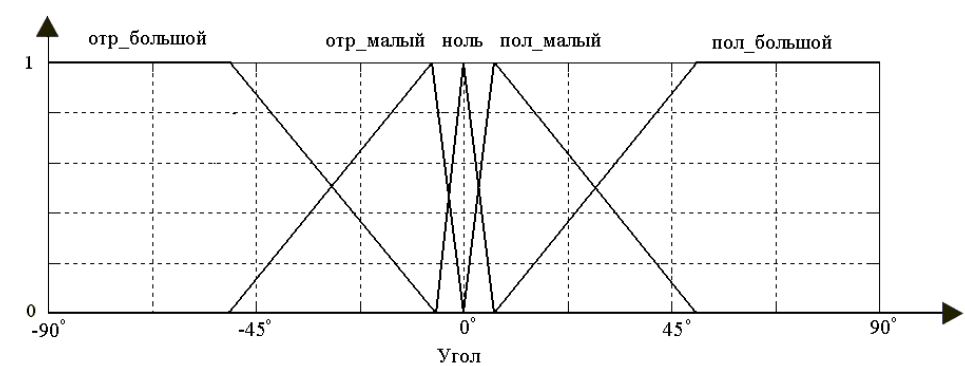


Рис. 7.25. Графики функций принадлежности для термов лингвистической переменной "*Угол*", измеряемый между положением контейнера и кабиной крана

Рис. 7.26. Графики функций принадлежности для термов лингвистической переменной "*Мощность*" двигателя

Далее необходимо определить методы агрегирования подусловий. Поскольку во всех правилах 1—6 в качестве логической связки для подусловий применяется только нечеткая конъюнкция (операция "И"), то в качестве метода агрегирования будем использовать операцию min-конъюнкции. В качестве схемы нечеткого вывода будем использовать метод Мамдани, поэтому методом активизации будет MIN, который рассчитывается по формуле (7.6). Для аккумуляции заключений правил будем использовать метод max-дизъюнкции, который также применяется в случае схемы нечеткого вывода методом Мамдани. Наконец, в качестве метода дефаззификации будем использовать метод центра тяжести для одноэлементных множеств (7.10), который можно также записать в виде ключевого слова COGS.

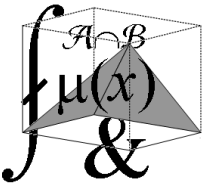
Рассмотрим пример выполнения нечеткого вывода для этой модели в случае, когда текущее расстояние до цели равно 7.5 м, а угол между контейнером и кабиной крана равен 10°.

В этом случае фаззификация первой входной лингвистической переменной приводит к значениям степени истинности: 0.5 для терма "близкое" и 0.5 для терма "среднее", а фаззификация второй нечеткой переменной приводит к значениям истинности: 0.1 для терма "пол_большой" и 0.9 для терма "пол_малый". Соответствующие подусловия совместно используются только в правиле нечеткой продукции с номером 5. Это единственное правило считается активным и используется в текущем процессе нечеткого вывода.

Агрегирование подусловий правила 5 дает в результате число 0.5. Следующим этапом нечеткого вывода является активизация заключения в правиле 5 нечеткой продукции. Поскольку это заключение единственное и задано в форме одноточечного множества, то активизация тривиальна и дает в результате степень принадлежности 0.5. Аккумулирование и дефаззификация также тривиальны и дают в результате, что мощность двигателя равна 12 киловатт.

В заключение следует отметить, что рассмотренные примеры будут использоваться далее в качестве тестовых задач в *частях II* и *III*.

Глава 8



Язык нечеткого управления — FCL

Язык нечеткого управления FCL (Fuzzy Control Language) описан в Стандарте IEC 1131-7, в котором определяются цели разработки этого языка, его базовая нотация и приводятся примеры записи моделей нечеткого управления с использованием нотации языка FCL. В настоящей главе рассматриваются элементы языка FCL в соответствии с указанным Стандартом.

Язык FCL разработан для представления нечетких моделей систем управления, в частности, моделей так называемых программируемых контроллеров (Programmable Controllers) или *программируемых логических контроллеров* (ПЛК) в форме структурируемого текста, который может быть интерпретирован как программа на языке высокого уровня. Хотя Стандарт IEC 1131-7 не определяет требования к вычислительным средам и устройствам, которые могут реализовывать трансляцию, компиляцию и выполнение программ на языке FCL, описанная в нем нотация основных компонентов систем нечеткого вывода позволяет достичь формального уровня строгости, необходимого для последующей разработки соответствующих инструментальных средств.

Изложению нотации языка предшествует краткое введение в специальную область прикладных исследований, которая получила название нечеткого управления. После этого рассматриваются элементы базовой нотации языка FCL. В заключение этой главы подробно рассматривается конкретный пример разработки нечеткой модели системы управления подъемным краном и запись этой модели в нотации языка FCL.

8.1. Концептуальные основы нечеткого управления

Под *нечетким управлением* (Fuzzy Control) понимается область применения общей методологии теории нечетких множеств и нечеткой логики для решения практических задач управления. Нечеткое управление возникло как технология, способная расширить возможности автоматизации производства и предназначенная для решения прикладных задач в области управления, которые в общем случае могут быть реализованы с помощью программируемых контроллеров.

Нечеткое управление базируется на использовании не столько аналитических или теоретических моделей, сколько на практическом применении знаний, которые можно представить в форме так называемых лингвистических баз правил. Нечеткое управление может использоваться в том случае, когда существует определенный опыт экспертов и его можно записать некоторым формальным образом. Все это позволяет воспользоваться доступными знаниями с целью улучшить процессы управления и решить ряд задач, например:

- управление (с обратной или без обратной связи, с одной или со многими переменными, для линейных или нелинейных систем);
- установка параметров систем управления в автономном режиме или в режиме реального времени;
- классификация и распознавание образов;
- оперативное принятие решения (Послать этот продукт на обработку устройством A или B?);
- помощь операторам в принятии решений или настройке параметров;
- определение и диагностика неисправностей в системах.

Широкий диапазон приложений и естественность подхода, основанного на опыте специалистов, делает нечеткое управление основным средством, которое в качестве стандарта должно стать доступным для всех пользователей программируемых контроллеров. Нечеткое управление может также непосредственно комбинироваться с классическими методами управления.

Применение нечеткого управления может быть наиболее эффективным в тех случаях, когда отсутствует явная модель процесса и аналитическая модель является слишком сложной для представления (например, системы с несколькими входами и несколькими выходами) или для получения решений в реальном масштабе времени.

Другое достоинство нечеткого управления заключается в непосредственном объединении опыта нескольких специалистов. При этом вовсе не нужно моделировать целиком весь контроллер с помощью нечеткого управления — иногда нечеткое управление может только интерполировать серию локально линейных моделей или динамически адаптировать параметры некоторого линейного регулятора. Тем самым становится возможным не только оперировать нелинейными моделями, но и сосредоточить внимание на рассмотрении тех параметров существующих регуляторов, которые следует улучшить.

Нечеткое управление, являясь многозначным управлением, больше ограничивается значениями высказываний "*истина*" или "*ложь*". Эта особенность делает нечеткое управление адекватным средством для моделирования эмпирического опыта экспертов, оперируя теми понятиями, в терминах которых формулируются управляющие воздействия на заданном множестве входов.

Ниже дается краткое введение в теорию нечеткого управления, насколько это необходимо для понимания нотации языка FCL. Поскольку используемая в нечетком управлении терминология вынесена в *глоссарий*, здесь рассматриваются

только основные особенности интеграции приложений нечеткого управления и их представления в форме диаграмм.

С точки зрения информационных технологий системы нечеткого управления являются продукционными экспертными системами. С точки зрения теории систем управления системы нечеткого управления являются контроллерами с нелинейными параметрами регулирования. При этом текущие значения выходных переменных зависят только от текущих значений входных переменных и не зависят от предыстории этих значений за исключением случаев, когда отсутствуют активные правила и не определены значения переменных по умолчанию. Если же контроллер должен быть реализован как динамическая система, то соответствующие динамические функции представляют собой внешние элементы для нечеткого функционального блока.

В системах автоматического регулирования обычно используются дифференцирующие и интегрирующие элементы (звенья) первого порядка. Выходные переменные таких элементов являются дополнительными входными переменными для системы нечеткого управления. Такими переменными также могут быть переменные, описывающие значения отклонения управляемых параметров от установленных значений.

Напротив, выходные переменные систем нечеткого управления могут использоваться операторами для выполнения коррекции управляемых параметров в различных системах управления.

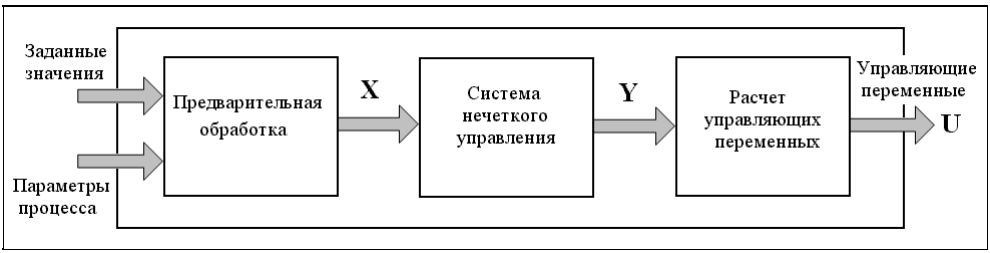
Общая структура систем с нечетким управлением изображена на рис. 8.1, *а*, пример реализации системы нечеткого управления изображен на рис. 8.1, *б*. В примере в качестве входной переменной используется разность x между заданным и реальным значениями контролируемого параметра. Эта разность совместно с ее производной по времени и интегралом по заданному интервалу времени передаются в собственно систему нечеткого управления как три входные переменные, не зависящие от своей предыстории. В то же время переменная для коррекции контролируемого параметра получается на основе интегрирования по заданному интервалу времени выходной переменной системы нечеткого управления.

Область применения нечеткого управления достаточно широка — от небольших и простых приложений до комплексных и сложных проектов. Чтобы охватить все возможные случаи, следует использовать Правила согласованности классов систем нечеткого управления, которые дополняют и расширяют базовую нотацию языка FCL. При этом Базовый Класс определяет минимальное множество требований, которым должны удовлетворять все согласованные системы, что обеспечивает переносимость программ нечеткого управления.

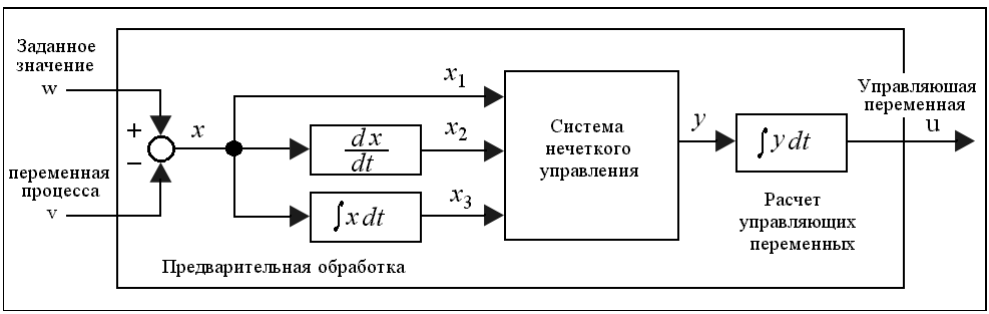
Существующая теория и системы, реализованные в области нечеткого управления, отличаются между собой по используемой терминологии, функциональным возможностям и особенностям реализации в инструментальных средствах.

Цель Стандарта IEC 1131-7 состоит в том, чтобы, с одной стороны, предоставить разработчикам и пользователям однозначные и доступные для понимания базовые средства для интеграции приложений нечеткого управления, которые могут быть записаны в нотации языков программируемых контроллеров, а, с

другой стороны, обеспечить возможность переносимости мобильных программ нечеткого управления между различными системами программирования.



а



б

Рис. 8.1. Общая структура систем нечеткого управления

Необязательные средства языка FCL определены в Классе Расширения. Программы нечеткого управления, использующие эти средства, могут переноситься с одной системы на другую только в том случае, если эти системы реализуют одинаковое множество этих средств. В противном случае может оказаться возможным лишь частичный перенос программ. Стандарт не требует, чтобы все согласованные системы реализовывали средства Класса Расширения в полном объеме. Хотя и допускается возможность частичного переноса, следует избегать использования нестандартных средств. Поэтому согласованная система не должна содержать нестандартные средства, которые не могут быть адекватно реализованы с использованием стандартных средств Базового Класса и Класса Расширения.

Чтобы не исключать из рассмотрения системы, использующие свои собственные и достаточно сложные средства, и не препятствовать процессу дальнейшего развития, Стандарт разрешает использование дополнительных нестандартных средств, которые не вошли в ни Базовый Класс, ни в Класс Расширения. Однако все такие средства должны быть перечислены стандартным способом, чтобы можно было бы легко установить их нестандартный характер.

Переносимость приложений нечеткого управления зависит от особенностей как систем программирования, так и от характеристик систем управления. Все эти

особенности указываются в Списке Проверки Данных, который разрабатывается производителями систем.

Интеграция программируемых контроллеров

Приложения нечеткого управления, разработанные в форме программ на языке FCL согласно Стандарту IEC1131-7, должны быть инкапсулированы в Функциональные Блоки (или Программы) на основе Стандарта IEC1131-3, который определяет использование языков программирования в программируемых контроллерах. Поэтому здесь будут использоваться понятия Типа Функционального Блока и Экземпляра Функционального Блока, которые определяются в Стандарте IEC 1131-3.

Примечание

Стандарт IEC 1131-3 (полное название — International standard IEC 1131-3. Programmable controllers. Part 3. Programming languages, 1993) заслуживает того, чтобы остановиться на нем более подробно. Этот Стандарт определяет синтаксис и семантику пяти языков программирования программируемых контроллеров: Ladder diagram (LD), Sequential Function Charts (SFC), Function Block Diagram (FBD), Structured Text (ST), Instruction List (IL). Оставляя в стороне дискуссию о достоинствах и недостатках этих языков и Стандарта в целом, ниже дается лишь краткая характеристика этих языков.

- Язык LD представляет собой некоторый графический язык программирования, являющийся стандартизованным вариантом класса языков релейно-контактных схем. Логические выражения на этом языке описываются в виде реле, которые широко применялись в области автоматизации в 1960-х годах.
- Язык SFC представляет собой графический язык, используемый для описания алгоритма в виде набора связанных пар: шаг (step) и переход (transition). Шаг представляет собой набор операций над переменными. Переход — набор логических условных выражений, определяющий передачу управления к следующей паре шаг-переход. По внешнему виду описание на языке SFC напоминает хорошо известные логические блок-схемы алгоритмов. SFC имеет возможность распараллеливания алгоритма. Однако SFC не имеет средств для описания шагов и переходов, которые могут быть выражены только средствами других языков стандарта.
- Язык FBD представляет собой графический язык, по своей сути похожий на LD. Вместо реле в этом языке используются функциональные блоки, по внешнему виду — микросхемы. Алгоритм работы некоторого устройства на этом языке выглядит как функциональная схема электронного устройства: элементы типа "логическое И", "логическое ИЛИ" и т. п., соединенные линиями.
- Язык ST представляет собой текстовый высокоуровневый язык общего назначения, по синтаксису ориентированный на Паскаль. Самостоятельного значения не имеет, используется только совместно с языком SFC.
- Язык IL представляет собой текстовый язык низкого уровня. Выглядит как типичный язык Ассемблера, что объясняется его происхождением: для

некоторых моделей ПЛК фирмы Siemens является языком Ассемблера. В рамках стандарта IEC 1131-3 к архитектуре конкретного процессора не привязан. Самостоятельного значения не имеет, используется только совместно с SFC.

Перечисленные языки IEC 1131-3 используются ведущими фирмами изготовителями ПЛК, имеют длительную историю применения, достаточно распространены и известны пользователям по тем или иным модификациям. Несмотря на то, что во многих случаях такие модификации несущественны, это влечет определенные неудобства при работе с ПЛК различных фирм-изготовителей. С этой точки зрения, стандарт IEC 1131-3 несомненно прогрессивен, поскольку позволяет привести бесчисленное число различных вариантов и интерпретаций языков ПЛК к единому знаменателю.

Тип Функционального Блока, определяемый в языке FCL, задается входными и выходными параметрами, специальными правилами и объявлениями нечеткого управления. Соответствующие *Экземпляры Функционального Блока* должны содержать значения данных конкретных приложений нечеткого управления.

Функциональные Блоки, записанные на языке FCL, могут быть использованы в программах и Функциональных Блоках, записанных на любом из языков Стандарта IEC1131-3. При этом типы данных входных и выходных параметров Функционального Блока или программы на языке FCL должны соответствовать аналогичным параметрам данной среды реализации, как это показано на рис. 8.2.

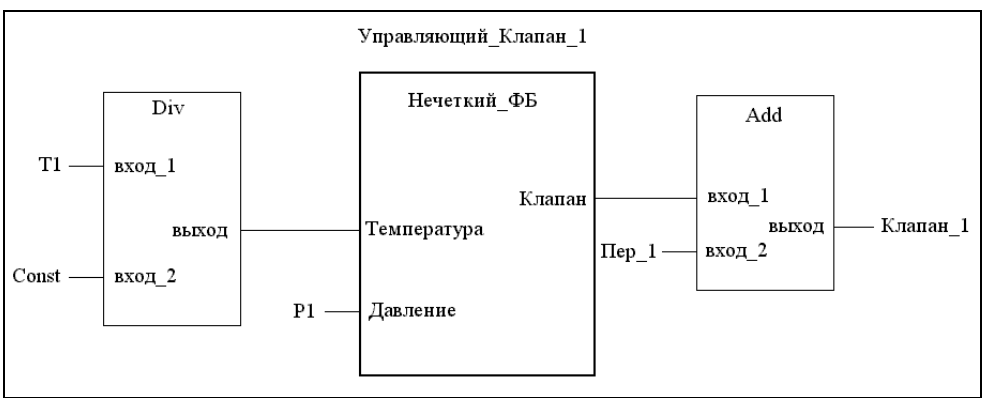


Рис. 8.2. Пример Функционального Блока нечеткого управления в форме диаграммы языка FBD

В этом примере *Управляющий_Клапан_1* использует конкретный экземпляр Функционального Блока, инстанцируемый от Типа Функционального Блока *Нечеткий_ФБ*. Тип Функционального Блока *Нечеткий_ФБ* может быть записан в форме программы на языке FCL в соответствии с его базовой нотацией, определенной в Стандарте IEC1131-7. Этот Функциональный Блок *Нечеткий_ФБ* может быть использован как в некоторой программе, так и в другом Функцио-

нальном Блоке, представленном в форме диаграммы на графическом языке FBD (Function Block Diagram, диаграммы функциональных блоков).

Перенос программ нечеткого управления

Определение нотации языка FCL базируется на определениях языков программирования Стандарта IEC1131-3. При этом взаимодействие алгоритма нечеткого управления со средой программирования должно быть скрыто от соответствующих программ. Именно поэтому алгоритм нечеткого управления следует представлять в форме Функционального Блока. При этом для представления собственно лингвистических аспектов Функциональных Блоков нечеткого управления оказываются необходимыми и такие элементы, как функции принадлежности, правила, операции и методы, которые и определяются в Стандарте IEC1131-7.

Стандартизация элементов языка FCL выполнена с целью достижения единого представления данных для обеспечения переноса программ между системами нечеткого управления различных производителей, как это изображено на рис. 8.3.

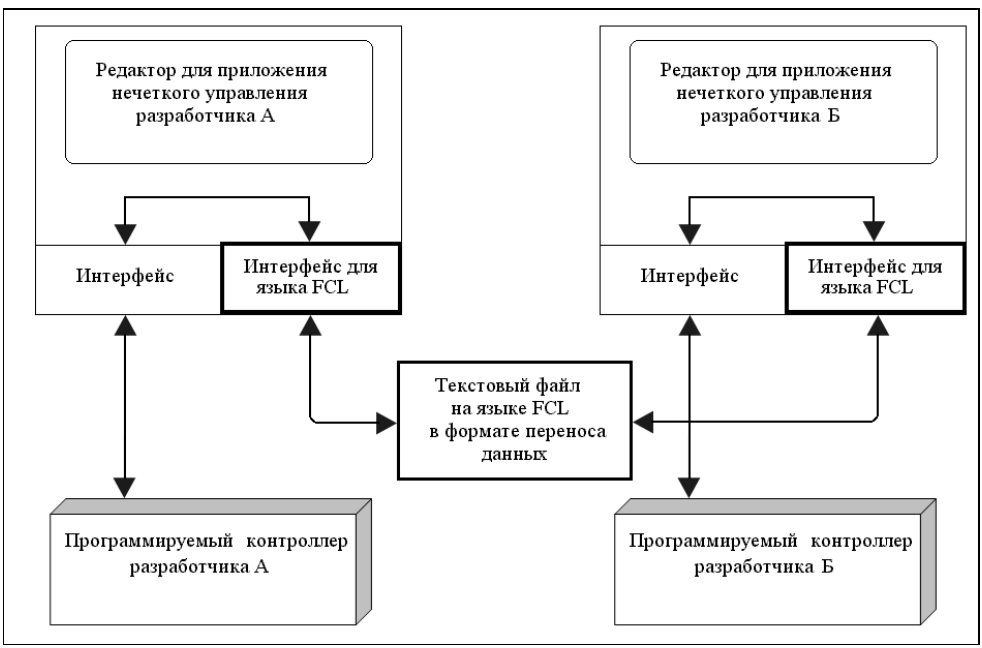


Рис. 8.3. Переносимость программ, написанных на языке FCL

Использование такого единого представления разработчиками программируемых контроллеров позволяет каждому из них иметь собственное аппаратное

обеспечение, редакторы и компиляторы программ. Разработчику достаточно лишь реализовать интерфейс данных для своего собственного редактора. В свою очередь заказчики и потребители получают возможность переноса соответствующих проектов нечеткого управления между различными разработчиками.

История разработки и стандартизации языка FCL

Необходимость унификации и стандартизации подходов к рассмотрению методологии нечеткого управления стала осознаваться в первой половине 1990-х гг. в связи с широкой разработкой и производством промышленных и бытовых устройств, реализованных на основе алгоритмов нечеткого управления.

С целью стандартизации методологии программирования, средств и функциональных характеристик нечеткого управления в области программируемых контроллеров в 1993 г. в США была организована специальная рабочая группа по рассмотрению соответствующих предложений (NWIP, New Working Item Proposal). В результате широкого международного обсуждения предложений была одобрена и принята в рамках Международной электротехнической комиссии IEC рабочая программа с названием "Стандартизация Нечеткого Управления".

Поскольку уже тогда предполагалось, что находившаяся на стадии становления технология будет включена в проектируемые программируемые контроллеры, разработка этого проекта была поручена в рамках подкомитета IEC SC65B (Устройства) специальной рабочей группе IEC 65B/WG7 (Программируемые контроллеры).

В декабре 1996 г. международным отделением TF8 рабочей группы IEC/65B/WG7 во Франкфурте, Германия был подготовлен проект Стандарта, получивший номер 1.0. По плану он должен стать частью 7 Международного Стандарта IEC 1131 "Программируемые контроллеры". Полное название этого Стандарта — Стандарт IEC 1131-7 CD1-2- IEC TC65/WG 7/TF8 Part 7 — Fuzzy Control Language (Release 19, Januar 1997). Для удобства будем использовать его сокращенное название: IEC 1131-7.

Основное назначение Стандарта IEC 1131-7 — определить язык FCL, предназначенный для программирования приложений нечеткого управления, которые могут быть использованы в программируемых контроллерах.

Этот Стандарт ссылается на другие стандартные документы, которые также утверждены в качестве Стандартов Международной организацией по стандартизации ISO по предложению Международной электротехнической комиссии. Ниже следует перечень этих документов:

- IEC 1-1581-FDIS 1996, International Electrotechnical Vocabulary, clause 351: Automatic Control
- IEC 1131 Part 1: 1992, General information
- IEC 1131 Part 2: 1992, Equipment requirements and tests

- IEC 1131 Part 3: 1993, Programming languages
- IEC 1131 Technical Report, Guidelines for users and implementers of IEC 1131-3

8.2. Базовая нотация языка нечеткого управления FCL

В настоящем разделе рассматриваются элементы языка FCL, их синтаксис и правила использования в нечетких моделях систем управления. Описание этих элементов занимает центральное место в Стандарте IEC 1131-7.

Основные элементы языка FCL

Метод спецификации текстового описания программируемых контроллеров приводится в Приложении A Стандарта IEC 1131-3 (IEC 1131 Part 3: 1993, Programming languages). Этот же метод спецификации используется при описании языка FCL.

Формальная спецификация элементов языка для текстовых языков программирования описывается в Приложении B Стандарта IEC 1131-3. Для языка FCL следует использовать следующее подмножество этих элементов языка:

- Буквы, цифры, идентификаторы
- Константы
- Типы данных
- Переменные

Нотация правил продукций

В дополнение к перечисленным выше элементам языка Части 3 Стандарта IEC 1131 для записи правил продукций следует использовать следующие элементы языка FCL:

```
объявление_функционального_блока ::= 'FUNCTION_BLOCK'
имя_функционального_блока
  {объявления_вх_вых_переменных_ФБ}
{объявления_других_переменных_ФБ}
тело_функционального_блока
'END_FUNCTION_BLOCK'
объявления_вх_вых_переменных_ФБ ::= объявления_входа | объявления_выхода
объявления_других_переменных_ФБ ::= объявления_переменных
тело_функционального_блока ::= {блок_фаззификации}
```

```
  {блок_дефаззификации}
  {блок_правил}
  {необязательный_блок}
блок_фаззификации ::= 'FUZZIFY' имя_переменной
  {лингвистический_терм}
'END_FUZZIFY
блок_дефаззификации ::= 'DEFUZZIFY' имя_нечеткой_переменной
  {лингвистический_терм}
  метод_дефаззификации
  значение_по_умолчанию
  [интервал]
  'END_FUZZIFY'
блок_правил ::=    'RULEBLOCK' имя_блока_правил
  определение_операции
  [метод_активизации]
  метод_аккумуляции
  {правило}
'END_RULEBLOCK'
необязательный_блок ::=   'OPTION'
любой параметр спецификации модели
'END_OPTION'
лингвистический_терм ::=  'TERM' имя_терма ':=' функция_принадлежности';'
функция_принадлежности ::=  одноэлементное_множество | точки
одноэлементное_множество ::=   число | имя_переменной
точки ::= {'('число | имя_переменной',' число')'}
метод_дефаззификации ::='METHOD' ':' 'COG' | 'COGS' | 'COA' | 'LM' | 'RM'
';'
значение_по_умолчанию ::=  'DEFAULT' ':=' число | 'NC' ';'
интервал ::= 'RANGE' ':=  ' '('число'..'число')' ';'
определение_операции ::=   ('OR' ':' 'MAX' | 'ASUM' | 'BSUM') |
('AND' ':' 'MIN' | 'PROD' | 'BDIF') ';'
метод_активизации ::=    'ACT' ':' 'PROD' | 'MIN' ';'
метод_аккумуляции ::=    'ACCU' ':' 'MAX' | 'BSUM' | 'NSUM' ';'
правило ::=  'RULE' целое_число':'
'IF' условие 'THEN' заключение [WITH весовой_коэффициент] ';'
условие ::=      (подусловие | имя_переменной) {AND' | 'OR'
(подусловие | имя_переменной)}
подусловие ::=    ('NOT' '('имя_переменной 'IS' ['NOT']) имя_терма')') |
(имя_переменной 'IS' ['NOT'] имя_терма)
```

```
заключение ::=    {(имя_переменной | (имя_переменной 'IS' имя_терма))
','}
(имя_переменной | имя_переменной 'IS' имя_терма)
весовой_коэффициент ::= переменная | число
имя_функционального_блока ::= идентификатор
имя_блока_правил ::= идентификатор
имя_терма ::= идентификатор
имя_нечеткой_переменной ::= идентификатор
имя_ переменной ::= идентификатор
число ::= целое_число | действительное_число
объявления_входа ::= see IEC 1131-3 Annex B
объявления_выхода ::= see IEC 1131-3 Annex B
объявления_переменных ::= see IEC 1131-3 Annex B
идентификатор ::= see IEC 1131-3 Annex B
```

Ключевые слова языка FCL

В табл. 8.1 приводится перечень ключевых слов языка FCL, которые разрешается использовать только в том значении, как оно определено в нотации языка FCL.

Таблица 8.1. *Зарезервированные ключевые слова языка FCL*

Ключевое слово	Описание ключевого слова
()	Скобки в условии, *терме*, *интервале*
ACCU	Метод аккумуляции
ACT	Метод активизации
AND	Операция логической конъюнкции (логическое "И")
ASUM	Операция логического "ИЛИ" по формуле алгебраической суммы
BDIF	Операция логического "И" по формуле ограниченной разности
BSUM	Метод аккумуляции по формуле ограниченной суммы
COA	Метод дефаззификации по формуле центра площади
COG	Метод дефаззификации по формуле центра тяжести
COGS	Метод дефаззификации по формуле центра тяжести для одноэлементных множеств
DEFAULT	Значение выхода по умолчанию в случае отсутствия активных правил

Таблица 8.1 *(продолжение)*

Ключевое слово	Описание ключевого слова
DEFUZZIFY	Дефаззификация выходной переменной
END_DEFUZZIFY	Конец спецификации дефаззификации
END_FUNCTION_BLOCK	Конец спецификации функционального блока
END_FUZZIFY	Конец спецификации фаззификации
END_OPTIONS	Конец спецификации дополнительных параметров
END_RULEBLOCK	Конец спецификации блока правил
END_VAR	Конец определения входных/выходных переменных
FUNCTION_BLOCK	Начало спецификации функционального блока
FUZZIFY	Фаззификация входной переменной
IF	Начало правила, после которого следует его условие
IS	Связка для значения лингвистической переменной в условии или заключении
LM	Метод дефаззификации по формуле левого модального значения
MAX	Метод аккумуляции по формуле максимума
METHOD	Метод дефаззификации
MIN	Минимум в качестве логической операции "И"
NC	Значение выходной переменной остается без изменения (No Change) в случае отсутствия активных правил
NOT	Операция логического отрицания (логическое "НЕ")
NSUM	Метод аккумуляции по формуле нормализованной суммы
OPTIONS	Начало спецификации дополнительных параметров
OR	Операция логической дизъюнкции (логическое "ИЛИ")
PROD	Произведение в качестве логической операции "И"
RANGE	Интервал переменной для шкалирования функции принадлежности
RM	Метод дефаззификации по формуле правого модального значения
RULE	Начало спецификации нечеткого правила
RULEBLOCK	Начало спецификации блока правил

Таблица 8.1 *(окончание)*

Ключевое слово	Описание ключевого слова
TERM	Определение лингвистического терма (функции принадлежности) для лингвистической переменной
THEN	Разделитель условия и заключения в нечетком правиле
VAR	Определение локальной переменной (переменных)
VAR_INPUT	Определение входной переменной (переменных)
VAR_OUTPUT	Определение выходной переменной (переменных)
WITH	Определение весового коэффициента

Интерфейс функционального блока (Function Block interface)

Согласно Стандарту IEC 1131-7, внешнее представление Нечеткого Функционального Блока требует использования следующих элементов языка FCL:

```
FUNCTION_BLOCK имя_функционального_блока   {Функциональный блок}
VAR_INPUT          {Объявление входных параметров}
имя-переменной: тип_данных;
...
END_VAR
VAR_OUTPUT                    {Объявление выходных параметров}
имя-переменной: тип_данных;
...
END_VAR
...
VAR                 {Локальные переменные}
имя-переменной: тип_данных;
END_VAR
END_FUNCTION_BLOCK
```

Эти элементы языка FCL позволяют описать интерфейс функционального блока. Интерфейс функционального блока определяется совместно с параметрами, которые могут использоваться внутри и вне этого функционального блока. Типы данных этих параметров должны быть определены в соответствии с требованиями Стандарта языка FCL. Ниже приводится пример объявления функционального блока на языке структурированного текста (ST, Structured Text), а на

рис. 8.4. изображен этот же функциональный блок на языке диаграмм функциональных блоков.

```
FUNCTION_BLOCK Нечеткий_ФБ
VAR_INPUT
   Температура : REAL;
   Давление : REAL;
END_VAR
VAR_OUTPUT
   Клапан : REAL;
END_VAR
END_FUNCTION_BLOCK
```

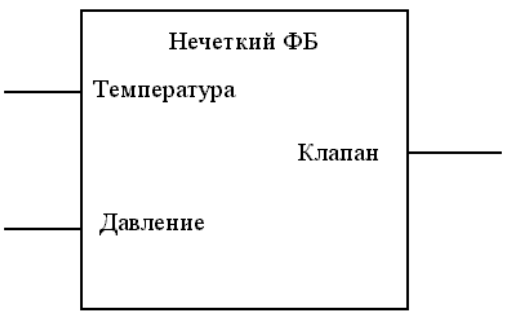


Рис. 8.4. Пример объявления интерфейса функционального блока на языке FBD

Фаззификация (Fuzzification)

Как было сказано в *главе 7*, цель фаззификации заключается в преобразовании числовых значений входных переменных в функции принадлежности, определенные для лингвистических значений соответствующих переменных. Это преобразование в языке FCL описывается между ключевыми словами: `FUZZIFY` и `END_FUZZIFY` следующим образом:

```
FUZZIFY имя_переменной
TERM имя_терма := функции_принадлежности;
...
END_FUZZIFY
```

После ключевого слова `FUZZIFY` должно быть записано имя переменной, которая используется для фаззификации. Это имя переменной предварительно должно быть определено в секции `VAR_INPUT`. Соответствующая лингвистиче-

ская переменная должна быть описана одним или несколькими лингвистическими термами.

Лингвистические термы определяются после ключевого слова TERM и описываются посредством функций принадлежности, которые фаззифицируют эту переменную. В качестве функции принадлежности может использоваться некоторая кусочно-линейная функция. Эта функция задается таблицей своих точек в форме:

```
функция_принадлежности ::= (точка i), (точка j),...
```

Каждая точка в списке представляет собой пару значений, разделенных запятой. Первое из них — количественное значение переменной, а второе — значение функции принадлежности для этого значения переменной. Эти пары значений заключаются в скобки и отделяются друг от друга запятыми:

```
точка i ::= значение входа i | имя_переменной на входе i,
значение функции принадлежности i
```

Таким способом могут быть определены все простые элементы, такие как наклонная прямая и треугольная функция принадлежности. Точки должны записываться в возрастающем порядке значений переменной. Функция принадлежности между последовательными точками является линейной. Остальные значения функции принадлежности для произвольного значения входа вычисляются посредством линейной интерполяции значений функции принадлежности для двух соседних точек.

Число точек может изменяться, но их максимальное количество ограничено согласно правилу 6 классов согласованности.

Ниже приводится пример функции принадлежности с 3 точками для лингвистического терма "теплый":

```
TERM теплый := (10°C, 0.0), (50°C, 1.0), (90°, 0.0);
```

Если значение лингвистической переменной меньше, чем первая базовая точка в таблице задания функции принадлежности, то соответствующее значение функции принадлежности принимается равным значению для этой первой точки в таблице. Если значение лингвистической переменной больше, чем последняя базовая точка в таблице задания функции принадлежности, то соответствующее значение функции принадлежности принимается равным значению для этой последней точки в таблице.

Примеры определения линейной *Z*-образной функции и линейной *S*-образной функции принадлежности:

```
FUZZIFY температура;
   TERM холодный := (-10°C, 1), (10°C, 0);
   TERM теплый := (10°C, 0.0), (50°C, 1.0);
END_FUZZIFY
```

Чтобы адаптировать модель нечеткого управления на этапе ее реализации, базовые точки функций принадлежности могут быть модифицированы. Это можно

сделать посредством определения дополнительных переменных, которые описываются как входные в функциональном блоке. Эти переменные необходимо объявить в секции VAR_INPUT функционального блока.

Примечание

Значения отдельных точек функций принадлежности во время выполнения программы могут выходить за пределы последовательности.

Ниже представлен пример задания точек функции принадлежности

```
VAR_INPUT
   температура : REAL; (* этот вход должен быть фаззифицирован *)
   давление : REAL; (*этот вход должен быть фаззифицирован *)
   bp_warm1, bp_warm2 : REAL; (* эти входы предназначены для
   интерактивной адаптации *)
END_VAR
   FUZZIFY температура
TERM теплый := (bp_warm1, 0.0), (40°C, 1.0), (bp_warm2, 0.0);
   ...
END_FUZZIFY
```

Дефаззификация (Defuzzification)

Лингвистическая переменная должна быть преобразована в количественное значение. Это преобразование описывается между ключевыми словами DEFUZZIFY и END_DEFUZZIFY.

Переменная, которая используется для дефаззификации, должна быть записана после ключевого слова DEFUZZIFY. Имя этой переменной должно быть объявлено ранее в секции VAR_OUTPUT.

```
DEFUZZIFY имя_переменной
   TERM имя_терма := функция_принадлежности;
   метод_дефаззификации;
   значение_по_умолчанию;
   [интервал;]
END_DEFUZZIFY
```

Определение лингвистических термов задается в секции фаззификации. С целью упрощения дефаззификации выходов могут быть использованы специальные функции принадлежности в форме значения для одноэлементного множества. Подобные функции принадлежности описываются посредством единственного значения для лингвистического терма.

Ниже приводится пример определения таких термов.

```
DEFUZZIFY клапан
   TERM открыт_на_слив := -100;
   TERM закрыт := 0;
   TERM открыт_на_наполнение := 100;
END_DEFUZZIFY
```

Метод дефаззификации должен быть определен посредством элемента языка `METOД` после ключевого слова `METHOD`.

```
METHOD : метод_дефаззификации;
```

В качестве метода дефаззификации могут быть использованы следующие методы (табл. 8.2)

Таблица 8.2. *Методы дефаззификации*

Ключевое слово	Описание	Формула расчета
`COG`	Центр тяжести (Centre of Gravity)	(7.9)
`COGC`	Центр тяжести для одноточечных множеств (Centre of Gravity for Singletons)	(7.10)
`COA`	Центр площади (Centre of Area)	(7.11)
`LM`	Левое модальное значение (Left Most Maximum)	(7.12)
`RM`	Правое модальное значение (Right Most Maximum)	(7.13)

Если значение функции принадлежности равно 0 для всех лингвистических термов некоторой выходной переменной, то это означает отсутствие активных правил для этой переменной. В этом случае дефаззификация не позволит получить адекватный результат. Именно по этой причине целесообразно определить для такой переменной некоторое значение по умолчанию. Это значение будет назначено выходной переменной только в случае отсутствия активных правил. Значение по умолчанию задается после ключевого слова `DEFAULT`.

```
DEFAULT := значение | NC;
```

После ключевого слова `DEFAULT` может быть указано значение по умолчанию или ключевое слово `NC` (no change). Последнее применяется для того, чтобы явно указать, что выход остается без изменения в случае, если отсутствуют активные правила.

Интервал определяется заданием после ключевого слова `RANGE` некоторых минимального и максимального значений, которые разделяются двумя точками.

```
RANGE := (minimum value.. maximum value);
```

Интервал используется для спецификации минимального и максимального значений некоторой выходной переменной. Если функции принадлежности термов выходной переменной заданы в форме одноэлементных множеств, то подобный интервал не может быть задан. В остальных случаях RANGE используется для ограничения каждой функции принадлежности интервалом по каждой из выходных переменных. Этот интервал следует явно определять, чтобы избежать непредсказуемых значений выходных переменных.

Если этот интервал не задан, то принимается интервал по умолчанию для типа данных соответствующей переменной, как это определено в части 3 Стандарта IEC 1131-3.

Блок правил (Rule block)

Для определения механизма нечеткого вывода используется один или несколько блоков правил. С целью удобства и предоставления возможности декомпозиции базы правил на отдельные модули разрешается использовать несколько блоков правил. При этом каждый блок правил должен иметь уникальное имя.

Правила должны быть определены между ключевыми словами RULEBLOCK и END_RULEBLOCK следующим образом.

```
RULEBLOCK имя_блока_правил
   определение_операции;
   [ метод_активизации;]
   метод_аккумуляции;
   правила;
END_RULEBLOCK
```

Внутри блока правил используются нечеткие операции

```
определение_операции ::= операция : алгоритм
```

Чтобы соответствовать законам Де Моргана следует использовать парные алгоритмы для операторов AND и OR, например, если для оператора AND используется алгоритм MIN, то для оператора OR следует использовать алгоритм MAX. Соответствующие парные алгоритмы определены в табл. 8.3.

Таблица 8.3. *Парные алгоритмы для операций*

Оператор OR (логическое "ИЛИ")		**Оператор AND (логическое "И")**	
Ключевое слово для алгоритма	**Алгоритм**	**Ключевое слово для алгоритма**	**Алгоритм**
MAX	$\max\{\mu_1(x), \mu_2(x)\}$	MIN	$\min\{\mu_1(x), \mu_2(x)\}$
ASUM	$\mu_1(x)+\mu_2(x) - \mu_1(x)\cdot\mu_2(x)$	PROD	$\mu_1(x)\cdot\mu_2(x)$
BSUM	$\min\{1, \mu_1(x) + \mu_2(x)\}$	BDIF	$\max\{0, \mu_1(x)+\mu_2(x)-1\}$

Ниже представлен пример записи блоков правил:

```
RULEBLOCK первый
   AND : MIN;
   ...
END_RULEBLOCK
RULEBLOCK второй
   AND : PROD;
   ...
END_RULEBLOCK
```

Метод активизации определяется после ключевого слова `ACT` следующим образом:

```
ACT : метод_активизации;
```

Методы активизации, которые допустимы в языке FCL, определены в табл. 8.4. При этом следует помнить, что метод активизации не применим для одноэлементного множества.

Таблица 8.4. *Методы активизации*

Название метода	Ключевое слово для метода активизации	Алгоритм
Произведение	`PROD`	$\mu_1(x)\cdot\mu_2(x)$
Минимум	`MIN`	$\min\{\mu_1(x), \mu_2(x)\}$

Метод аккумуляции определяется после ключевого слова `ACCU` следующим образом:

```
ACCU : метод_аккумуляции;
```

Методы аккумуляции, которые допустимы в языке FCL, определены в табл. 8.5.

Таблица 8.5. *Методы аккумуляции*

Название метода	Ключевое слово для метода аккумуляции	Формула
Максимум	`MAX`	$\max\{\mu_1(x), \mu_2(x)\}$
Ограниченная сумма	`BSUM`	$\min\{1, \mu_1(x) + \mu_2(x)\}$
Нормализованная сумма	`NSUM`	$\dfrac{\mu_1(x)+\mu_2(x)}{\max\{1, \max\limits_{y\in X}\{\mu_1(y)+\mu_2(y)\}\}}$

Входами блока правил являются лингвистические переменные со своими множествами лингвистических термов. Каждый из термов имеет соответствующие значения функции принадлежности.

Правила определяются внутри блока правил. Каждое правило должно иметь в качестве своего имени уникальный номер внутри блока. Этот номер записывается после ключевого слова RULE и заканчивается двоеточием.

```
RULE номер_правила : IF условие THEN заключение
[WITH весовой_коэффициент];
```

Само правило должно начинаться ключевым словом IF, после которого записывается условие правила. После условия должно следовать заключение, которое записывается после ключевого слова THEN.

В одном правиле допускается использовать несколько отдельных подусловий и входных переменных, чтобы иметь возможность оперировать различными нечеткими степенями принадлежности, которые могут быть включены в нечеткий функциональный блок FFB. Все такие подусловия должны быть записаны между ключевыми словами IF и THEN и соединены логическими операциями с ключевыми словами AND, OR, NOT.

Приоритет этих логических операций определяется согласно табл. 8.6.

Таблица 8.6. *Приоритет логических операций*

Приоритет	Операция	Приоритет	Операция
1	Скобки ()	3	AND (И)
2	NOT (НЕ)	4	OR (ИЛИ)

Ниже приводится простой пример записи некоторого правила:

```
RULE 1 : IF подусловие1 AND переменная1 OR переменная2 THEN заключение;
```

В соответствии с Базовым Уровнем согласованности операция OR в условии некоторого правила может быть заменена определением двух отдельных правил. Например, правило

```
RULE 3 : IF подусловие1 OR подусловие2 THEN заключение;
```

можно заменить на правила:

```
RULE 3a : IF условие1 THEN заключение;
RULE 3b : IF условие2 THEN заключение;
```

Подусловия начинаются с имени некоторой лингвистической переменной, после которого следует ключевое слово IS с необязательным ключевым словом NOT. После этих ключевых слов записывается один лингвистический терм для лингвистической переменной, которая определяется в соответствующем условии:

```
подусловие := лингвистическая_переменная IS [NOT] лингвистический_терм
```

Примечание

Используемые в условии лингвистические термы должны соответствовать лингвистической переменной в том же условии. При этом используемый терм должен быть заранее определен с ключевым словом TERM.

Ниже приводятся примеры записи подусловий в форме структурируемого текста:

```
температура IS горячая
температура IS NOT горячая
```

Допускается использование ключевого слова NOT перед записью подусловия. В этом случае следует применять круглые скобки, например:

```
IF NOT (температура IS горячий) THEN...
```

Заключение может быть разделено на несколько подзаключений и выходные переменные. Подзаключения начинаются с имени некоторой лингвистической переменной, после которого следует ключевое слово IS с одним лингвистическим термом для этой лингвистической переменной:

```
подзаключение := лингвистическая_переменная IS лингвистический_терм
```

Ниже приводится пример нескольких подзаключений, записанных в одну строку:

```
IF температура IS холодная AND давление IS низкое THEN переменная1,
клапан1 IS открыт_на_наполнение, клапан2 IS закрыт;
```

или в несколько строк:

```
IF температура IS холодная AND давление IS низкое
THEN переменная1,
   клапан1 IS открыт_на_наполнение,
   клапан2 IS закрыт;
```

При необходимости, но не обязательно, для каждого подзаключения можно задать весовой коэффициент, который представляет собой некоторое число типа данных REAL и со значением из интервала [0, 1]. Это число записывается после ключевого слова WITH:

```
IF условие THEN подзаключение1 [WITH весовой_коэффициент],
подзаключение2;
```

Выражение предназначено для уменьшения степени принадлежности (значения функции принадлежности) подзаключения посредством умножения результата подзаключения на этот весовой коэффициент.

Чтобы иметь возможность внешнего изменения параметров приложений нечеткого управления, весовой коэффициент может быть задан как переменная. При этом соответствующая переменная должна быть объявлена в секции VAR_INPUT. Это предоставит возможность изменить весовой коэффициент в ходе выполнения программы и тем самым адаптировать программу нечеткого управления к особенностям решаемой задачи.

Если подзаключение не имеет записи с ключевым словом WITH, то используется значение весового коэффициента по умолчанию, которое равно 1. Ниже приводится пример записи постоянного весового коэффициента:

```
IF температура IS холодная AND давление IS низкое
THEN   клапан1 IS открыт_на_наполнение WITH 0.5,
   клапан2 IS закрыт;
```

Пример записи переменного весового коэффициента:

```
VAR_INPUT
вес_правила1 : REAL := 0.8;
END_VAR
RULEBLOCK правило_для_температуры
   RULE 1: IF температура IS холодная AND давление IS низкое
   THEN   клапан1 IS открыт_на_наполнение WITH вес_правила1,
   ...
END_RULEBLOCK
```

Простой пример записи модели нечеткого управления с использованием нотации языка FCL

```
FUNCTION_BLOCK Нечеткий_ФБ
VAR_INPUT
   температура : REAL;
   давление : REAL;
END_VAR
VAR_OUTPUT
   клапан : REAL;
END_VAR
FUZZIFY температура
   TERM холодная := (-10°C, 1.0) (10°C, 0.0);
   TERM горячая := (20°C, 0.0) (70°C, 1.0);
END_FUZZIFY
FUZZIFY давление
   TERM низкое := (55, 1.0) (95, 0.0);
   TERM высокое := (55, 0.0) (95, 1.0);
END_FUZZIFY
DEFUZZIFY клапан
```

```
   TERM открыт_на_слив := -100;
   TERM закрыт := 0;
   TERM открыт_на_наполнение := 100;
   ACCU : MAX;
   METHOD : COGS;
   DEFAULT := 0;
END_DEFUZZIFY
RULE_BLOCK номер_1
   AND : MIN;
   RULE 1 : IF температура IS холодная AND давление IS низкое
   THEN клапан IS открыт_на_слив
   RULE 2 : IF температура IS холодная AND давление IS высокое
   THEN клапан IS закрыт WITH 0.8;
   RULE 3 : IF температура IS горячая AND давление IS низкое
   THEN клапан IS закрыт;
   RULE 4 : IF температура IS горячая AND давление IS высокое
   THEN клапан IS открыт_на_наполнение;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

Необязательные параметры (Optional parameters)

При построении нечетких моделей может возникнуть необходимость определить некоторую дополнительную информацию для той или иной системы, чтобы иметь возможность наиболее адекватного преобразования приложений нечеткого управления. Такая дополнительная информация может относиться к соответствующему элементу языка FCL и помещаться между ключевыми словами `OPTIONS` и `END_OPTIONS` следующим образом.

```
OPTIONS
   параметры_спецификации_приложения
END_OPTIONS
```

Соответствующие элементы языка FCL должны удовлетворять правилам согласованности классов открытого уровня (правило 6).

Согласованность классов языка FCL

Модели систем управления, записанные в нотации языка FCL, должны быть согласованными по так называемым *уровням согласованности классов*. Эти уровни согласованности изображены на рис. 8.5 в форме так называемой *диаграммы па-*

кетов в нотации языка UML. Эта диаграмма представляет данные уровни в качестве основных подпакетов языка FCL, между которыми имеются зависимости, изображенные пунктирными линиями со стрелками.

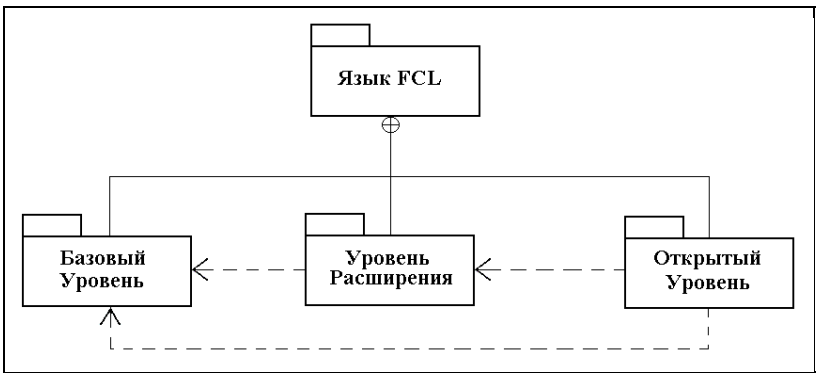


Рис. 8.5. Уровни согласованности классов языка FCL

Примечание

В Стандарте IEC 1131-7 уровни согласованности классов изображены иным способом, хотя по своей сути они отражают структуру языка FCL в форме именно *пакетов*, как они определяются в нотации унифицированного языка моделирования UML. Желание более точно отобразить эту структуру и семантику зависимостей между пакетами послужило основной причиной для представления иерархии уровней согласованности в более адекватном виде — в форме диаграммы пакетов.

Согласно Стандарту IEC 1131-7, уровни согласованности классов представляют собой иерархию из трех уровней.

- Базовый Уровень, включающий в себя определения функционального блока и типы данных.
- Уровень Расширения, включающий в себя необязательные параметры.
- Открытый Уровень, включающий в себя дополнительные параметры.

Модель системы управления, записанная в нотации языка FCL, должна соответствовать настоящим требованиям Стандарта, которые представлены в форме следующих *Правил согласованности*:

1. Для реализации функциональности в модели нечеткого управления следует использовать Функциональный Блок, как он определен в настоящем Стандарте. Поэтому определения Функциональных Блоков и Типов Данных для входных и выходных параметров функционального блока должны согласовываться с базовой нотацией языка FCL.

2. Все параметры функциональности модели нечеткого управления, определенные в табл. 8.7, должны быть заданы в соответствии с базовой нотацией языка FCL. Указанная таблица определяет множество элементов Базового Уровня, которые должны содержать в общем случае все модели стандартных систем.
3. Подмножество элементов Уровня Расширения, определенных в табл. 8.8, составляет дополнительные элементы, которые можно использовать в необязательном порядке. Модели систем в нотации языка FCL должны в точности отвечать соответствующим требованиям Стандарта IEC 1131-7. Все необязательные параметры должны относиться к Уровню Расширения, а перечень используемых необязательных параметров в форме табл. 8.8 должен стать частью документации на модель системы.
4. Другие параметры, выходящие за пределы Базового Уровня и Уровня Расширения, допускается использовать в модели таким образом, чтобы эти параметры не имели таких же или аналогичных параметров среди стандартных элементов функциональности языка FCL. Тем самым должны быть исключены любые возможные недоразумения. Эти дополнительные параметры должны относиться к Открытому Уровню, а перечень используемых дополнительных параметров в форме табл. 8.9 должен стать частью документации на модель системы.
5. Перенос прикладных программ между различными системами нечеткого управления следует осуществлять в текстовой форме в соответствии с нотацией языка FCL, определенной в Стандарте IEC 1131-7. При этом все системы нечеткой логики, претендующие на соответствие этому Стандарту, должны обязательно использовать в качестве своего входного и выходного формата нотацию Базового Уровня языка FCL (см. табл. 8.7), включая определение Функционального Блока и Типов Данных из Стандарта IEC 1131-3, а также Уровень Расширения с дополнительными средствами (см. табл. 8.8) и Открытый Уровень с необязательными средствами (см. табл. 8.9).

Таблица 8.7. *Элементы языка FCL Базового Уровня (обязательные)*

Элемент языка FCL	Ключевое слово	Примечание
Объявление функционального блока	`VAR_INPUT`, `VAR_OUTPUT`	Содержит входные и выходные переменные
Функция принадлежности	Входная переменная : `TERM`	Максимум — 3 точки (координата степени принадлежности равна 0 или 1)
	Выходная переменная : `TERM`	Только одноэлементное множество
Агрегирование условий	Оператор : `AND`	Алгоритм : `MIN`

Таблица 8.7 *(окончание)*

Элемент языка FCL	Ключевое слово	Примечание
Активизация	—	Не применяется в случае использования только одноэлементных множеств
Аккумуляция (результат агрегирования)	Оператор : `ACCU`	Алгоритм : `MAX`
Дефаззификация	`METHOD`	Алгоритм : `COGS`
Значение по умолчанию		Чтобы использовать этот параметр, в Базовом Классе должны быть подготовлены предварительные определения
Условие	`IF... IS ...`	*N* подусловий
Заключение	`THEN`	Только одно подзаключение
Весовой коэффициент	`WITH`	Только значение

Элементы, представленные в табл. 8.8, расширяют множество параметров, которые являются необязательными для представления в стандартной модели нечеткого управлении (например, для логической операции "И" может быть использован алгоритм `PROD` или `BDIF`).

Таблица 8.8. *Элементы языка FCL Уровня Расширения (необязательные)*

Элемент языка FCL	Ключевое слово	Примечание
Объявление функционального блока	`VAR`	Содержит локальные переменные
Функция принадлежности	Входная переменная :`TERM`	Максимум — 4 точки (координата степени принадлежности равна 0 или 1)
	Выходная переменная :`TERM`	Максимум — 4 точки (координата степени принадлежности равна 0 или 1)
Агрегирование условий	Оператор : `AND`	Алгоритм : `PROD`, `BDIF`
	Оператор : `OR`	Алгоритм : `ASUM`, `BSUM`
	Оператор : `NOT`	1 — `{аргумент}`

Таблица 8.8 *(окончание)*

Элемент языка FCL	Ключевое слово	Примечание
	Скобки	`()`
Активизация	Оператор : `ACT`	Алгоритм : `MIN`, `PROD`
Аккумуляция	Оператор : `ACCU`	Алгоритм : `BSUM`, `NSUM`
Метод дефаззификации	Оператор : `METHOD`	Алгоритм : `COG`, `COA`, `LM`, `RM`
Значение по умолчанию	`DEFAULT`	`NC`, значение
Условие	`IF`	*n* подусловий, *N* входных переменных
Заключение	`THEN`	*n* подзаключений, *N* выходных переменных
Весовой коэффициент	`WITH`	Значение назначается переменной в секции объявления `VAR_INPUT...END_VAR`

В табл. 8.9 представлен пример списка элементов языка FCL Открытого Уровня. Этот список также должен быть частью документации на систему.

Таблица 8.9. *Пример списка элементов языка FCL Открытого Уровня*

Произвольные функции принадлежности для входов/выходов (например, функция Гаусса, экспоненциальная функция)
Более чем 4 точки для функции принадлежности
Координата значения функции принадлежности от 0 до 1
Произвольные элементы, заданные программистом

Список проверки данных

Этот список проверки данных должен прилагаться к технической документации. В этом списке разработчик программных контроллеров, средств нечеткого программирования и прикладных программ должен описать специальные параметры реализации своей системы нечеткого управления. Чтобы обеспечить передачу приложений нечеткого управления между системами различных производителей, необходимо представить следующий Список проверки данных (табл. 8.10), который является средством установления правильности возможного переноса программ.

Таблица 8.10. *Список проверки данных*

Технические данные	**Значения, установленные разработчиком (примеры)**
Типы данных входов и выходов функционального блока	`REAL, INT`
Строки комментариев в программах на языке FCL	`YES, NO`
Длина идентификаторов (например, имена переменных, блоков правил, термов)	6, 8
Максимальное число входных переменных для фаззификации	6, 8
Максимальное число функций принадлежности термов для входной переменной	5, 7
Максимальное общее число функций принадлежности термов для всех входных переменных	30, 56
Максимальное число точек функции принадлежности для каждого из термов входных переменных	3, 4, 10
Максимальное общее число точек функции принадлежности для всех термов входных переменных	90, 224
Максимальное число выходных переменных для дефаззификации	6, 8
Максимальное число функций принадлежности термов для выходной переменной	5, 7
Максимальное общее число функций принадлежности термов для всех выходных переменных	30, 56
Максимальное число точек функции принадлежности для каждого из термов выходных переменных	1, 4, 10
Максимальное общее число точек функции принадлежности для всех термов выходных переменных	90, 224
Максимальное число блоков правил	1, 10
Максимальное число правил в блоке	10
Максимальное число подусловий в правиле	4, 10
Максимальное число всех правил	15
Максимальное число подзаключений в правиле	4
Глубина вложенности скобок "`()`"	1, 3

8.3. Пример разработки и записи нечетких моделей на языке FCL

Нечеткая модель управления смесителем воды при принятии душа

Соответствующий пример был рассмотрен ранее в *главе 7*. Если в качестве алгоритма нечеткого вывода используется алгоритм Мамдани, то методом активизации будет MIN, который рассчитывается по формуле (7.6). Для аккумуляции заключений правил будем использовать метод MAX, который наиболее часто применяется в случае схемы нечеткого вывода методом Мамдани. Наконец, в качестве метода дефаззификации будем использовать метод центра тяжести, который задается с помощью ключевого слова COG.

Таким образом, нечеткая модель управления смесителем воды может быть записана в нотации языка FCL следующим образом:

```
FUNCTION_BLOCK смеситель_воды
VAR_INPUT
   температура: REAL;
END_VAR
VAR_OUTPUT
   угол: REAL;
END_VAR
FUZZIFY температура
   TERM холодная := (10, 1) (30, 0);
   TERM прохладная := (20, 0) (35, 1) (50, 0);
   TERM теплая := (40, 0) (50, 1) (60, 0);
   TERM не_очень_горячая := (50, 0) (60, 1) (70, 0);
   TERM горячая := (60, 0) (70, 1);
END_FUZZIFY
DEFUZZIFY угол
   TERM большой_влево := (-72, 1) (-36, 0);
   TERM небольшой_влево := (-54, 0) (-27, 1) (0, 0);
   TERM нуль := (-18, 0) (0, 1) (18, 0);
   TERM небольшой_вправо := (0, 0) (27, 1) (54, 0);
   TERM большой_вправо := (36, 0) (72, 1);
   ACCU: MAX;
```

```
    METHOD : COG
    DEFAULT := 0
  END_DEFUZZIFY
  RULEBLOCK Номер_1
    AND : MIN;
    RULE 1: IF температура IS горячая THEN угол IS большой_вправо;
    RULE 2: IF температура IS не_очень_горячая THEN угол IS неболь-
    шой_вправо;
    RULE 3: IF температура IS теплая THEN угол IS нуль;
    RULE 4: IF температура IS прохладная THEN угол IS небольшой_
    влево"
    RULE 5: IF температура IS холодная THEN угол IS большой_ влево;
  END_RULEBLOCK
END_FUNCTION_BLOCK
```

Нечеткая модель управления кондиционером воздуха в помещении

Соответствующий пример был рассмотрен ранее в *главе 7*. Если в качестве алгоритма нечеткого вывода используется алгоритм Мамдани, то методом активизации будет `MIN`, который рассчитывается по формуле (7.6). Для аккумуляции заключений правил будем использовать метод `MAX`, который наиболее часто применяется в случае схемы нечеткого вывода методом Мамдани. Наконец, в качестве метода дефаззификации будем использовать метод центра площади, который задается с помощью ключевого слова `COA`.

Таким образом, нечеткая модель управления кондиционером может быть записана в нотации языка FCL следующим образом:

```
  FUNCTION_BLOCK кондиционер
  VAR_INPUT
    температура: REAL;
    скорость_изм: REAL;
  END_VAR
  VAR_OUTPUT
    угол: REAL;
  END_VAR
  FUZZIFY температура
    TERM NB := (5, 1) (10, 0);
    TERM NS := (5, 0) (10, 1) (16, 0);
```

```
   TERM Z := (15, 0) (17.5, 1) (20, 0);
   TERM PS := (19, 0) (25, 1) (30, 0);
   TERM PB := (25, 0) (30, 1);
END_FUZZIFY
FUZZIFY скорость_изм
   TERM NS := (-1, 1) (0, 0);
   TERM Z := (-0.5, 0) (0, 1) (0.5, 0);
   TERM PS := (0, 0) (1, 1);
END_FUZZIFY
DEFUZZIFY угол
   TERM NB := (-67.5, 1) (-45, 0);
   TERM NM := (-67.5, 0) (-45, 1) (-22.5, 0);
   TERM NS := (-45, 0) (-22.5, 1) (0, 0);
   TERM Z := (-10, 0) (0, 1) (10, 0);
   TERM PS := (0, 0) (22.5, 1) (45, 0);
   TERM PM := (22.5, 0) (45, 1) (67.5, 0);
   TERM PB := (45, 0) (67.5, 1);
   ACCU: MAX;
   METHOD : COA
   DEFAULT := 0
END_DEFUZZIFY
RULEBLOCK Номер_1
   AND : MIN;
   RULE 1: IF температура IS PB AND скорость_изм IS PS
   THEN угол IS NB;
   RULE _2: IF температура IS PB AND скорость_изм IS NS
   THEN угол IS NS:
   RULE _3: IF температура IS PS AND скорость_изм IS PS
   THEN угол IS NM;
   RULE _4: IF температура IS PS AND скорость_изм IS NS
   THEN угол IS Z;
   RULE _5: IF температура IS NB AND скорость_изм IS NS
   THEN угол IS PB;
   RULE _6: IF температура IS NB AND скорость_изм IS PS
   THEN угол IS PS;
   RULE _7: IF температура IS NS AND скорость_изм IS NS
```

```
      THEN угол IS PM;
      RULE _8: IF температура IS NS AND скорость_изм IS PS
      THEN угол IS Z;
      RULE _9: IF температура IS PB AND скорость_изм IS Z
      THEN угол IS NM;
      RULE _10: IF температура IS PS AND скорость_изм IS Z
      THEN угол IS NS;
      RULE _11: IF температура IS NB AND скорость_изм IS Z
      THEN угол IS PM;
      RULE _12: IF температура IS NS AND скорость_изм IS Z
      THEN угол IS PS;
      RULE _13: IF температура IS Z AND скорость_изм IS PS
      THEN угол IS NS;
      RULE _14: IF температура IS Z AND скорость_изм IS NS
      THEN угол IS PS;
      RULE _15: IF температура IS Z AND скорость_изм IS Z
      THEN угол IS Z;
   END_RULEBLOCK
END_FUNCTION_BLOCK
```

Нечеткая модель управления контейнерным краном

Соответствующий пример был рассмотрен ранее в *главе 7*. Если в качестве алгоритма нечеткого вывода используется алгоритм Мамдани, то методом активизации будет `MIN`, который рассчитывается по формуле (7.6). Для аккумуляции заключений правил будем использовать метод `MAX`, который наиболее часто применяется в случае схемы нечеткого вывода методом Мамдани. Наконец, в качестве метода дефаззификации будем использовать метод центра тяжести для одноэлементных множеств, который задается с помощью ключевого слова `COGS`.

Таким образом, нечеткая модель управления контейнерным краном может быть записана в нотации языка FCL следующим образом:

```
FUNCTION_BLOCK контейнерный_кран
VAR_INPUT
      расстояние: REAL;
      угол: REAL;
```

```
END_VAR
VAR_OUTPUT
      мощность: REAL;
END_VAR
FUZZIFY расстояние
      TERM очень_далекое := (-5, 1) (0, 0);
      TERM нуль := (-5, 0) (0, 1) (5, 0);
      TERM близкое := (0, 0) (5, 1) (10, 0);
      TERM среднее := (5, 0) (10, 1) (22, 0);
      TERM далекое := (10, 0) (22, 1);
END_FUZZIFY
FUZZIFY угол
      TERM отр_большой := (-50, 1) (-5, 0);
      TERM отр_малый := (-50, 0) (-5, 1) (0, 0);
      TERM нуль := (-5, 0) (0, 1) (5, 0);
      TERM пол_малый := (0, 0) (5, 1) (50, 0);
      TERM пол_большой := (5, 0) (50, 1);
END_FUZZIFY
DEFUZZIFY мощность
      TERM отр_высокая := -27;
      TERM отр_средняя := -12;
      TERM нуль := 0;
      TERM пол_средняя := 12;
      TERM пол_высокая := 27;
      ACCU: MAX;
      METHOD : COGS
      DEFAULT := 0
END_DEFUZZIFY
RULEBLOCK Номер_1
      AND : MIN;
      RULE 1: IF расстояние IS далекое AND угол IS ноль
      THEN мощность IS пол_средняя;
      RULE 2: IF расстояние IS далекое AND угол IS отр_малый
      THEN мощность IS пол_большая;
      RULE 3: IF расстояние IS далекое AND угол IS отр_большой
      THEN мощность IS пол_средняя;
```

```
        RULE 4: IF расстояние IS среднее AND угол IS отр_малый
        THEN мощность IS отр_средняя;
        RULE 5: IF расстояние IS близкое AND угол IS пол_малый
        THEN мощность IS пол_средняя;
        RULE 6: IF расстояние IS ноль AND угол IS ноль
        THEN мощность IS ноль;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

Поскольку этот функциональный блок записан в нотации языка FCL, который является языком программирования высокого уровня, то рассмотренная программа по своей структуре полностью соответствует некоторой процедуре. Формальными параметрами такой процедуры являются входные переменные блока правил, а возвращаемыми значениями — значения выходных переменных после дефаззификации.

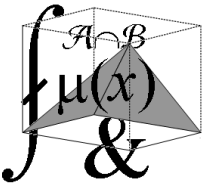
Вызов такой процедуры в некоторой внешней программе может быть реализован, например, следующим образом:

```
контейнерный_кран (расстояние :=12, угол :=15°);
varP := контейнерный_кран.мощность;
```

В этом случае переменной с именем `varP` присваивается значение нечеткого вывода для блока правил, определенного в Функциональном Блоке `контейнерный_кран`, при конкретных значениях входных переменных: 12 *м* для расстояния и 15° для угла.

Особенности реализации систем нечеткого вывода и выполнение соответствующих расчетов для конкретных значений входных переменных и параметров функциональных блоков в инструментальных средах MATLAB и fuzzyTECH будут рассмотрены далее в *частях II* и *III* книги.

Глава 9



Основы общей теории нечеткой меры

В начале книги рассматривалось несколько содержательных примеров, которые иллюстрировали особенности нечеткого представления знаний и различия в нечетком и вероятностном описании различных явлений и объектов. В настоящей главе излагаются основы теории нечеткой меры, которая позволяет с единых концептуальных позиций рассматривать широкий класс задач нечеткого моделирования, устанавливая при этом взаимосвязь между теорией нечетких множеств, теорией вероятностей и теорией свидетельств.

Необходимость введения в рассмотрение и изучение различных вариантов нечетких мер обуславливается наличием различных аспектов неопределенности, которые встречаются при решении задач нечеткого моделирования. В этом случае построение адекватных нечетких моделей оказывается возможным на основе анализа доступной информации с точки зрения общей концепции нечеткой меры и представления ее в виде, в наибольшей степени соответствующем получению достоверного решения задачи.

9.1. Нечеткие меры и их основные свойства

Нечеткие меры предназначены для представления неточной или неясной информации относительно принадлежности тех или иных элементов одному или нескольким обычным множествам. Чтобы пояснить эту идею, рассмотрим следующий простой пример.

П р и м е р 9.1. Предположим, имеется два утверждения в форме обычных высказываний: "*Движущийся по автостраде автомобиль белого или серого цвета*" и "*Движущийся по автостраде автомобиль белого или серебристого цвета*", принадлежащие двум различным наблюдателям. Вопрос заключается в том, чтобы по возможности точно определить цвет движущегося по автостраде автомобиля.

Если предположить, что оба эти высказывания истинны, то достаточно преобразовать эти высказывания, представив их в форме некоторого предиката $P(x)$, который бы описывал каждое из них. В этом случае обозначим множество авто-

мобилей белого цвета через A_1, множество автомобилей серого цвета через A_2, а множество автомобилей серебристого цвета через A_3. Пусть предикат $P(x)$ описывает цвет автомобиля таким образом, что он принимает значение "*истина*" в случае, когда автомобиль x принадлежит одному из введенных в рассмотрение множеств. А именно, если цвет автомобиля белый, то $P(x)$ принимает значение "*истина*" только при $x \in A_1$, если цвет автомобиля серый, то $P(x)$ принимает значение "*истина*" только при $x \in A_2$ и, наконец, если цвет автомобиля серебристый, то $P(x)$ принимает значение "*истина*" только при $x \in A_3$.

Тогда истинность первого высказывания будет соответствовать истинности данного предиката $P(x)$ при $x \in A_1 \cup A_2$, истинность второго из высказываний будет соответствовать истинности данного предиката $P(x)$ при $x \in A_1 \cup A_3$. Что касается поставленного вопроса, то истинность соответствующего высказывания определяется множеством: $(A_1 \cup A_2) \cap (A_1 \cup A_3)$, которое равно A_1 в случае, когда три множества A_1, A_2, A_3 взаимно не пересекаются. Таким образом, ответом на поставленный вопрос будет утверждение о том, что "*Движущийся по автостраде автомобиль белого цвета*".

Ситуация существенно усложняется в ситуации, когда исходные высказывания сформулированы в форме нечетких высказываний. Именно в этом случае наиболее адекватный ответ позволяет дать теория нечеткой меры. Рассмотрение основ этой теории начнем с формального определения аксиоматики нечеткой меры, которая впоследствии конкретизируется с целью систематизации отдельных мер возможности и необходимости, доверия и правдоподобия, а также классической вероятности.

Общее определение нечеткой меры

Формальное определение нечеткой меры основывается на рассмотрении некоторого произвольного универсума $\boldsymbol{X}$ и множества всех его подмножеств (булеана) $\mathbb{B}(\boldsymbol{X})$.

Нечеткая мера. *Нечеткой мерой* или *квазимерой* называется произвольное отображение $G: \mathbb{B}(\boldsymbol{X}) \to [0, 1]$, удовлетворяющее следующим условиям *(аксиомам нечеткой меры)*:

$$G(\varnothing)=0, \qquad G(\boldsymbol{X})=1, \qquad (\textit{ограниченность}) \tag{9.1}$$

если $A, B \in \mathbb{B}(\boldsymbol{X})$ и $A \subset B$, то

$$G(A) \le G(B), \qquad (\textit{монотонность}) \tag{9.2}$$

если $F_i \in \mathbb{B}(\boldsymbol{X})$ и $\{F_i\}_{i \in \mathbb{N}}$ является монотонной относительно включения последовательностью множеств, то

$$\lim_{i \to \infty} G(F_i) = G(\lim_{i \to \infty} F_i) \qquad (\textit{непрерывность}) \tag{9.3}$$

Примечание

Формальное определение нечеткой меры в форме аксиом (9.1)—(9.3) не удовлетворяет строгим критериям конструктивности, принятым в некоторых школах

математики. С этой целью вместо булеана $\mathbb{B}(\boldsymbol{X})$ необходимо использовать более "хорошую" конструкцию — так называемую σ-*алгебру* или *Борелевское поле* подмножеств $\boldsymbol{X}$. Хотя булеан является более общим понятием, чем σ-алгебра подмножеств, его использование для ограниченных или компактных универсумов не приводит к концептуальным противоречиям.

По аналогии с вероятностным пространством общее определение нечеткой меры позволяет определить математическую структуру, соответствующую пространству с нечеткой мерой.

Пространством с нечеткой мерой называется математическая структура вида $(\boldsymbol{X}, \Im, G)$, где $\boldsymbol{X}$ — базовое множество (универсум); $\Im$ — σ-алгебра на $\boldsymbol{X}$; G — некоторая нечеткая мера, т. е. функция множества, удовлетворяющая аксиомам (9.1)—(9.3).

С учетом сделанного выше примечания и с целью систематизации последующего изложения будем использовать конкретизацию булеана в форме σ-алгебры $\Im$ на $\boldsymbol{X}$. Эти обозначения будут применяться и далее.

Математическая структура пространства с нечеткой мерой $(\boldsymbol{X}, \Im, G)$ является слишком абстрактной для своего использования и допускает различные конкретизации на основе введения дополнительных аксиом, усиливающих отдельные свойства функции G. Получаемые в результате математические структуры являются порожденными от математической структуры пространства с нечеткой мерой и наследуют все характерные свойства последней.

Меры доверия и правдоподобия

Конкретизацией общего понятия нечеткой меры являются мера доверия (от англ. *belief*) и мера правдоподобия (от англ. *plausibility*).

М е р а д о в е р и я. *Мерой доверия* называется такая нечеткая мера G_*, для которой вместо аксиомы (9.3) используется более сильная аксиома:

если $A_1, A_2, \ldots, A_n \in \Im$, то

$$G_*(A_1 \cup A_2 \cup \ldots \cup A_n) \geq \sum_{i=1}^{n} G_*(A_i) - \sum_{i<j} G_*(A_i \cap A_j) + \ldots \qquad (9.4)$$

$$\ldots + (-1)^{n+1} G_*(A_1 \cap A_2 \cap \ldots \cap A_n).$$

Меру доверия иногда называют также *нижней вероятностью* и обозначают через $b(A)$ или $bel(A)$.

Пространством с мерой доверия называется математическая структура $(\boldsymbol{X}, \Im, G_*)$, функция нечеткой меры G_* которого удовлетворяет аксиомам (9.1), (9.2) и (9.4).

В случае 2-х множеств (n=2) формула (9.4) превращается в формулу: $G_*(A_1 \cup A_2) \geq G_*(A_1) + G_*(A_2) - G_*(A_1 \cap A_2)$.

М е р а п р а в д о п о д о б и я. *Мерой правдоподобия* называется такая нечеткая мера G^*, для которой вместо аксиомы (9.3) используется более сильная аксиома:

если $A_1, A_2, ..., A_n \in \Im$, то

$$G^*(A_1 \cap A_2 \cap ... \cap A_n) \le \sum_{i=1}^{n} G^*(A_i) - \sum_{i<j} G^*(A_i \cup A_j) + ... \qquad (9.5)$$

$$... + (-1)^{n+1} G^*(A_1 \cup A_2 \cup ... \cup A_n).$$

Меру правдоподобия иногда называют также *верхней вероятностью* и обозначают через $pl(A)$.

Пространством с мерой правдоподобия называется математическая структура $(X, \Im, G^*)$, функция нечеткой меры G^* которой удовлетворяет условиям (9.1), (9.2) и (9.5).

В случае 2-х множеств (n=2) формула (9.5) превращается в формулу: $G^*(A_1 \cap A_2) \le G^*(A_1) + G^*(A_2) - G^*(A_1 \cup A_2)$.

Эти две меры удовлетворяют свойству обратного дополнения: $G^*(A) = 1 - G_*(\overline{A})$ и $G_*(A) = 1 - G^*(\overline{A})$.

Аксиоматика пространств с мерой доверия и мерой правдоподобия остается все еще слишком бедной для их практического использования в задачах нечеткого моделирования. Наибольшее применение при построении нечетких моделей находят меры возможности, необходимости и вероятности, определения которых приводятся далее в этой главе.

Меры возможности, необходимости и вероятности

М е р а в о з м о ж н о с т и. *Мерой возможности* называется такая нечеткая мера G^0, для которой вместо аксиомы (9.3) используется более сильная аксиома:

$$\text{если } A_1, A_2, ..., A_i, ... \in \Im, \text{ то } G^0(A_1 \cup A_2 \cup ... \cup A_i ...) = \sup_{i \in N} G^0(A_i) \qquad (9.6)$$

Пространством с мерой возможности называется математическая структура $(X, \Im, G^0)$, функция нечеткой меры G^0 которой удовлетворяет условиям (9.1), (9.2) и (9.6).

Мера возможности в некотором контексте отражает физические или иные ограничения на значения, которые может принимать та или иная переменная. Так, например, если рассмотреть число пассажиров, которые могут поместиться в обычном легковом автомобиле, то с каждым натуральным числом $n \in \mathbb{N}$ можно связать число из интервала [0, 1], которое содержательно будет соответствовать возможности размещения n пассажиров (не считая водителя) в салоне обычного легкового автомобиля.

Если обозначить эту функцию через $g(x)$, то очевидно, $g(1) = 1$, $g(2) = 1$, $g(3) = 1$, $g(4) = 1$, $g(5) = 0.9$ (бывает ведь и так), $g(6) = 0.5$ (вспомним свою юность), $g(7) = 0.2$ (случалось один раз в салоне ГАЗ 21), $g(8) = g(9) = \ldots = 0$. (Впрочем, кто-нибудь из читателей может не согласиться и будет по своему прав, поскольку оценка возможности — процедура субъективная).

Это распределение возможности может быть использовано для некоторых количественных оценок. Например, если имеется свидетельство, что в проехавшем легковом автомобиле находилось несколько пассажиров, то тем самым нечеткое множество, описывающее свойство "несколько", сопоставляется с рассмотренным распределением возможностей. Это сопоставление может быть выполнено с помощью так называемого нечеткого интеграла, определение которого приводится далее в *разд. 9.2*.

Для меры возможности справедливо следующее свойство: $G^0(A)+G^0(\overline{A})\geq 1$.

М е р а н е о б х о д и м о с т и. *Мерой необходимости* называется такая нечеткая мера G_0, для которой вместо аксиомы (9.3) используется более сильная аксиома:

$$\text{если } A_1, A_2, \ldots, A_i, \ldots \in \Im, \text{ то } G_0(A_1 \cap A_2 \cap \ldots \cap A_i \ldots) = \inf_{i \in N} G_0(A_i). \qquad (9.7)$$

Меру необходимости иногда называют согласованной мерой доверия.

Для меры необходимости справедливы следующие свойства: $G_0(A)+G_0(\overline{A})\leq 1$, а также: $G^0(A)=1-G_0(\overline{A})$ и $G_0(A)=1-G^0(\overline{A})$. Если $G_0(A)>0$, то $G^0(A)=1$. Напротив, если $G^0(A)<1$, то $G_0(A)=0$.

Пространством с мерой необходимости называется математическая структура $(X, \Im, G_0)$, функция нечеткой меры G_0 которого удовлетворяет условиям (9.1), (9.2) и (9.7).

В е р о я т н о с т н а я м е р а. *Вероятностной мерой* называется такая нечеткая мера P, для которой вместо аксиомы (9.3) используется более сильная аксиома:

$$\text{если } A_1, A_2, \ldots, A_i, \ldots \in \Im \text{ и } A_i \cap A_j = \varnothing \ (i \neq j), \text{ то } P(A_1 \cup A_2 \cup \ldots \cup A_i \ldots) = \sum_{i \in N} P(A_i). \qquad (9.8)$$

Пространством с вероятностной мерой или *вероятностным пространством* называется математическая структура $(X, \Im, P)$, функция нечеткой меры P которого удовлетворяет условиям (9.1), (9.2) и (9.8).

Вероятностное пространство обладает наиболее сильной аксиоматикой, обусловленной необходимостью выполнения условия счетной аддитивности (9.8). С одной стороны это способствовало становлению и развитию самостоятельной математической дисциплины — теории вероятностей с весьма обширной областью приложений и глубиной полученных теоретических результатов. Познакомиться с предметом этой дисциплины и методами соответствующих исследований можно, обратившись к специальной литературе.

В контексте нечеткого моделирования следует помнить, что аксиома счетной аддитивности требует, чтобы исследуемый объект или процесс удовлетворял целому ряду требований, в частности, некоторому закону распределения вероятностей. В отдельных случаях, особенно при отсутствии прецедентов или другой

достоверной информации о свойствах исследуемых объектов или систем, подобные предположения могут привести к построению неадекватных математических моделей.

Именно по этой причине необходимо с большой осторожностью подходить к высказываниям типа: "Предположим, что случайная величина подчиняется равномерному закону распределения". Подобные утверждения, особенно при недостатке информации относительно полного пространства событий, могут оказаться причиной получения необоснованных результатов с использованием вероятностных моделей.

С точки зрения прикладного системного анализа в подобных ситуациях при недостатке статистической информации или при отсутствии предпосылок выполнения аксиом вероятностного пространства, целесообразно использовать более слабую аксиоматику пространств с мерой возможности и необходимости. Построенные на их основе нечеткие модели могут оказаться более конструктивными для решения широкого класса практических задач.

λ-нечеткие меры

В завершение рассмотрения нечетких мер приводится их параметрическое представление, которое оказывается очень удобным в случае построения и использования так называемых *адаптивных* нечетких моделей, в которых некоторые модели обладают возможностью итеративного уточнения либо в ходе предварительного обучения или итеративной настройки. Примером подобных моделей являются нечеткие нейронные сети, которые рассматриваются далее в *главе 15*. Здесь приводится лишь формальное определение λ-нечеткой меры, которое необходимо для общей систематизации пространств с нечеткой мерой.

λ-нечеткая мера. *λ-нечеткой мерой* называется такая нечеткая мера P, для которой вместо аксиомы (9.3) используется более сильная аксиома:

$$\text{если } A, B\in\Im \text{ и } A\cap B=\varnothing, \text{ то } G_\lambda(A\cup B)=G_\lambda(A)+G_\lambda(B)+\lambda\cdot G_\lambda(A)\cdot G_\lambda(B), \qquad (9.9)$$

где число $\lambda\in(-1,+\infty)$ называется *параметром нормировки*.

Пространством с λ-нечеткой мерой называется математическая структура $(\boldsymbol{X}, \Im, G_\lambda)$, функция нечеткой меры G_λ которого удовлетворяет условиям (9.1), (9.2) и (9.9).

Пространство с λ-нечеткой мерой $(\boldsymbol{X}, \Im, G_\lambda)$ в свою очередь допускает несколько конкретизаций в зависимости от значения параметра λ. Наиболее важными с точки зрения классификации пространств с нечеткой мерой являются случаи $\lambda\in(-1, 0)$, $\lambda=1$ и $\lambda\in(0,+\infty)$, для которых получаются конкретизации введенных в рассмотрение пространств с нечеткой мерой.

Пространством с λ-нечеткой мерой доверия называется математическая структура $(\boldsymbol{X}, \Im, G_{*\lambda})$, функция нечеткой меры $G_{*\lambda}$ которого удовлетворяет условиям (9.1), (9.2) и дополнительному условию (9.9) при $\lambda\in(0,+\infty)$.

Пространством с λ-нечеткой мерой правдоподобия называется математическая структура $(X, \Im, G^*_\lambda)$, функция нечеткой меры G^*_λ которого удовлетворяет условиям (9.1), (9.2) и дополнительному условию (9.9) при $\lambda \in (-1, 0)$.

Следует заметить, что вероятностное пространство $(X, \Im, P)$ является также пространством с λ-нечеткой мерой при $\lambda=0$.

Классификация пространств с нечеткими мерами

Рассмотренные выше пространства с различными вариантами нечетких мер представляют собой математические структуры, каждую из которых можно считать некоторым абстрактным классом. Важной особенностью подобной точки зрения является определенная взаимосвязь классов пространств с неопределенностью, основанная на их конкретизации посредством усиления соответствующих аксиом.

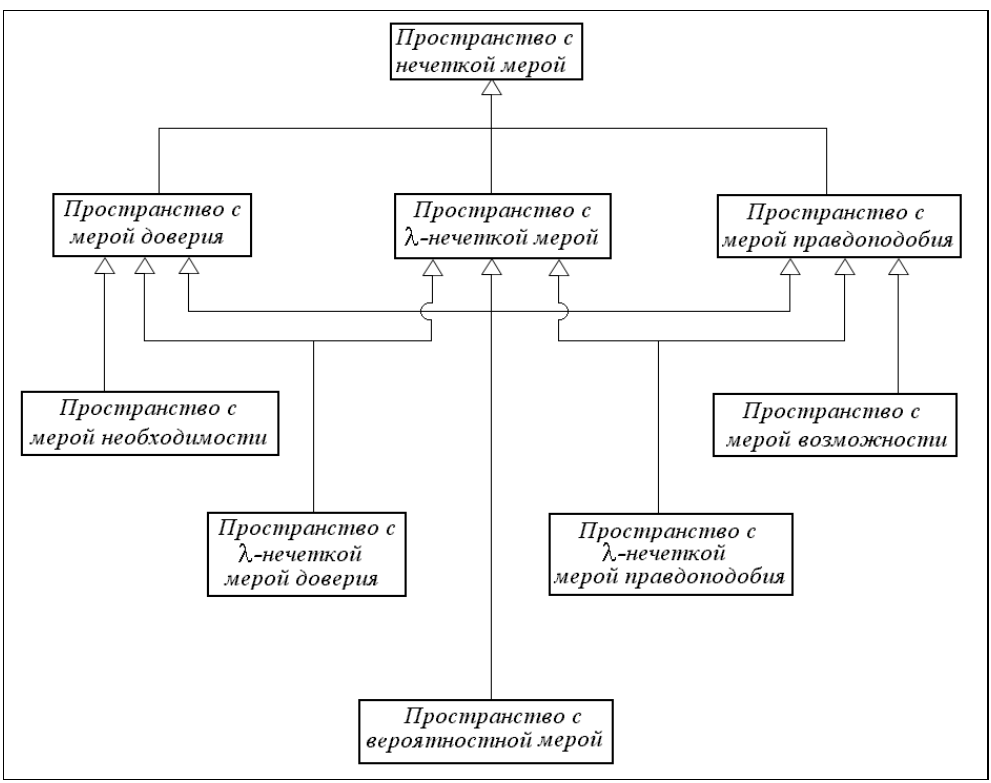


Рис. 9.1. Диаграмма классов математических структур пространств с нечеткой мерой в нотации языка UML

Таким образом, может быть получена классификация различных классов пространств с нечеткой мерой, которая представлена на рис. 9.1 в форме *диаграммы классов* языка UML. На этой диаграмме базовая математическая структура пространства с нечеткой мерой может использоваться в качестве своеобразного шаблона, параметрами которого являются аксиомы рассмотренных нечетких мер. При этом отношение обобщения соответствует усилению отдельных аксиом.

Если между двумя математическими структурами пространств с нечеткой мерой имеется отношение обобщения, то математическая структура нижнего уровня является конкретизацией соответствующей математической структуры верхнего уровня. При этом все математические структуры нижних уровней наследуют аксиоматику соответствующих математических структур верхних уровней.

Так, например, математическая структура пространства с мерой необходимости является конкретизацией или частным случаем математической структуры пространства с мерой доверия. Математическая структура вероятностного пространства является конкретизацией как математической структуры пространства с мерой доверия, так и математической структуры пространства с мерой правдоподобия, тем самым реализуется так называемое *множественное наследование*. Указанные взаимосвязи имеют существенное значение при рассмотрении математических структур пространств с нечеткой мерой.

Теория нечеткой меры находит также применение при определении нечеткого интеграла, который успешно используется для решения задач нечеткого моделирования.

9.2. Нечеткий интеграл и примеры его вычисления

Пусть $(\boldsymbol{X}, \Im, g)$ — некоторое пространство с нечеткой мерой, для которого $\boldsymbol{X}$ — базовое множество (универсум); $\Im$ — σ-алгебра на $\boldsymbol{X}$; g — некоторая нечеткая мера, т. е. функция множества, удовлетворяющая аксиомам (9.1)—(9.3). Рассмотрим некоторую измеримую на $\Im$ функцию h: $\boldsymbol{X}\to[0, 1]$.

Нечеткий интеграл. *Нечетким интегралом от функции h на множестве* $A\in\Im$ *по нечеткой мере g* называется выражение следующего вида:

$$\int_A h(x)\} \bullet g = \sup_{\alpha\in[0,1]} \{\min\{\alpha, g(A\cap H_\alpha\}\}, \tag{9.10}$$

где $H_\alpha=\{x \mid h(x)\geq\alpha\}$. Нечеткий интеграл также называют нечетким ожиданием (FEV, *Fuzzy Expected Value*). Наибольший интерес при решении практических задач нечеткого моделирования приобретает обобщение этого понятия на случай, когда вместо множества A используется нечеткое множество $\mathcal{A}$.

Нечетким интегралом от функции h на нечетком множестве $\mathcal{A}=\{<x, \mu_{\mathcal{A}}(x)>\}$, *заданного на универсуме* $\boldsymbol{X}$, *по нечеткой мере g* называется выражение следующего вида:

$$\int_X \min\{\mu_A(x), h(x)\} \bullet g, \tag{9.11}$$

где интегрирование осуществляется по всем значениям универсума $\boldsymbol{X}$, а знак нечеткого интеграла здесь обозначен в форме обычного интеграла.

Пример 9.2. В качестве примера рассмотрим использование нечеткого интеграла для оценки модели проезжающих по автостраде автомобилей.

С этой целью определим некоторую функцию нечеткой меры $g(x)$, которая содержательно описывает распределения возможности достижения предельной скорости на автостраде некоторой моделью автомобиля, для определенности, предположим ВАЗ 2110. При этом в качестве универсума $\boldsymbol{X}$ будем рассматривать диапазон скоростей в интервале [30 км/ч, 180 км/ч]. Функция нечеткой меры $g(x)$ задается исходя из физической возможности данной модели автомобиля развить максимальную скорость движения по автостраде. График этой функции изображен на рис. 9.2, *а*.

Если в качестве автомагистрали рассматривается автобан, то каждый из автомобилей стремится двигаться по ней с высокой скоростью. Поэтому в качестве функции $h(x)$ возьмем некоторую функцию, описывающую высокую скорость движения автомобилей по автомагистрали. Эта функция может быть определена на основе анализа скоростей проезжающих автомобилей. График этой функции изображен на рис. 9.2, *б*.

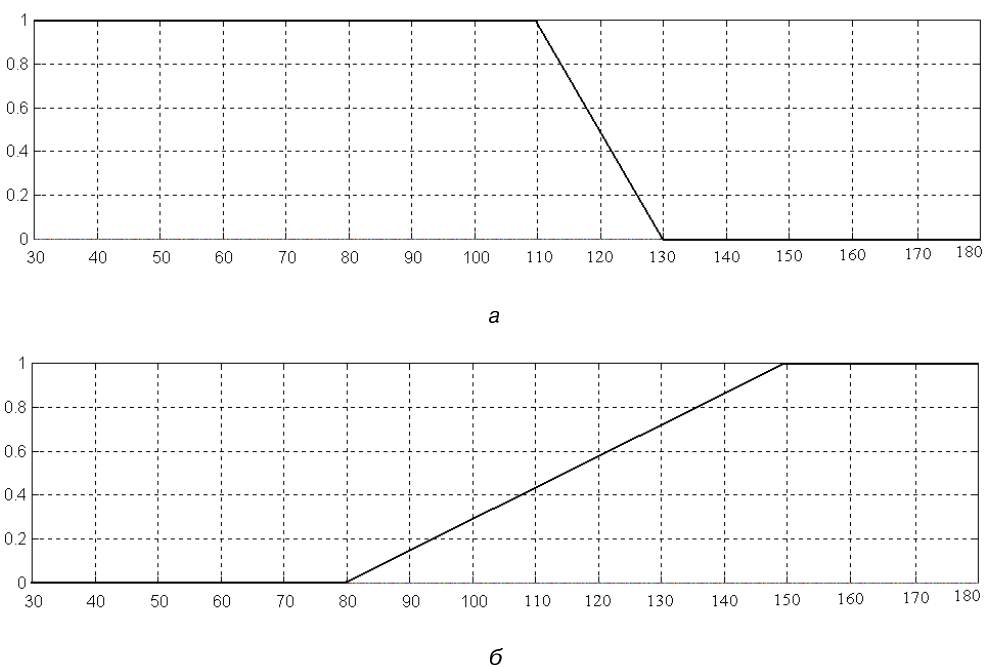


Рис. 9.2. Графики функции распределения возможностей (*а*) и функции высокой скорости движения автомобилей (*б*)

Далее предположим, что мимо пункта наблюдения проехало два автомобиля, один со скоростью около 90 км/ч, а другой — со скоростью около 120 км/ч. При этом скорость движения определяется визуально наблюдателем без технических приборов. Этим значениям скоростей соответствуют два нечетких множества $\mathcal{A}$ и $\mathcal{B}$, функции принадлежности которых изображены на рис. 9.3, *а*. Требуется выполнить оценку скорости и определить возможность проезда мимо пункта контроля автомобиля марки ВАЗ 2110.

Для решения этой задачи используем определенное ранее понятие нечеткого интеграла. Для этого необходимо найти два его значения: одно — для функции $h(x)$ на нечетком множестве $\mathcal{A}$ по нечеткой мере $g(x)$, а второе — для функции $h(x)$ на нечетком множестве $\mathcal{B}$ по нечеткой мере $g(x)$. Выполнив соответствующие расчеты, получим следующие значения: для первого автомобиля — 0.27, для второго автомобиля — 0.55 (значения приближенные). Процесс решения задачи иллюстрируется графиками на рис. 9.3, *б*.

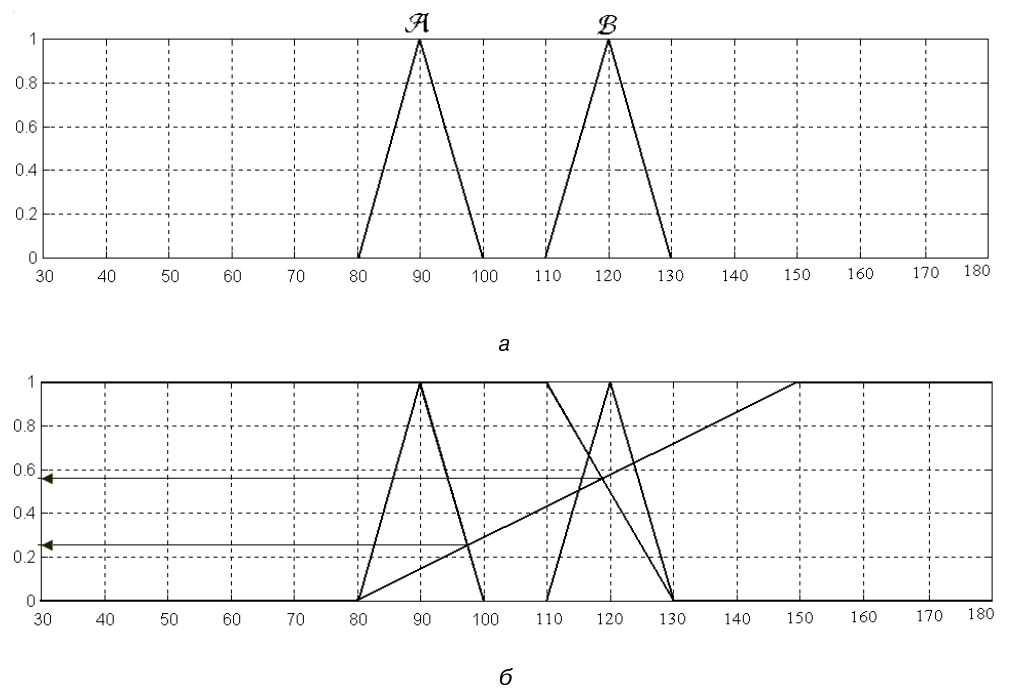


Рис. 9.3. Графики функций принадлежности нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ (*а*) и результатов вычисления нечеткого интеграла (*б*)

Таким образом, возможность того, что первым проехал мимо пункта наблюдения автомобиль модели ВАЗ 2110 равна 0.27, а возможность того, что автомобиль модели ВАЗ 2110 был вторым равна 0.55. Если же есть дополнительная ин-

формация о том, что один из двух автомобилей есть модель ВАЗ 2110, то результат решения задачи явно свидетельствует в пользу второго автомобиля.

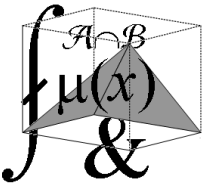
Данная нечеткая модель может быть модифицирована на основе установления более точного вида функции оценки скорости движения по автомагистрали только для моделей интересуемого нас класса. В этом случае результат решения задачи будет соответствовать более высоким значениям величин возможности при установлении модели автомобиля.

Примечание

В качестве упражнения предлагается построить аналогичную нечеткую модель для другой марки автомобиля, например, Ford Escort 4W, а результаты сравнить. Впрочем, большинство из читателей могут это сделать и без построения нечеткой модели.

В настоящее время теория нечетких мер находит применение для многокритериальной оценки альтернатив в нечетких моделях принятия решений. Хотя теоретические исследования в этой области показали высокую адекватность нечетких моделей при решении отдельных задач, реализация соответствующих алгоритмов в коммерческих программных средствах и инструментариях пока еще оставляет надежды на будущее.

Глава 10



Нечеткие сети Петри

Сети Петри (СП) и их многочисленные модификации являются одним из классов моделей, неоспоримым достоинством которых является возможность адекватного представления не только структуры сложных организационно-технологических систем и комплексов, но также и логико-временных особенностей процессов их функционирования. Сети Петри представляют собой математическую модель для представления структуры и анализа динамики функционирования систем в терминах "условие-событие". Эта модель может быть успешно использована для описания так называемых *динамических дискретных систем* различных классов, таких как: вычислительные процессы и программы, технологические процессы, информационные, экономические, биологические, социальные и технические системы.

Модели сетей Петри позволяют исследовать работоспособность моделируемых систем, оптимальность их структуры, эффективность процесса их функционирования, а также возможность достижения в процессе функционирования определенных состояний. Сети Петри и их обобщения являются удобным и мощным средством моделирования асинхронных, параллельных распределенных и недетерминированных процессов, позволяют наглядно представить динамику функционирования систем и составляющих их элементов. Свойство иерархического вложения сетей Петри позволяет рассматривать модели различной степени детализации, обеспечивая тем самым необходимую декомпозицию сложных систем и процессов.

К настоящему времени известно большое количество разновидностей и обобщений классического формализма сетей Петри, к которым, в первую очередь, следует отнести временные СП, СП с разноцветными маркерами и дугами, алгебраические СП, Е-сети. Данные классы моделей позволяют представить структуру и динамику функционирования моделируемых систем в условиях отсутствия влияния тех или иных факторов неопределенности. Указанное предположение о детерминированном характере структурных взаимосвязей и динамики функционирования СП существенно ограничивает возможности практического использования моделей данных классов и не отражает адекватным образом отдельные аспекты знаний о предметной области.

Включение описания неопределенности в различные детерминированные разновидности и обобщения СП может быть осуществлено, вообще говоря, различным образом по каждому из основных компонентов исходного формализма со-

ответствующего класса СП. Поскольку при этом можно рассматривать различные формы неопределенности (стохастическая, нечеткая, комбинированная), то, следуя по этому пути, можно получить чрезвычайно большое количество вариантов формализма соответствующих классов СП с неопределенностью. В настоящей главе рассмотрены только основные из известных подклассов нечетких СП, которые имеют непосредственное отношение к тематике нечеткого моделирования и находят наибольшее применение при решении прикладных задач.

10.1. Базовый формализм классических сетей Петри

Сети Петри получили свое название в честь немецкого математика Карла А. Петри (C. A. Petri), который в 1962 г. опубликовал свою работу с описанием нового класса сетей, использованных им в своей диссертации. В середине 1980-х гг. произошел буквально всплеск активности в области разработки новых классов сетей Петри и формального анализа их свойств, следствием чего стали ежегодные международные конференции и семинары, посвященные данной тематике. Хотя сетям Петри и их многочисленным разновидностям присущ серьезный недостаток, связанный с отсутствием эффективных алгоритмов решения так называемой проблемы достижимости, этот формализм представляет и сейчас важное направление прикладных исследований в области системного моделирования.

Формально СП представляет собой граф специального вида *(см. приложение 1)* с дополнительными правилами, которые определяют динамику процесса функционирования СП. Ниже приводится определение сетей Петри, представляющее собой базовый формализм СП, на основе которого строятся различные обобщения и расширения СП. При этом следует помнить, что тот или иной класс СП определяется не только своим формальным или графическим представлением, но также и правилами функционирования СП.

Обобщенная маркированная СП. *Обобщенная маркированная сеть Петри* (или кратко *сеть Петри*) — это пятерка $C = (P, T, I, O, \boldsymbol{m}_0)$, где:

- $P=\{p_1, p_2,..., p_n\}$ — конечное множество *позиций* СП;
- $T=\{t_1, t_2,..., t_u\}$ — конечное множество *переходов* СП;
- I — *входная функция переходов*, которая определяется как отображение I: $P\times T\rightarrow \mathbb{N}_0$;
- O — *выходная функция переходов*, которая определяется как отображение O: $T\times P\rightarrow \mathbb{N}_0$;
- $\boldsymbol{m}_0=(m_1^0, m_2^0,..., m_n^0)$ — вектор *начальной маркировки* СП, при этом $m_i^0\in\mathbb{N}_0$ $(\forall i\in\{1, 2,..., n\})$ и m_i^0 — *компонент* вектора начальной маркировки СП, соответствующий позиции $p_n\in P$.

Здесь и далее через $\mathbb{N}_0$ будет обозначаться множество натуральных чисел и ноль, т. е. $\mathbb{N}_0=\{0, 1, 2, 3,...\}$.

В определении сети Петри часто выделяют первые четыре компонента, которые задают *структуру* СП: $N=(P, T, I, O)$. В этом случае СП может быть записана в эквивалентной форме: $C = (N, \boldsymbol{m}_0)$.

Примечание

Эквивалентное определение обобщенной маркированной сети Петри $C = (P, T, I, O, \boldsymbol{m}_0)$, а, точнее, структуры СП может быть дано с использованием понятия мультимножества или комплекта, являющегося, в свою очередь, обобщением понятия множества *(см. приложение 1)*. В этом случае входная и выходная функции переходов СП определяются как I: $P\to\mathbb{B}^*(T)$ и O: $T\to\mathbb{B}^*(P)$ соответственно. Здесь через $\mathbb{B}^*(T)$ и $\mathbb{B}^*(P)$ обозначены множества всех комплектов, построенных на универсумах, в качестве которых используются множества переходов T и позиций P соответственно.

Более простым случаем обобщенной маркированной сети Петри является так называемая *ординарная маркированная сеть Петри*, которая также определяется как пятерка $(P, T, I, O, \boldsymbol{m}_0)$, где в качестве входной функции переходов I и выходной функции переходов O используются отображения I: $T\times P\to\{0, 1\}$ и O: $T\times P\to\{0, 1\}$ соответственно.

Графически ординарная маркированная сеть Петри изображается ориентированным двудольным графом специального вида, а обобщенная маркированная сеть Петри — ориентированным двудольным мультиграфом специального вида. Множество вершин графа или мультиграфа СП есть $V=P\cup T$, а множество дуг определяется входной и выходной функциями переходов. Особенностью графического представления сети Петри является изображение позиций в виде кружочков, а переходов — в виде черточек (узких прямоугольников). Дуги графа или мультиграфа соединяют переходы только с позициями, а позиции — только с переходами, поэтому граф или мультиграф является двудольным. Начальная маркировка изображается точками или натуральным числом внутри кружочков, соответствующих отдельным позициям СП. Количество таких точек, получивших название *маркеров*, в позициях СП равно значению соответствующего компонента вектора начальной маркировки.

П р и м е р 10.1. Изображенная на рис. 10.1, *а* ординарная маркированная сеть Петри имеет: множество позиций $P=\{p_1, p_2, p_3, p_4, p_5\}$, множество переходов $T=\{t_1, t_2, t_3, t_4\}$, значения входной функции переходов $I(p_1, t_1)=1$, $I(p_1, t_2)=1$, $I(p_2, t_2)=1$, $I(p_3, t_3)=1$, $I(p_4, t_3)=1$, $I(p_4, t_4)=1$, остальные $I(p_i, t_j)=0$; значения выходной функции переходов $O(t_1, p_3)=1$, $O(t_2, p_4)=1$, $O(t_3, p_3)=1$, $O(t_4, p_5)=1$, остальные значения $O(t_i, p_j)=0$; вектор начальной маркировки $\boldsymbol{m}_0=(1, 1, 0, 0, 0)$. Маркеры в позициях этой СП изображены точками, при этом отсутствие точки в позиции означает отсутствие в ней маркера.

Изображенная на рис. 10.1, *б* обобщенная маркированная сеть Петри имеет: множество позиций $P=\{p_1, p_2, p_3, p_4\}$, множество переходов $T=\{t_1, t_2, t_3, t_4\}$, значения входной функции переходов $I(p_1, t_2)=3$, $I(p_2, t_1)=1$, $I(p_3, t_3)=1$, $I(p_4, t_4)=2$, остальные $I(p_i, t_j)=0$; значения выходной функции переходов $O(t_1, p_1)=1$, $O(t_2, p_3)=1$, $O(t_2, p_4)=1$, $O(t_3, p_2)=1$, $O(t_3, p_4)=1$, $O(t_4, p_1)=1$, остальные значения $O(t_i, p_j)=0$;

вектор начальной маркировки $\boldsymbol{m}_0=(3, 0, 0, 1)$. Количество маркеров в позициях этой СП изображено цифрами.

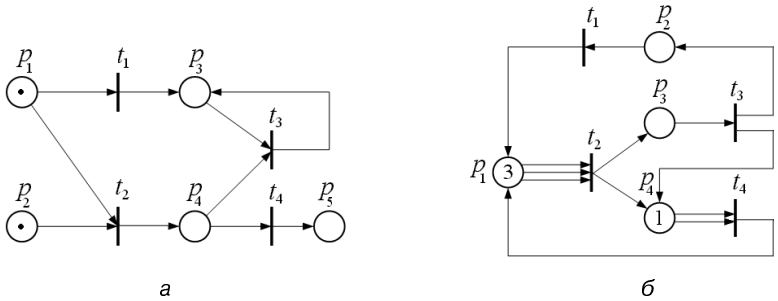


Рис. 10.1. Графическое изображение ординарной (*а*) и обобщенной (*б*) маркированных сетей Петри

Позиции, дуги из которых ведут в переход $t_j \in T$ СП, называются *входными позициями* перехода t_j; аналогично, позиции, в которые ведут дуги из перехода $t_j \in T$, называются *выходными позициями* этого перехода. Значения входной и выходной функций переходов в этом случае удобно представлять в виде соответствующих *матрицы входных позиций* **I** и *матрицы выходных позиций* **O**. Для СП, изображенной на рис. 10.1, *а*, матрицы входных и выходных позиций соответственно равны:

$$\mathbf{I} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{O} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Для СП, изображенной на рис. 10.1, *б*, матрицы входных и выходных позиций соответственно равны:

$$\mathbf{I} = \begin{bmatrix} 0 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \qquad \mathbf{O} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Примечание

Как можно заметить, структура N маркированной сети Петри (как ординарной, так и обобщенной) полностью определяется матрицами входных и выходных позиций переходов. Ниже обобщенная маркированная сеть Петри, если не возникает неоднозначности понимания, будет сокращенно именоваться сетью Петри (СП). При этом все утверждения, справедливые для обобщенных сетей Петри, естественным образом оказываются справедливыми и для ординарных СП, но не наоборот.

Наибольшее значение с прикладной точки зрения, во многом обусловившей интерес к развитию исследований в области СП, имеет динамика изменения начальной и последующих маркировок СП. При этом сам процесс изменения маркировок СП происходит в результате запуска переходов, который определяется условием активности и правилом срабатывания переходов.

Динамика изменения начальной и последующих маркировок СП после момента ее запуска подчиняется следующим *Правилам* $\mathbb{P}(C)$.

- ($\mathbb{P}_1$) ***Правило определения текущего состояния СП.*** Любое состояние СП $C = (P, T, I, O, \boldsymbol{m}_0)$ определяется некоторой маркировкой СП, которая представляет собой вектор $\boldsymbol{m}=(m_1, m_2,..., m_n)$. При этом m_i является компонентом вектора маркировки СП, соответствующим позиции $p_n \in P$, где $m_i \in \mathbb{N}_0$ $(\forall i \in \{1, 2,..., n\})$. Очевидно, начальное состояние СП, т. е. состояние сети Петри до момента ее запуска или начала процесса функционирования, определяется вектором начальной маркировки $\boldsymbol{m}_0$.

- ($\mathbb{P}_2$) ***Правило (условие) активности переходов.*** Переход $t_j \in T$ СП $C = (P, T, I, O, \boldsymbol{m}_0)$ называется *активным* (разрешенным или возбужденным) при некоторой маркировке $\boldsymbol{m}=(m_1, m_2,..., m_n)$, если выполнено следующее условие:

$$m_i \geq I(t_j, p_i) \qquad (\forall p_i \in P). \tag{10.1}$$

  Другими словами, некоторый переход СП является активным, если в каждой из его входных позиций имеется такое количество маркеров, которое больше или равно количеству дуг, соединяющих соответствующую входную позицию с данным переходом.

- ($\mathbb{P}_3$) ***Правило срабатывания перехода.*** Если некоторый переход $t_j \in T$ СП $C = (P, T, I, O, \boldsymbol{m}_0)$ активен при маркировке $\boldsymbol{m}=(m_1, m_2,..., m_n)$, т. е. для данного перехода выполнено условие (10.1), то срабатывание этого перехода, осуществляемое мгновенным образом, приводит к новой маркировке $\boldsymbol{m}_v=(m_1^v, m_2^v,..., m_n^v)$, компоненты вектора которой определяются по следующей формуле:

$$m_i^v = m_i + O(t_j, p_i) - I(p_i, t_j) \qquad (\forall p_i \in P). \tag{10.2}$$

  Другими словами, срабатывание некоторого активного перехода СП "перемещает" маркеры из входных позиций данного перехода в его выходные позиции таким образом, что во всех его входных позициях "исчезает" столько маркеров, сколько дуг соединяет эту входную позицию с данным переходом. Соответственно, во всех его выходных позициях "появляется" столько маркеров, сколько дуг соединяет данный переход с соответствующей выходной позицией. При этом срабатывание перехода считается неделимым актом, т. е. предполагается, что изъятие маркеров из всех входных позиций и их перемещение во все выходные позиции осуществляется мгновенно, с нулевой задержкой.

Проиллюстрируем применение описанных Правил $\mathbb{P}(C)$ изменения начальной маркировки $\boldsymbol{m}_0$ СП $C = (P, T, I, O, \boldsymbol{m}_0)$ для СП из примера 10.1. Так, для СП, изображенной на рис. 10.1, *а*, при данной начальной маркировке $\boldsymbol{m}_0=(1, 1, 0, 0, 0)$ согласно правилу $\mathbb{P}_2$ активными являются сразу два перехода: t_1, t_2, поскольку для каждого из них выполнено условие (10.1). Для СП, изображенной на рис. 10.1, *б*, активным при начальной маркировке $\boldsymbol{m}_0=(3, 0, 0, 1)$ является единст-

венный переход t_2, т. к. только для него выполнено условие (10.1). Для остальных переходов этих СП условие (10.1) не выполняется, следовательно, они не являются активными при соответствующих начальных маркировках.

Процесс запуска или функционирования СП начинается с определения активных переходов при начальной маркировке и их срабатывания. При этом срабатывание активного перехода t_1 для СП, изображенной на рис. 10.1, *а*, приводит к новой маркировке $\boldsymbol{m}_v=(0, 1, 1, 0, 0)$, компоненты вектора которой определяются по правилу $\mathbb{P}_3$ следующим образом: $m_1^v=1+0-1=0$, $m_2^v=1+0-0=1$, $m_3^v=0+1-0=1$, $m_4^v=0+0-0=0$, $m_5^v=0+0-0=0$. Срабатывание активного перехода t_2 этой же СП приводит к новой маркировке $\boldsymbol{m}_w=(0, 0, 0, 1, 0)$, компоненты вектора которой определяются по правилу $\mathbb{P}_3$ следующим образом: $m_1^w=1+0-1=0$, $m_2^w=1+0-1=0$, $m_3^w=0+0-0=0$, $m_4^w=0+1-0=1$, $m_5^w=0+0-0=0$.

Примечание

Следует заметить, что наличие в СП двух и более активных при некоторой маркировке переходов приводит к конфликту, суть которого заключается в том, что срабатывание одного из активных переходов может превратить другие переходы в неактивные. Следствием этого является альтернативное ветвление последовательности достижимых маркировок. Желание исключить подобные конфликты переходов приводит к необходимости дополнительного определения последовательности или задания приоритета их срабатывания. Поскольку в базовом формализме СП и Правилах $\mathbb{P}(C)$ ничего не сказано о способе разрешения этого конфликта, все активные переходы следует считать равноправными с точки зрения возможности их срабатывания.

Срабатывание единственного активного перехода t_2 для СП, изображенной на рис. 10.1, *а*, приводит к новой маркировке $\boldsymbol{m}_v=(0, 0, 1, 2)$, компоненты вектора которой определяются по правилу $\mathbb{P}_3$ следующим образом: $m_1^v=3+0-3=0$, $m_2^v=0+0-0=0$, $m_3^v=0+1-0=1$, $m_4^v=1+1-0=2$. Ординарная сеть Петри, полученная в результате срабатывания активного перехода t_1, изображена на рис. 10.2, *а*, а обобщенная сеть Петри, полученная в результате срабатывания активного перехода t_2, изображена на рис. 10.2, *б*.

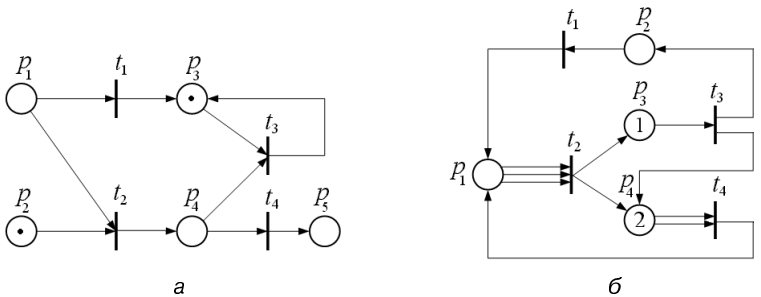


Рис. 10.2. Графическое изображение СП после срабатывания перехода t_1 для СП, изображенной на рис. 10.1, *а*, и после срабатывания перехода t_2 для СП, изображенной на рис. 10.1, *б*

Непосредственная достижимость маркировок. Маркировка $\boldsymbol{m}_v$ СП $C = (P, T, I, O, \boldsymbol{m}_0)$ *непосредственно достижима из маркировки* $\boldsymbol{m}$ (непосредственно следует за маркировкой $\boldsymbol{m}$) этой же СП, если маркировка $\boldsymbol{m}_v$ получается в результате срабатывания некоторого перехода $t_i \in T$, активного при маркировке $\boldsymbol{m}$. Тем самым определяется отношение непосредственной достижимости маркировок, которое будем обозначать: $\boldsymbol{m}\text{–}(t_i)\rightarrow\boldsymbol{m}_v$.

Очевидно, отношение непосредственной достижимости маркировок является бинарным транзитивным отношением на декартовом произведении множества всех возможных маркировок фиксированной СП. Что касается свойств рефлексивности и симметричности, то в общем случае нельзя сделать никаких утверждений относительно их наличия или отсутствия для той или иной СП.

Достижимость маркировок. Маркировка $\boldsymbol{m}_w$ СП $C = (P, T, I, O, \boldsymbol{m}_0)$ называется *достижимой из маркировки* $\boldsymbol{m}$ этой же СП, если существует конечная упорядоченная последовательность маркировок $M_k = <\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_q>$, в которой $\boldsymbol{m}_1 = \boldsymbol{m}$ и $\boldsymbol{m}_q = \boldsymbol{m}_w$, и соответствующая ей конечная упорядоченная последовательность переходов $\pi_l = <t_1, t_2, \ldots, t_{q-1}>$, такие, что любая пара соседних маркировок в M_k удовлетворяет отношению непосредственной достижимости: $\boldsymbol{m}_j\text{–}(t_j)\rightarrow\boldsymbol{m}_{j+1}$ $(\forall j \in \{1, 2, \ldots, q-1\})$. Отношение достижимости маркировок СП будем обозначать: $\boldsymbol{m}\text{–}(\pi_l)\rightarrow\boldsymbol{m}_w$.

Отношение достижимости маркировок также является бинарным транзитивным отношением на декартовом произведении множества всех возможных маркировок фиксированной СП. Более того, данное отношение является транзитивным замыканием *(см. приложение 1)* отношения непосредственной достижимости маркировок, в чем читателю предлагается убедиться самостоятельно.

Примечание

Следует заметить, что при достижимости маркировки $\boldsymbol{m}_w$ из маркировки $\boldsymbol{m}$ для заданной СП может, вообще говоря, существовать несколько последовательностей маркировок M_k и, соответственно, последовательностей переходов π_l. Более того, даже для фиксированной последовательности M_k может существовать несколько последовательностей переходов π_l, удовлетворяющих условию определения достижимости маркировок. Данное обстоятельство существенно усложняет анализ отношения достижимости маркировок в СП, который, как следует из дальнейшего изложения, является центральной задачей при исследовании характеристик конкретных СП.

Обозначим через $D(\boldsymbol{m})$ множество маркировок, достижимых из некоторой фиксированной маркировки $\boldsymbol{m}$ для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$, т. е. $D(\boldsymbol{m}) = \{\boldsymbol{m}_w \mid (\exists\, \pi_l)\colon \boldsymbol{m}\text{–}(\pi_l)\rightarrow\boldsymbol{m}_w\}$. В действительности, это множество определяется конструктивным образом, поскольку для его построения может быть использован рекурсивный алгоритм построения отношения достижимости для маркировки $\boldsymbol{m}$ и СП $C = (P, T, I, O, \boldsymbol{m}_0)$, основанный на определении отношения достижимости: $\boldsymbol{m}\text{–}(\pi_l)\rightarrow\boldsymbol{m}_w$.

Множество достижимых маркировок СП. Множество $D(\boldsymbol{m}_0)$ будем называть *множеством достижимых маркировок* для СП $C = (P, T, I, O, \boldsymbol{m}_0)$ или просто — *множеством достижимости* СП.

Поскольку в прикладных исследованиях формализм сети Петри используется в качестве модели дискретных динамических систем, то каждую маркировку СП можно рассматривать в качестве отдельного состояния такой системы (Правило $\mathbb{P}_1$). В этом случае для рассматриваемой СП особый интерес представляет множество достижимых маркировок $D(\boldsymbol{m}_0)$, которое соответствует пространству достижимых состояний некоторой дискретной динамической системы. При этом мощность множества достижимых маркировок $\mathrm{card}(D(\boldsymbol{m}_0))$ может быть конечной или счетной. В случае конечного множества $D(\boldsymbol{m}_0)$ теоретически может быть получено его конструктивное представление в форме перечисления всех последовательностей M_k и π_l, удовлетворяющих определению отношения достижимости. Если же множество $D(\boldsymbol{m}_0)$ счетное, то возникают принципиальные трудности его конструктивного определения и представления.

Для наглядного представления множества $D(\boldsymbol{m}_0)$ в случае его конечной мощности используется графическое изображение в виде специального графа в форме так называемого дерева достижимых маркировок или кратко — дерева достижимости СП. *Дерево достижимых маркировок* представляет собой ориентированный граф, вершины которого соответствуют достижимым маркировкам, а дуги — отношению непосредственного следования маркировок, причем каждая дуга помечается тем переходом, срабатывание которого привело к смене данных маркировок: $\boldsymbol{m}-(t_j)\rightarrow\boldsymbol{m}_v$. Корнем дерева является вершина, соответствующая начальной маркировке.

Примечание

При графическом представлении множества достижимости $D(\boldsymbol{m}_0)$ в дереве достижимости предполагается, что равные маркировки $\boldsymbol{m}_v$, которым соответствуют различные последовательности переходов π_l, изображаются различными вершинами на дереве. В этом случае в каждую вершину дерева будет вести единственный ориентированный путь от корневой вершины $\boldsymbol{m}_0$, которому соответствует единственная последовательность переходов π_l. Если же с целью удобства и сокращения геометрических размеров дерева достижимости равным маркировкам ставить в соответствие одну вершину, то полученный в результате граф получил название *диаграммы достижимых маркировок* СП или диаграммы состояний СП. В последнем случае от корневой вершины диаграммы $\boldsymbol{m}_0$ в каждую из ее вершин может вести несколько ориентированных путей, каждому из которых будет соответствовать своя последовательность переходов π_l. Для нахождения множества достижимости $D(\boldsymbol{m}_0)$ и его представления в форме дерева или диаграммы достижимости предложены два основных алгоритма, один из которых основан на процедуре исчерпывающего поиска в ширину, а другой — на процедуре исчерпывающего поиска в глубину. Эти алгоритмы позволяют сформировать все множество достижимых маркировок и отличаются лишь последовательностью выбора маркировок для анализа: по ширине или глубине дерева. С этими алгоритмами можно познакомиться в соответствующей литературе.

Примеры двух диаграмм достижимости, построенных для сетей Петри из примера 10.1, представлены на рис. 10.3. Данные диаграммы изображены в форме так называемых *диаграмм деятельности* в нотации унифицированного языка моде-

лирования, с основами которого можно познакомиться по одному из специальных руководств.

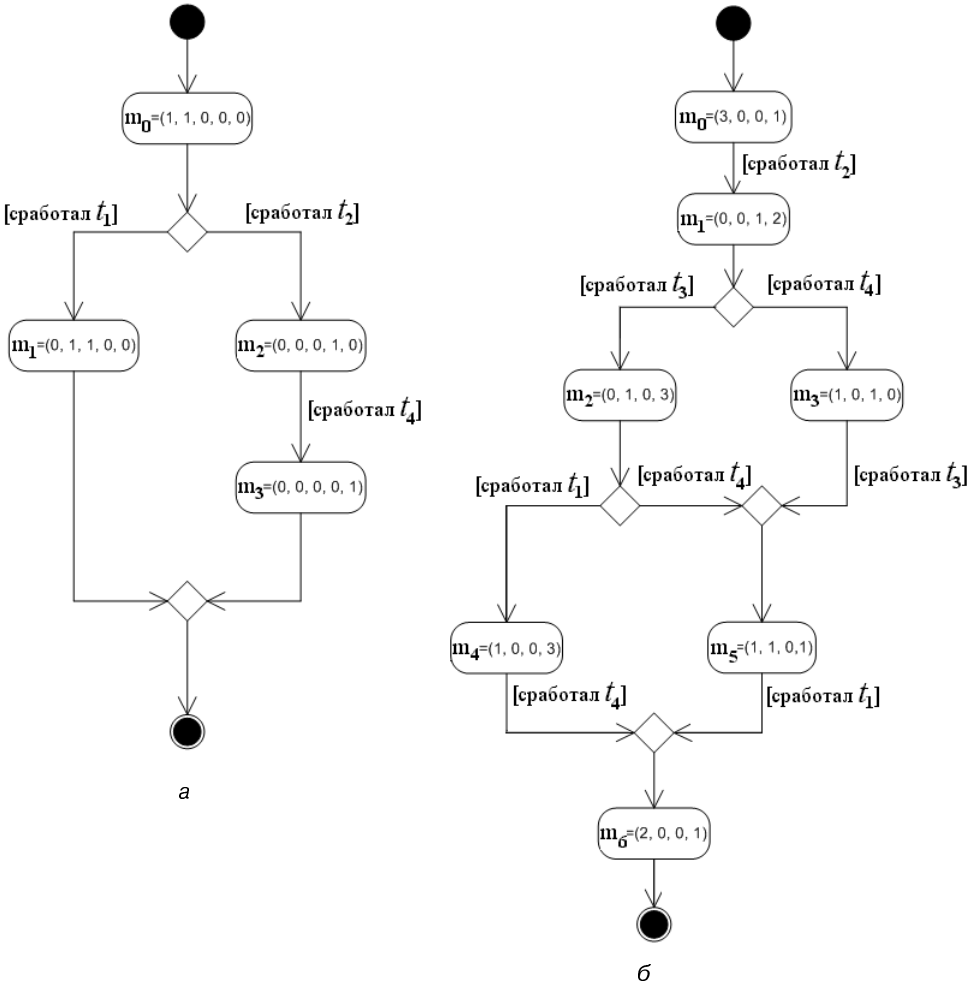


Рис. 10.3. Диаграммы достижимости для СП, изображенной на рис. 10.1, *а* (*а*), и для СП, изображенной на рис. 10.1, *б* (*б*)

Замечательным свойством СП является тот факт, что структура и начальная маркировка СП $C = (P, T, I, O, \boldsymbol{m}_0)$ вполне однозначным образом *порождают* или определяют множество достижимости $D(\boldsymbol{m}_0)$. Именно это обстоятельство, сближая некоторым образом СП с теорией конечных автоматов и теорией формальных грамматик, позволяет строить наглядные модели, адекватно интегрирующие структурные и функциональные аспекты моделируемых систем и объектов.

Свойства сетей Петри и задачи их анализа

Проведение исследований с использованием сетей Петри предполагает установление целого ряда конкретных свойств моделей с последующей интерпретацией полученных результатов применительно к решаемой задаче и проблемной области. При этом сети Петри могут различаться не только своей структурой, но даже при одинаковой структуре, но различными начальными маркировками — множеством достижимых маркировок. Поэтому исследование моделей СП предполагает анализ не только статических (структурных) характеристик, но и целого ряда динамических характеристик.

Что касается структурных или статических свойств СП, их анализ предполагает рассмотрение СП с точки зрения теории графов. Поскольку при исследовании моделей СП на первый план выдвигаются задачи анализа таких свойств сетей Петри, как безопасность, ограниченность, сохраняемость, живость и достижимость маркировок, наибольший интерес представляют функциональные или динамические характеристики СП. Именно наличие или отсутствие этих свойств у конкретной модели СП позволяет сформулировать некоторые требования по выбору рациональной структуры исследуемой системы или процесса. Ниже приводятся определения основных динамических свойств СП, имеющих непосредственное отношение к процессу функционирования или запуска СП.

Ограниченность позиций и СП. Позиция $p_i \in P$ СП $C = (P, T, I, O, \boldsymbol{m}_0)$ называется *k–ограниченной*, если для любой достижимой маркировки $\boldsymbol{m} \in D(\boldsymbol{m}_0)$ выполняется условие $m_i \le k$ для некоторого фиксированного значения $k \in \{1, 2, 3,...\}$. Или, другими словами, количество маркеров в данной позиции не превосходит натурального числа k для всех маркировок, достижимых из начальной маркировки $\boldsymbol{m}_0$. СП $C = (P, T, I, O, \boldsymbol{m}_0)$ называется *k-ограниченной*, если все позиции множества P являются k-ограниченными.

Безопасность позиций и СП. 1-ограниченная позиция СП называется *безопасной* позицией. 1-ограниченная СП называется *безопасной* СП.

Сохраняемость СП. СП $C = (P, T, I, O, \boldsymbol{m}_0)$ называется *сохраняющей*, если существует вектор $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ с неотрицательными вещественными компонентами, такой, что выполняется условие:

$$\sum_{i=1}^{n} w_i \cdot m_i = \text{const} \qquad (\forall \boldsymbol{m} \in D(\boldsymbol{m}_0)). \tag{10.3}$$

Если данное условие выполняется при $w_i = 1$ $(\forall i \in \{1, 2, \ldots, n\})$, то соответствующая СП называется *строго сохраняющей* или *консервативной*.

Устойчивость переходов и СП. Переход t_i СП $C = (P, T, I, O, \boldsymbol{m}_0)$ называется *устойчивым*, если выполнено следующее логическое условие:

$$\begin{gathered}((\boldsymbol{m} \ge (I(p_1, t_i), I(p_2, t_i), \ldots, I(p_n, t_i)) \wedge (\boldsymbol{m} \ge (I(p_1, t_r), I(p_2, t_r), \ldots, I(p_n, t_r))) \supset \\ \supset (\boldsymbol{m} \ge (I(p_1, t_i) + I(p_1, t_r), I(p_2, t_i) + I(p_2, t_r), \ldots, I(p_n, t_i) + I(p_n, t_r))) \\ (\forall t_r \in T \setminus \{t_i\}, \forall \boldsymbol{m} \in D(\boldsymbol{m}_0)\end{gathered} \tag{10.4}$$

Другими словами, если переход t_i является активным при маркировке $\boldsymbol{m} \in D(\boldsymbol{m}_0)$, то никакой другой переход t_r, тоже активный при маркировке $\boldsymbol{m}$, не может, сработав, сделать переход t_i неактивным, т. е. лишить его возможности срабатывания. СП $C = (P, T, I, O, \boldsymbol{m}_0)$ называется устойчивой, если все ее переходы являются устойчивыми.

Примечание

По содержанию логическое условие (10.4) представляет собой некоторый двухместный предикат $P(t, \boldsymbol{m})$, который записан в традиционной математической форме, т. е. с постфиксной записью кванторов общности. Так же как и в случае инфиксной записи кванторов, истинность предиката $P(t, \boldsymbol{m})$ эквивалентна истинности импликации для любых переходов t_r и достижимых маркировок $\boldsymbol{m}$, входящих в область действия квантора общности *(см. приложение 2)*.

Уровни активности переходов СП. Пусть задана СП $C = (P, T, I, O, \boldsymbol{m}_0)$.

- Переход $t_i \in T$ обладает *активностью уровня* 0 и называется *пассивным* или *мертвым*, если не существует ни одной достижимой маркировки $\boldsymbol{m} \in D(\boldsymbol{m}_0)$, при которой этот переход был бы активным.
- Переход $t_i \in T$ обладает *активностью уровня* 1 и называется *потенциально активным* или *живым*, если существует достижимая в СП маркировка $\boldsymbol{m} \in D(\boldsymbol{m}_0)$, при которой этот переход является активным.
- Переход $t_i \in T$ обладает *активностью уровня* 2, если для любого натурального $s \in \mathbb{N}$ в СП существует последовательность срабатываемых переходов π_k, в которой t_i присутствует по крайней мере s раз.
- Переход $t_i \in T$ обладает *активностью уровня* 3, если существует последовательность срабатываемых переходов π_k, в которой переход t_i присутствует неограниченное число раз.
- Переход $t_i \in T$ обладает *активностью уровня* 4 и называется *активным* или *живым*, если для любой достижимой в СП маркировки $\boldsymbol{m} \in D(\boldsymbol{m}_0)$ существует последовательность срабатываемых переходов π_k, так что t_i является активным при некоторой маркировке $\boldsymbol{m}_r \in D(\boldsymbol{m})$, получаемой в результате последовательного срабатывания переходов из π_k.
- Переход $t_i \in T$ называется *потенциально мертвым*, если существует некоторая достижимая маркировка $\boldsymbol{m} \in D(\boldsymbol{m}_0)$, такая, что при любой достижимой из $\boldsymbol{m}$ маркировке $\boldsymbol{m}_w \in D(\boldsymbol{m})$ переход t_i не является активным.
- СП $C = (P, T, I, O, \boldsymbol{m}_0)$ обладает активностью уровня k, где $k \in \{0,1,2,3,4\}$, если все ее переходы обладают активностью уровня k.

Тупиковая маркировка. Достижимая в СП маркировка $\boldsymbol{m} \in D(\boldsymbol{m}_0)$ называется t_i-*тупиковой* ($t_i \in T$), если переход t_i потенциально мертвый для маркировки $\boldsymbol{m}$. Если достижимая маркировка $\boldsymbol{m}$ t_i-тупиковая для всех переходов $t_i \in T$, то она называется *тупиковой*.

Рассмотрим использование введенных определений для СП из примера 10.1. Так, для СП, изображенной на рис. 10.1, *а*, все позиции являются 1-ограниченными, а значит и вся сеть Петри является 1-ограниченной или безопасной. Переход t_4 является устойчивым, а переходы t_1 и t_2 устойчивыми не являются, следовательно, данная СП не является устойчивой. Переходы t_1, t_2 и t_4 обладают активностью уровня 1, а переход t_3 является мертвым. Следовательно, данная СП обладает активностью уровня 0. Достижимые маркировки $\boldsymbol{m}_1=(0, 1, 1, 0, 0)$ и $\boldsymbol{m}_3=(0, 0, 0, 0, 1)$ являются тупиковыми, поскольку ни один из переходов СП не является активным при данных маркировках.

Так, для СП, изображенной на рис. 10.1, *б*, все позиция p_1 и p_4 являются 3-огрниченными, позиции p_2 и p_3 являются 1-ограниченными, а значит и вся сеть Петри является 1-ограниченной или безопасной. Все переходы этой СП являются устойчивыми, следовательно, данная СП является устойчивой. Все переходы СП обладают активностью уровня 1. Следовательно, данная СП обладает активностью уровня 1. Достижимая маркировка $\boldsymbol{m}_6=(2, 0, 0, 1)$ является тупиковой, поскольку при данной маркировке ни один из переходов СП не является активным.

При моделировании процессов функционирования дискретных динамических систем одно из центральных мест занимает установление рассмотренных выше свойств СП. В общем случае исследование характеристик СП предполагает решение следующих задач:

1. *Задача достижимости маркировки.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и заданной некоторой $\boldsymbol{m}_r$ установить выполнение условия: $\boldsymbol{m}_r \in D(\boldsymbol{m}_0)$. Или, другими словами, достижима ли маркировка $\boldsymbol{m}_r$ из начальной маркировки $\boldsymbol{m}_0$ для заданной СП?
2. *Задача достижимости подмаркировки.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и заданной маркировки $\boldsymbol{m}_r$ и подмножества позиций $P' \subset P$ определить: существует ли достижимая маркировка $\boldsymbol{m}_w \in D(\boldsymbol{m}_0)$, такая что $\boldsymbol{m}_i^r = \boldsymbol{m}_i^w$ $(\forall p_i \in P')$? Другими словами, существует ли достижимая маркировка $\boldsymbol{m}_w$, компоненты вектора которой с номерами позиций из подмножества P' равны соответствующим компонентам вектора маркировки $\boldsymbol{m}_r$? Остальные компоненты вектора $\boldsymbol{m}_r$ могут принимать произвольные значения.
3. *Задача покрываемости маркировки.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и заданной маркировки $\boldsymbol{m}_r$ определить, существует ли достижимая маркировка $\boldsymbol{m}_w \in D(\boldsymbol{m}_0)$, такая что $\boldsymbol{m}_w \geq \boldsymbol{m}_r$?
4. *Задача достижимости нуля.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ определить, достижима ли нулевая маркировка $\boldsymbol{m} = (0, 0,..., 0)$ из начальной маркировки $\boldsymbol{m}_0$, т. е. выполняется ли условие: $\boldsymbol{m} \in D(\boldsymbol{m}_0)$?
5. *Задача достижимости нуля в одной позиции.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и некоторой фиксированной позиции $p_i \in P$ определить, существует ли достижимая маркировка $\boldsymbol{m} \in D(\boldsymbol{m}_0)$, в которой $\boldsymbol{m}_i = 0$ (i-ый компонент вектора маркировки равен нулю)?

6. *Задача равенства.* Для двух СП $C' = (P', T', I', O', \boldsymbol{m}_0')$ и $C'' = (P'', T'', I'', O'', \boldsymbol{m}_0'')$, таких что $P' = P''$, определить, равны ли соответствующие им множества достижимых маркировок $D(\boldsymbol{m}_0') = D(\boldsymbol{m}_0'')$?
7. *Задача подмножества.* Для двух СП $C' = (P', T', I', O', \boldsymbol{m}_0')$ и $C'' = (P'', T'', I'', O'', \boldsymbol{m}_0'')$ определить, является ли одно из множеств достижимых маркировок подмножеством другого, т. е. выполняется ли отношение $D(\boldsymbol{m}_0') \subseteq D(\boldsymbol{m}_0'')$ или отношение $D(\boldsymbol{m}_0'') \subseteq D(\boldsymbol{m}_0')$?
8. *Задача k-ограниченности позиций и СП в целом.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и фиксированной позиции $p_i \in P$ определить, существует ли натуральное число k, для которого позиция p_i является k-ограниченной (является ли СП в целом k-ограниченной)?
9. *Задача безопасности позиций и СП в целом.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и фиксированной позиции $p_i \in P$ определить, является ли позиция p_i безопасной (является ли безопасной СП в целом)?
10. *Задача устойчивости переходов.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и заданного подмножества переходов $T' \subseteq T$ определить, являются ли переходы из подмножества T' устойчивыми?
11. *Задача активности переходов.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ и заданного подмножества переходов $T' \subseteq T$ определить уровни активности переходов подмножества T'.
12. *Задача сохраняемости СП.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ определить, является ли она сохраняющей?
13. *Задача строгой сохраняемости (консервативности) СП.* Для заданной СП $C = (P, T, I, O, \boldsymbol{m}_0)$ определить, является ли она строго сохраняющей (консервативной)?

При изучении данных задач оказалось, что они определенным образом взаимосвязаны между собой. Так, например, решение задачи достижимости (1) предопределяет решение задачи достижимости нулевой маркировки (4), а решение задачи достижимости подмаркировки (2) предопределяет решение задачи достижимости нуля в одной позиции (5). Решение задачи k-ограниченности позиции (8) предопределяет решение задачи безопасности позиций (9), а решение задачи сохраняемости (12) предопределяет решение задачи консервативности (13).

Таким образом, исчерпывающая информация о решении указанных задач содержится в дереве или диаграмме достижимых маркировок СП. Поэтому одной из основных проблем в рамках классического формализма СП является построение диаграммы достижимых маркировок и ее анализ для конкретных СП. Решение этой проблемы позволяет дать ответ на решение всех перечисленных выше задач (1—13) и установить все основные свойства СП, содержательная интерпретация которых зависит от проблемной области и рассматриваемых моделей прикладных систем.

10.2. Основные подклассы нечетких сетей Петри

Нечеткие сети Петри (НСП), являясь разновидностью СП с неопределенностью, позволяют конструктивно решать задачи нечеткого моделирования и нечеткого управления, в которых неопределенность имеет нестохастический или субъективный характер. В данной главе внимание сосредоточено на рассмотрении только трех подклассов НСП, являющихся естественной конкретизацией базовой математической структуры СП с неопределенностью, которая может быть построена на основе использования общей концепции нечеткой меры *(см. главу 9)*.

Нечеткие сети Петри типа V_f

Первый из рассматриваемых ниже подклассов НСП, называемый в дальнейшем НСП типа V_f или кратко — НСП V_f, получается в результате введения описания неопределенности нечеткого характера в начальную маркировку и правила изменения маркировок базового формализма классических СП.

Нечеткая сеть Петри типа V_f. *Нечеткая сеть Петри типа* V_f (НСП V_f) определяется как $V_f = (N, \mathbf{M}_0)$, где:

- $N = (P, T, I, O)$ — *структура* НСП V_f, в которой: $P = \{p_1, p_2,..., p_n\}$ — конечное множество позиций; $T = \{t_1, t_2,..., t_u\}$ — конечное множество переходов; I: $P \times T \to \mathbb{N}_0$ — входная функция переходов; O: $T \times P \to \mathbb{N}_0$ — выходная функция переходов;
- $\mathbf{M}_0$ — *матрица начальной маркировки*, размерность которой равна $(n \times (d+1))$. Каждый элемент m_{ij}^0 этой матрицы равен значению функции принадлежности наличия $j-1$ числа маркеров в позиции p_i НСП на момент начала ее запуска. По определению функции принадлежности элементы матрицы начальной маркировки должны удовлетворять следующему условию:

$$m_{ij}^0 \in [0, 1] \qquad (\forall i \in \{1, 2,..., n\}, \forall j \in J). \tag{10.5}$$

Здесь, как и ранее, $\mathbb{N}_0 = \{0, 1, 2, 3,...\}$ обозначает множество натуральных чисел и ноль. Множество J определяется как: $J = \{1,..., d, d+1\} \subset \mathbb{N}_0$, т. е. как некоторое конечное подмножество $\mathbb{N}_0$, состоящее из $d+1$ первых натуральных чисел. При этом общее количество столбцов матрицы начальной маркировки определяется максимальным количеством вводимых в рассмотрение маркеров в позициях НСП V_f, которое в общем случае принимается равным d.

Примечание

В определении НСП V_f каждая строка матрицы $\mathbf{M}_0$ может рассматриваться как значения функции принадлежности нечеткого множества маркеров для соответствующей позиции НСП V_f. Характерным свойством указанного нечеткого множества является его дискретный характер, что позволяет задать его конечным множеством положительных значений функции принадлежности для носи-

теля данного множества. В этом случае базовый формализм классических СП может быть получен в качестве частного случая НСП V_f, если в определении НСП V_f условие (10.5) заменить на обычное условие для допустимых значений компонентов начальной маркировки СП: $m_{ij}^0 \in \{0, 1\}$ $(\forall i \in \{1, 2,..., n\}, \forall j \in J)$ и $\sum_{j=1}^{d+1} m_{ij}^0 \le 1$ $(\forall i \in \{1, 2,..., n\})$. С другой стороны, вводя дополнительно к (10.5) условие: $\sum_{j=1}^{d+1} m_{ij}^0 = 1$ $(\forall i \in \{1, 2,..., n\})$, получим определение стохастических СП типа V_s. Таким образом, НСП V_f являются обобщением не только обычных СП, но и обобщением стохастических СП V_s.

Из определения введенного в рассмотрение подкласса НСП V_f можно видеть, что структура N НСП V_f полностью идентична структуре базового формализма классических СП. Поэтому НСП V_f графически также изображаются ориентированным двудольным мультиграфом аналогично обычным СП.

П р и м е р 10.2. Структура изображенной на рис. 10.4 НСП типа V_f, имеет множество позиций $P=\{p_1, p_2, p_3, p_4\}$ и множество переходов $T=\{t_1, t_2, t_3, t_4\}$. Начальная маркировка для этой НСП V_f может быть задана, например, следующей матрицей:

$$\mathbf{M_0} = \begin{bmatrix} 0.1 & 0.2 & 0.6 & 0.8 \\ 0.8 & 0 & 0 & 0 \\ 0.7 & 0 & 0 & 0 \\ 0.6 & 0.5 & 0 & 0 \end{bmatrix}.$$

Строки матрицы начальной маркировки образованы из векторов $\boldsymbol{m}_i^0 = (m_{i1}^0, m_{i2}^0, m_{i3}^0, m_{i4}^0)$, $(\forall i \in \{1, 2,..., n\})$, компоненты которых определяют степени принадлежности $m_{ij}^0 = \mu_i(j\text{-}1)$ нечеткого наличия $j-1$ количества маркеров в позиции p_i данной НСП. При этом общее количество вводимых в рассмотрение маркеров равно $d=3$. Так, для позиции p_1 значение $m_{11}^0 = 0.1$ означает, что степень принадлежности нечеткого отсутствия маркеров в данной позиции для начальной маркировки равна 0.1. Значение $m_{12}^0 = 0.2$ означает, что степень принадлежности нечеткого наличия одного маркера в позиции p_1 для начальной маркировки равно 0.2. Значение $m_{13}^0 = 0.6$ означает степень принадлежности нечеткого наличия двух, а значение $m_{14}^0 = 0{,}8$ — степень принадлежности нечеткого наличия трех маркеров в позиции p_1 данной НСП для начальной маркировки. Значения $m_{15}^0 = m_{16}^0 = ... = 0$ в матрице $\mathbf{M_0}$ не приводятся.

Другими словами, в матрице начальной маркировки $\mathbf{M_0}$ указываются лишь значения степеней принадлежности для носителя нечеткого множества количества маркеров в позициях НСП V_f.

Динамика изменения начальной и последующих маркировок НСП V_f после момента ее запуска подчиняется следующим *Правилам* $\mathbb{P}(V_f)$:

- ($\mathbb{P}_1$) ***Правило определения текущей маркировки***. Любое текущее состояние НСП V_f определяется некоторой матрицей $\mathbf{M}$ размерности $(n \times (d+1))$, элемен-

ты которой удовлетворяют условию (10.5) и интерпретируются как значения степеней принадлежности нечеткого наличия $\{0, 1, 2,..., d\}$ маркеров в соответствующих позициях $p_i \in P$ НСП V_f. Начальное состояние НСП определяется матрицей начальной маркировки $\mathbf{M}_0$.

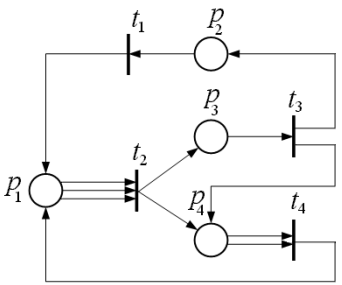


Рис. 10.4. Графическое изображение структуры НСП типа V_f

- ($\mathbb{P}_2$) ***Правило (условие) активности перехода***. Переход $t_k \in T$ НСП V_f называется *активным (разрешенным, возбужденным)* при некоторой текущей маркировке $\mathbf{M}$, если выполнено следующее условие:

$$\boldsymbol{\sigma} \geq (I(p_1, t_k),\ I(p_2, t_k),..., I(p_n, t_k)), \tag{10.6}$$

где компоненты вектора $\boldsymbol{\sigma} = (\sigma_1, \sigma_2,..., \sigma_n)$ определяются по формуле:

$$\sigma_i = \max_{(j \in J) \wedge (m_{ij} > 0)} \{j\} - 1 \qquad (\forall p_i \in P), \tag{10.7}$$

т. е. значение σ_i равно максимальному индексу отличной от нуля степени принадлежности нечеткого наличия маркеров в позициях НСП V_f. Другими словами, некоторый переход $t_k \in T$ НСП V_f является активным или разрешенным, если для текущей маркировки $\mathbf{M}$ во всех его входных позициях имеется нечеткое количество маркеров, которое больше или равно количеству дуг, соединяющих соответствующие входные позиции с рассматриваемым переходом.

- ($\mathbb{P}_3$) ***Правило срабатывания перехода***. Если переход $t_k \in T$ НСП V_f является активным при некоторой текущей маркировке $\mathbf{M}$ (т. е. выполнено условие (10.6)), то срабатывание данного перехода, осуществляемое мгновенным образом, приводит к новой маркировке $\mathbf{M}'$, элементы которой определяются следующим образом:

  - для каждой из входных позиций $p_i \in P$, для которых $I(p_i, t_k) \neq 0$, по формулам:

$$m_{i1}' = \max_{(j \in \{1, 2,..., I(p_i, t_k)+1\})} \{m_{ij}\}; \tag{10.8}$$

$$m_{ij}' = m_{i, j+I(p_i, t_k)} \qquad (\forall j \in J \setminus \{1\}). \tag{10.9}$$

- для каждой из выходных позиций $p_i \in P$, такой что: $O(t_k, p_i) \neq 0$ по формулам:

$$m_{ij}^{'} = \min\{m_{ij}, 1 - q_k\} \qquad (\forall j \in \{1, 2, \ldots, O(t_k, p_i)\}); \tag{10.10}$$

$$m_{ij}^{'} = \max\{\min\{m_{ij}, 1 - q_k\}, \min\{m_{i, j-O(t_k, p_i)}, q_k\}\}, \tag{10.11}$$
$$(\forall j \in J) \wedge (j > O(t_k, p_i))$$

где q_k — степень принадлежности или мера возможности *нечеткого срабатывания* (запуска) перехода $t_k \in T$, рассчитываемая по формуле:

$$q_k = \min_{(i \in \{1, 2, \ldots, n\})} \{ \max_{(j \in J) \wedge (j > I(p_i, t_k))} \{m_{ij}\}\} \qquad (\forall t_k \in T). \tag{10.12}$$

Если некоторые из позиций $p_i \in P$ являются одновременно входными и выходными для разрешенного перехода $t_k \in T$, то для них элементы матрицы новой маркировки $\mathbf{M}'$ рассчитываются последовательно, вначале по формулам (10.8)-(10.9), а затем — по формулам (10.10)—(10.11).

Проиллюстрируем применение описанных Правил $\mathbb{P}(V_f)$ на примере изменения начальной маркировки $\mathbf{M_0}$ НСП V_f, изображенной на рис. 10.4. Согласно правилу $\mathbb{P}_2$, при данной маркировке $\mathbf{M_0}$ разрешенным является единственный переход t_2, поскольку только для него выполнено условие (10.6): $(3, 0, 0, 1) \geq (3, 0, 0, 0)$. Срабатывание данного перехода приводит к новой маркировке $\mathbf{M_1}$, элементы матрицы которой определяются по правилу $\mathbb{P}_3$ следующим образом:

- для позиции p_1, являющейся входной для перехода t_2 со значением $I(p_1, t_2) = 3$, с использованием соотношений (10.8)—(10.9): $m_{11}^{1} = \max\{0.1, 0.2, 0.6, 0.8\} = 0.8$, $m_{12}^{1} = m_{13}^{1} = m_{14}^{1} = 0$;
- для позиций p_3 и p_4, являющимися выходными для перехода t_2 со значениями $O(t_2, p_3) = 1$ и $O(t_2, p_4) = 1$, с использованием соотношений (10.10)—(10.11), при этом степень принадлежности нечеткого срабатывания перехода t_2 равна $q_2 = 0.8$: $m_{31}^{1} = \min\{m_{11}^{0}, 1-0.8\} = \min\{0.7, 0.2\} = 0.2$; $m_{32}^{1} = \max\{\min\{m_{32}^{0}, 1-0.8\}, \min\{m_{31}^{0}, 0.8\}\} = \max\{min\{0, 0.2\}, \min\{0.7, 0.8\}\} = 0.7$; $m_{33}^{1} = m_{34}^{1} = 0$, $m_{41}^{1} = \min\{m_{41}^{0}, 1-0.8\} = \min\{0.6, 0.2\}\} = 0.2$, $m_{42}^{1} = \max\{\min\{m_{42}^{0}, 1-0.8\}, \min\{m_{41}^{0}, 0.8\}\} = \max\{\min\{0.5, 0.2\}, \min\{0.6, 0.8\}\} = 0.6$, $m_{43}^{1} = \max\{\min\{m_{43}^{0}, 1-0.8\}, \min\{m_{42}^{0}, 0.8\}\} = \max\{\min\{0, 0.2\}, \min\{0.5, 0.8\}\} = 0.5$, $m_{44}^{1} = 0$;
- для позиции p_2, не являющейся для перехода t_2 ни входной, ни выходной, компоненты новой маркировки $\mathbf{M_1}$ будут равны компонентам исходной маркировки $\mathbf{M_0}$, т. е. $\boldsymbol{m_2} = (0.8, 0, 0, 0)$.

Таким образом, после нечеткого срабатывания перехода t_2 начальная маркировка $\mathbf{M_0}$ НСП V_f изменится на маркировку $\mathbf{M_1}$, матрица которой равна:

$$\mathbf{M_1} = \begin{bmatrix} 0.8 & 0 & 0 & 0 \\ 0.8 & 0 & 0 & 0 \\ 0.2 & 0.7 & 0 & 0 \\ 0.2 & 0.6 & 0.5 & 0 \end{bmatrix}.$$ Полученная маркировка $\mathbf{M_1}$ разрешает переходы t_3 и t_4, в

чем можно убедиться, используя правило $\mathbb{P}_2$. Поэтому процесс изменения марки-

ровок для данной НСП V_f может быть продолжен, что предлагается выполнить самостоятельно в качестве упражнения.

Поскольку процесс изменения маркировок НСП V_f во многом аналогичен соответствующему процессу для базового формализма классических СП с такой же структурой, хотелось бы представить динамику изменения маркировок НСП V_f также в виде дерева или диаграммы достижимых маркировок. Строгое определение диаграммы достижимых маркировок НСП V_f базируется на отношениях непосредственного следования и достижимости маркировок, определения которых приводятся ниже.

Непосредственная достижимость маркировок. Маркировка $\mathbf{M}_v$ НСП $V_f = (N, \mathbf{M}_0)$ *непосредственно достижима из маркировки* $\mathbf{M}$ (непосредственно следует за маркировкой $\mathbf{M}$) этой же НСП, если маркировка $\mathbf{M}_v$ получается в результате срабатывания некоторого разрешенного при $\mathbf{M}$ перехода $t_k \in T$. Отношение непосредственного следования маркировок НСП V_f будем обозначать через: $\mathbf{M}\text{–}(t_k)\rightarrow\mathbf{M}_v$, где $\mathbf{M}$ и $\mathbf{M}_v$ — матрицы соответствующих маркировок.

Очевидно, отношение непосредственной достижимости маркировок НСП V_f является бинарным транзитивным отношением на декартовом произведении множества всех возможных маркировок фиксированной НСП V_f. Что касается свойств рефлексивности и симметричности, то в общем случае нельзя сделать никаких утверждений относительно их наличия или отсутствия для той или иной НСП V_f.

Достижимость маркировок. Маркировка $\mathbf{M}_w$ НСП $V_f = (N, \mathbf{M}_0)$ называется *достижимой из маркировки* $\mathbf{M}$ этой же НСП, если существует конечная упорядоченная последовательность маркировок $M_k = <\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_q>$, где $\mathbf{M}_1=\mathbf{M}$, $\mathbf{M}_q=\mathbf{M}_w$, и соответствующая ей конечная упорядоченная последовательность переходов $\pi_l = <t_1, t_2, \ldots, t_{q-1}>$ такие, что любая пара соседних маркировок в M_k удовлетворяет отношению непосредственного следования маркировок: $\mathbf{M}_j\text{–}(t_j)\rightarrow\mathbf{M}_{j+1}$ $(\forall j \in \{1, 2, \ldots, q-1\})$. Отношение достижимости для НСП V_f будем обозначать через $\mathbf{M}\text{–}(\pi_l)\rightarrow\mathbf{M}_w$.

Отношение достижимости маркировок НСП V_f также является бинарным транзитивным отношением на декартовом произведении множества всех возможных маркировок фиксированной НСП. Более того, данное отношение является транзитивным замыканием *(см. приложение 1)* отношения непосредственной достижимости маркировок, в чем читателю предлагается убедиться самостоятельно.

Обозначим через $D(\mathbf{M})$ множество маркировок, достижимых из некоторой фиксированной маркировки $\mathbf{M}$ для заданной НСП $V_f = (N, \mathbf{M}_0)$, т. е. $D(\mathbf{M}) = \{\mathbf{M}_w \mid (\exists\, \pi_l)\ \mathbf{M}\text{–}(\pi_l)\rightarrow\mathbf{M}_w\}$.

Множество достижимых маркировок НСП V_f. Множество $D(\mathbf{M}_0)$ будем называть *множеством достижимых маркировок* для НСП $V_f = (N, \mathbf{M}_0)$ или просто — *множеством достижимости* НСП V_f.

Диаграммой достижимых маркировок (графом достижимости) НСП V_f будем называть ориентированный мультиграф $D(V, E, \gamma)$, где $V = \{v_1, v_2, \ldots, v_q\}$ — множест-

во вершин мультиграфа, которое соответствует множеству достижимых маркировок $D(\mathbf{M}_0)$; $E=\{e_{ij}^{k}\}$ $(\forall v_i, v_j \in V)\}$ — в общем случае мультимножество (комплект) дуг, которое соответствует отношению непосредственной достижимости маркировок НСП V_f; $\gamma: E \to T$ — отображение, помечающее каждую дугу мультиграфа переходом, срабатывание которого обеспечивает выполнение соответствующего отношения непосредственной достижимости для рассматриваемой пары маркировок.

Диаграмма достижимых маркировок, построенная для НСП из примера 10.2 в форме диаграммы деятельности нотации UML, представлена на рис. 10.5.

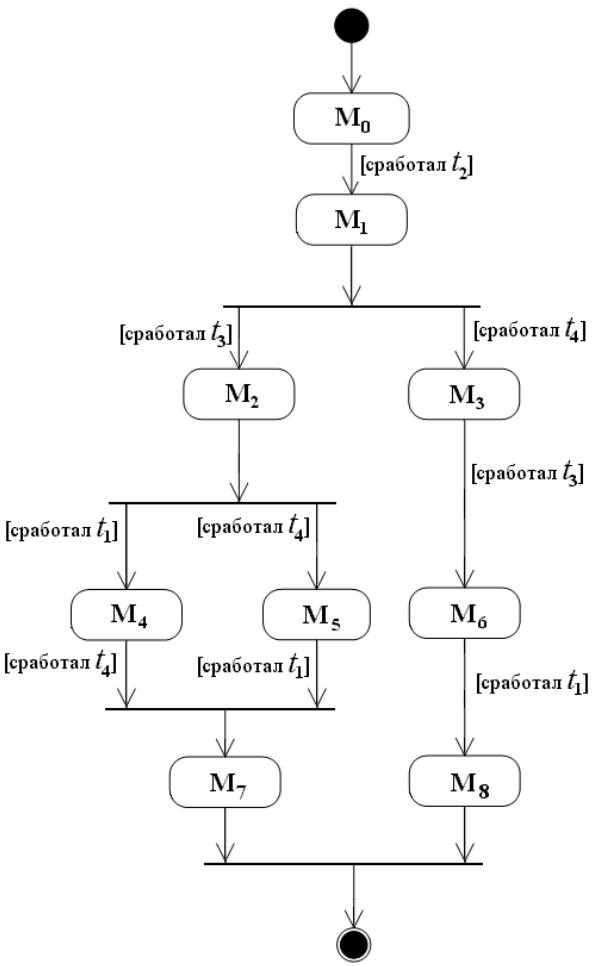


Рис. 10.5. Диаграмма достижимых маркировок для НСП V_f из примера 10.2

Для наглядности значения матриц всех достижимых маркировок записаны отдельно от диаграммы:

$$\mathbf{M_1}=\begin{bmatrix}0.8 & 0 & 0 & 0\\ 0.8 & 0 & 0 & 0\\ 0.2 & 0.7 & 0 & 0\\ 0.2 & 0.6 & 0.5 & 0\end{bmatrix}\quad \mathbf{M_2}=\begin{bmatrix}0.8 & 0 & 0 & 0\\ 0.3 & 0.7 & 0 & 0\\ 0.7 & 0 & 0 & 0\\ 0.2 & 0.3 & 0.6 & 0\end{bmatrix}\quad \mathbf{M_3}=\begin{bmatrix}0.5 & 0.5 & 0 & 0\\ 0.8 & 0 & 0 & 0\\ 0.2 & 0.7 & 0 & 0\\ 0.6 & 0 & 0 & 0\end{bmatrix}$$

$$\mathbf{M_4}=\begin{bmatrix}0.3 & 0.7 & 0 & 0\\ 0.7 & 0 & 0 & 0\\ 0.7 & 0 & 0 & 0\\ 0.2 & 0.3 & 0.6 & 0.5\end{bmatrix}\quad \mathbf{M_5}=\begin{bmatrix}0.4 & 0.6 & 0 & 0\\ 0.3 & 0.7 & 0 & 0\\ 0.7 & 0 & 0 & 0\\ 0.6 & 0.5 & 0 & 0\end{bmatrix}\quad \mathbf{M_6}=\begin{bmatrix}0.5 & 0.5 & 0 & 0\\ 0.3 & 0.7 & 0 & 0\\ 0.7 & 0 & 0 & 0\\ 0.3 & 0.6 & 0 & 0\end{bmatrix}$$

$$\mathbf{M_7}=\begin{bmatrix}0.3 & 0.4 & 0.6 & 0\\ 0.7 & 0 & 0 & 0\\ 0.7 & 0 & 0 & 0\\ 0.6 & 0.5 & 0 & 0\end{bmatrix}\quad \mathbf{M_8}=\begin{bmatrix}0.3 & 0.5 & 0.5 & 0\\ 0.7 & 0 & 0 & 0\\ 0.7 & 0 & 0 & 0\\ 0.3 & 0.6 & 0 & 0\end{bmatrix}.$$

Как показывает анализ данной диаграммы достижимости (см. рис. 10.5), для рассматриваемой НСП V_f справедливы следующие отношения: $\mathbf{M_0}-(t_2)\rightarrow\mathbf{M_1}$, $\mathbf{M_1}-(t_3)\rightarrow\mathbf{M_2}$, $\mathbf{M_1}-(t_4)\rightarrow\mathbf{M_3}$ и $\mathbf{M_0}-(<t_2, t_3>)\rightarrow\mathbf{M_2}$.

Для НСП V_f могут быть введены в рассмотрение свойства, соответствующие приведенным выше свойствам классических СП. В этом случае при решении задач системного моделирования с использованием рассматриваемых моделей НСП V_f основной интерес также представляют множества $D(\mathbf{M_0})$ и соответствующие им диаграммы достижимости, которые содержат исчерпывающую информацию об соответствующих свойствах НСП V_f.

Нечеткие сети Петри типа C_f

Второй из рассматриваемых подклассов НСП, называемый в дальнейшем НСП типа C_f или кратко — НСП C_f, получается в результате введения нечеткости в начальную маркировку и правила срабатывания переходов формализма ординарных СП.

Нечеткая сеть Петри типа C_f. *Нечеткая сеть Петри типа* C_f (НСП C_f) определяется как $C_f = (N, \boldsymbol{f}, \lambda, \boldsymbol{m_0})$, где:

- $N = (P, T, I, O)$ — структура НСП C_f, которая аналогична структуре ординарных СП и для которой I: $P\times T\rightarrow\{0, 1\}$ и O: $T\times P\rightarrow\{0, 1\}$ — входная и выходная функции переходов соответственно;
- $\boldsymbol{f} = (f_1, f_2,..., f_u)$ — вектор значений *функции принадлежности нечеткого срабатывания переходов*, при этом $f_j\in[0, 1]$ $(\forall j\in\{1, 2,..., u\})$;

- $\lambda = (\lambda_1, \lambda_2, ..., \lambda_u)$ — вектор значений порога срабатывания переходов, при этом $\lambda_j \in [0, 1]$ $(\forall j \in \{1, 2, ..., u\})$;
- $\boldsymbol{m}_0 = (m_1^0, m_2^0, ..., m_n^0)$ — вектор *начальной маркировки*, каждая компонента которого определяется значением функции принадлежности нечеткого наличия одного маркера в соответствующей позиции данной НСП C_f, при этом $m_i^0 \in [0, 1]$ $(\forall i \in \{1, 2, ..., n\})$.

Примечание

Определение для НСП C_f несколько отличается от известных в литературе определений НСП, т. к. последние дополнительно содержат в своем определении некоторое множество высказываний $D = \{d_1, d_2, ..., d_n\}$ и биективное отображение α: $D \to P$. Поскольку дополнительные компоненты (D, α) подобных определений отражают специфику моделируемой предметной области и не оказывают принципиального влияния на математические аспекты анализа свойств НСП C_f, в нашем определении они не используются, что не влечет потери в строгости всех последующих рассуждений. Из методических соображений следует отметить, что ординарные СП можно считать конкретизацией НСП C_f при фиксации следующих ограничений на компоненты в определении НСП C_f: $f_j = 1$, $\lambda_j = 0$ $(\forall j \in \{1, 2, ..., u\})$ и $m_i^0 \in \{0, 1\}$ $(\forall i \in \{1, 2, ..., n\})$. Данное обстоятельство делает данное определение НСП C_f корректным в плане нечеткого обобщения классических СП.

Структура N введенного в рассмотрение подкласса НСП C_f также имеет обычный (не нечеткий) вид, определяемый матрицами входных **I** и выходных **O** позиций. Поэтому графически НСП C_f изображаются ориентированным двудольным графом аналогично ординарным СП.

П р и м е р 10.3. Изображенная на рис. 10.6 НСП C_f имеет множество позиций $P = \{p_1, p_2, p_3, p_4, p_5\}$ и множество переходов $T = \{t_1, t_2, t_3, t_4\}$, ниже каждого из которых $(\forall t_j \in T)$ указываются значение функции принадлежности нечеткого срабатывания f_j. С целью удобства вектор значений порогов нечеткого срабатывания переходов $\lambda = (0.6, 0.5, 0.4, 0.5)$ и вектор начальной маркировки $\boldsymbol{m}_0 = (0.95, 0.9, 0, 0, 0)$ указаны отдельно.

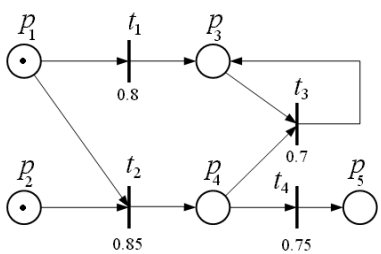


Рис. 10.6. Графическое изображение НСП C_f с начальной маркировкой $\boldsymbol{m}_0 = (0.95, 0.9, 0, 0, 0)$ и вектором значений порога нечеткого срабатывания $\lambda = (0.6, 0.5, 0.4, 0.5)$

Динамика изменения начальной и последующих маркировок НСП C_f после момента ее запуска подчиняется следующим *Правилам* $\mathbb{P}(C_f)$:

- ($\mathbb{P}_1$) ***Правило определения текущей маркировки***. Любое текущее состояние НСП C_f определяется вектором $\boldsymbol{m} = (m_1, m_2, ..., m_n)$, компоненты которого ($m_i \in [0, 1]$) интерпретируются как значения функции принадлежности нечеткого наличия одного маркера в соответствующих позициях $p_i \in P$ НСП C_f. Начальное состояние НСП определяется вектором начальной маркировки $\boldsymbol{m}_0$.

- ($\mathbb{P}_2$) ***Правило (условие) активности перехода***. Переход $t_k \in T$ НСП C_f называется *активным* (разрешенным, возбужденным) при некоторой текущей маркировке $\boldsymbol{m}$, если выполнено следующее условие:

$$\min_{(i \in \{1, 2, ..., n\}) \wedge (I(p_i, t_k) > 0)} \{m_i\} \geq \lambda_k , \qquad (10.13)$$

  где λ_k — значения порога срабатывания перехода $t_k \in T$. Другими словами, переход $t_k \in T$ НСП C_f является активным, если во всех его входных позициях имеются ненулевые значения компонентов вектора текущей маркировки, а минимальное из них — не меньше порога срабатывания рассматриваемого перехода.

- ($\mathbb{P}_3$) ***Правило нечеткого срабатывания перехода***. Если переход $t_k \in T$ НСП C_f является активным при некоторой текущей маркировке $\boldsymbol{m}$, (т. е. для него выполнено условие (10.13)), то нечеткое срабатывание данного перехода, осуществляемое мгновенным образом, приводит к новой маркировке $\boldsymbol{m}_v = (m_1^v, m_2^v, ..., m_n^v)$, компоненты вектора которой определяются по следующим формулам:

  - для каждой из входных позиций $p_i \in P$, для которых $I(p_i, t_k) > 0$:

$$m_i^v = 0, \qquad (\forall p_i \in P) \wedge (I(p_i, t_k) > 0); \qquad (10.14)$$

  - для каждой из выходных позиций $p_j \in P$, для которых $O(t_k, p_j) > 0$:

$$m_j^v = \max\{m_j, \min_{(i \in \{1, 2, ..., n\}) \wedge (I(p_i, t_k) > 0)} \{m_i, f_k\}\} \quad (\forall p_j \in P) \wedge (O(t_k, p_j) > 0), \qquad (10.15)$$

    где f_k — значение функции принадлежности или мера возможности нечеткого срабатывания (запуска) перехода $t_k \in T$, которое задается при определении конкретной НСП C_f.

    Если некоторые из позиций $p_i \in P$ являются одновременно входными и выходными для разрешенного перехода $t_k \in T$, то для них компоненты вектора новой маркировки рассчитываются последовательно, вначале по формуле (10.14), а затем — по формуле (10.15).

Рассмотрим применение описанных Правил $\mathbb{P}(C_f)$ на примере изменения начальной и последующих маркировок НСП C_f, изображенной на рис. 10.6.

При начальной маркировке $\boldsymbol{m}_0 = (0.95, 0.9, 0, 0, 0)$ активными являются два перехода: t_1 и t_2, поскольку для каждого из них выполненоусловие (10.13), т. е.

$0.95>0.6$ и $\min\{0.95, 0.9\} = 0.9>0.5$, где $\lambda_1 = 0.6$ и $\lambda_2 = 0.5$ — значения порогов срабатывания переходов t_1 и t_2 соответственно. Других разрешенных переходов при данной начальной маркировке нет.

Нечеткое срабатывание перехода t_1 приводит к новой маркировке $\boldsymbol{m}_1$, компоненты вектора которой определяются по формулам (10.14)—(10.15) следующим образом. Поскольку позиция $p_1 \in P$ является входной для данного перехода, то для нее: $m_1^1=0$. Для позиции p_3, которая является выходной для перехода t_1: $m_3^1 = \max\{0, \min\{0.95, 0.8\}\} = 0.8$, где $f_1 = 0.8$. Значения $m_2^1 = m_2^0$, $m_4^1 = m_4^0$, $m_5^1 = m_5^0$ остаются без изменения, поскольку позиции p_2, p_4, p_5 не является ни входными, ни выходными для рассматриваемого активного перехода t_1. Таким образом, нечеткое срабатывание перехода t_1 приводит к изменению начальной маркировки $\boldsymbol{m}_0$ на новую маркировку $\boldsymbol{m}_1 = (0, 0.9, 0.8, 0, 0)$.

Нечеткое срабатывание перехода t_2 приводит к новой маркировке $\boldsymbol{m}_2$, компоненты вектора которой определяются по формулам (10.14)—(10.15) следующим образом. Поскольку позиции $p_1, p_2 \in P$ являются входными для данного перехода, то для них: $m_1^2=0$ и $m_2^2=0$. Для выходной позиции p_4 компонент новой маркировки равен: $m_4^2 = \max\{0, \min\{0.95, 0.9, 0.85\}\} = 0.85$. Значения $m_3^2=m_5^2=0$ остаются без изменения, поскольку позиции p_3 и p_5 не являются ни входными, ни выходными для рассматриваемого активного перехода t_2. Таким образом, нечеткое срабатывание перехода t_2 приводит к изменению начальной маркировки $\boldsymbol{m}_0$ на новую маркировку $\boldsymbol{m}_2 = (0, 0, 0, 0.85, 0)$.

Примечание

Следует заметить, что нечеткое срабатывание перехода t_1 делает неактивным переход t_2, а нечеткое срабатывание перехода t_2 в свою очередь делает неактивным переход t_1. Таким образом, данные переходы являются конфликтными, что приводит к их альтернативному срабатыванию.

При текущей маркировке $\boldsymbol{m}_1$ нет активных переходов, а при маркировке $\boldsymbol{m}_2$ активным является переход $t_4 \in T$, поскольку для него выполнено условие (10.13): $0.85>0.5$, где $\lambda_4 = 0.5$. Нечеткое срабатывание перехода t_4 изменит маркировку $\boldsymbol{m}_2$ на новую маркировку $\boldsymbol{m}_3 = (0, 0, 0, 0, 0.75)$, где значение $m_5^3 = 0.75$ получается с использованием формулы (10.15). При полученной в результате маркировке $\boldsymbol{m}_3$ нет активных переходов, поэтому рассмотрение этого примера можно закончить.

Строгое определение диаграммы достижимых маркировок НСП C_f базируется на отношениях непосредственного следования и достижимости маркировок, определения которых приводятся ниже.

Непосредственная достижимость маркировок. Маркировка $\boldsymbol{m}_v$ НСП $C_f = (N, \boldsymbol{m}_0)$ *непосредственно достижима из маркировки* $\boldsymbol{m}$ (непосредственно следует за маркировкой $\boldsymbol{m}$) этой же НСП, если маркировка $\boldsymbol{m}_v$ получается в результате нечеткого срабатывания некоторого активного при $\boldsymbol{m}$ перехода

$t_k \in T$. Отношение непосредственного следования маркировок НСП C_f будем обозначать через: $\boldsymbol{m}\text{–}(t_k)\rightarrow\boldsymbol{m}_v$, где $\boldsymbol{m}$ и $\boldsymbol{m}_v$ — векторы соответствующих маркировок.

Очевидно, отношение непосредственной достижимости маркировок НСП C_f является бинарным транзитивным отношением на декартовом произведении множества всех возможных маркировок фиксированной НСП C_f.

Достижимость маркировок. Маркировка $\boldsymbol{m}_w$ НСП $C_f = (N, \boldsymbol{m}_0)$ называется *достижимой из маркировки* $\boldsymbol{m}$ этой же НСП, если существует конечная упорядоченная последовательность маркировок $M_k = <\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_q>$, где $\boldsymbol{m}_1=\boldsymbol{m}$, $\boldsymbol{m}_q=\boldsymbol{m}_w$, и соответствующая ей конечная упорядоченная последовательность переходов $\pi_l = <t_1, t_2, ..., t_{q-1}>$ такие, что любая пара соседних маркировок в M_k удовлетворяет отношению непосредственного следования маркировок: $\boldsymbol{m}_j\text{–}(t_j)\rightarrow\boldsymbol{m}_{j+1}$ $(\forall j \in \{1, 2, ..., q-1\})$. Отношение достижимости для НСП C_f будем обозначать через $\boldsymbol{m}\text{–}(\pi_l)\rightarrow\boldsymbol{m}_w$.

Отношение достижимости маркировок НСП C_f также является бинарным транзитивным отношением на декартовом произведении множества всех возможных маркировок фиксированной НСП. Как и в случае классических СП, данное отношение является транзитивным замыканием отношения непосредственной достижимости маркировок, в чем читателю предлагается убедиться самостоятельно.

Для НСП C_f (см. рис. 10.6) справедливы соотношения: $\boldsymbol{m}_0\text{–}(t_1)\rightarrow\boldsymbol{m}_1$, $\boldsymbol{m}_0\text{–}(t_2)\rightarrow\boldsymbol{m}_2$, $\boldsymbol{m}_2\text{–}(t_4)\rightarrow\boldsymbol{m}_3$ и $\boldsymbol{m}_0\text{–}(<t_2, t_4>)\rightarrow\boldsymbol{m}_3$.

Как и ранее, через $D(\boldsymbol{m})$ будем обозначать множество маркировок, достижимых из некоторой фиксированной маркировки $\boldsymbol{m}$ для заданной НСП $C_f = (N, \boldsymbol{m}_0)$, т. е. $D(\boldsymbol{m}) = \{\boldsymbol{m}_w \mid (\exists\, \pi_l)\ \boldsymbol{m}\text{–}(\pi_l)\rightarrow\boldsymbol{m}_w\}$.

Множество достижимых маркировок НСП C_f. Множество $D(\boldsymbol{m}_0)$ будем называть *множеством достижимых маркировок* для НСП $C_f = (N, \boldsymbol{m}_0)$ или просто — *множеством достижимости* НСП C_f.

Диаграммой достижимых маркировок (графом достижимости) НСП C_f будем называть ориентированный граф $D(V, E, \gamma)$, где $V=\{v_1, v_2, ..., v_q\}$ — множество вершин графа, которое соответствует множеству достижимых маркировок $D(\boldsymbol{m}_0)$; $E=\{e_{ij}\}$ $(\forall v_i, v_j \in V)\}$ — множество дуг, которое соответствует отношению непосредственной достижимости маркировок НСП C_f; γ: $E\rightarrow T$ — отображение, помечающее каждую дугу данного графа переходом, нечеткое срабатывание которого обеспечивает выполнение соответствующего отношения непосредственной достижимости для рассматриваемой пары маркировок.

Диаграмма достижимых маркировок, построенная для НСП C_f из примера 10.3 в форме диаграммы деятельности нотации UML, представлена на рис. 10.7. На этой диаграмме срабатывание переходов понимается в смысле нечеткого срабатывания, как это определяется правилом $\mathbb{P}_3$.

Некоторые примеры построения нечетких моделей продукционных систем с использованием нечетких сетей Петри типа C_f приводятся в *разд. 10.4*.

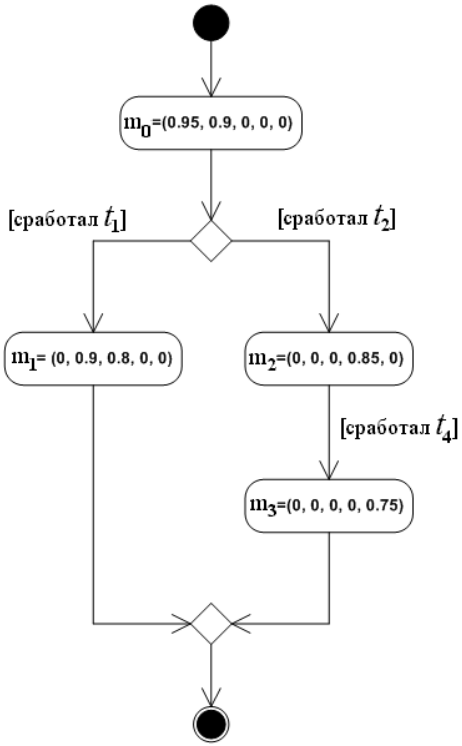


Рис. 10.7. Диаграмма достижимых маркировок для НСП C_f из примера 10.3

Обобщенные нечеткие временные сети Петри типа C_{PT}^f

Третий и наиболее общий подкласс НСП, называемый здесь обобщенными нечеткими временными СП типа C_{PT}^f или кратко — ОНВСП C_{PT}^f, может быть получен введением описания неопределенности нечеткого характера в формализм обобщенных временных СП типа C_{PT}, что также позволяет сформулировать определения различных более конкретных подклассов ОНВСП в качестве частных случаев. Однако более детальное рассмотрение последних выходит за рамки данной книги и может служить предметом специальных исследований.

Обобщенная нечеткая временная сеть Петри типа C_{PT}^f (ОНВСП C_{PT}^f). *Обобщенная нечеткая временная сеть Петри типа* C_{PT}^f может быть определена как $C_{PT}^f = (N, \boldsymbol{m}_0(\omega), \boldsymbol{z}(\omega), \boldsymbol{s}(\omega))$, где:

- $N = (P, T, I, O)$ — структура ОНВСП C_{PT}^f, которая аналогична структуре обычных СП и для которой $I: P \times T \to \mathbb{N}_0$ — входная функция переходов; $O: T \times P \to \mathbb{N}_0$ — выходная функция переходов;

- $\boldsymbol{m}_0(\omega)= (m_1^0, m_2^0,..., m_n^0)$ — вектор начальной маркировки, с каждым компонентом m_i^0 которого связано некоторое пространство с нечеткой мерой $(\Omega_i, \Im_i, G_i)$ $(\forall i \in \{1, 2,..., n\})$;
- $\boldsymbol{z}(\omega) = (z_1, z_2,..., z_n)$ — вектор *параметров временных задержек маркеров* в позициях ОНВСП СРТ^f, с каждым компонентом z_i которого связано некоторое пространство с нечеткой мерой $(\Omega_i', \Im_i', G_i')$ $(\forall i \in \{1, 2,..., n\})$;
- $\boldsymbol{s}(\omega) = (s_1, s_2,..., s_u)$ — вектор *параметров времен срабатывания активных переходов*, с каждым компонентом s_j которого также связано некоторое пространство с нечеткой мерой $(\Omega_j'', \Im_j'', G_j'')$ $(\forall j \in \{1, 2,..., u\})$.

Данное определение ОНВСП СРТ^f является слишком общим и абстрактным, что затрудняет практическое использование соответствующих моделей при решении прикладных задач нечеткого моделирования. Однако математическая структура данного класса СРТ^f допускает различные конкретизации на основе рассмотрения конкретных способов задания нечетких мер. Наиболее перспективным и интересным в контексте нечеткого моделирования представляется рассматриваемый ниже подкласс ОНВСП СРТ^f, который получается на основе задания параметров векторов $\boldsymbol{z}$ и $\boldsymbol{s}$ в виде нечетких величин *(см. главу 5)*.

Нечеткая временная сеть Петри типа СРТ^v (НВСП СРТ^v). *Нечеткой временной сетью Петри типа* СРТ^v будем называть подкласс ОНВСП СРТ^f, который определяется как $\text{СРТ}^v=(N, \mu_0, \boldsymbol{z}_v, \boldsymbol{s}_v)$, где:

- $N =(P, T, I, O)$ — структура НВСП СРТ^v, которая аналогична структуре ординарных СП и для которой I: $P\times T\rightarrow\{0, 1\}$ — входная функция переходов; O: $T\times P\rightarrow\{0, 1\}$ — выходная функция переходов;
- $\boldsymbol{m}_0= (m_1^0, m_2^0,..., m_n^0)$ — вектор начальной маркировки, каждый компонент m_i^0 которого представляет собой некоторую неотрицательную нечеткую величину;
- $\boldsymbol{z}_v= (z_1^v, z_2^v,..., z_n^v)$ — вектор параметров *временных задержек маркеров в позициях* НВСП СРТ^v, каждый компонент z_i^v которого представляет собой некоторую неотрицательную нечеткую величину;
- $\boldsymbol{s}_v= (s_1^v, s_2^v,..., s_u^v)$ — вектор параметров *времен срабатывания разрешенных переходов* НВСП СРТ^v, каждый компонент s_j^v которого также представляет собой некоторую неотрицательную нечеткую величину.

Примечание

Напомним, что используемое в данном определении понятие неотрицательной нечеткой величины представляет собой нечеткое множество $\mathcal{B}=\{x, \mu_{\mathcal{B}}(x)\}$, заданное на множестве неотрицательных действительных чисел $\mathbb{R}_+$, т. е. для которого универсумом $\boldsymbol{X}$ служит множество $\mathbb{R}_+$. Другими словами, функции принадлежности компонентов векторов $\boldsymbol{m}_0$, $\boldsymbol{z}_v$ и $\boldsymbol{s}_v$ определяются как μ_m: $\mathbb{R}_+\rightarrow[0, 1]$, μ_z: $\mathbb{R}_+\rightarrow[0, 1]$ и μ_s: $\mathbb{R}_+\rightarrow[0, 1]$ соответственно.

Поскольку понятие нечеткой величины допускает конкретизацию в виде нечеткого интервала и нечеткого числа, которые наиболее удобно с вычислительной точки зрения использовать в форме (L-R)-представления, то рассматриваемый подкласс НВСП C_{PT}^{v} также может быть конкретизирован применительно к указанной форме представления $\boldsymbol{m}_0$, $\boldsymbol{z}_v$ и $\boldsymbol{s}_v$. Следуя в этом направлении, можно определить наиболее удобный с вычислительной точки зрения подкласс НВСП, основанный на задании указанных векторов в форме трапециевидных нечетких интервалов (ТНИ) и треугольных нечетких чисел (ТНЧ), которые были определены в *главе 5*.

Нечеткая временная сеть Петри типа C_{PT}^{T} (НВСП C_{PT}^{T}). *Нечеткой временной сетью Петри типа* C_{PT}^{T} будем называть подкласс НВСП C_{PT}^{v}, который определяется как $C_{PT}^{T} = (N, \boldsymbol{m}_0, \boldsymbol{z}_T, \boldsymbol{s}_T)$, где:

- $N = (P, T, I, O)$ — структура НВСП C_{PT}^{T}, которая аналогична структуре ординарных СП и для которой I: $P \times T \to \{0, 1\}$ — входная функция переходов; O: $T \times P \to \{0, 1\}$ — выходная функция переходов;
- $\boldsymbol{m}_0 = (m_1^0, m_2^0, ..., m_n^0)$ — вектор начальной маркировки, каждый компонент m_i^0 которого представляет собой ТНИ $\mathcal{M}_{T_i} = \langle a_i, b_i, \alpha_i, \beta_i \rangle_T$ $(\forall i \in \{1, 2, ..., n\})$;
- $\boldsymbol{z}_T = (z_1, z_2, ..., z_n)$ — вектор параметров *временных задержек маркеров в позициях* НВСП C_{PT}^{T}, каждый компонент z_i которого представляет собой ТНИ $\mathcal{Z}_{T_i} = \langle a_i, b_i, \alpha_i, \beta_i \rangle_T$ $(\forall i \in \{1, 2, ..., n\})$;
- $\boldsymbol{s}_T = (s_1, s_2, ..., s_u)$ — вектор параметров *времен срабатывания разрешенных переходов* НВСП C_{PT}^{T}, каждый компонент s_j которого также представляет собой ТНИ $\mathcal{S}_{T_j} = \langle a_j, b_j, \alpha_j, \beta_j \rangle_T$ $(\forall j \in \{1, 2, ..., u\})$.

Примечание

Если в данном определении НВСП C_{PT}^{T} все ТНИ в векторах $\boldsymbol{m}_0$, $\boldsymbol{z}_T$, $\boldsymbol{s}_T$ заменить на ТНЧ $\mathcal{M}_{\Delta i} = \langle a_i, \alpha_i, \beta_i \rangle_\Delta$, $\mathcal{Z}_{\Delta i} = \langle a_i, \alpha_i, \beta_i \rangle_\Delta$ и $\mathcal{S}_{\Delta j} = \langle a_j, \alpha_j, \beta_j \rangle_\Delta$ соответственно, то получим определение НВСП $C_{PT}^{\Delta} = (N, \boldsymbol{m}_0, \boldsymbol{z}_\Delta, \boldsymbol{s}_\Delta)$, которая является частным случаем НВСП C_{PT}^{T}. На основе НВСП C_{PT}^{T} и C_{PT}^{Δ} могут быть также получены определения отдельных подклассов НВСП C_P^{T}, C_P^{Δ} и НВСП C_T^{T} и C_T^{Δ} путем исключения нечеткости из рассмотрения векторов $\boldsymbol{s}_T$, $\boldsymbol{s}_\Delta$ и $\boldsymbol{z}_T$, $\boldsymbol{z}_\Delta$ соответственно, что предлагается выполнить читателям самостоятельно в качестве упражнения. В то же время полное исключение неопределенности при задании векторов $\boldsymbol{s}_T$, $\boldsymbol{s}_\Delta$, $\boldsymbol{z}_T$, $\boldsymbol{z}_\Delta$ приводит к определению обычных временных СП. В случае частичного исключения неопределенности при задании векторов $\boldsymbol{s}_T$, $\boldsymbol{s}_\Delta$ и $\boldsymbol{z}_T$, $\boldsymbol{z}_\Delta$, т. е. при задании их компонентов в виде неотрицательных целых или действительных чисел, могут быть получены определения комбинированных НВСП, рассмотрение которых выходит за рамки настоящей книги.

Графически НВСП C_{PT}^{T} и C_{PT}^{Δ} представляются в виде ориентированного двудольного графа, аналогично классическим ординарным СП.

Пример 10.4. Изображенная на рис. 10.8 НВСП C_{PT}^{T} имеет множество позиций $P = \{p_1, p_2, p_3, p_4, p_5\}$ и множество переходов $T = \{t_1, t_2, t_3,\}$. Для каждой из пози-

ций дополнительно следует указать времена задержек маркеров в виде ТНИ: $z_1=\mathcal{Z}_{T1}=<2, 3, 1, 1>_T$, $z_2=\mathcal{Z}_{T2}=<3, 5, 1, 1>_T$, $z_3=\mathcal{Z}_{T3}=<1, 2, 0, 0>_T$, $z_4=\mathcal{Z}_{T4}=<2, 4, 1, 1>_T$, $z_5=\mathcal{Z}_{T5}=<0, 0, 0, 0>_T$. Так, например, $z_1^T=<2, 3, 1, 1>_T$ означает, что временная задержка маркера в позиции $p_1 \in P$ представляет собой ТНИ $\mathcal{Z}_{T1}$, для которого: $a=2$, $b=3$, $\alpha=1$, $\beta=1$. Для каждого из переходов дополнительно следует указать времена срабатывания активных переходов в виде ТНИ: $s_1=\mathcal{S}_{T1}=<1, 2, 0, 1>_T$, $s_2=\mathcal{S}_{T2}=<0, 1, 0, 1>_T$, $s_3=\mathcal{S}_{T3}=<2, 3, 1, 0>_T$. Так, например, $s_1=<1, 2, 0, 1>_T$ означает, что время срабатывания перехода $t_1 \in T$ в случае его активности представляет собой ТНИ $\mathcal{S}_{T1}$, для которого: $a=1$, $b=2$, $\alpha=0$, $\beta=1$.

Начальная маркировка для этой НВСП также должна быть задана в виде вектора с компонентами в форме ТНИ: $m_1^0=\mathcal{M}_{T1}=<1, 2, 1, 1>_T$, $m_2^0=\mathcal{M}_{T2}=<2, 3, 0, 1>_T$, $m_3^0=\mathcal{M}_{T3}=<0, 0, 0, 0>_T$, $m_4^0=\mathcal{M}_{T4}=<0, 0, 0, 0>_T$, $m_5^0=\mathcal{M}_{T5}=<0, 0, 0, 0>_T$. Так, например, $m_1^0=<1, 2, 1, 1>_T$ означает, что возможность наличия одного маркера в начальной маркировке $\boldsymbol{m}_0$ в позиции $p_1 \in P$ представляет собой ТНИ $\mathcal{M}_{T1}$, для которого: $a=1$, $b=2$, $\alpha=1$, $\beta=1$. Остальные компоненты векторов $\boldsymbol{m}_0$, $\boldsymbol{z}_T$, $\boldsymbol{s}_T$ интерпретируются аналогичным образом.

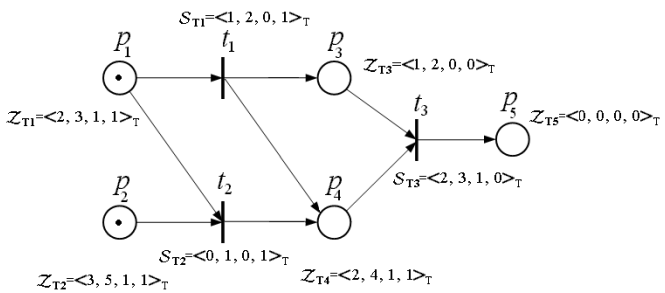


Рис. 10.8. Графическое изображение НВСП C_{PT}^T с начальной маркировкой $m_1^0=\mathcal{M}_{T1}=<1, 2, 1, 1>_T$, $m_2^0=\mathcal{M}_{T2}=<2, 3, 0, 1>_T$, $m_3^0=\mathcal{M}_{T3}=<0, 0, 0, 0>_T$, $m_4^0=\mathcal{M}_{T4}=<0, 0, 0, 0>_T$, $m_5^0=\mathcal{M}_{T5}=<0, 0, 0, 0>_T$

Примечание

Изображенная на рис. 10.8 структура НВСП C_{PT}^T может представлять собой пример НВСП C_{PT}^Δ в случае задания компонентов векторов $\boldsymbol{m}_0$, $\boldsymbol{z}_T$, $\boldsymbol{s}_T$ в виде ТНЧ $\mathcal{A}_\Delta=<a, \alpha, \beta>_\Delta$. Так, например, если вместо векторов $\boldsymbol{m}_0$, $\boldsymbol{z}_T$, $\boldsymbol{s}_T$ задать вектора $\boldsymbol{m}_0$, $\boldsymbol{z}_\Delta$, $\boldsymbol{s}_\Delta$ в виде: $z_1=\mathcal{Z}_{\Delta 1}=<2, 1, 1>_\Delta$, $z_2=\mathcal{Z}_{\Delta 2}=<3, 1, 2>_\Delta$, $z_3=\mathcal{Z}_{\Delta 3}=<1, 0, 1>_\Delta$, $z_4=\mathcal{Z}_{\Delta 4}=<2, 1, 1>_\Delta$, $z_5=\mathcal{Z}_{\Delta 5}=<0, 0, 0>_\Delta$, $s_1=\mathcal{S}_{\Delta 1}=<1, 0, 1>_\Delta$, $s_2=\mathcal{S}_{\Delta 2}=<1, 0, 2>_\Delta$, $s_3=\mathcal{S}_{\Delta 3}=<2, 1, 1>_\Delta$ и начальной маркировкой, например, $m_1^0=\mathcal{M}_{\Delta 1}=<1, 1, 1>_\Delta$, $m_2^0=\mathcal{M}_{\Delta 2}=<2, 2, 1>_\Delta$, $m_3^0=\mathcal{M}_{\Delta 3}=<0, 0, 0>_\Delta$, $m_4^0=\mathcal{M}_{\Delta 4}=<0, 0, 0>_\Delta$, $m_5^0=\mathcal{M}_{\Delta 5}=<0, 0, 0>_\Delta$, то получим пример НВСП C_{PT}^Δ со структурой N из примера 10.4. В этом случае значение $z_1=<2, 1, 1>_\Delta$ означает, что временная задержка маркера в позиции $p_1 \in P$ представляет собой ТНЧ $\mathcal{Z}_{\Delta 1}$, для которого $a=2$, $\alpha=1$, $\beta=1$. Значение

s_1=<1, 0, 1>$_\Delta$ означает, что время срабатывания перехода $t_1 \in T$ в случае его активности представляет собой ТНЧ $\mathcal{S}_{\Delta 1}$, для которого: a=1, α=0, β=1. Остальные компоненты векторов $\boldsymbol{m}_0$, $\boldsymbol{z}_\Delta$, $\boldsymbol{s}_\Delta$ интерпретируются аналогичным образом.

Для описания процесса изменения начальной и последующих маркировок ОНВСП $C_{PT}{}^f$ можно использовать различные правила, определяющие конкретные условия активности и правила срабатывания переходов, аналогично правилам $\mathbb{P}(C_f)$ и $\mathbb{P}(V_f)$. Однако, поскольку данный класс ОНВСП $C_{PT}{}^f$ является наиболее общим и абстрактным среди всех ранее рассмотренных, ниже приводится конкретизация данных правил только применительно для его подкласса безопасных НВСП $C_{PT}{}^T$. При этом безопасность НВСП понимается в смысле возможности наличия в каждой из позиций сети не более одного маркера, задаваемого соответствующим ТНИ.

Динамика изменения начальной и последующих маркировок безопасных НВСП $C_{PT}{}^T$ (далее просто — НВСП $C_{PT}{}^T$) после момента их запуска подчиняется следующим *Правилам* $\mathbb{P}(C_{PT}{}^T)$.

- ($\mathbb{P}_1$) ***Правило определения текущей маркировки***. Любое текущее, в том числе и начальное состояние НВСП $C_{PT}{}^T$ определяется вектором $\boldsymbol{m} = (m_1, m_2, ..., m_n)$, компоненты которого m_i представляют собой ТНИ $\mathcal{M}_{Ti} = <a_i, b_i, \alpha_i, \beta_i>_T$ ($\forall i \in \{1, 2, ..., n\}$) и интерпретируются как значения функции принадлежности нечеткого наличия одного маркера в соответствующих позициях $p_i \in P$ относительно времени, отсчитываемого от момента запуска данной НВСП $C_{PT}{}^T$. Начальное состояние НВСП $C_{PT}{}^T$ определяется вектором начальной маркировки $\boldsymbol{m}_0$.

- ($\mathbb{P}_2$) ***Правило (условие) активности перехода***. Переход $t_k \in T$ НВСП $C_{PT}{}^T$ называется *активным* (разрешенным, возбужденным) при некоторой доступной маркировке $\boldsymbol{m}$, если выполнено следующее условие:

$$\mathcal{M}_{Ti} > 0 \qquad (\forall p_i \in P) \wedge (I(p_i, t_k) > 0), \qquad (10.16)$$

  т. е. во всех входных позициях рассматриваемого перехода на момент времени τ должны быть *доступные* маркеры, представленные отличными от нуля НТНИ. При этом неравенство (10.16) следует понимать применительно к функции принадлежности соответствующего ТНИ для аргумента $\tau \in \mathbb{R}_+$. Доступность маркеров в позициях НВСП определяется согласно правилу $\mathbb{P}_4$.

- ($\mathbb{P}_3$) ***Правило нечеткого срабатывания перехода***. Если переход $t_k \in T$ НВСП $C_{PT}{}^T$ активен при некоторой доступной маркировке $\boldsymbol{m}$, т. е. для него выполнено условие (10.16), то нечеткое срабатывание данного перехода, осуществляемое за время $s_k = \mathcal{S}_{Tk}$, приводит к новой маркировке $\boldsymbol{m}_v$, компоненты вектора которой определяются следующим образом:

  - для каждой из входных позиций $p_i \in P$, для которых $I(p_i, t_k) > 0$, по формуле:

$$m_i{}^v = <0, 0, 0, 0>_T \qquad (\forall p_i \in P) \wedge (I(p_i, t_k) > 0); \qquad (10.17)$$

- для каждой из выходных позиций $p_j \in P$, для которых $O(t_k, p_j)>0$, по формуле:

$$m_j^{v} = \min\{\max_{(i\in\{1, 2, \ldots, n\})\wedge(I(p_i, t_k)>0)}\{m_i\}+s_k, m_j\} \qquad (10.18)$$
$$(\forall p_j \in P)\wedge(O(t_k, p_j)>0)\wedge(\mathcal{M}_{T_j}\neq<0, 0, 0, 0>_T),$$

  где "**max**" — расширенная операция максимума для ТНИ, определяемая по формуле (5.27), "**min**" — расширенная операция минимума для ТНИ, определяемая по формуле (5.28), а "+" — операция сложения ТНИ, определяемая по формуле (5.20);

- для каждой позиции $p_i \in P$, не являющейся ни входной, ни выходной, по формуле:

$$m_i^{v} = m_i \qquad (\forall p_i \in P)\wedge(I(p_i, t_k)=0)\ (O(t_k, p_j)=0). \qquad (10.19)$$

Если некоторые из позиций $p_j \in P$ являются одновременно входными и выходными для активного перехода $t_k \in T$, то для них компоненты вектора новой маркировки рассчитываются последовательно, вначале по формуле (10.17), а затем — по формуле (10.18).

❒ ($\mathbb{P}_4$) ***Правило нечеткой задержки маркеров в позициях***. После нечеткого срабатывания активного перехода согласно правилу $\mathbb{P}_3$ маркеры в выходных позициях для новой маркировки $\boldsymbol{m}_v$ в общем случае являются *недоступными.* На них начинают действовать временные задержки в соответствующих выходных позициях сработавшего перехода, определяемые вектором $\boldsymbol{z}_T$. Соответствующие маркеры становятся *доступными* только после окончания действия временных задержек, которые определяют доступную маркировку $\boldsymbol{m}_v'$ по формуле:

$$\mathcal{M}_{T_j}' = \mathcal{M}_{T_j} + \mathcal{Z}_{T_j} \qquad (\forall p_j \in P)\wedge(O(t_k, p_j)>0)\wedge(\mathcal{M}_{T_i}\neq<0, 0, 0, 0>_T), \qquad (10.20)$$

где операция сложения понимается как сложение ТНИ по формуле (5.20). Для выходных позиций p_j перехода t_k с $\mathcal{M}_{T_j}=<0, 0, 0, 0>_T$ по определению $\mathcal{M}_{T_j}'=<0, 0, 0, 0>_T$.

Примечание

Указанные правила $\mathbb{P}(C_{PT}^{T})$ могут быть расширены на общий случай небезопасных НВСП C_{PT}^{T}. Однако возможность наличия в позициях НВСП C_{PT}^{T} нескольких маркеров приведет к необходимости рассмотрения в качестве текущей маркировки некоторой матрицы, элементами которой будут ТНИ. Интерпретация последних будет соответствовать значениям функции принадлежности нечеткого наличия соответствующего количества маркеров в позициях данного подкласса НВСП для определенных моментов времени. Данное обстоятельство существенно усложняет процесс анализа свойств небезопасных НВСП C_{PT}^{T}, поэтому в дальнейшем наше внимание будет ограничено рассмотрением только безопасных НВСП C_{PT}^{T} и C_{PT}^{Δ}. Следует также отметить, что если в правилах $\mathbb{P}(C_{PT}^{T})$ всюду заменить ТНИ на ТНЧ, то получим соответствующие правила $\mathbb{P}(C_{PT}^{\Delta})$, которые описывают динамику изменения начальной и последующих маркировок безопасных НВСП C_{PT}^{Δ}.

Проиллюстрируем применение описанных правил $\mathbb{P}(C_{PT}{}^{T})$ на примере изменения начальной маркировки НВСП $C_{PT}{}^{T}$, изображенной на рис. 10.8.

Согласно правилу $\mathbb{P}_1$ начальное состояние данной НВСП $C_{PT}{}^{T}$ определяется вектором $\boldsymbol{m}_0=(m_1{}^0, m_2{}^0, m_3{}^0, m_4{}^0, m_5{}^0)$, компоненты которого являются ТНИ: $m_1{}^0=\mathcal{M}_{T1}=<1, 2, 1, 1>_T$, $m_2{}^0=\mathcal{M}_{T2}=<2, 3, 0, 1>_T$, $m_3{}^0=\mathcal{M}_{T3}=<0, 0, 0, 0>_T$, $m_4{}^0=\mathcal{M}_{T4}=$ $=<0, 0, 0, 0>_T$, $m_5{}^0=\mathcal{M}_{T5}=<0, 0, 0, 0>_T$. Согласно правилу $\mathbb{P}_4$ на начальную маркировку $\boldsymbol{m}_0$ начинают действовать временные задержки маркеров в позициях данной НВСП. Поэтому доступной начальной маркировкой будет вектор $\boldsymbol{m}_0'$, компоненты которого равны: $m_1'=\mathcal{M}_{T1}=<3, 5, 2, 2>_T$, $m_2'=\mathcal{M}_{T2}=<5, 8, 1, 2>_T$, $m_3'=$ $=\mathcal{M}_{T3}=<0, 0, 0, 0>_T$, $m_4'=\mathcal{M}_{T4}=<0, 0, 0, 0>_T$, $m_5'=\mathcal{M}_{T5}=<0, 0, 0, 0>_T$. Согласно правилу $\mathbb{P}_2$ для доступной маркировки $\boldsymbol{m}_0'$ активными будут переходы t_1 и t_2, поскольку для них выполнено условие (10.16).

Нечеткое срабатывание перехода t_1 приводит к новой маркировке $\boldsymbol{m}_v$, компоненты вектора которой определяются согласно правилу $\mathbb{P}_3$ и равны соответственно: $m_1{}^v=\mathcal{M}_{T1}=<0, 0, 0, 0>_T$, поскольку позиция $p_1\in P$ является входной для данного перехода; $m_2{}^v=\mathcal{M}_{T2}=<5, 8, 1, 2>_T$, поскольку позиция p_2 не является выходной для рассматриваемого активного перехода t_1; $m_3{}^v=\mathcal{M}_{T3}=<3, 5, 2, 2>_T+<1, 2, 0, 1>_T=$ $=<4, 7, 2, 3>_T$, поскольку позиция p_3 является выходной для перехода t_1; $m_4{}^v=\mathcal{M}_{T4}=<3, 5, 2, 2>_T+<1, 2, 0, 1>_T=<4, 7, 2, 3>_T$, поскольку позиция p_4 также является выходной для перехода t_1; $m_5{}^v=\mathcal{M}_{T5}=<0, 0, 0, 0>_T$, поскольку позиция $p_5\in P$ не является выходной для рассматриваемого разрешенного перехода t_1.

Нечеткое срабатывание перехода t_2, также активного при маркировке $\boldsymbol{m}_0'$ приводит к новой маркировке $\boldsymbol{m}_w$. Компоненты вектора этой маркировки определяются по правилу $\mathbb{P}_3$ и равны соответственно: $m_1{}^w=\mathcal{M}_{T1}=<0, 0, 0, 0>_T$, $m_2{}^w=\mathcal{M}_{T2}=<0, 0, 0, 0>_T$, т. к. позиции p_1 и p_2 — входные для перехода t_2; $m_4{}^w=\mathcal{M}_{T4}=\mathbf{max}\{<3, 5, 2, 2>_T, <5, 8, 1, 2>_T\}+<0, 1, 0, 1>_T=<5, 8, 1, 2>_T+<0, 1, 0, 1>_T=$ $=<5, 9, 1, 3>_T$, т. к. позиция p_4 — выходная для перехода t_2; $m_3{}^w=\mathcal{M}_{T3}=$ $=<0, 0, 0, 0>_T$, $m_5{}^0=\mathcal{M}_{T5}=<0, 0, 0, 0>_T$ остаются без изменения, т. к. p_3, $p_5\in P$ не являются выходными для перехода t_2.

После нечеткого срабатывания переходов t_1 и t_2 маркировки $\boldsymbol{m}_v$ и $\boldsymbol{m}_w$ являются недоступными, поскольку на маркеры в выходных позициях p_3 и p_4 действуют временные задержки. Согласно правилу $\mathbb{P}_4$, доступными будут маркировки $\boldsymbol{m}_v'$ и $\boldsymbol{m}_w'$, компоненты которых равны соответственно: $m_1{}^{v\prime}=<0, 0, 0, 0>_T$, $m_2{}^{v\prime}=$ $=<5, 8, 1, 2>_T$, $m_3{}^{v\prime}=<5, 9, 2, 3>_T$, $m_4{}^{v\prime}=<6, 11, 3, 4>_T$, $m_5{}^{v\prime}=<0, 0, 0, 0>_T$ и $m_1{}^{w\prime}=<0, 0, 0, 0>_T$, $m_2{}^{w\prime}=<0, 0, 0, 0>_T$, $m_3{}^{w\prime}=<0, 0, 0, 0>_T$, $m_4{}^{w\prime}=<7, 13, 2, 4>_T$, $m_5{}^{w\prime}=<0, 0, 0, 0>_T$.

Рассматривая маркировки $\boldsymbol{m}_v'$ и $\boldsymbol{m}_w'$ в качестве доступных, можно заметить, что при маркировке $\boldsymbol{m}_v'$ активным будет единственный переход t_3, а при маркировке $\boldsymbol{m}_w'$ активных переходов в данной НВСП нет. Закончить рассмотрение этого примера изменения маркировок предлагается читателям самостоятельно в качестве упражнения.

Анализ данного примера показывает, что для НВСП $C_{PT}{}^{T}$ можно определить отношения непосредственной достижимости и достижимости маркировок.

Непосредственная достижимость маркировок. Маркировка $\boldsymbol{m}_v$ НВСП $C_{PT}^{T}=(N, \boldsymbol{m}_0, \boldsymbol{z}_T, \boldsymbol{s}_T)$ *непосредственно достижима из маркировки* $\boldsymbol{m}$ (непосредственно следует за маркировкой $\boldsymbol{m}$) этой же НСП, если маркировка $\boldsymbol{m}_v$ получается в результате нечеткого срабатывания некоторого активного при $\boldsymbol{m}$ перехода $t_k \in T$ и является доступной для активности других переходов. Отношение непосредственного следования маркировок НВСП C_{PT}^{T} будем обозначать через: $\boldsymbol{m}\text{–}(t_k)\rightarrow\boldsymbol{m}_v$, где $\boldsymbol{m}$ и $\boldsymbol{m}_v$ — векторы соответствующих маркировок.

Достижимость маркировок. Маркировка $\boldsymbol{m}_w$ НВСП $C_{PT}^{T}=(N, \boldsymbol{m}_0, \boldsymbol{z}_T, \boldsymbol{s}_T)$ называется *достижимой из маркировки* $\boldsymbol{m}$ этой же НСП, если существует конечная упорядоченная последовательность маркировок $M_k=<\boldsymbol{m}_1, \boldsymbol{m}_2,..., \boldsymbol{m}_q>$, где $\boldsymbol{m}_1=\boldsymbol{m}$, $\boldsymbol{m}_q=\boldsymbol{m}_w$, и соответствующая ей конечная упорядоченная последовательность переходов $\pi_l=<t_1, t_2,..., t_{q-1}>$, такие, что любая пара соседних маркировок в M_k удовлетворяет отношению непосредственного следования маркировок: $\boldsymbol{m}_j\text{–}(t_j)\rightarrow\boldsymbol{m}_{j+1}$ $(\forall j\in\{1, 2,..., q-1\})$. Отношение достижимости для НВСП C_{PT}^{T} будем обозначать через $\boldsymbol{m}\text{–}(\pi_l)\rightarrow\boldsymbol{m}_w$.

Для НВСП C_{PT}^{T} (рис. 10.8) справедливы соотношения: $\boldsymbol{m}_0'\text{–}(t_1)\rightarrow\boldsymbol{m}_v'$ и $\boldsymbol{m}_0'\text{–}(t_2)\rightarrow\boldsymbol{m}_w'$.

Как и ранее, через $D(\boldsymbol{m})$ будем обозначать множество маркировок, достижимых из некоторой фиксированной маркировки $\boldsymbol{m}$ для заданной НВСП $C_{PT}^{T}=(N, \boldsymbol{m}_0, \boldsymbol{z}_T, \boldsymbol{s}_T)$, т. е. $D(\boldsymbol{m})=\{\boldsymbol{m}_w \mid (\exists\,\pi_l)\ \boldsymbol{m}\text{–}(\pi_l)\rightarrow\boldsymbol{m}_w\}$.

Множество достижимых маркировок НВСП C_{PT}^{T}. Множество $D(\boldsymbol{m}_0)$ будем называть *множеством достижимых маркировок* для НСП $C_f=(N, \boldsymbol{m}_0)$ или просто — *множеством достижимости* НСП C_f.

Диаграммой достижимых маркировок (графом достижимости) НВСП C_{PT}^{T} будем называть ориентированный граф $D(V, E, \gamma)$, где $V=\{v_1, v_2,..., v_q\}$ — множество вершин графа, которое соответствует множеству достижимых маркировок $D(\boldsymbol{m}_0)$; $E=\{e_{ij}\}$ $(\forall v_i, v_j\in V)\}$ — множество дуг, которое соответствует отношению непосредственной достижимости маркировок НВСП C_{PT}^{T}; γ: $E\rightarrow T$ — отображение, помечающее каждую дугу данного графа переходом, нечеткое срабатывание которого обеспечивает выполнение соответствующего отношения непосредственной достижимости для рассматриваемой пары маркировок.

Свойства нечетких сетей Петри

Проведение исследований с использованием нечетких сетей Петри также предполагает установление целого ряда конкретных свойств моделей с последующей интерпретацией полученных результатов применительно к решаемой задаче и проблемной области. Ниже приводятся определения только тех из основных динамических свойств НСП рассмотренных подклассов, которые допускают обобщение аналогичных свойств базового формализма классических СП.

Ограниченность позиций. Позиция $p_i\in P$ НСП $V_f=(N, \mathbf{M}_0)$ называется *k-ограниченной*, если для любой достижимой маркировки $\mathbf{M}\in D(\mathbf{M}_0)$ выполняется

условие: $\max\{m_{ij} \mid m_{ij}>0\} \le k-1$ для некоторого фиксированного значения $k \in \{1, 2, 3,...\}$. Здесь максимум берется по всем компонентам вектора $\boldsymbol{m_i}$, соответствующего i-ый строке матрицы достижимой маркировки **M**.

НСП $V_f = (N, \mathbf{M}_0)$ называется *k-ограниченной*, если все позиции множества P являются k-ограниченными.

Безопасность позиций. 1-ограниченная позиция НСП $V_f = (N, \mathbf{M}_0)$ называется *безопасной* позицией. 1-ограниченная НСП $V_f = (N, \mathbf{M}_0)$ называется *безопасной* НСП.

Устойчивость переходов. Переход t_i НСП V_f (а также НСП C_f и НВСП $C_{PT}{}^T$) называется *устойчивым*, если этот переход t_i остается активным при срабатывании любого другого активного при этой же маркировке перехода t_j. НСП называется устойчивой, если все ее переходы являются устойчивыми.

Уровни активности переходов НСП. Пусть задана НСП V_f (НСП C_f или НВСП $C_{PT}{}^T$).

- Переход $t_i \in T$ обладает *активностью уровня* 0 и называется *пассивным* или *мертвым*, если не существует ни одной достижимой в этой НСП маркировки, при которой этот переход был бы активным.
- Переход $t_i \in T$ обладает *активностью уровня* 1 и называется *потенциально активным* или *живым*, если существует достижимая в этой НСП маркировка, при которой этот переход является активным.
- Переход $t_i \in T$ обладает *активностью уровня* 2, если для любого натурального $s \in \mathbb{N}$ в этой НСП существует последовательность срабатываемых переходов π_k, в которой t_i присутствует по крайней мере s раз.
- Переход $t_i \in T$ обладает *активностью уровня* 3, если в этой НСП существует последовательность срабатываемых переходов π_k, в которой переход t_i присутствует неограниченное число раз.
- Переход $t_i \in T$ обладает *активностью уровня* 4 и называется *активным* или *живым*, если для любой достижимой в этой НСП маркировки существует последовательность срабатываемых переходов π_k, так что t_i является активным при некоторой другой достижимой в этой НСП маркировке, получаемой в результате последовательного срабатывания переходов из π_k.
- Переход $t_i \in T$ называется *потенциально мертвым*, если существует некоторая достижимая в этой НСП маркировка, такая что при любой достижимой из данной маркировки переход t_i не является активным.
- Рассматриваемая НСП обладает активностью уровня k, где $k \in \{0,1,2,3,4\}$, если все ее переходы обладают активностью уровня k.

Тупиковая маркировка. Достижимая в НСП маркировка называется *t_i-тупиковой* ($t_i \in T$), если переход t_i потенциально мертвый для данной маркировки. Если рассматриваемая достижимая в этой НСП маркировка t_i-тупиковая для всех переходов $t_i \in T$, то она называется *тупиковой*.

При моделировании процессов функционирования дискретных динамических систем с использованием НСП рассмотренных подклассов также могут быть сформулированы некоторые задачи, аналогичные задачам базового формализма СП. Так, например, *задача достижимости маркировки в НСП* состоит в установлении отношения достижимости некоторой заданной маркировки из начальной маркировки для рассматриваемой НСП.

Классификация нечетких сетей Петри

Ниже представлены основы теории математических структур СП с неопределенностью и схема порождения конкретных подклассов НСП, использующие понятие математической структуры пространств с нечеткой мерой. Дополнительное включение в схему порождения классов математических структур СП с неопределенностью концепции нечеткой меры позволяет существенно обогатить получаемые при этом модели НСП, увеличить их моделирующие способности и повысить их адекватность при решении практических задач. Полученная в результате классификация НСП фиксирует основные компоненты соответствующих моделей и позволяет систематизировать представление достаточно широкого класса математических структур НСП на базе единой аксиоматики.

При этом в основу систематизации классов СП с неопределенностью, включая НСП и НВСП, положена концепция порождения конкретных математических структур детерминированных СП путем введения в их компоненты различных видов неопределенности.

Для иллюстрации этой идеи рассмотрим схему порождения различных классов СП с неопределенностью, образуемых из базовой математической структуры класса *временных* СП (ВСП), которая определяется как $<C_{PT}, \mathbb{P}>$. Здесь $C_{PT} = (N, \boldsymbol{m}_0, \boldsymbol{z}, \boldsymbol{s})$ — базовый формализм ВСП, в котором:

- $N = (P, T, I, O)$ — структура ВСП C_{PT}, которая аналогична структуре обобщенных СП и для которой I: $P\times T\to\mathbb{N}_0$ — входная функция переходов; O: $T\times P\to\mathbb{N}_0$ — выходная функция переходов;
- $\boldsymbol{m}_0 = (m_1{}^0, m_2{}^0,..., m_n{}^0)$ — вектор *начальной маркировки*, каждый компонент $m_i{}^0$ которого представляет собой целое неотрицательное число: $m_i\in\mathbb{N}_0$ $(\forall i\in\{1, 2,..., n\})$;
- $\boldsymbol{z} = (z_1, z_2,..., z_n)$ — вектор *временных задержек маркеров в позициях* ВСП C_{PT}, каждый компонент z_i которого представляет собой целое неотрицательное число: $z_i\in\mathbb{N}_0$ $(\forall i\in\{1, 2,..., n\})$ $(\forall i\in\{1, 2,..., n\})$;
- $\boldsymbol{s} = (s_1, s_2,..., s_u)$ — вектор *времен срабатывания разрешенных переходов* ВСП C_{PT}, каждый компонент s_j которого также представляет собой целое неотрицательное число: $s_j\in\mathbb{N}_0$ $(\forall i\in\{1, 2,..., u\})$.

$\mathbb{P}$ — совокупность Правил $\mathbb{P}(C_{PT})$, определяющих процесс запуска и функционирования ВСП, которые включают в себя условия активности и срабатывания

переходов, доступности маркеров в позициях ВСП, изменения начальной и последующих маркировок, а также, возможно, другие условия.

В общем случае введение неопределенности в описание исходной математической структуры ВСП $C_{PT} = (N, \boldsymbol{m}_0, \boldsymbol{z}, \boldsymbol{s})$ предполагает задание одной или нескольких структур с неопределенностью $\mathbb{S}(\omega)$, которая может отражать стохастический, нечеткий или комбинированный характер ее проявления. Последовательно вводя описание неопределенности $\mathbb{S}(\omega)$ в отдельные компоненты базовой математической структуры (МС) $<C_{PT}, \mathbb{P}>$ можно получить следующие обобщенные классы ВСП $C_{PT} = (N, \boldsymbol{m}_0, \boldsymbol{z}, \boldsymbol{s})$ с неопределенностью согласно следующей *Схеме порождения*:

- $<N(\omega), \boldsymbol{m}_0, \boldsymbol{z}, \boldsymbol{s}, \mathbb{P}'>$ — МС ВСП с неопределенностью задания структуры $(P, T, I, O)_\omega$ ВСП C_{PT}, при этом $\mathbb{P}'$ обозначает модификацию исходных Правил функционирования ВСП $\mathbb{P}$, отражающих содержательный смысл вводимой неопределенности в структуру ВСП;
- $<N, \boldsymbol{m}_0(\omega), \boldsymbol{z}, \boldsymbol{s}, \mathbb{P}'>$ — МС ВСП с неопределенностью задания начальной маркировки $\boldsymbol{m}_0$. Конкретизацией данной МС являются рассмотренные выше НСП V_f при $\boldsymbol{z} = \mathbf{0}$, $\boldsymbol{s} = \mathbf{0}$, где $\mathbb{S}(\omega)$ определяется аксиоматикой нечеткой меры;
- $<N, \boldsymbol{m}_0, \boldsymbol{z}(\omega), \boldsymbol{s}(\omega), \mathbb{P}'>$ — МС ВСП с неопределенностью задания времен задержки маркеров в позициях и времен срабатывания активных переходов. Данная МС представляет собой определенный выше класс ОНВСП C_{PT}^f, применительно к которому $\mathbb{S}(\omega)$ определяется аксиоматикой пространства с нечеткой мерой $(\Omega, \Im, G)$;
- $<N, \boldsymbol{m}_0, \boldsymbol{z}, \boldsymbol{s}, \mathbb{P}'(\omega)>$ — МС ВСП с неопределенностью в задании Правил $\mathbb{P}$, определяющих процесс функционирования сети. Конкретизацией этой МС являются различные разновидности ВСП и СП, например, с заданием вероятностных или нечетких мер на множествах конфликтных переходов с целью исключения альтернативного ветвления на диаграмме достижимых маркировок.

Если в рассмотренной схеме порождения МС СП с неопределенностью под $\mathbb{S}(\omega)$ понимать математическую структуру с аксиомами нечеткой математики, то полученные подобным образом обобщения базового формализма классических МС естественно принять за определения соответствующих подклассов нечетких СП, что вполне согласуется с известными в литературе формальными определениями последних.

И, наконец, рассматривая различные комбинации совместного введения описания неопределенности $\mathbb{S}(\omega)$ в отдельные компоненты базовой МС ВСП и МС других классов детерминированных СП (например, в МС СП с разноцветными маркерами и дугами, в МС сетей предикат-переход), а также используя различные структуры $\mathbb{S}(\omega)$ стохастической, нечеткой или комбинированной природы, можно получить определение весьма богатых в математическом отношении МС СП различных классов, анализ свойств которых далеко выходит за рамки тематики книги и может служить предметом самостоятельных исследований.

Взаимосвязь различных подклассов нечетких сетей Петри, которые могут быть получены с использованием предложенной выше схемы порождения МС СП с неопределенностью, представлена на рис. 10.9 в форме *диаграммы классов* языка UML. На этой диаграмме базовая математическая структура временных СП используется в качестве своеобразного шаблона, параметрами которого являются те ее компоненты, в которые вводится описание неопределенности согласно рассмотренной выше схеме порождения подклассов СП.

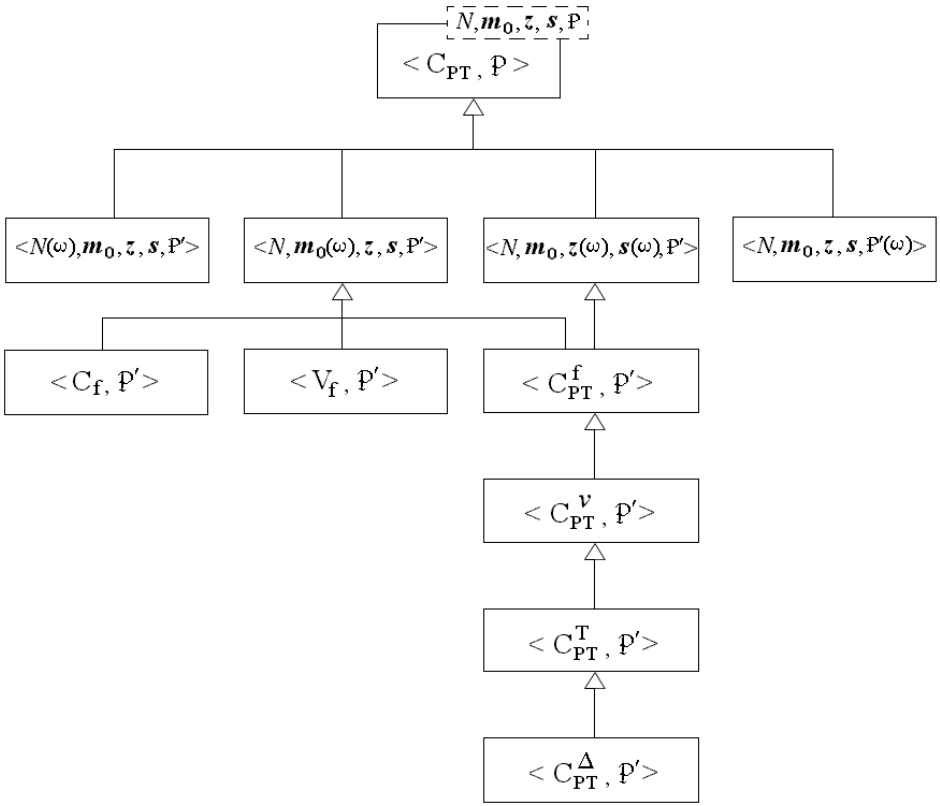


Рис. 10.9. Диаграмма классов нечетких сетей Петри, полученная на основе схемы порождения математических структур сетей Петри с неопределенностью

При изучении подклассов НСП основное внимание уделяется не столько теоретическому анализу отмеченных свойств НСП, сколько анализу конкретных нечетких моделей, построенных на их основе. Ниже рассматриваются некоторые конкретные особенности построения нечетких моделей на основе подклассов НСП и интерпретация их свойств с учетом специфики той или иной проблемной области.

10.3. Использование нечетких сетей Петри для представления правил нечетких продукций

При решении прикладных задач нечеткого моделирования и выполнения процесса приближенных рассуждений используются модифицированные НСП $C_f' = (N, \boldsymbol{f}, \lambda, \boldsymbol{m}_0)$, для которых правила $\mathbb{P}_1$, $\mathbb{P}_2$ такие же, как и рассмотренные выше для $\mathbb{P}(C_f)$, а правило $\mathbb{P}_3$ модифицировано и принимает следующий вид:

($\mathbb{P}_3'$) При расчете компонентов вектора новой маркировки $\boldsymbol{m}$ как для входных, так и для выходных позиций используется единая формула (10.15).

Это правило обусловлено тем обстоятельством, что НСП C_f' используют для интерпретации маркеров в позициях понятие нечеткой истинности высказывания. Значение последнего не становится равным нулю для высказываний в левой части правил продукций после их выполнения при данной интерпретации.

Одним из наиболее известных приложений НСП является их использование для наглядного представления правил нечетких продукций и выполнения на их основе вывода нечетких заключений.

В этом случае используется следующая интерпретация позиций и переходов НСП. Правило нечеткой продукции вида "ПРАВИЛО i: ЕСЛИ $\mathcal{A}$, ТО $\mathcal{B}$" представляется как некоторый переход $t_i \in T$ НСП $(N, \boldsymbol{f}, \lambda, \boldsymbol{m}_0)$, при этом условию $\mathcal{A}$ этого правила соответствует входная позиция $p_j \in P$ этого перехода t_i, а заключению — выходная позиция $p_k \in P$ этого перехода t_i (рис. 10.10, *а*).

Если условие правила нечеткой продукции состоит из нескольких подусловий, соединенных операцией нечеткой конъюнкции: $\mathcal{A} = \mathcal{A}_1 \wedge \mathcal{A}_2 \wedge \ldots \wedge \mathcal{A}_l$, то все эти подусловия представляются как входные позиции соответствующего перехода (рис. 10.10, *б* для случая $l=3$).

Если заключение правила нечеткой продукции состоит из нескольких подзаключений, соединенных операцией нечеткой конъюнкции: $\mathcal{B} = \mathcal{B}_1 \wedge \mathcal{B}_2 \wedge \ldots \wedge \mathcal{B}_l$, то все эти подзаключения также представляются как выходные позиции соответствующего перехода (рис. 10.10, *в* для случая $l=3$).

Более сложный случай соответствует дизъюнкции подусловий и подзаключений. Так, если условие правила нечеткой продукции состоит из нескольких подусловий, соединенных операцией нечеткой дизъюнкции: $\mathcal{A} = \mathcal{A}_1 \vee \mathcal{A}_2 \vee \ldots \vee \mathcal{A}_l$, то все эти подусловия представляются как входные позиции отдельных переходов t_i для $i \in \{1, 2, \ldots, l\}$ (рис. 10.10, *г* для случая $l=3$).

Если же заключение правила нечеткой продукции состоит из нескольких подзаключений, соединенных операцией нечеткой дизъюнкции: $\mathcal{B} = \mathcal{B}_1 \vee \mathcal{B}_2 \vee \ldots \vee \mathcal{B}_l$, то все эти подзаключения представляются как выходные позиции отдельных переходов t_i для $i \in \{1, 2, \ldots, l\}$ (рис. 10.10, *д* для случая $l=3$).

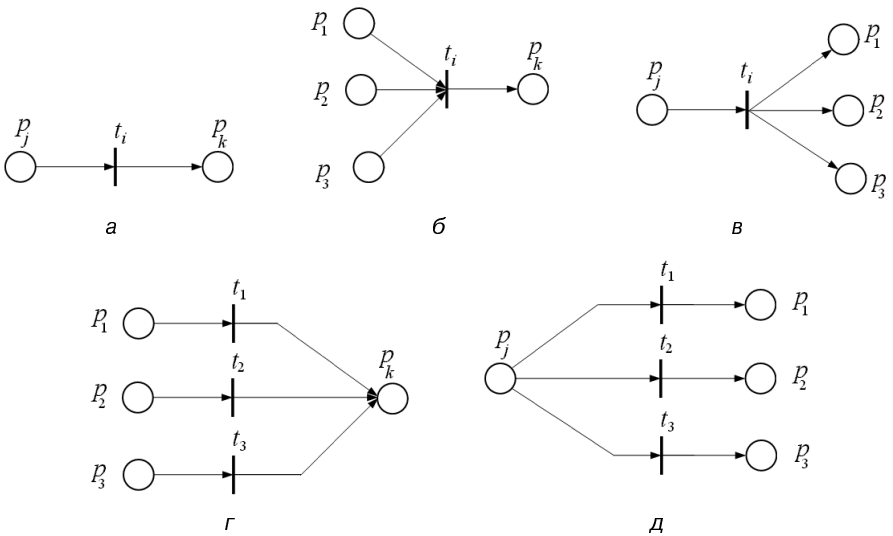


Рис. 10.10. Фрагменты нечетких сетей Петри для представления различных вариантов правил нечетких продукций

Таким образом, любое правило нечеткой продукции может быть представлено в виде фрагмента НСП. При этом веса или коэффициенты определенности F_i правил нечетких продукций преобразуются в вектор $\boldsymbol{f} = (f_1, f_2,..., f_u)$ значений функции принадлежности нечеткого срабатывания переходов, а степеням истинности подусловий правил соответствуют значения компонентов начальной маркировки $\boldsymbol{m}_0 = (m_1^0, m_2^0,..., m_n^0)$, которая в этом случае описывает текущую ситуацию моделируемой проблемной области.

Примечание

Следует заметить, что в дополнение к базовому формализму правил нечетких продукций в НСП можно учесть возможность активизации каждого из правил заданием вектора $\boldsymbol{\lambda} = (\lambda_1, \lambda_2,..., \lambda_u)$, компоненты которого определяют значения порога срабатывания переходов.

П р и м е р 10.4. Представим базу правил нечетких продукций из примера 6.5 в форме модифицированной НСП $C_f' = (N, \boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{m}_0)$ и рассмотрим пример вывода заключений с ее использованием на основе Правил $\mathbb{P}(C_f')$. Напомним, что в качестве системы нечетких продукций используется следующее множество правил нечетких продукций:

- ПРАВИЛО_1: ЕСЛИ "Гражданин не является высокопоставленным чиновником", ТО "он подвергается таможенному досмотру" (F_1=1.0)
- ПРАВИЛО_2: ЕСЛИ "Гражданин является высокопоставленным чиновником", ТО "он не подвергается таможенному досмотру" (F_2=0.9)

- ПРАВИЛО_3: ЕСЛИ "Гражданин не подвергается таможенному досмотру", ТО "не исключается возможность провоза наркотиков" (F_3=0.8)
- ПРАВИЛО_4: ЕСЛИ "Количество граждан, проходящих таможенный досмотр, велико", ТО "контролер испытывает чувство усталости" (F_4=0.6)
- ПРАВИЛО_5: ЕСЛИ "Контролер испытывает чувство усталости", ТО "не исключается возможность провоза наркотиков" (F_5=0.7)
- ПРАВИЛО_6: ЕСЛИ "Гражданин подвергается таможенному досмотру" И "в отношении этого гражданина имеется агентурная информация", ТО "исключается возможность провоза наркотиков" (F_6=0.95)
- ПРАВИЛО_7: ЕСЛИ "Гражданин подвергается таможенному досмотру" И "контролер использует новейшие технические средства", ТО "исключается возможность провоза наркотиков" (F_7=0.95)

Этой системе правил нечетких продукций соответствует НСП $C_f' = (N, \boldsymbol{f}, \lambda, \boldsymbol{m}_0)$, структура которой изображена на рис. 10.11. При этом все компоненты вектора λ принимаются равными нулю.

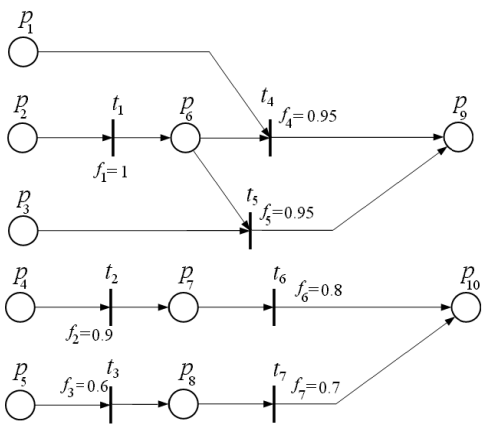


Рис. 10.11. Структура нечеткой сети Петри для представления базы правил нечетких продукций примера 10.4

Для удобства визуализации этой НСП принята следующая интерпретация позиций, каждая из которых соответствует отдельному нечеткому высказыванию:

- p_1 — "В отношении гражданина имеется агентурная информация";
- p_2 — "Гражданин не является высокопоставленным чиновником";
- p_3 — "Контролер использует новейшие технические средства";
- p_4 — "Гражданин является высокопоставленным чиновником";
- p_5 — "Количество граждан, проходящих таможенный досмотр, велико"
- p_6 — "Гражданин подвергается таможенному досмотру";

- p_7 — "Гражданин не подвергается таможенному досмотру";
- p_8 — "Контролер испытывает чувство усталости";
- p_9 — "Исключается возможность провоза наркотиков";
- p_{10} — "Не исключается возможность провоза наркотиков".

При этом каждый из переходов соответствует отдельному правилу нечеткой продукции: t_1 — правилу 1, t_2 — правилу 2, t_3 — правилу 4, t_4 — правилу 6, t_5 — правилу 7, t_6 — правилу 3, t_7 — правилу 5.

Как и ранее, предположим, что на таможенном пункте контроля сложилась следующая текущая ситуация. Среди граждан, въезжающих в страну, находятся высокопоставленные чиновники (T=0.2). Количество граждан, проходящих таможенный досмотр, невелико (T=0.1). Таможенный пункт контроля оснащен новейшими техническими средствами (T=0.8). Какая-либо предварительная информация о наличии наркотиков у отдельных граждан отсутствует (T=0.9). Здесь в скобках указаны степени истинности соответствующих нечетких высказываний. Тем самым задается вектор начальной маркировки: $\boldsymbol{m}_0$=(0.1, 0.8, 0.8, 0.2, 0.9, 0, 0, 0, 0, 0). Этой ситуации соответствует НСП, изображенная на рис. 10.12.

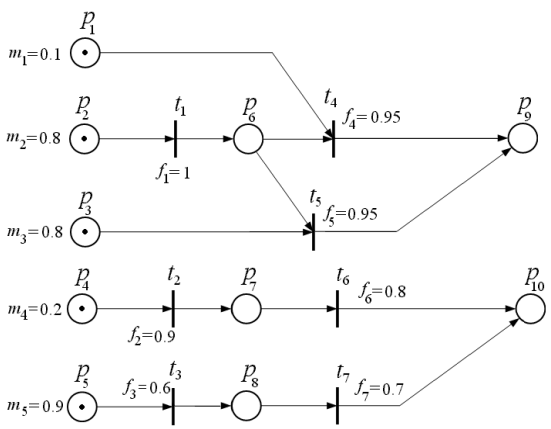


Рис. 10.12. Нечеткая сеть Петри, представляющая рассматриваемую текущую ситуацию на таможенном пункте контроля

Формально задача заключается в том, чтобы, используя построенную НСП и модифицированные Правила ее функционирования $\mathbb{P}(C_f')$, оценить наличие одного маркера в позициях p_9 и p_{10}.

С этой целью последовательно рассмотрим все активные переходы и результаты их нечеткого срабатывания. На первом шаге решения этой задачи следует определить активные переходы в НСП.

Срабатывание активного при начальной маркировке перехода t_1 приводит к изменению маркировки в позиции p_6, которая становится равной m_6=0.8. Сраба-

тывание активного при начальной маркировке перехода t_2 приводит к изменению маркировки в позиции p_7, которая становится равной m_7=0.2. Срабатывание активного при начальной маркировке перехода t_3 приводит к изменению маркировки в позиции p_8, которая становится равной m_8=0.6.

Новые значения маркеров в этих позициях приводят к активности переходов t_4-t_7. При этом срабатывание активного перехода t_4 приводит к изменению маркировки в позиции p_9, которая становится равной m_9=0.1. Срабатывание активного перехода t_5 приводит к новому изменению маркировки в позиции p_9, которая становится равной m_9=0.8.

Срабатывание активного перехода t_6 приводит к изменению маркировки в позиции p_{10}, которая становится равной m_{10}=0.2. Наконец, срабатывание активного перехода t_6 приводит к новому изменению маркировки в позиции p_{10}, которая становится равной m_{10}=0.6. Поскольку дальнейшие срабатывания активных переходов не изменяют маркировку: $\boldsymbol{m}$=(0.1, 0.8, 0.8, 0.2, 0.9, 0.8, 0.2, 0.6, 0.8, 0.6), то на этом процесс анализа функционирования данной НСП можно закончить.

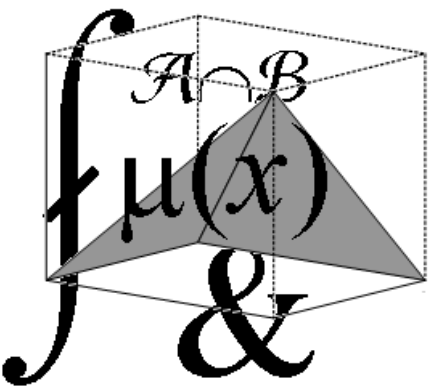
Таким образом, для рассматриваемой ситуации степень истинности нечеткого высказывания "исключается возможность провоза наркотиков" равна значению m_9=0.8, а степень истинности нечеткого высказывания "не исключается возможность провоза наркотиков" равна значению m_{10}= 0.6.

Анализ этих значений истинности показывает, что они в точности совпадают со значениями степеней истинности соответствующих высказываний, полученных при решении задачи методом прямого вывода заключений в примере 6.5. Несомненным достоинством представления базы правил в форме НСП и последующее решение задач вывода заключений на их основе является наглядность и визуализация всех промежуточных результатов.

Примечание

Можно предложить в качестве упражнения получить нечеткие заключения для НСП с той же структурой, но с использованием измененных Правил функционирования НСП, отражающих возможность применения других методов нечеткой композиции (6.23)—(6.28), а полученные результаты сравнить.

В заключение следует отметить, что решение практических задач нечеткого вывода в системах нечетких продукций характеризуется высокой трудоемкостью выполнения численных расчетов. Поэтому в таких случаях не вызывает сомнения необходимость использования для этой цели специальных программных инструментариев и программных средств, позволяющих существенно упростить создание и анализ соответствующих нечетких моделей. Далее, в *частях II* и *III*, детально рассматриваются особенности разработки и выполнения исследований с нечеткими моделями в двух программных средах — MATLAB и fuzzyTECH, которые обладают всеми необходимыми возможностями реализации систем нечеткого вывода и визуализации результатов нечеткого моделирования.



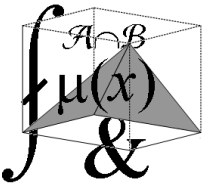
ЧАСТЬ II

НЕЧЕТКОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ MATLAB

Система MATLAB, разработанная и постоянно обновляемая компанией MathWorks Inc. (США), является одной из наиболее известных систем компьютерной математики. К последним принято относить специализированные компьютерные программы, которые предназначены для решения широкого класса задач, связанных с тем или иным разделом теоретической или прикладной математики. При этом отдельные классы задач, которые позволяет решать система MATLAB, имеют весьма условное отношение к классической математике, поскольку в настоящее время представляют собой узко специализированные области научных и прикладных исследований.

Использование системы MATLAB и связанных с ней методик моделирования и процедур выполнения численных расчетов стало стандартом *de facto* для широчайшего круга специалистов из самых различных областей науки, техники, экономики и образования. Содержащая специальные средства нечеткого моделирования, система MATLAB позволяет выполнять весь комплекс исследований по разработке и применению нечетких моделей. Именно по этим причинам система MATLAB была выбрана в качестве одного из программных средств, в рамках которых можно реализовать рассмотренные ранее теоретические концепции нечетких множеств и процедуры нечеткого вывода.

Глава 11



Общая характеристика программы MATLAB

Система MATLAB (сокращение от англ. *MATrix LABoratory* — матричная лаборатория) представляет собой интегрированную программную среду для выполнения численных расчетов, компьютерного моделирования и вычислительных экспериментов, охватывающих в том или ином объеме различные области классической и современной математики, а также широчайший спектр инженерных приложений.

Архитектурно система MATLAB состоит из базовой программы и нескольких десятков так называемых пакетов расширения, которые в своей совокупности обеспечивают исключительно широкий диапазон решаемых задач. Интеграция всех этих средств в единой рабочей среде обеспечивает необходимую гибкость использования сотен встроенных функций, реализующих разнообразные математические процедуры и вычислительные алгоритмы.

Нечеткое моделирование в среде MATLAB осуществляется с использованием пакета расширения Fuzzy Logic Toolbox, в котором реализованы десятки функций нечеткой логики и нечеткого вывода. Перечень этих функций с их краткой характеристикой и примерами использования приводится в *приложении 3*.

Примечание

В настоящей книге рассматривается система MATLAB версии 6.1 (Release 12.1), которая является последней на момент написания книги. Практически вся информация о нечетком моделировании с использованием пакета Fuzzy Logic Toolbox оказывается справедливой для предыдущих версий системы MATLAB 6.0 (Release 12) и 5.3 (Release 11).

11.1. Основные элементы системы MATLAB

Прежде всего, необходимо сказать несколько слов о системных требованиях к компьютеру. Компьютер, на который предполагается установить систему MATLAB 6.1, должен удовлетворять следующим системным требованиям.

- Процессор Pentium, Pentium Pro, Pentium II, Pentium III, Pentium IV или AMD Athlon.

- Операционная система MS Windows 95/98/ME, MS Windows 4.0 (SP 5/6a) MS Windows 2000 (для ОС UNIX/Linux имеется специальный дистрибутив, который здесь не рассматривается).
- Оперативная память (RAM) минимум 64 Мбайт, рекомендуется 128 Мбайт.
- Дисковод компакт-дисков CD-ROM (необходим для инсталляции).
- Свободное место на жестком диске, необходимое для установки системы MATLAB 6.1, определяется количеством устанавливаемых пакетов расширений и документацией. Вариант установки всех пакетов и полной документации к ним может потребовать до 1.5 Гбайт. В процессе инсталляции при выборе пакетов расширений пользователю выдается информация о необходимом свободном месте на жестком диске.
- 8-битовый графический адаптер и монитор, поддерживающий не менее 256 цветов.
- Один из браузеров Интернета для просмотра документации в формате HTML и доступа к сайту компании MathWorks по адресу: **www.mathworks.com**.
- Программа Adobe Acrobat Reader для просмотра документации в формате PDF.

При установке отдельных пакетов расширений могут потребоваться следующие дополнительные средства.

- Графический адаптер с ускорителем и звуковая карта.
- Текстовый процессор MS Word из MS Office 95/97/2000 для работы с пакетом расширения MATLAB Notebook.
- Поддержка протокола TCP/IP для работы в сети.

Для программирования в среде MATLAB и разработки собственных MEX-файлов могут оказаться необходимыми следующие среды разработки программ:

- Compaq Visual Fortran 5.0/6.1
- MS Visual C++ 5.0/6.0
- Borland C/C++ 5.0/5.02
- Borland C++ Builder 3.0/4.0/5.0
- Lcc 2.4 (поставляется с MATLAB)

Особенности инсталляции системы MATLAB на компьютер пользователя

Для инсталляции системы MATLAB 6.1 необходимо использовать 2 компакт-диска, первый из которых содержит дистрибутив с программами и пакетами расширения, а второй — обширную документацию по всем разделам системы MATLAB в форматах HTML и PDF.

Процесс инсталляции системы MATLAB достаточно традиционен для ОС MS Windows 9*x*/2000 и включает в себя следующие этапы:

1. Установить компакт-диск с дистрибутивом программы в привод CD-ROM и выполнить щелчок на кнопке с предложением начать инсталляцию.
2. Ввести пароль персональной лицензии (PLP, Personal License Password).
3. Ознакомиться с Соглашением о лицензировании данного программного обеспечения.
4. Ввести свое имя и название компании.
5. Выбрать устанавливаемые компоненты, пакеты расширения и состав документации из числа доступных для данной лицензии.

После выполнения последнего этапа начинается собственно установка системы MATLAB и документации на компьютер пользователя. При установке документации необходимо вставить второй компакт-диск в привод CD-ROM, что будет предложено сделать пользователю после окончания инсталляции исполняемых модулей программы. Установка всей системы MATLAB и документации осуществляется в отдельную папку (каталог) на жестком диске (по умолчанию — C:\MATLAB6p1).

Примечание

Имеются сообщения о том, что версии MATLAB 6.0 и ниже не работают на ПК с процессором Intel Pentium IV. Автор склонен не согласиться с этими мнениями, поскольку все описанные в книге операции и действия проверялись, в том числе, для системы MATLAB версии 5.3, установленной на ПК с процессором Intel Pentium IV. Возможно эта проблема может быть связана с установкой на один ПК нескольких версий MATLAB. В последнем случае для корректной работы каждой из них необходимо использовать отдельный файл autoexec.bat, в котором должны быть прописаны пути запускаемой версии MATLAB (либо вручную отредактировать этот файл до перезагрузки ОС MS Windows).

Запуск системы MATLAB и элементы ее графического интерфейса

После инсталляции в главном меню ОС MS Windows 9*x*/2000 появляется новая программная группа с именем MATLAB 6.1. Для запуска системы MATLAB 6.1 необходимо в этой программной группе выбрать название программы "MATLAB 6.1", в результате чего на экране монитора появится рабочий интерфейс программы MATLAB 6.1 (рис. 11.1). При этом каждый отдельный запуск системы MATLAB называется *сеансом* или сессией работы с ней.

В верхней части окна MATLAB расположена строка заголовка, в которой указывается название системы. В правой части строки заголовка находятся стандартные кнопки управления графическим окном программы.

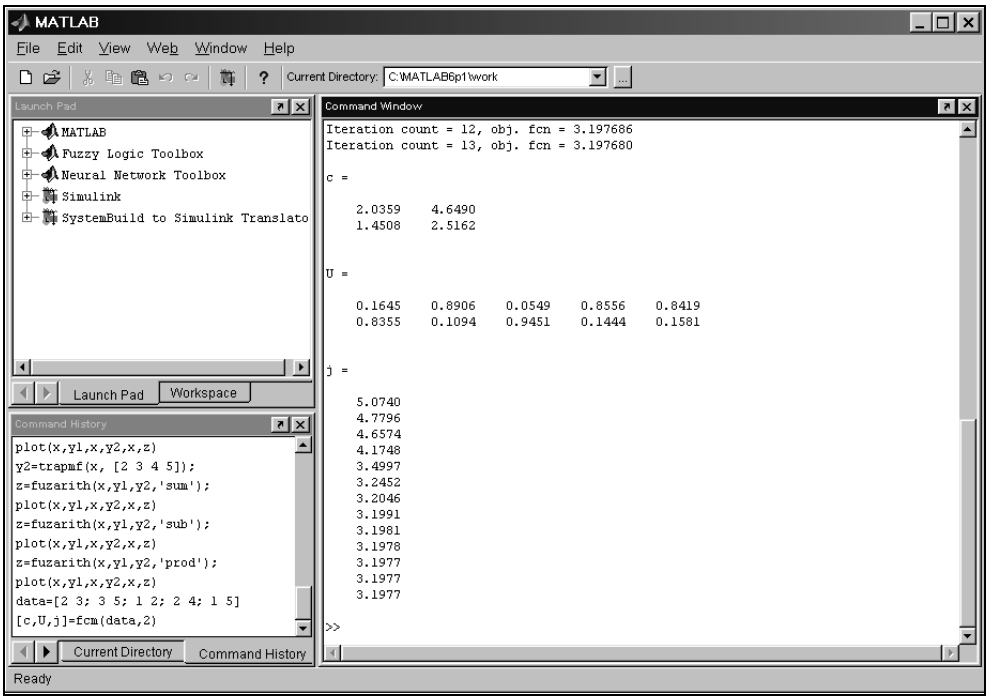


Рис. 11.1. Общий вид графического интерфейса системы MATLAB 6.1

Под строкой заголовка находится строка главного меню системы MATLAB и строка панели инструментов (рис. 11.2). Выбор той или иной операции меню выполняется стандартным для всех приложений MS Windows образом. Панель инструментов содержит кнопки, позволяющие осуществить доступ к наиболее часто используемым командам и операциям.

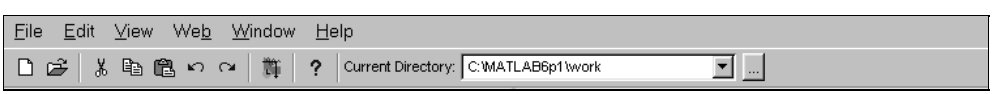


Рис. 11.2. Главное меню и панель инструментов системы MATLAB 6.1

Основную часть окна системы MATLAB занимает окно команд (рис. 11.3), которое предназначено для взаимодействия с системой в режиме командной строки. В последних версиях MATLAB 6.0/6.1 режим командной строки (режим команд) продолжает оставаться основным для многих практических задач. Однако следует признать, что с каждой новой версией увеличивается количество специальных графических средств, позволяющих реализовать интерактивное взаимодействие пользователя с программой в более удобной и наглядной форме.

Окно команд можно "открепить" от остальных окон программы, нажав соответствующую кнопку справа вверху.

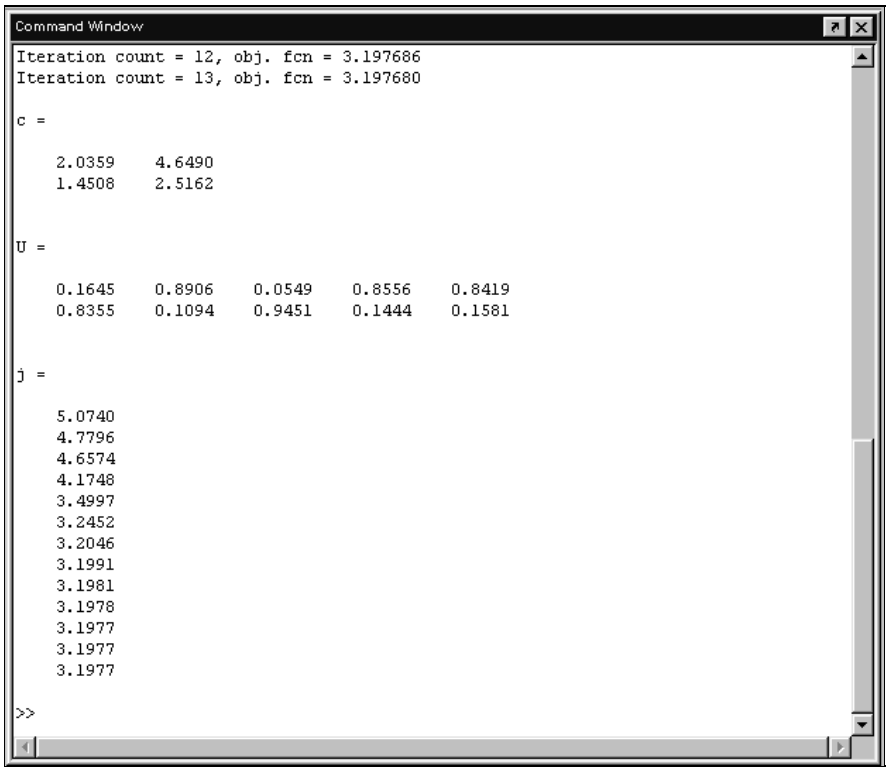


Рис. 11.3. Окно команд системы MATLAB 6.1

Окно команд используется для ввода команд и функций с необходимыми аргументами, задания значений переменным и отображения результатов выполненных расчетов. По своему назначению окно команд аналогично режиму командной строки операционных систем UNIX/Linux, поскольку обладает возможностями прямого вызова огромного числа функций MATLAB. Этот факт отражает историю появления первых версий системы MATLAB, в которых окно команд было единственным средством взаимодействия с системой.

С появлением новых версий MATLAB были разработаны специальные графические интерфейсы пользователя (GUI — Graphic User Interface), ориентированные на решение отдельных классов задач. Не является исключением и пакет Fuzzy Logic Toolbox, который сейчас содержит около 10 самостоятельных GUI. Тем не менее, окно команд продолжает играть ключевую роль при работе с системой MATLAB, поскольку обеспечивает полный контроль над процессом решения тех или иных задач. Более того, все простейшие операции по заданию исходных зна-

чений расчетных переменных и выполнению большого числа специальных математических функций по-прежнему осуществляются в окне команд.

Для гибкого и полного использования возможностей системы MATLAB следует знать функциональные возможности окна команд. Говоря о процессе работы в данном окне, будем называть этот режим — *режимом командной строки* или сокращенно — *режимом команд*. Напротив, использование специальных графических интерфейсов GUI для решения отдельных классов задач будем называть *графическим* или *интерактивным режимом* работы.

Продолжая знакомство с рабочим интерфейсом системы MATLAB, отметим две вкладки, расположенные в левой части главного окна системы MATLAB. Одна из них представляет собой панель доступа к компонентам системы (Launch Pad), а другая — окно просмотра рабочей области системы (Workspace).

Панель доступа к компонентам системы (рис. 11.4) организована в стиле обычного Проводника и предназначена для быстрого вызова справки по выбранному разделу MATLAB, демонстрационных примеров, специальных графических средств и получения дополнительной информации из Интернета. Выбор необходимого действия осуществляется после нажатия правой кнопки мыши, установленной на том или ином компоненте системы. При этом можно также выбрать режим редактирования исходных файлов компонентов или режим их обновления через Интернет.

Рис. 11.4. Окно доступа к компонентам системы MATLAB

Окно просмотра рабочей области системы MATLAB (рис. 11.5) позволяет визуализировать содержимое рабочей области и, что не менее важно, редактировать значения отдельных переменных (элементов массивов и структур). При этом следует помнить о важнейшей особенности системы MATLAB — все объявленные и заданные переменные система рассматривает как матрицы или

массивы. Поэтому в окне просмотра рабочей области указывается не только имя переменной, но и ее размерность как матрицы, занимаемый ею размер в байтах и тип этой переменной.

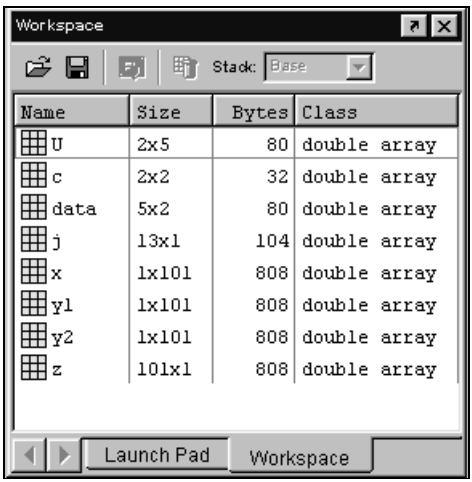


Рис. 11.5. Окно просмотра рабочей области системы MATLAB

Для редактирования значений переменных необходимо дважды щелкнуть на строке с выделенной переменной, в результате чего будет открыто диалоговое окно со значениями этой переменной в отдельных ячейках. Окно просмотра рабочей области позволяет также выполнять целый ряд других операций: удалить выбранную переменную из рабочей области, очистить всю рабочую область, загрузить переменные в рабочую область из внешнего файла или сохранить значения переменных рабочей области во внешнем файле. Все эти операции становятся доступными в контекстном меню, которое может быть вызвано нажатием правой кнопки мыши, установленной на выделенной переменной в этом окне.

В правой нижней части главного окна системы MATLAB расположены две вкладки, одна из которых содержит окно истории команд (Command History), а другая — окно текущего каталога или папки системы (Current Directory).

Окно истории команд (рис. 11.6) позволяет визуализировать все команды, которые были введены пользователем не только в текущем сеансе работы с системой, но и в предшествующих сеансах. Если необходимо выполнить одну из ранее введенных команд, то для этого достаточно дважды щелкнуть на ее имени в активном окне истории команд. После этого выбранная команда последует на исполнение.

В случае необходимости предварительно отредактировать одну из введенных ранее команд можно скопировать в буфер выбранную в окне истории команду и потом вставить ее в окно команд, используя для этой цели правую кнопку мыши. Аналогичный результат может быть достигнут и простым "перетаскиванием" с помощью мыши выбранной команды в окно команд.

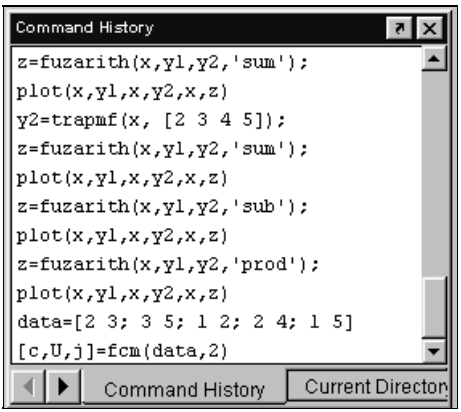


Рис. 11.6. Окно истории команд системы MATLAB

Заметим, что просмотреть ранее выполненные команды можно находясь в окне команд. Для этого достаточно использовать клавиши клавиатуры со стрелками <↑> и <↓>.

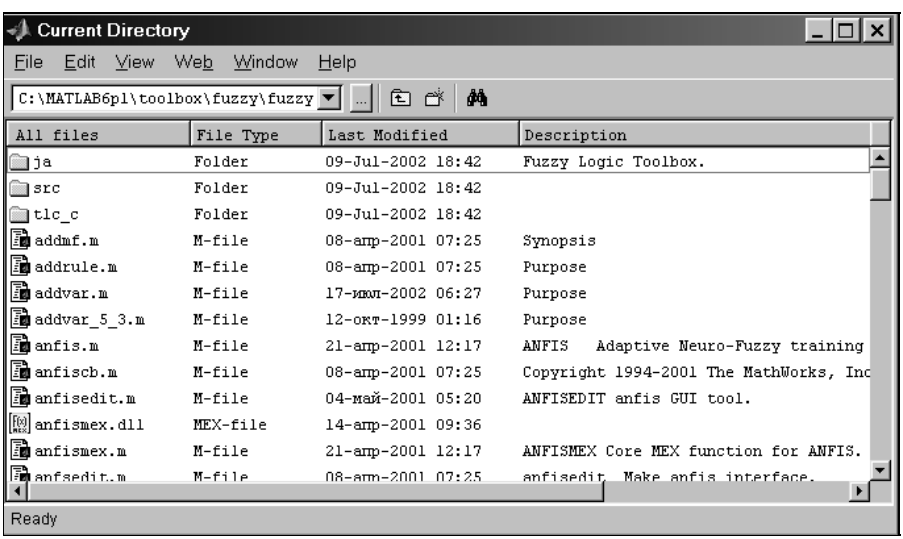


Рис. 11.7. Окно текущего каталога системы MATLAB

Окно текущего каталога (рис. 11.7) отображает файлы выбранного каталога системы MATLAB, который при необходимости можно изменить. При этом следует помнить, что файлы с расширением m, dat, fis являются обычными текстовыми файлами, которые можно просматривать и редактировать в любом ASCII-

редакторе. Однако более удобным для этой цели является встроенный редактор системы MATLAB, так называемый редактор-отладчик m-файлов. Этот редактор открывается после двойного щелчка на имени любого m-файла, при этом в него оказывается загруженным выбранный файл. Более подробно работа с редактором m-файлов рассматривается в *главе 14*.

В состав системы MATLAB входит встроенная справочная система и обширная документация в форматах HTML и PDF, которые содержат необходимую информацию по системе в целом, ее отдельным функциям и многочисленным пакетам расширения.

Встроенная справочная система и документация, поставляемая с системой MATLAB

Для просмотра документации по системе MATLAB (в случае ее инсталляции) необходимо нажать кнопку **?** панели инструментов или выбрать команду меню: **Help>MATLAB Help**. В результате выполнения этой команды на экране появится дополнительное окно с программой просмотра справочной системы MATLAB (рис. 11.8).

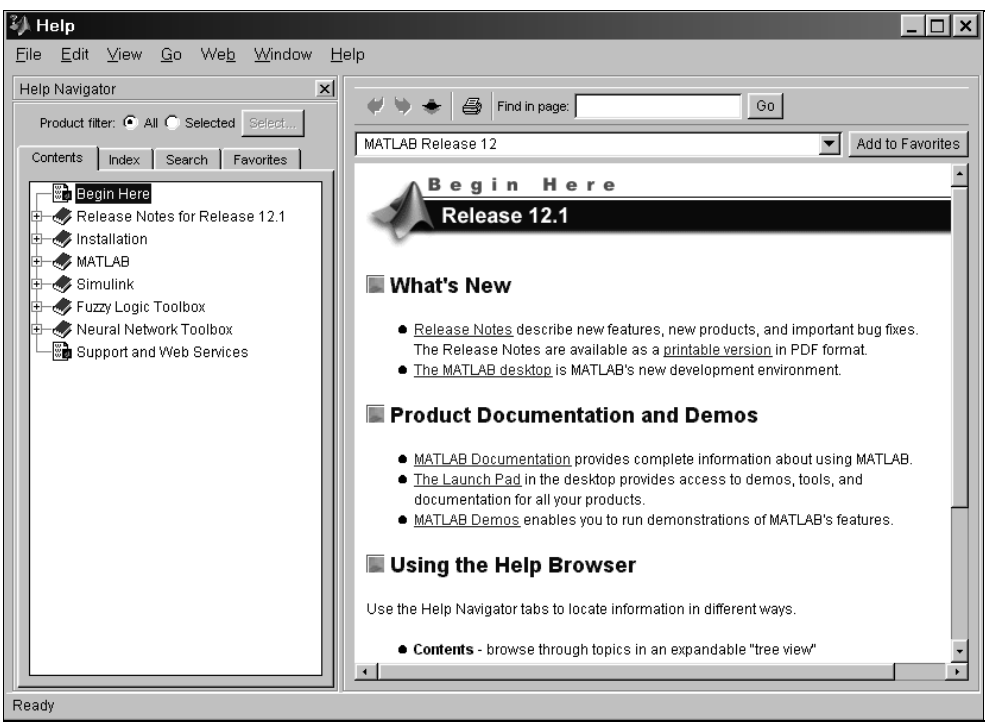


Рис. 11.8. Окно справочной системы MATLAB

В левой части этого окна содержится панель навигации с несколькими вкладками, которые позволяют просматривать различные разделы документации, осуществлять поиск информации по ключевым словам и запоминать ранее просмотренные разделы. Следует отметить, что информация здесь организована по иерархическому принципу.

В правой части окна справочной системы MATLAB расположено основное окно просмотра документации, организованное по технологии гипертекста. Соответствующие ссылки на другие разделы выделены традиционным образом: синим цветом и подчеркнуты. При необходимости здесь можно выбрать вариант документации в формате PDF или распечатать интересуемый раздел справки на принтере. История просмотра документации сохраняется, для ее анализа можно воспользоваться соответствующими кнопками.

Весьма удобной является вкладка **Index**, которая позволяет осуществлять поиск информации по ключевым словам (рис. 11.9). Для этого следует ввести ключевое слово в поле ввода **Search index for**. В процессе ввода символов ключевого слова система MATLAB переходит к справке по первому подходящему слову, которое следует в алфавитном порядке после введенных символов.

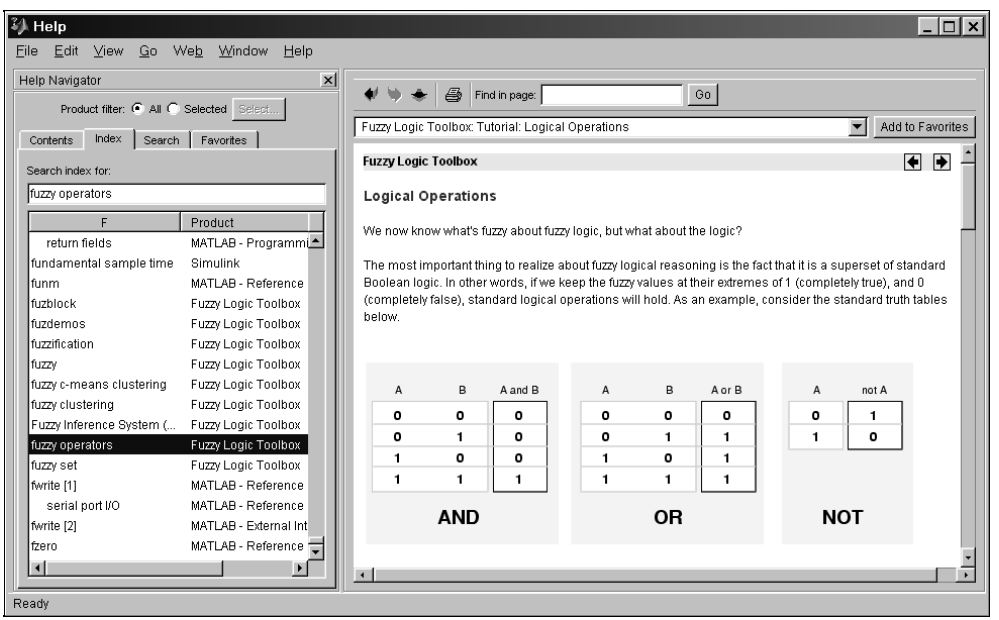


Рис. 11.9. Окно справочной системы MATLAB с открытой вкладкой поиска по ключевым словам

В дополнение к документации практически все функции системы MATLAB оснащены собственной справочной информацией. Если необходимо получить такую справочную информацию по отдельной функции, имя которой известно, то

в окне команд достаточно набрать слово "help" с именем этой функции. После выполнения этой команды в окне команд отобразится справочная информация по выбранной функции (рис. 11.10).

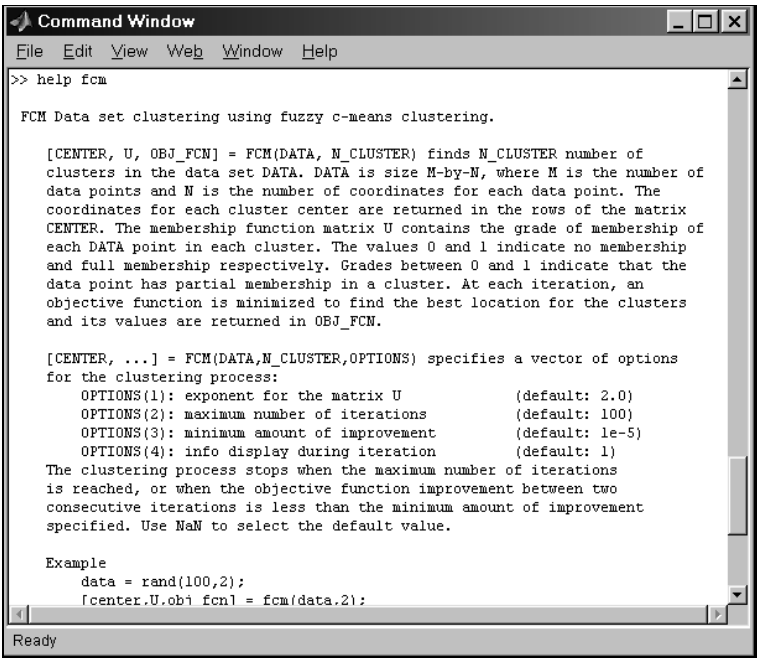


Рис. 11.10. Окно команд со справочной информацией по выбранной функции `fcm`

В окне команд также доступна справочная информация по функциям целого раздела или пакета расширения системы MATLAB. Для этого следует после имени команды `help` набрать имя интересующего раздела или пакета расширения, установленного в системе MATLAB. После выполнения этой команды в окне команд отобразится краткая информация о составе и назначении всех функций выбранного раздела.

11.2. Основные приемы работы в системе MATLAB

Прежде всего рассмотрим операции главного меню системы MATLAB и назначение отдельных кнопок панели инструментов. После этого остановимся на основных командах, которые могут оказаться необходимыми в процессе работы в режиме командной строки.

Назначение операций главного меню

Окно системы MATLAB имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой MATLAB, загружать и сохранять информацию во внешних файлах, изменять внешний вид элементов графического интерфейса, вызывать справочную информацию и т. д. Рассмотрим назначение отдельных пунктов главного меню системы MATLAB.

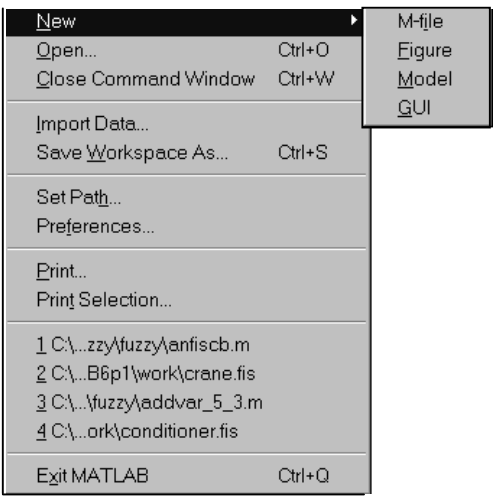


Рис. 11.11. Операции пункта меню **File** главного меню

❒ Пункт меню **File** (Файл) главного меню содержит следующие операции (рис. 11.11):

- **New** — позволяет выбрать тип нового объекта системы MATLAB и содержит дополнительное вложенное подменю с подпунктами: **M-file** — вызов редактора-отладчика m-файлов; **Figure** — открытие пустого окна графиков функций; **Model** — открытие пустого окна для создания новой Simulink-модели; **GUI** — вызов редактора для разработки элементов графического интерфеса пользователя. При этом новые объекты по умолчанию имеют имя Untitled;
- **Open...** — вызывает стандартное диалоговое окно открытия внешнего файла с диска. Если выбирается некоторый текстовый файл (например, m-файл), то выбранный файл загружается в редактор-отладчик m-файлов. Если выбирается файл Simulink-модели (файл с расширением mdl), то выбранный файл загружается в окно системы моделирования Simulink;
- **Close Command Window** — закрывает окно команд системы MATLAB. После выполнения этой операции вместо нее появляется операция с предложением закрыть следующее окно доступа к компонентам системы: **Close Launch Pad** и т. д.;

- **Import Data…** — позволяет импортировать информацию из внешних файлов различных форматов, включая графические, звуковые и мультимедиа. При этом вызывается мастер импорта данных, выполняющий предварительный просмотр изображений;
- **Save Workspace As…** — позволяет сохранить рабочую область системы MATLAB во внешнем файле с расширением mat на диске. При этом вызывается стандартное диалоговое окно сохранения файла на диске с предложением задать имя соответствующего файла;
- **Set Path…** — вызывает окно задания путей доступа к файлам системы MATLAB. Как правило, значения путей, установленные по умолчанию, следует изменять только в случае крайней необходимости;
- **Preferences…** — вызывает окно настройки системы MATLAB, которое позволяет изменить шрифт и цвет отображения данных в различных окнах, а также изменить некоторые другие характеристики элементов рабочего интерфейса;
- **Print…** — позволяет распечатать на принтере информацию о текущем документе системы MATLAB. В этом случае вызывается стандартное диалоговое окно настройки свойств печати на подключенном к данному компьютеру принтеру;
- **Print Selection…** — позволяет распечатать на принтере информацию о выделенной части текущего документа системы MATLAB. При этом также вызывается стандартное диалоговое окно настройки свойств печати на подключенном к данному компьютеру принтеру;
- секция с именами последних файлов, с которыми осуществлялась работа в системе MATLAB;
- **Exit MATLAB** — закрывает систему MATLAB, при этом содержимое рабочей области оказывается не сохраненным, если предварительно не была выполнена соответствующая операция.

❒ Пункт меню **Edit** (Редактирование) содержит следующие операции (рис. 11.12):

- **Undo** — отменяет выполнение последнего действия;
- **Redo** — отменяет выполнение последней операции **Undo**;
- **Cut** — вырезает выделенный фрагмент из текущей строки окна команд и помещает его в буфер обмена;
- **Copy** — копирует выделенный фрагмент из окна команд и помещает его в буфер обмена;
- **Paste** — вставляет фрагмент в текущую строку окна команд из буфера обмена;
- **Paste Special...** — вызывает мастер импорта данных, который позволяет выполнить предварительный просмотр информации, хранящейся в буфере обмена;

- **Select All** — позволяет выделить всю информацию окна команд текущего сеанса работы;
- **Delete** — удаляет выделенный объект (бывает активной весьма редко);
- **Clear Command Window** — очищает окно команд от информации текущего сеанса работы;
- **Clear Command History** — очищает окно истории команд от информации о введенных ранее командах;
- **Clear Workspace** — очищает рабочее пространство системы MATLAB от всей имеющейся в ней информации о переменных.

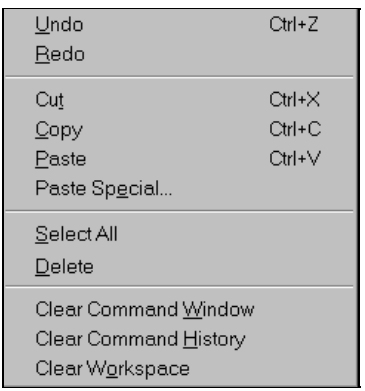


Рис. 11.12. Операции пункта меню **Edit** главного меню

❒ Пункт меню **View** (Вид) позволяет отображать на экране различные компоненты системы и содержит следующие операции (рис. 11.13):

- **Desktop Layout** — позволяет настроить внешний вид графического интерфейса системы MATLAB и содержит дополнительные подпункты: **Default** — располагает все окна системы MATLAB по умолчанию (рис. 11.1); **Command Window Only** — устанавливает интерфейс для системы MATLAB, состоящий только из окна команд; **Simple** — устанавливает упрощенный интерфейс для системы MATLAB, состоящий из окна команд и окна истории команд; **Short History** — устанавливает узкое окно истории команд; **Tall History** — устанавливает широкое окно истории команд; **Five Panel** — делает видимыми на экране все 5 основных окон системы MATLAB;
- **Undock Command Window** — позволяет открепить и сделать "плавающим" окно команд системы MATLAB;
- **Command Window** — делает видимым/невидимым окно команд;
- **Command History** — делает видимым/невидимым окно истории команд;

- **Current Directory** — делает видимым/невидимым окно текущего каталога;
- **Workspace** — делает видимым/невидимым окно просмотра рабочей области;
- **Launch Pad** — делает видимым/невидимым окно доступа к компонентам системы MATLAB;
- **Help** — вызывает браузер справочной системы MATLAB;
- **Current Directory Filter** — позволяет установить фильтры для отображения файлов текущего каталога;
- **Workspace View Options** — позволяет отображать отдельные свойства переменных рабочей области и сортировать их по различным параметрам.

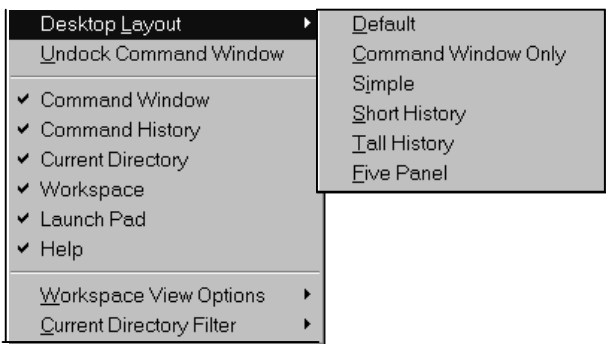


Рис. 11.13. Операции пункта меню **View** главного меню

- Пункт меню **Web** (Интернет) вызывает установленный в операционной системе по умолчанию браузер Интернета и делает попытку соединиться с Web-сайтом компании MathWorks (в случае наличия доступа в Интернет). Выбор отдельной операции этого пункта меню определяет загрузку той или иной Web-страницы компании, предназначенной для выполнения специальных действий по дополнительной поддержке системы MATLAB или загрузке имеющихся обновлений.
- Пункт меню **Window** (Окно) содержит операцию **Close All**, которая позволяет закрыть все дополнительные окна с графиками, графическими редакторами и другими компонентами системы MATLAB, открытыми в текущем сеансе работы. Если дополнительные окна отсутствуют, то эта операция является недоступной. Если открыты дополнительные окна, то в этом пункте меню появляется новая строка с именем каждого из дополнительных окон, выбрав которую, можно сразу перейти в нужное окно.
- Пункт меню **Help** (Справка) содержит следующие операции (рис. 11.14):
  - **Full Product Family Help** — вызывает браузер (программу просмотра) справочной системы, установленный на начало знакомства с MATLAB (см. рис. 11.8);

- **MATLAB Help** — вызывает браузер справочной системы, установленный на раздел общей справки о MATLAB;
- **Using the Desktop** — вызывает браузер справочной системы, установленный на раздел справки об элементах и окнах рабочего интерфейса системы MATLAB;
- **Using the Command Window** — вызывает браузер справочной системы, установленный на раздел справки об окне команд системы MATLAB;
- **Demos** — вызывает окно с демонстрационными примерами, поставляемыми с системой MATLAB;
- **About MATLAB** — отображает информацию о текущей рабочей версии системы MATLAB.

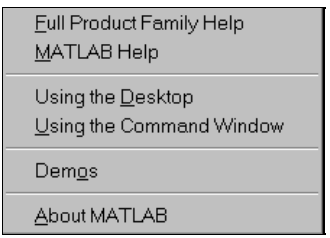


Рис. 11.14. Операции пункта меню **Help** главного меню

Назначение операций панели инструментов

Панель инструментов содержит набор кнопок, которые дублируют наиболее часто выполняемые операции главного меню. Рассмотрим назначение отдельных кнопок панели инструментов (табл. 11.1).

Таблица 11.1. *Назначение кнопок панели инструментов*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	New M-File	Вызывает редактор-отладчик m-файлов с пустым окном редактирования и именем файла Untitled, заданным по умолчанию
	Open File	Вызывает стандартное диалоговое окно открытия внешнего файла с диска. Если выбирается некоторый текстовый файл (например, m-файл), то выбранный файл загружается в редактор-отладчик m-файлов. Если выбирается файл Simulink-модели (файл с расширением mdl), то выбранный файл загружается в окно системы моделирования Simulink

Таблица 11.1 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Cut	Вырезает выделенный фрагмент из текущей строки окна команд и помещает его в буфер обмена
	Copy	Копирует выделенный фрагмент из окна команд и помещает его в буфер обмена
	Paste	Вставляет фрагмент в текущую строку окна команд из буфера обмена
	Undo	Отменяет выполнение последнего действия
	Redo	Отменяет выполнение последней операции **Undo**
	Simulink	Вызывает окно библиотеки системы моделирования Simulink
	Help	Вызывает окно справочной системы MATLAB
	Current Directory	Позволяет установить текущий каталог (папку), содержимое которого будет отображаться в диалоговых окнах открытия и сохранения файлов на диске

Основные приемы работы в окне команд

Как уже отмечалось ранее, окно команд (см. рис. 11.3) используется для ввода команд и функций с необходимыми аргументами, задания значений отдельным переменным и отображения результатов выполненных расчетов.

При этом используются следующие основные правила работы. Ввод команды осуществляется набором соответствующих символов команды в отдельной строке активного окна команд после символов приглашения >>. Запуск команды на исполнение после ее набора осуществляется нажатием клавиши <Enter>. Аналогично набираются и отдельные значения переменных, при этом после имени переменной необходимо указать знак равенства =, который в системе MATLAB играет роль знака присваивания.

Следует отметить, что символы верхнего и нижнего регистров воспринимаются системой MATLAB как различные. При вводе произвольной комбинации символов система MATLAB вначале пытается сопоставить эти символы с именем одной из переменных рабочей области. Если переменной с подобным именем в рабочем пространстве нет, то система MATLAB сопоставляет символы с именем одной из встроенных функций. Если и функция с таким именем отсутствует, то в окно команд будет выведено сообщение об ошибке.

Примечание

Окно команд системы MATLAB воспринимает ввод символов с клавиатуры в основной кодировке ASCII (англоязычный регистр или латиница). Что же касается использования символов дополнительной кодировки (кириллицы), то при попытке ввести русскоязычные символы в качестве строк или комментариев могут возникнуть проблемы, зависящие от версии MATLAB и операционной системы. В частности, версия MATLAB 6.1, установленная в среде MS Windows 98, не воспринимает строчную букву "я", а при вводе в окно команд строчной буквы "с" происходит самопроизвольный переход к следующей строке без выполнения набранного фрагмента и без его сохранения. В то же время в версии MATLAB 5.3 эти проблемы отсутствуют. Наконец, на диаграммах и графиках может неверно отображаться символ подчеркивания "_" и некоторые другие символы, набранные в русскоязычном регистре. Во всех подобных случаях следует помнить, что система MATLAB не является локализованной программой, и поэтому было бы странным требовать от нее корректного восприятия кириллицы.

Если имя команды или значения слишком длинное, то следует использовать символ продолжения ввода — `...` *(три точки)*. В этом случае все набранные в новой строке символы считаются продолжением символов предыдущей строки. Если после набора имени команды или задания значений переменной поставлен символ `;` *(точка с запятой)*, то после исполнения данной команды результат ее выполнения в окне команд не отображается. В противном случае (завершение набора команды без символа `;`) происходит отображение результата ее выполнения, что может перегружать окно команд излишней информацией.

Если результат выполнения отдельной команды имеет некоторое значение, которое не назначено никакой переменной, то в системе для этой цели предусмотрена переменная по умолчанию — переменная с именем `ans`. Переменная с этим именем может появиться и в рабочей области, важно помнить, что ее значение может изменяться по мере выполнения отдельных команд.

При задании значений переменным в системе MATLAB задавать тип самих переменных нет необходимости, т. к. тип каждой переменной определяется самим задаваемым значением. Так, например, если справа от имени переменной записано целое число без разделительной точки, то тип этой переменной будет целочисленный. Более того, как уже отмечалось ранее, система MATLAB каждый объект рассматривает как матрицу. При этом обычное число или символ считается матрицей размерности 1×1, а массив из нескольких элементов — матрицей размерности $1\times n$. В этом заключается одна из ключевых особенностей системы, которая обеспечивает высокую производительность вычислений и удобство работы пользователя.

Поскольку во всех численных расчетах система MATLAB оперирует матрицами, для задания значений их элементов используются специальные символы: `[` — для начала и `]` — для окончания отдельной матрицы. При этом матрицы записываются по строкам в возрастающем порядке. Каждый элемент матрицы отделяется от другого элемента пробелом, строки отделяются друг от друга символом `;`. После задания значений переменных (матриц) все они запоминаются и хранятся

в так называемой *рабочей области* (workspace) системы MATLAB. При задании матриц размерности 1×1 (обычных скаляров) знаки начала и окончания матрицы можно не указывать.

Например, если необходимо задать переменной с именем `a` некоторое количественное значение, скажем 10, то, находясь в окне команд (мигающий курсор после знака приглашения), следует ввести следующую строку:

```
>> x=10
```

После нажатия клавиши <Enter> в окне команд появится следующая информация:

```
x =
    10
```

Аналогично можно задать значение строки некоторой переменной с именем `b`:

```
>> b='String'
b =
       String
```

При этом сама строка текста заключается в апострофы. Чтобы задать матрицу, например, матрицу $\mathbf{M}_P$ нечеткого отношения из примера 4.3 (напомним, что она равна

$$\begin{bmatrix} 1 & 0.1 & 0.2 \\ 0.8 & 0.9 & 1 \\ 0.7 & 0.8 & 0.5 \\ 1 & 0.5 & 0.2 \end{bmatrix}),$$

следует ввести ее значения по строкам следующим образом:

```
>> m=[1 0.1 0.2; 0.8 0.9 1; 0.7 0.8 0.5; 1 0.5 0.2]
m =
    1.0000    0.1000    0.2000
    0.8000    0.9000    1.0000
    0.7000    0.8000    0.5000
    1.0000    0.5000    0.2000
```

Далее для этой матрицы могут быть применены различные унарные функции (функции с единственным аргументом), например, транспонирования и нахождения обратной матрицы, нахождения минимальных и максимальных элементов в каждом из столбцов матрицы и др. Например, для нахождения минимальных элементов в каждом из столбцов матрицы `m` следует воспользоваться встроенной функцией `min`:

```
>> min(m)
ans =
    0.7000    0.1000    0.2000
```

Основные алгебраические операции матричной арифметики приводятся в табл. 11.2. При этом следует отметить двойственность этих операций, поскольку наряду с символами операций для их выполнения можно использовать и встроенные функции. Так, например, для сложения двух матриц **A** и **B** одинаковой размерности можно ввести: `A+B` или `plus(A, B)`, что совершенно эквивалентно.

Таблица 11.2. *Основные алгебраические операции системы MATLAB*

Символ операции	Имя функции	Название	Примечание
`+`	`plus`	Сложение	Применяется для матриц и скаляров. В случае сложения двух матриц они должны быть одной размерности
`-`	`minus`	Вычитание	Применяется для матриц и скаляров. В случае разности двух матриц они должны быть одной размерности
`.*`	`times`	Умножение	Поэлементное умножение двух матриц, при этом матрицы должны быть одной размерности
`./`	`rdivide`	Правое деление	Поэлементное деление первой матрицы на вторую. В случае деления двух матриц они должны быть одной размерности
`.\`	`ldivide`	Левое деление	Поэлементное деление второй матрицы на первую. В случае деления двух матриц они должны быть одной размерности
`+`	`uplus`	Унарный плюс	Применяется для матриц и скаляров
`-`	`uminus`	Унарный минус	Применяется для матриц и скаляров
`.^`	`power`	Возведение в степень	Возведение в степень каждого из элементов матрицы
`.'`	`transpose`	Транспонирование	Обычное транспонирование матрицы, при котором ее строки и столбцы меняются местами
`'`	`ctranspose`	Транспонирование с комплексным сопряжением	В дополнение к обычному транспонированию матрицы выполняется комплексное сопряжение ее элементов
`*`	`mtimes`	Умножение матриц	Умножение двух матриц (строка на столбец), при котором размерности матриц должны быть согласованными
`/`	`mrdivide`	Правое деление матриц	Эквивалентно умножению первой матрицы на обратную второй матрицы

Таблица 11.2 *(окончание)*

Символ операции	Имя функции	Название	Примечание
\	mldivide	Левое деление матриц	Эквивалентно умножению обратной первой матрицы на вторую матрицу
^	mpower	Возведение в степень матрицы	Применяется только для квадратных матриц

Для обращения к отдельным элементам матрицы следует после имени матрицы указать в круглых скобках индексы соответствующего элемента, разделенные запятой. Например, для обращения к элементу первой строки и второго столбца матрицы m следует ввести:

```
>> m(1,2)
ans =
    0.1000
```

В некоторых случаях для задания значений элементам массивов и матриц весьма удобным оказывается использование оператора *двоеточие* (:), действие которого аналогично присваиванию значений переменной в цикле типа for. При этом достаточно указать значение первого элемента массива, затем после двоеточия — шаг изменения элементов и, наконец, после двоеточия — значение последнего элемента массива. Если значение шага отрицательное, то элементы массива будут следовать в убывающем порядке. Например, для изображения графиков функций одной переменной необходимо определить диапазон значений соответствующей независимой переменной. Это удобно сделать с помощью рассмотренного оператора следующим образом:

```
>> x=0:0.1:10;
```

В результате ввода этой строки будет определен массив x, состоящий из 101 элемента, первым из которых является число 0, а последним — число 10.

Для задания некоторых специальных матриц в системе MATLAB предусмотрены отдельные функции. Так, например, для того, чтобы определить матрицу размерности 3×4, состоящую из нулей, в окне команд достаточно ввести имя соответствующей функции и ее параметры в круглых скобках:

```
>> z=zeros(3, 4)
z =
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Для задания специальных матриц можно воспользоваться специальными функциями, простейшие из которых приводятся в табл. 11.3. С другими аналогичными функциями можно познакомиться по справочной системе MATLAB.

Таблица 11.3. *Простейшие функции для задания специальных матриц*

Имя функции	Назначение	Примечание
`zeros`	Задание матрицы, состоящей из нулей	В качестве аргументов указывается размерность матрицы
`ones`	Задание матрицы, состоящей из единиц	В качестве аргументов указывается размерность матрицы
`rand`	Задание матрицы, элементы которой распределены по равномерному закону в интервале [0,1]	В качестве аргументов указывается размерность матрицы
`randn`	Задание матрицы, элементы которой распределены по нормальному закону	В качестве аргументов указывается размерность матрицы
`eye`	Задание единичной матрицы, главная диагональ которой состоит из единиц	В качестве аргументов указывается размерность матрицы
`magic`	Задание "магической" матрицы, сумма элементов которой по строкам, по столбцам и по диагоналям одинаковая	В качестве единственного аргумента указывается размерность квадратной матрицы

В процессе работы в системе MATLAB следует знать о системных переменных, используемых для хранения и представления отдельных значений переменных или специальных математических констант. Одной из подобных системных переменных является рассмотренная ранее переменная `ans`. Общий перечень системных переменных представлен в табл. 11.4.

Таблица 11.4. *Системные переменные MATLAB*

Имя переменной	Назначение	Примечание
`ans`	Результат выполнения последней операции	
`i` или `j`	Мнимая единица, используемая при задании комплексных чисел	Определяется как `sqrt(-1)`
`computer`	Тип операционной системы компьютера, на котором используется система MATLAB	
`[str, maxsize]=computer`	Первый аргумент слева — тип операционной системы, второй аргумент — максимально допустимое число элементов в задаваемых матрицах	

Таблица 11.4 *(окончание)*

Имя переменной	Назначение	Примечание
`eps`	Значение порога для функций, реализующих приближенные итеративные процедуры вычислений	Определяется как `2^(-52)`
`Inf`	Положительная бесконечность для численных расчетов	Это значение возвращается при делении на 0
`NaN`	Неопределенное значение (Not a Number)	Это значение возвращается, например, при выполнении деления 0 на 0
`nargin`	Число входных аргументов вызываемой функции	
`narout`	Число выходных аргументов вызываемой функции	
`pi`	Транспонирование с комплексным сопряжением	В дополнение к обычному транспонированию матрицы выполняется комплексное сопряжение ее элементов
`realmax`	Наибольшее положительное действительное число для операций машинной арифметики	Положительная бесконечность по своей величине превышает это значение
`realmin`	Наименьшее положительное действительное число для операций машинной арифметики	Значение порога по своей величине превышает это значение

Примечание

Следует отметить, что указанным в табл. 11.4 системным переменным можно присвоить произвольные значения, при этом система MATLAB не выдаст даже предупреждения. Аналогично, в качестве имен переменных можно использовать и имена встроенных функций MATLAB (`sin`, `cos`, `abs` и т. д.), однако в этом случае соответствующие функции окажутся недоступными.

Информация о всех функциях системы MATLAB содержится в справочной системе. При этом операции сравнения и логические операции рассматриваются в *главе 14*. Для работы в режиме командной строки с пакетом Fuzzy Logic Toolbox могут быть использованы специальные функции, общий перечень которых с описанием их назначения и примерами задания представлен в *приложении 3*.

11.3. Графические возможности системы MATLAB

Система MATLAB обладает удобными, гибкими и мощными средствами визуализации и графического представления самых разнообразных математических объектов типа кривых, поверхностей и диаграмм на плоскости и в 3-мерном пространстве. При этом используются различные системы координат, стили и способы цветового выделения изображений, что обеспечивает наглядность получаемых рисунков. Построенные графики функциональных зависимостей можно экспортировать в большинство известных графических форматов. Подчеркивая достоинства графических возможностей системы MATLAB, следует отметить, что большинство изображенных в книге графиков получены с помощью системы MATLAB.

Детально рассмотреть все графические средства системы MATLAB, включая такие как средства анимации, объекты дескрипторной графики и собственный графический редактор пакета расширения Images, не представляется возможным, поскольку этой теме посвящены отдельные тома документации. В настоящей книге преследуется более скромная цель — познакомиться с основными функциями системы MATLAB, которые могут быть использованы для визуализации их отдельных свойств нечетких моделей и конечных результатов нечеткого моделирования.

Для построения простейшего графика функции одной переменной прежде всего необходимо определить множество (массив) значений независимой переменной и соответствующее множество значений зависимой (функциональной) переменной. После этого можно воспользоваться функцией системы MATLAB `plot` для изображения графика заданной функциональной зависимости. При этом задание значений независимой переменной удобно выполнить с помощью рассмотренной выше операции `:`. Ниже приводится последовательность команд для построения графика функции $y = \sin(x)$:

```
>> x = -10:0.1:10;
>> y = sin(x);
>> plot(x, y)
```

После выполнения данной последовательности команд возникнет новое окно с графиком этой функции (рис. 11.15). Это окно имеет собственное главное меню и панель команд. Соответствующие команды меню позволяют выполнить редактирование свойств изображенного графика, такие как: внести дополнительный текст (легенду), изменить цвет, тип линий и масштаб изображения, а также выполнить вращение изображения в любом из направлений.

Наиболее полный контроль и форматирование изображений графиков позволяет выполнить редактор свойств, состоящий из отдельных диалоговых окон. Так, редактор свойств осей **(Property Editor - Axes)**, вызываемый двойным щелчком мыши на изображении осей или командой меню **Edit>Axes Properties** (рис. 11.16), представляет собой самостоятельное диалоговое окно с набором вкладок, обеспечивающих доступ к отдельным осям графического изображения.

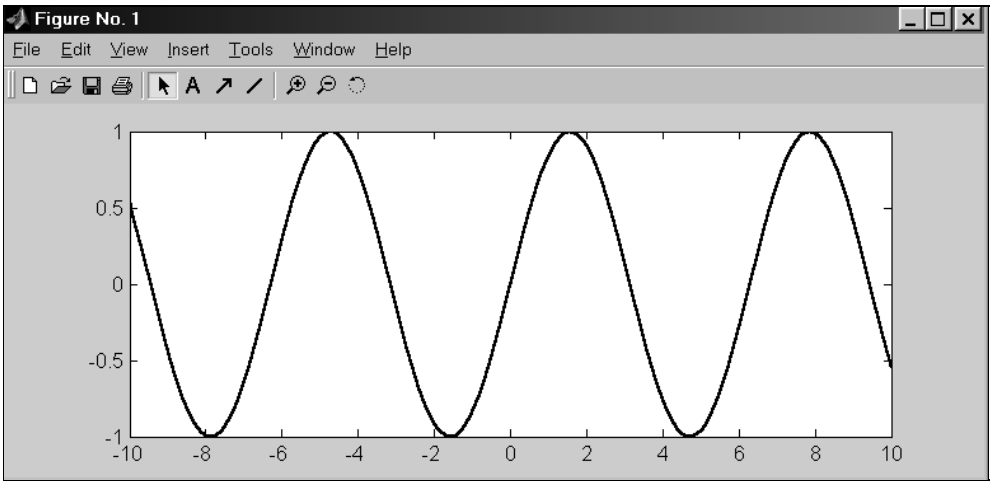


Рис. 11.15. Окно с построенным графиком функции $y = \sin(x)$

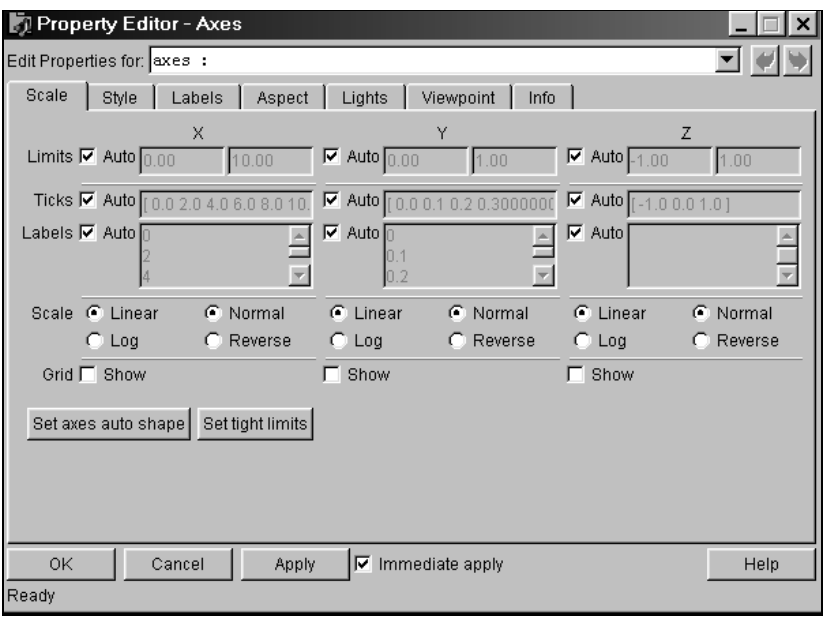


Рис. 11.16. Окно редактора свойств построенного графика с открытой вкладкой **Scale** для осей

Первая из вкладок **Scale** (Шкалирование) предназначена для форматирования осей графика. Здесь можно редактировать пределы изменения переменных осей, интервалы изображения меток переменных, а также указать линейный (по умол-

чанию) или логарифмический масштаб для отдельных из осей. При необходимости можно сделать видимой сетку для отдельных переменных осей, а также изменить направление оси на противоположное.

Вторая вкладка предназначена для редактирования стиля изображения осей (рис. 11.17). Здесь можно изменить заданные по умолчанию толщину и цвет линий отдельных осей, свойства шрифтов для записи легенд осей, а также геометрическое расположение линий осей относительно графика.

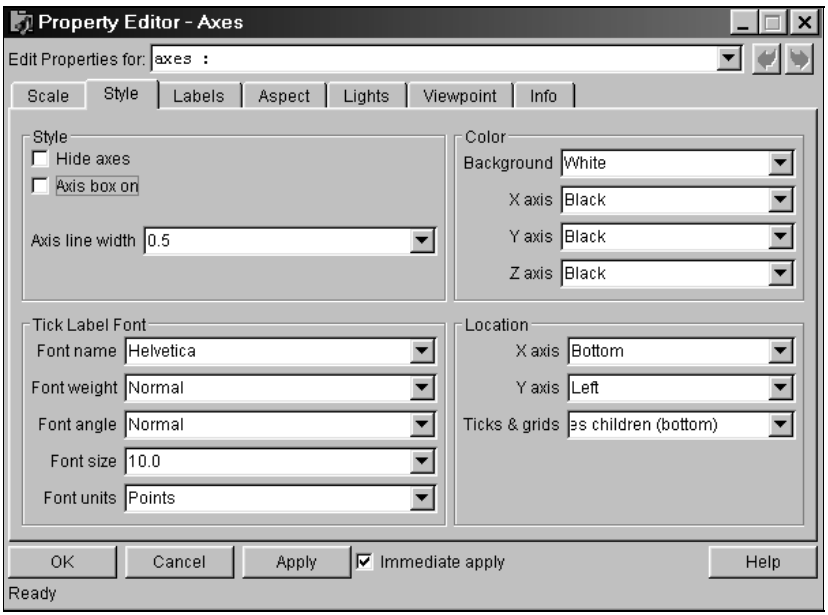


Рис. 11.17. Окно редактора свойств построенного графика с открытой вкладкой **Style** для осей

Остальные вкладки этого окна позволяют редактировать дополнительные свойства осей, связанные с пространственной перспективой и освещенностью графиков. Назначение меню команд этих вкладок вполне очевидно, поэтому подробно они рассматриваться не будут.

Редактор свойств линий (**Property Editor - Lines**), вызываемый двойным щелчком мыши на изображении линии графика или из редактора свойств осей выбором соответствующего пункта (**line:**) раскрывающегося списка **Edit Properties for** (рис. 11.18), представляет собой самостоятельное диалоговое окно с набором вкладок, обеспечивающих доступ к отдельным свойствам линии.

Здесь на вкладке **Style** (Стиль) можно изменить стиль изображения кривой, для чего следует выбрать необходимый пункт списка **Color**. На этой же вкладке

аналогичным образом можно изменить толщину и цвет изображения кривой (рис. 11.19). Следует отметить, что указанные свойства характерны для интерактивного форматирования элементов так называемой дескрипторной графики системы MATLAB, более подробно познакомиться с которой можно с помощью отдельных разделов справочной документации.

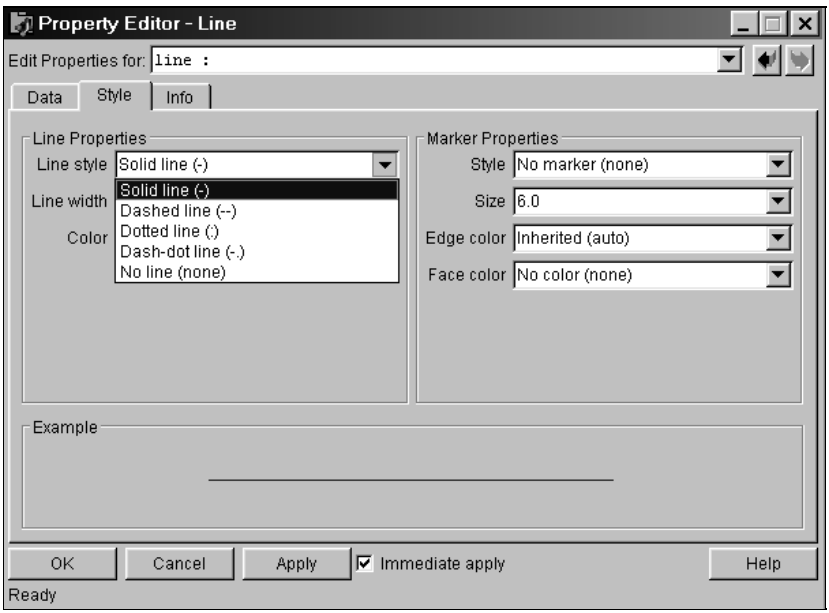


Рис. 11.18. Окно редактора свойств линии построенного графика с открытой вкладкой **Style** для выбора стиля изображения кривой

Для построения графиков нескольких функций одной переменной в одном окне можно также воспользоваться функцией системы MATLAB `plot`. Ниже приводится последовательность команд для построения графиков двух функций $y = \sin(x)$ и $y = \cos(x)$ в одном окне:

```
>> x = -10:0.1:10;
>> y = sin(x);
>> z = cos(x);
>> plot(x,y,x,z);
```

После выполнения данной последовательности команд появится окно с графиками этих функций (рис. 11.20). Используя вкладки свойств осей и линий, можно задать отображение сетки, изменить цвет кривых и добавить надписи (легенду) к этим кривым.

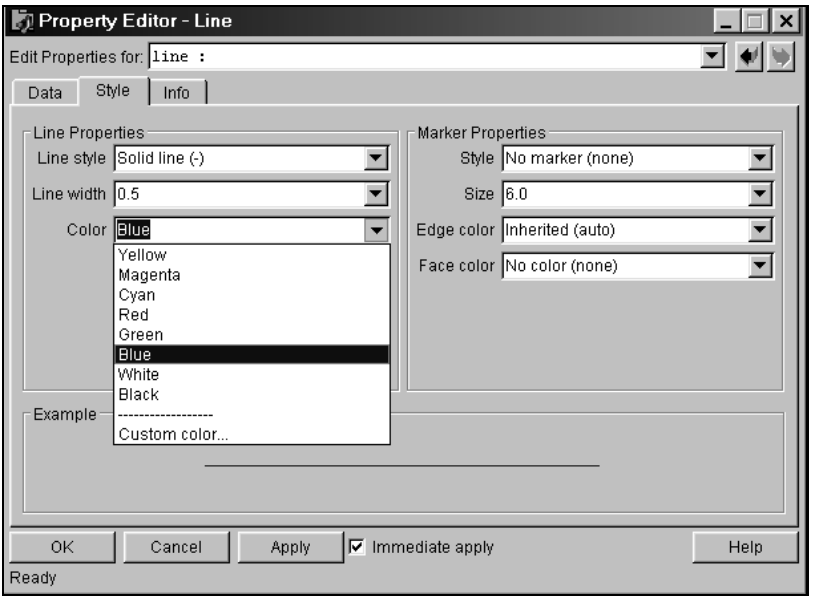


Рис. 11.19. Окно редактора свойств линии построенного графика с открытой вкладкой **Style** для выбора цвета изображения кривой

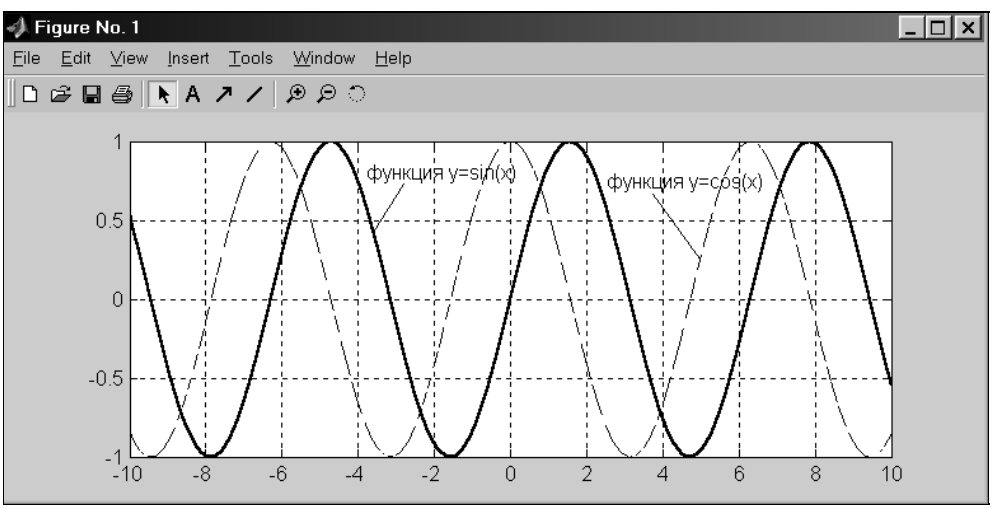


Рис. 11.20. Окно с построенными графиками функций $y = \sin(x)$ и $y = \cos(x)$

Для построения графиков 3-мерных поверхностей и кривых, представляющих собой функции двух переменных, также необходимо определить множество (массив) значений независимых переменных и соответствующее множество зна-

чений зависимой (функциональной) переменной. После этого для изображения графика заданной функциональной зависимости можно воспользоваться несколькими функциями системы MATLAB:

- `plot3` — функция построения изображений 3-мерных поверхностей линиями;
- `mesh` — функция построения изображений 3-мерных поверхностей с функциональной окраской образующих линий контурной сетки;
- `surf` — функция построения изображений 3-мерных поверхностей с функциональной закраской ячеек контурной сетки.

Задание значений независимых переменных удобно выполнить с помощью функции `meshgrid`. Ниже приводится последовательность команд для построения графика поверхности $z = \min(x, y)$ с помощью функции `plot3`, которая позволяет изображать графики кривых в перспективной проекции:

```
>> [x,y] = meshgrid([-1:0.1:1]);
>> z = min(x,y);
>> plot3(x,y,z)
```

После выполнения данной последовательности команд появится новое окно с графиком поверхности этой функции (рис. 11.21).

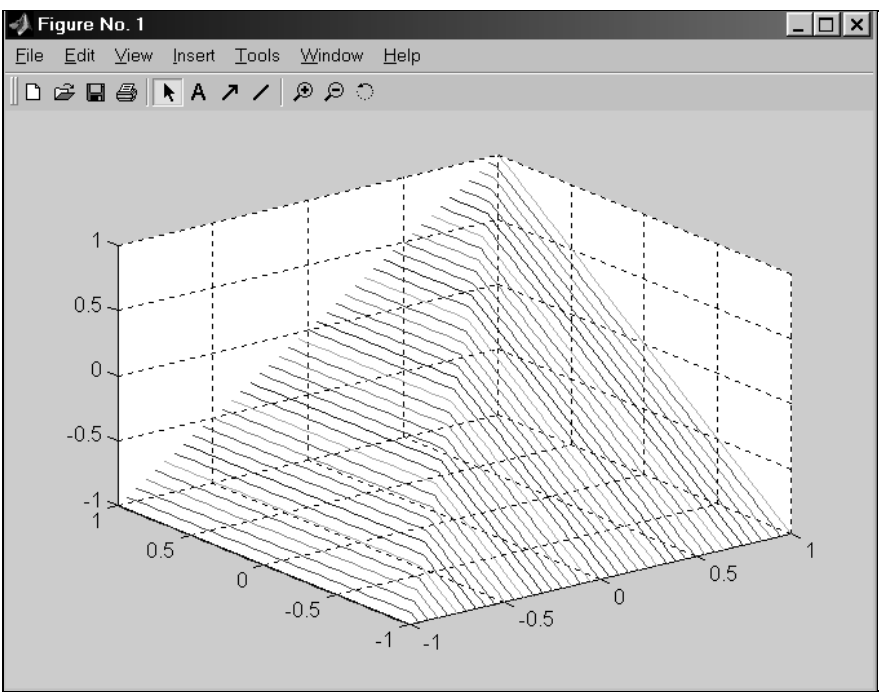


Рис. 11.21. Окно с построенной поверхностью $z = \min(x, y)$ с помощью функции `plot3`

Для сравнения ниже приводится последовательность команд для построения графика этой же поверхности $z = \min(x, y)$ с помощью функции `mesh`:

```
>> [x,y] = meshgrid([-1:0.1:1]);
>> z = min(x,y);
>> mesh(x,y,z)
```

После выполнения данной последовательности команд появится новое окно с графиком этой функции (рис. 11.22) в форме сетчатой поверхности.

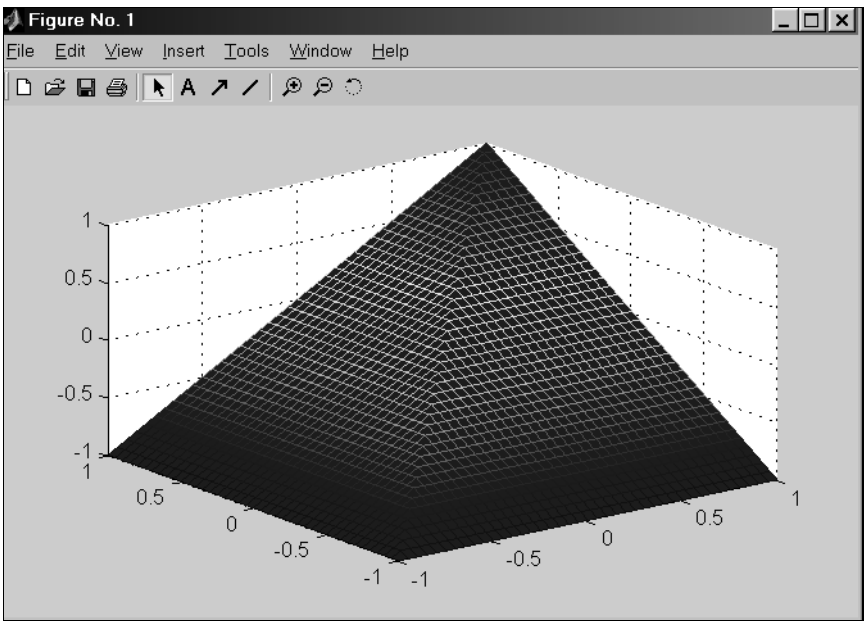


Рис. 11.22. Окно с построенной поверхностью $z = \min(x, y)$ с помощью функции `mesh`

Знакомство с функциями построения графиков в 3-мерном пространстве завершим еще одним примером визуализации рассматриваемой поверхности. Ниже приводится последовательность команд для построения графика поверхности $y = \min(x, y)$ с помощью функции `surf()`:

```
>> [x,y] = meshgrid([-1:0.1:1]);
>> z = min(x,y);
>> surf(x,y,z)
```

После выполнения данной последовательности команд появится новое окно с графиком этой функции (рис. 11.23) в форме расцвеченной поверхности.

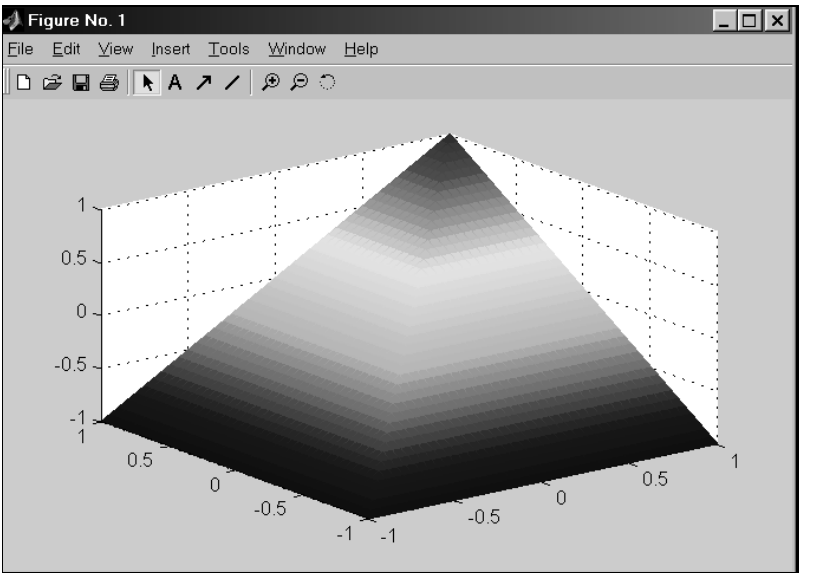


Рис. 11.23. Поверхность $z = \min(x, y)$, построенная с помощью функции `surf`

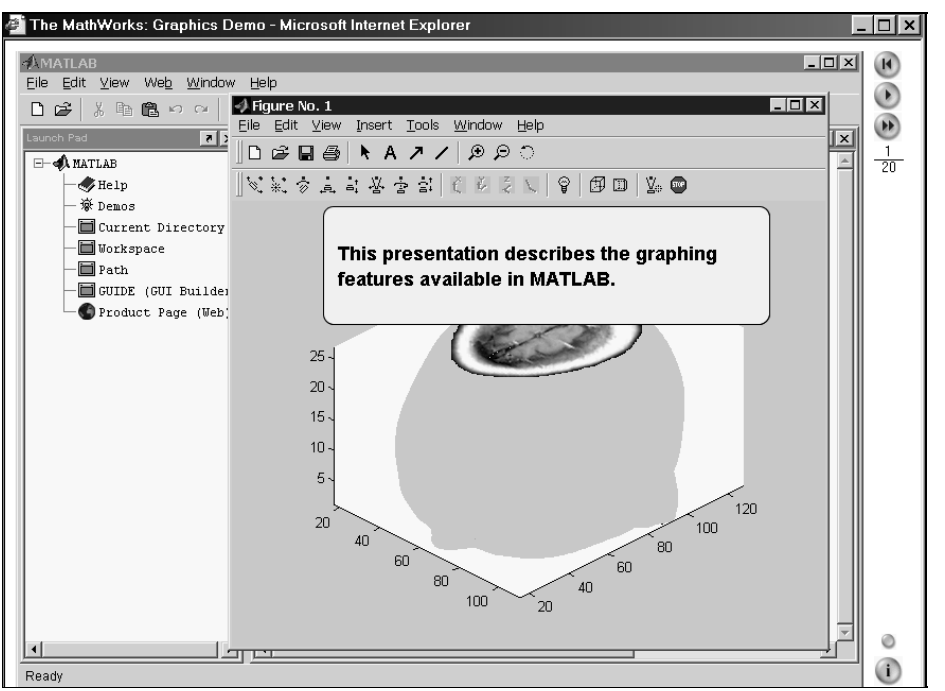


Рис. 11.24. Демонстрационный обучающий пример для ознакомления с графическими возможностями системы MATLAB

Для построения графиков кривых и поверхностей также могут быть использованы функции `meshc`, `meshz`, `surfc`, `surfl` и целый ряд других. Для построения столбиковых диаграмм можно использовать функция `bar`, `barh`, `hist`. Для построения плоской круговой диаграммы предназначена функция `pie`, а для построения 3-мерной круговой диаграммы — функция `pie3`.

Для дополнительного знакомства с графическими возможностями системы MATLAB могут оказаться полезными два обучающих мультипликационных примера. Первый из них, который вызвается из окна с демо-примерами операцией **Graphics>Overviw of features (playback)**, иллюстрирует процесс настройки свойств графического окна (рис. 11.24).

Второй обучающий пример, который можно вызвать из окна с демо-примерами операцией **Graphics>Plotting and printing (playback)**, иллюстрирует процесс настройки свойств изображений для печати в системе MATLAB (рис. 11.25).

В последующих главах приводятся другие варианты графических изображений, построенные с помощью средств MATLAB, а также рассматриваются способы построения нескольких графиков в одном графическом окне.

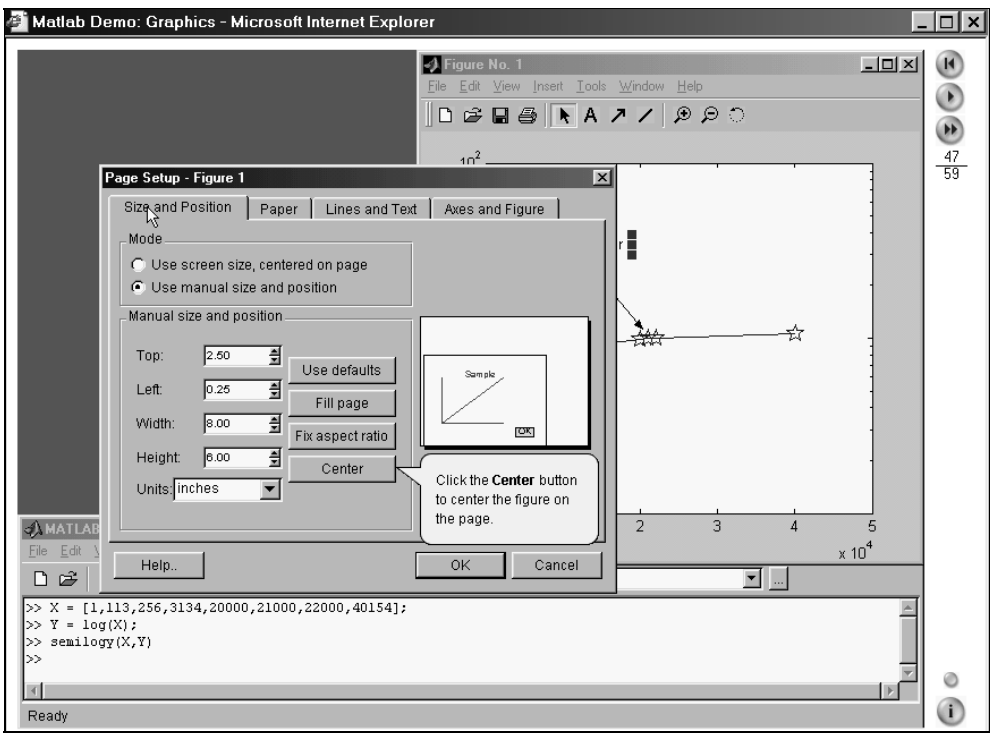
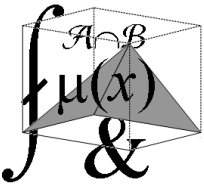


Рис. 11.25. Демонстрационный обучающий пример для ознакомления с возможностями настройки изображений для печати в системе MATLAB

Глава 12



Процесс нечеткого моделирования в среде MATLAB

Для реализации процесса нечеткого моделирования в среде MATLAB предназначен специальный пакет расширения Fuzzy Logic Toolbox. В рамках этого пакета, который расположен в папке C:\MATLAB6p1\toolbox\fuzzy (если система MATLAB установлена по умолчанию на диске C:), пользователь может выполнять необходимые действия по разработке и использованию нечетких моделей в одном из следующих режимов:

- в интерактивном режиме с помощью графических средств редактирования и визуализации всех компонентов систем нечеткого вывода;
- в режиме команд с помощью ввода имен соответствующих функций с необходимыми аргументами непосредственно в окно команд системы MATLAB

Ниже рассматриваются особенности разработки систем нечеткого вывода в каждом их этих режимов и даются рекомендации по выполнению необходимой последовательности действий.

12.1. Процесс разработки системы нечеткого вывода в интерактивном режиме

Для разработки и дальнейшего применения систем нечеткого вывода в интерактивном режиме могут быть использованы следующие графические средства, входящие в состав пакета Fuzzy Logic Toolbox.

- Редактор систем нечеткого вывода FIS (FIS Editor) или сокращенно — редактор FIS.
- Редактор функций принадлежности системы нечеткого вывода (Membership Function Editor) или сокращенно — редактор функций принадлежности.
- Редактор правил системы нечеткого вывода (Rule Editor) или сокращенно — редактор правил.
- Программа просмотра правил системы нечеткого вывода (Rule Viewer) или сокращенно — просмотрщик правил вывода.

- Программа просмотра поверхности системы нечеткого вывода (Surface Viewer) или сокращенно — просмотрщик поверхности вывода.

Кроме этих графических средств в состав пакета Fuzzy Logic Toolbox также входят следующие специальные программы.

- Редактор адаптивных систем нейро-нечеткого вывода (Adaptive Neuro-Fuzzy Inference System Editor) или сокращенно — редактор гибридных сетей или редактор ANFIS.
- Программа нечеткой кластеризации методом нечетких с-средних (fuzzy c-means clustering).

В табл. 12.1 представлены функции MATLAB, которые могут быть использованы для вызова соответствующих графических средств. Более подробно особенности работы с редактором гибридных сетей ANFIS изложены в *главе 15*, а особенности работы с программой нечеткой кластеризации — в *главе 13*.

Таблица 12.1. *Функции графического интерфейса пользователя*

Функция	Назначение
`anfisedit`	Редактор гибридных сетей ANFIS
`findcluster`	Программа нечеткой кластеризации
`fuzzy`	Редактор системы нечеткого вывода FIS
`mfedit`	Редактор функций принадлежности
`ruleedit`	Редактор правил нечеткого вывода
`ruleview`	Программа просмотра правил и диаграммы нечеткого вывода
`surfview`	Программа просмотра поверхности нечеткого вывода

Рассмотрим особенности каждого из графических средств, которые следует использовать для разработки и исследования систем нечеткого вывода в среде MATLAB.

Редактор систем нечеткого вывода FIS

Редактор систем нечеткого вывода FIS (или просто редактор FIS) является основным средством, которое используется для создания или редактирования систем нечеткого вывода в графическом режиме. Редактор FIS может быть открыт с помощью ввода функции `fuzzy` или `fuzzy('fismat')` в окне команд. Эта функция предоставляет пользователю возможность задавать и редактировать на высоком уровне свойства системы нечеткого вывода, такие как число входных и выходных переменных, тип системы нечеткого вывода, используемый метод дефаззификации и т. д.

Если функция `fuzzy` вызывается без аргументов, то редактор FIS вызывается для вновь создаваемой системы нечеткого вывода с именем Untitled по умолчанию (рис. 12.1). При этом по умолчанию также задается целый ряд параметров, таких как тип системы нечеткого вывода (Мамдани), нечеткие логические операции, методы импликации, агрегирования и дефаззификации и некоторые другие. Пользователь может согласиться с этими значениями или изменить их.

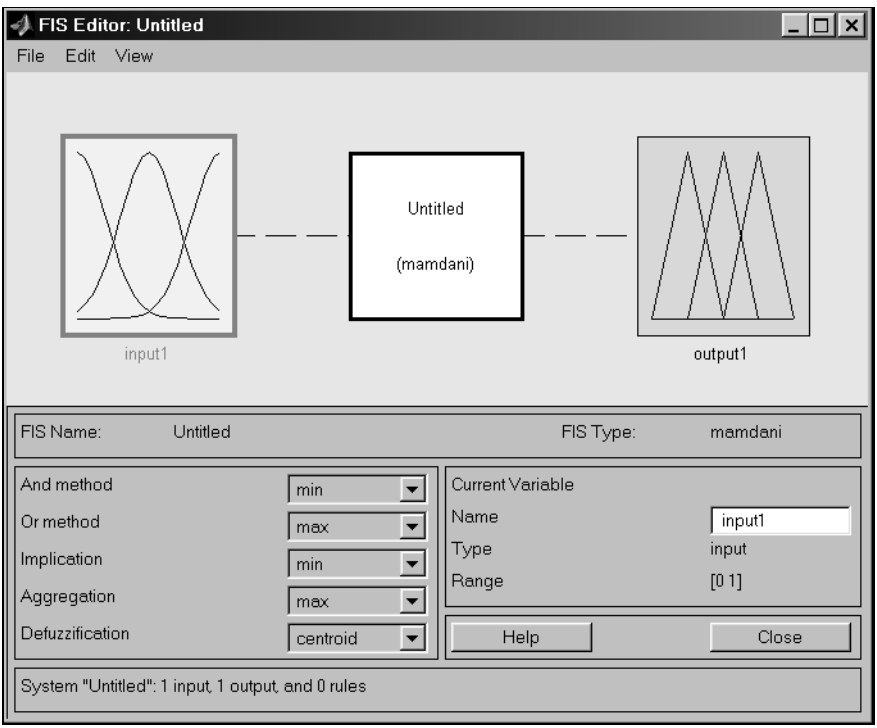


Рис. 12.1. Графический интерфейс редактора FIS, вызываемый функцией `fuzzy`

Если функция `fuzzy` вызывается с аргументом в форме `fuzzy('fismat')`, где `fismat` — имя внешнего файла с расширением fis с уже разработанной системой нечеткого вывода, то редактор вызывается с уже загруженной системой FIS с именем `fismat` (рис. 12.2).

Примечание

Возможен также вызов редактора FIS с помощью этой же функции в формате `fuzzy(fismat)`, где `fismat` — имя структуры FIS в рабочей области MATLAB. В этом случае соответствующая структура нечеткого вывода должна быть предварительно создана (например, средствами командного режима) либо загружена в рабочую область с помощью функции `readfis('fismat')`, где

`fismat` — имя внешнего fis-файла с разработанной ранее системой нечеткого вывода. В частности, вызов редактора функцией `fuzzy('tipper')` загружает в него систему нечеткого вывода для демонстрационного примера "*Чаевые в ресторане*", который более подробно будет рассмотрен ниже в качестве примера 12.1.

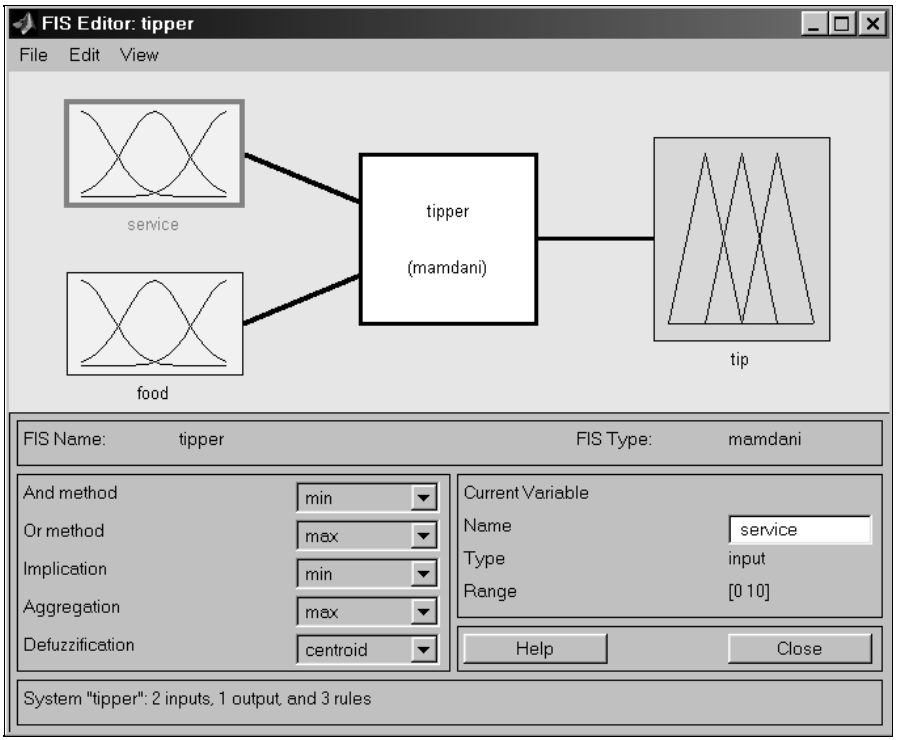


Рис. 12.2. Графический интерфейс редактора FIS, вызываемый функцией `fuzzy('tipper')`

Редактор FIS обладает графическим интерфейсом и позволяет вызывать все другие редакторы и программы просмотра систем нечеткого вывода. Графический интерфейс этого редактора обладает максимальным удобством и гибкостью, необходимой для интерактивной работы с отдельными компонентами системы нечеткого вывода.

В верхней части рабочего интерфейса редактора FIS изображается диаграмма, представляющая в визуальной форме входы и выходы системы нечеткого вывода, в центре которых находится так называемый *процессор нечетких правил*. Щелчок на прямоугольнике с изображением входа или выхода выделяет соответствующую переменную и делает ее текущей. Прямоугольник текущей переменной при этом выделяется красным цветом.

Двойной щелчок на прямоугольнике с изображением входной или выходной переменной вызывает редактор функций принадлежности с загруженной в него соответствующей переменной. Двойной щелчок на изображении процессора нечетких правил вызывает редактор правил для соответствующей системы нечеткого вывода. Если некоторая переменная существует в системе нечеткого вывода, но не используется в правилах вывода, то связь ее с процессором нечетких правил изображается не сплошной, а пунктирной линией.

Редактор FIS имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д. Рассмотрим назначение пунктов меню редактора FIS.

- Пункт меню **File** (Файл) редактора FIS содержит следующие операции:
  - **New FIS...** — позволяет выбрать тип задаваемой новой системы нечеткого вывода: **Mamdani** — типа Мамдани или **Sugeno** — типа Сугено. При этом задаваемая система нечеткого вывода не имеет ни входных, ни выходных переменных, а ее имя задается по умолчанию как Untitled;
  - **Import** — позволяет загрузить в редактор FIS существующую систему нечеткого вывода одним из следующих способов: **From Workspace...** — из рабочего пространства программы MATLAB или **From Disk...** — из внешнего файла. В последнем случае вызывается стандартное диалоговое окно открытия внешнего файла с диска;
  - **Export** — позволяет сохранить редактируемую систему нечеткого вывода одним из следующих способов: **To Workspace...** — в рабочем пространстве программы MATLAB или **To Disk...** — во внешнем файле. В последнем случае вызывается стандартное диалоговое окно сохранения файла на диске;
  - **Print** — позволяет распечатать на принтере редактируемую систему нечеткого вывода. В этом случае вызывается стандартное диалоговое окно настройки свойств печати на подключенном к данному компьютеру принтере;
  - **Close** — закрывает редактор FIS, при этом вызывается диалоговое окно с предложениями сохранить или отказаться от сохранения редактируемой системы нечеткого вывода.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **Add Variable...** — позволяет добавить в редактируемую систему нечеткого вывода переменную одного из следующих типов: **Input** — входную переменную или **Output** — выходную переменную;
  - **Remove Selected Variable** — удаляет выбранную переменную из редактируемой системы нечеткого вывода;
  - **Membership Functions...** — вызывает редактор функций принадлежности;
  - **Rules** — вызывает редактор правил нечеткого вывода.

- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил нечеткого вывода;
  - **Surface** — вызывает программу просмотра поверхности нечеткого вывода.

В левой нижней части рабочего интерфейса редактора FIS имеется 5 всплывающих меню:

- **And method** (Метод логической конъюнкции) — позволяет задать один из следующих методов для выполнения логической конъюнкции в условиях нечетких правил:
  - **min** — метод минимального значения (6.2);
  - **prod** — метод алгебраического произведения (6.3);
  - **Custom** — метод, определенный пользователем.
- **Or method** (Метод логической дизъюнкции) — позволяет задать один из следующих методов для выполнения логической дизъюнкции в условиях нечетких правил:
  - **max** — метод максимального значения (6.6);
  - **probor** — метод алгебраической суммы (6.7);
  - **Custom** — метод, определенный пользователем.
- **Implication method** (Метод вывода заключения) — позволяет задать один из следующих методов для выполнения (активизации) логического заключения в каждом из нечетких правил:
  - **min** — метод минимального значения (7.6);
  - **prod** — метод алгебраического произведения (7.7);
  - **Custom** — метод, определенный пользователем. Это меню не используется для систем нечеткого вывода типа Сугено.
- **Aggregation method** (Метод агрегирования) — позволяет задать один из следующих методов для агрегирования значений функции принадлежности каждой из выходных переменных в заключениях нечетких правил:
  - **max** — метод максимального значения (6.6);
  - **sum** — метод граничной суммы (6.8);
  - **probor** — метод алгебраической суммы (6.7).
  - **Custom** — метод, определенный пользователем. Это меню не используется для систем нечеткого вывода типа Сугено.
- **Defuzzification method** (Метод дефаззификации) — позволяет задать один из следующих методов для выполнения дефаззификации выходных переменных в системе нечеткого вывода типа Мамдани:
  - **centroid** — метод центра тяжести для дискретного множества значений функции принадлежности (7.10);

- **bisector** — метод центра площади (модификация (7.11));
- **mom** (middle of maximum) — метод среднего максимума, определяемый как среднее арифметическое левого и правого модальных значений;
- **som** (smallest of maximum) — метод наименьшего (левого) модального значения (7.12);
- **lom** (largest of maximum) — метод наибольшего (правого) модального значения (7.13);
- **Custom** — метод, определенный самим пользователем.

Для систем нечеткого вывода типа Сугено можно выбрать один из следующих методов дефаззификации:

- **wtaver** (weighted average) — метод взвешенного среднего (7.15);
- **wtsum** (weighted sum) — метод взвешенной суммы (7.16).

Примечание

Методы, определенные пользователем, должны быть заданы либо в существующих m-файлах, либо в форме разработанных пользователем отдельных m-файлов, размещенных в папке C:\MATLAB6p1\toolbox\fuzzy или в папке C:\MATLAB6p1\work (если система MATLAB установлена по умолчанию на диске C). Поскольку особенности написания и редактирования m-файлов относятся к области программирования, эта тематика будет рассмотрена далее в *главе 14*.

В правом нижнем углу находятся кнопка вызова встроенной справочной системы MATLAB (**Help**) и кнопка закрытия редактора FIS (**Close**).

Редактор функций принадлежности

Редактор функций принадлежности, как следует из его названия, предназначен для задания и редактирования функций принадлежности отдельных термов системы нечеткого вывода в графическом режиме. Редактор функций принадлежности может быть открыт с помощью ввода функции `mfedit`, а также `mfedit('a')` или `mfedit(a)`, в окне команд либо с помощью главного меню редактора FIS (командой меню **Edit>Membership Functions...** или нажатием клавиш <Ctrl>+<2>).

Эта функция, записанная в формате `mfedit`, просто вызывает редактор функций принадлежности без загрузки какой бы то ни было системы нечеткого вывода. Функция в формате `mfedit('a')` вызывает редактор функций принадлежности, который позволяет пользователю в графическом режиме анализировать и модифицировать все функции принадлежности некоторой структуры FIS, сохраненной во внешнем файле с именем a.fis. Функция в формате `mfedit(a)` работает с переменной рабочего пространства MATLAB, соответствующей структуре FIS с именем `a`. Для каждой функции принадлежности можно изменить ее имя, тип и параметры. Редактор предоставляет пользователю не только возможность вы-

брать любую из 11 встроенных функций принадлежности, но и задать собственную функцию принадлежности.

Результат вызова редактора функций принадлежности с помощью функции `mfedit('tipper')` изображен на рис. 12.3.

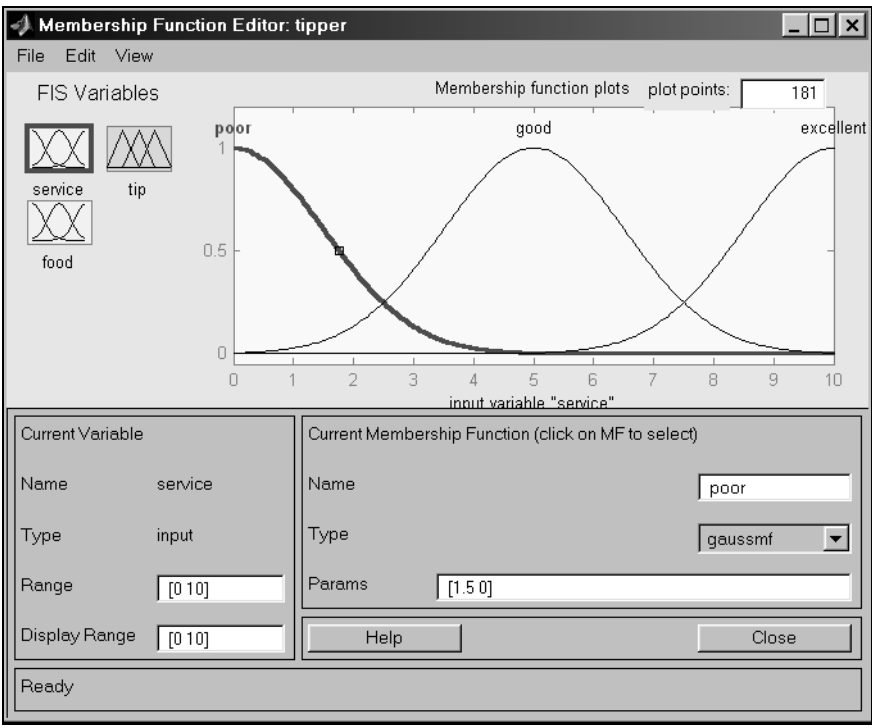


Рис. 12.3. Редактор функций принадлежности, вызываемый функцией `mfedit('tipper')`

Для отображения графиков функций принадлежности следует выбрать необходимую переменную в левой части графического интерфейса редактора под заголовком **FIS Variables** (Переменные FIS). Чтобы выбрать нужную функцию принадлежности, следует щелкнуть на ней или ее метке в основном окне с графиками функций принадлежности.

Редактор функций принадлежности имеет главное меню программы, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора функций принадлежности содержит такие же операции, что и соответствующий пункт меню редактора FIS.

- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **Add MF...** — позволяет добавить встроенную функцию принадлежности термов для выбранной переменной;
  - **Add Custom MF...** — позволяет добавить пользовательскую функцию принадлежности для отдельной переменной;
  - **Remove Current MF** — позволяет удалить отдельную функцию принадлежности;
  - **Remove All MFs** — позволяет удалить все функции принадлежности для отдельной переменной;
  - **FIS Properties...** — вызывает редактор FIS;
  - **Rules...** — вызывает редактор правил нечеткого вывода.
- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил нечеткого вывода;
  - **Surface** — вызывает программу просмотра поверхности нечеткого вывода.

Раскрывающийся список типов функций принадлежности позволяет выбрать одну из 11 встроенных функции принадлежности. Используя соответствующие поля ввода, можно изменить имена термов выбранной переменной в поле ввода **Name**, модифицировать параметры встроенных функций принадлежности в поле ввода **Params**.

Примечание

Поскольку данный редактор не позволяет задать функцию принадлежности, определенную пользователем, в случае подобной необходимости следует воспользоваться соответствующими функциями командного режима. Тем не менее, встроенных типов функций принадлежности оказывается вполне достаточно для большинства практических приложений.

Изменить вид функции принадлежности можно также с помощью мыши. Для этого следует выделить изменяемую функцию принадлежности на графике (она будет изображена красным цветом) и, не отпуская нажатую левую кнопку мыши, перемещать маркер в нужную сторону. При этом будут изменяться график соответствующей функции принадлежности и ее параметры. Этой возможностью следует пользоваться с осторожностью, поскольку выполненные изменения функции принадлежности уже не удастся отменить. В правом нижнем углу находятся кнопка вызова встроенной справочной системы MATLAB **(Help)** и кнопка закрытия редактора функций принадлежности **(Close)**.

Редактор правил системы нечеткого вывода

Редактор правил системы нечеткого вывода, как следует из его названия, предназначен для задания и редактирования отдельных правил системы нечеткого вывода в графическом режиме. Редактор правил может быть открыт с помощью ввода функции `ruleedit('a')` или `ruleedit(a)` в окне команд либо с помощью главного меню редактора FIS (командой меню **Edit>Rules...** или нажатием клавиш <Ctrl>+<3>).

Эта функция, записанная в формате `ruleedit('a')`, вызывает редактор правил, который позволяет пользователю в графическом режиме анализировать и модифицировать правила продукций системы нечеткого вывода FIS, сохраненной во внешнем файле с именем a.fis. Эта функция позволяет также выполнять грамматический анализ правил, которые используются в некоторой системе нечеткого вывода FIS.

Чтобы использовать данный редактор для создания правил, необходимо предварительно определить все входные и выходные переменные, для чего можно воспользоваться редактором системы нечеткого вывода FIS и редактором функций принадлежности. При этом задать правила можно с помощью выбора соответствующих значений термов входных и выходных переменных.

Результат вызова функции `ruleedit('tipper')` изображен на рис. 12.4.

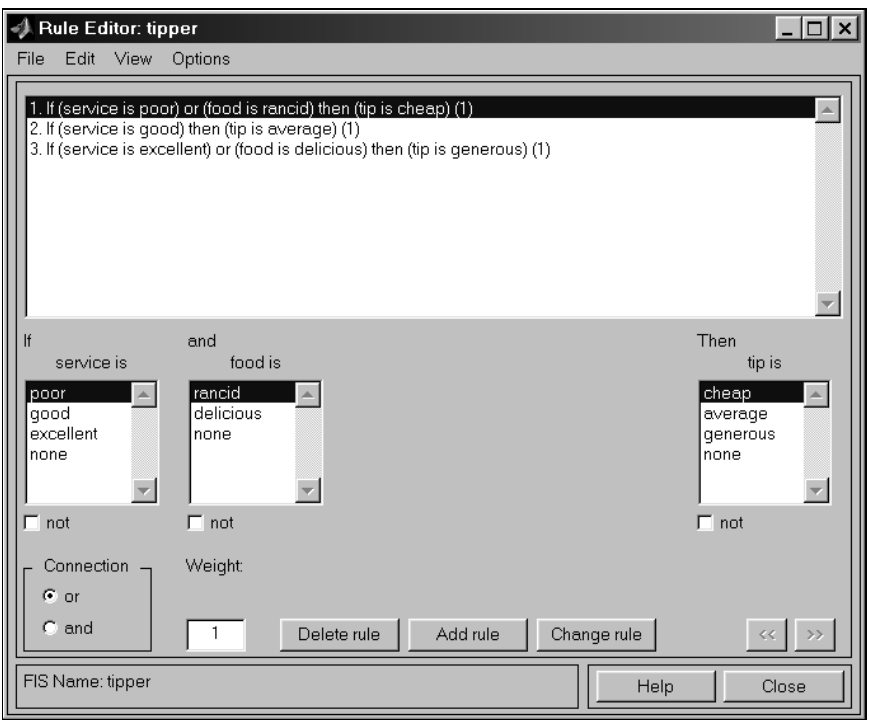


Рис. 12.4. Редактор правил, вызываемый функцией `ruleedit('tipper')`

Функция в формате `ruleedit(a)` вызывает редактор правил для переменной рабочего пространства MATLAB, соответствующей структуре FIS с именем `a`.

Редактор правил имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора правил содержит такие же операции, что и соответствующий пункт меню редактора FIS.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **FIS Properties...** — вызывает редактор FIS;
  - **Membership Functions...** — вызывает редактор функций принадлежности.
- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил;
  - **Surface** — вызывает программу просмотра поверхности вывода.
- Пункт меню **Options** (Сервис) содержит следующие операции:
  - **Language** — позволяет выбрать язык для записи правил в форме текста: **English** (английский), **Deutsch** (немецкий) или **Francais** (французский);
  - **Format** — позволяет выбрать формат записи правил системы нечеткого вывода: **Verbose** (в форме текста), **Symbolic** (в символической форме) или **Indexed** (в цифровой форме).

При записи правил в форме текста для создания законченных предложений используются служебные слова "if", "then", "is", "AND", "OR" и т. д. При записи правил в символической форме эти служебные слова заменяются символами соответствующих операций. Например, правило "if (A is $\mathcal{A}$) and (B is $\mathcal{B}$) then (C is $\mathcal{C}$)" преобразуется к виду: "(A = = $\mathcal{A}$) & (B = = $\mathcal{B}$) => (C = = $\mathcal{C}$)". Правила нечеткого вывода, записанные в цифровой форме, соответствуют формату их представления в структуре FIS, который рассматривается в *разд. 12.3*.

Поля ввода в средней части графического интерфейса редактора правил позволяют задать новое правило в системе нечеткого вывода. Для этого необходимо выделить имя терма соответствующей переменной, которая должна быть предварительно определена с помощью редактора функций принадлежности. Если некоторый терм не входит в правило, то для него следует выбрать значение "none". Если в условии правила используется логическое отрицание некоторого терма, то для этого терма следует отметить соответствующий флажок с меткой "not" ("выставить галочку").

Редактор правил позволяет также задать логические связки для подусловий правила (переключатель **Connection**) и вес правила (поле ввода **Weight**). Кнопки в нижней части графического интерфейса редактора правил, как следует из их названий, служат для удаления выделенного в окне правила (**Delete rule**), добавления созданного правила в систему (**Add rule**) и внесения изменений в выделенное

в окне правило (**Change rule**). В правом нижнем углу находятся кнопки вызова встроенной справочной системы MATLAB (**Help**) и кнопка закрытия редактора правил (**Close**).

Программа просмотра правил системы нечеткого вывода

Главное назначение программы просмотра правил заключается в возможности визуализировать результаты нечеткого вывода и получать значения выходных переменных в зависимости от исходных значений входных переменных. Графический интерфейс программы просмотра правил может быть открыт с помощью ввода функции `ruleview('a')` или `ruleview(a)` в окне команд либо с помощью главного меню редактора FIS, редактора функций принадлежности или редактора правил (командой меню **View>Rules** или нажатием клавиш <Ctrl>+<5>).

Функция, записанная в формате `ruleview('a')`, вызывает программу просмотра правил, которая изображает диаграмму нечеткого вывода для структуры FIS, сохраненной во внешнем файле с именем a.fis. Функция в формате `ruleview(a)` вызывает программу просмотра правил для переменной рабочего пространства MATLAB, соответствующей структуре FIS с именем `a`.

Программа просмотра правил не позволяет редактировать правила и функции принадлежности термов переменных и используется после разработки системы нечеткого вывода на этапе ее анализа и оценки. Функцию также целесообразно использовать в том случае, когда необходимо визуально представить весь процесс нечеткого вывода от начала до конца. При этом пользователь имеет возможность оценить значения выходных переменных нечеткой модели и влияние каждого из правил на результат нечеткого вывода посредством изменения значений входных переменных.

Графический интерфейс программы просмотра правил изображен на рис. 12.5.

Программа просмотра правил имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора правил содержит такие же операции, что и соответствующий пункт меню редактора FIS.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **FIS Properties...** — вызывает редактор FIS;
  - **Membership Functions...** — вызывает редактор функций принадлежности;
  - **Rules...** — вызывает программу редактирования правил.
- Пункт меню **View** (Вид) содержит следующие операции:
  - **Surface** — вызывает программу просмотра поверхности вывода.

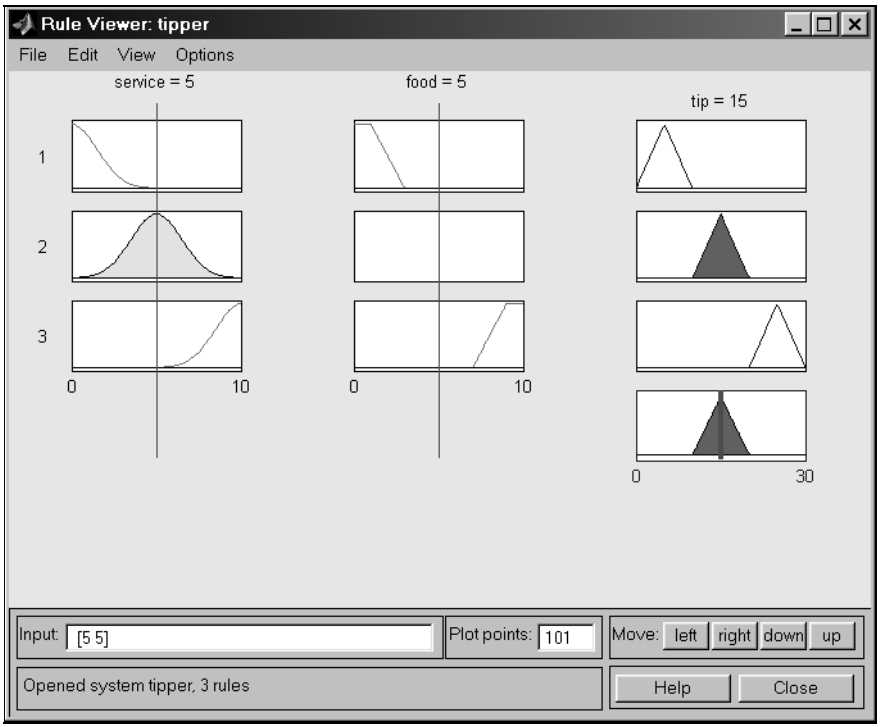


Рис. 12.5. Программа просмотра правил, вызванная функцией `ruleview('tipper')`

❒ Пункт меню **Options** (Сервис) содержит следующие операции:

- **Format** — позволяет выбрать формат записи правил системы нечеткого вывода: **Verbose** (в форме текста), **Symbolic** (в символической форме) или **Indexed** (в цифровой форме).

В центральной части графического интерфейса программы просмотра правил расположены прямоугольники, соответствующие отдельным входным переменным (функции принадлежности желтого цвета) и выходным переменным (функции принадлежности синего цвета) правил нечеткого вывода. При этом каждому правилу соответствует отдельная строка из этих прямоугольников. Номера правил указаны в левой части графического интерфейса.

В правой нижней части графического интерфейса расположен прямоугольник, изображающий дефаззификацию выходной переменной после аккумулирования всех заключений правил нечеткого вывода. Полученное в результате дефаззификации значение выходной переменной указывается в верхней части столбца с именем этой выходной переменной (`tip=15` на рис. 12.5).

Прямоугольники входных переменных пересекает вертикальная прямая красного цвета, положение которой соответствует конкретному значению входной пе-

ременной соответствующего столбца. Задать конкретные значения входных переменных можно либо с помощью их записи в поле ввода **Input**, либо с помощью мыши, перемещая вертикальные прямые в нужном направлении. В последнем случае можно щелкнуть на той или иной вертикальной прямой и, удерживая нажатой левую кнопку мыши, переместить прямую вправо или влево, либо просто щелкнуть в необходимой точке внутри прямоугольника соответствующей входной переменной.

Полученные после изменения значения входных переменных непосредственно отображаются в верхней части прямоугольников после имени входных переменных и в поле ввода с меткой **Input**. Более того, система MATLAB реагирует на каждое изменение значения отдельной входной переменной выполнением процедуры нечеткого вывода, получением и отображением соответствующих результирующих значений выходных переменных.

Программа просмотра поверхности системы нечеткого вывода

Программа просмотра поверхности системы нечеткого вывода позволяет просматривать поверхность системы нечеткого вывода и визуализировать графики зависимости выходных переменных от отдельных входных переменных. Графический интерфейс программы просмотра правил может быть открыт с помощью ввода функции `surfview('a')` или `surfview(a)` в окне команд либо с помощью главного меню редактора FIS, редактора функций принадлежности или редактора правил (командой меню **View>Surface** или нажатием клавиш <Ctrl>+<6>).

Функция, записанная в формате `surfview('a')`, вызывает программу просмотра поверхности, которая изображает поверхность нечеткого вывода для структуры FIS, сохраненной во внешнем файле с именем a.fis, для любой одной или двух из ее входных переменных. Функция в формате `surfview(a)` вызывает программу просмотра поверхности вывода для переменной рабочего пространства MATLAB, соответствующей структуре FIS с именем `a`.

Графический интерфейс программы просмотра поверхности изображен на рис. 12.6.

Программа просмотра поверхности вывода имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора правил содержит такие же операции, что и соответствующий пункт меню редактора FIS.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **FIS Properties...** — вызывает редактор FIS;
  - **Membership Functions...** — вызывает редактор функций принадлежности;
  - **Rules...** — вызывает программу редактирования правил.

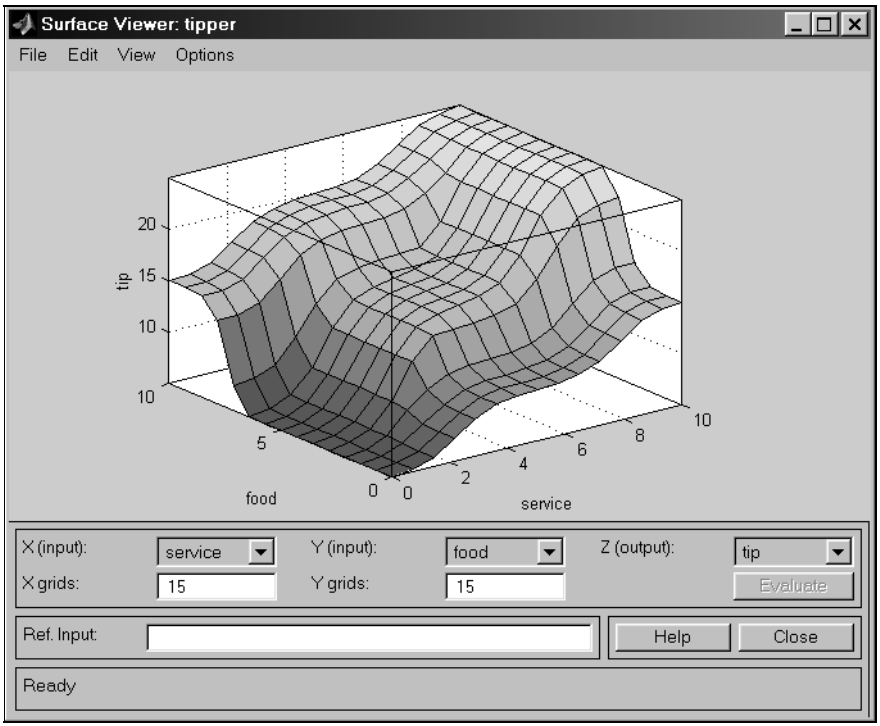


Рис. 12.6. Программа просмотра поверхности вывода, вызываемая функцией `surfview('tipper')`

- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил.
- Пункт меню **Options** (Сервис) содержит следующие операции:
  - **Plot** — позволяет выбрать один из 8 стилей изображения графика поверхности вывода;
  - **Color Map** — позволяет выбрать одну из 4 цветовых схем изображения графика поверхности вывода;
  - **Always evaluate** — пометка галочкой этого пункта вложенного меню приводит к автоматическому формированию новой поверхности вывода всякий раз, когда вносятся изменения в систему нечеткого вывода, влияющие на форму графика поверхности вывода (такие как изменение количества точек сетки графика). Это значение принято по умолчанию. Чтобы его отменить, необходимо снять галочку у этого пункта вложенного меню, щелкнув на этой позиции меню.

Программа просмотра поверхности вывода не позволяет вносить изменения в систему нечеткого вывода и соответствующую ей структуру FIS. Используя

главное меню программы, пользователь может выбрать входные переменные и соответствующие им горизонтальные оси системы координат (X и Y), а также выходную переменную, которой соответствует вертикальная ось системы координат (Z).

Щелкнув и удерживая левую кнопку мыши на осях графика поверхности, посредством последующего перемещения курсора мыши в том или ином направлении можно изменить угол просмотра поверхности вывода. Если рассматривается система нечеткого вывода с более чем двумя входными переменными, то для невизуализируемых входных переменных следует задать некоторые постоянные значения (константы).

Далее в этой главе описываются конкретные приемы и операции с рассмотренными средствами пакета Fuzzy Logic Toolbox, используя которые, пользователь может разработать и выполнить анализ системы нечеткого вывода в графическом режиме.

12.2. Пример разработки системы нечеткого вывода в интерактивном режиме

В качестве примера разработки системы нечеткого вывода в интерактивном режиме с помощью графических средств пакета Fuzzy Logic Toolbox рассмотрим следующую нечеткую модель, которая входит в число демонстрационных примеров системы MATLAB.

П р и м е р 12.1. "*Чаевые в ресторане*". Рассмотрим ситуацию в ресторане, при которой, согласно принятым в США традициям, после окончания обслуживания посетителя принято оставлять официанту чаевые. Основываясь на устоявшихся в этой стране обычаях и интуитивных представлениях посетителей ресторанов величина суммы чаевых не является постоянной и зависит, например, от качества обслуживания и качества приготовления заказанных блюд.

Задача состоит в том, чтобы разработать некоторую экспертную систему, которая была бы реализована в виде системы нечеткого вывода и позволяла бы определять величину чаевых на основе субъективных оценок посетителей качества обслуживания и качества приготовления заказанных блюд.

Эмпирические знания о рассматриваемой проблемной области могут быть представлены в форме следующих эвристических правил продукций:

1. Если обслуживание плохое или ужин подгоревший, то чаевые — малые.
2. Если обслуживание хорошее, то чаевые — средние.
3. Если обслуживание отличное или ужин превосходный, то чаевые — щедрые.

Примечание

Приведенные выше правила действительно субъективны и не свободны от критики. В частности, для многих посетителей наших ресторанов может показаться

странным правило 1, согласно которому следует оставлять чаевые в случае плохого обслуживания или подгоревшего ужина, и правило 2, согласно которому следует оставлять средние чаевые даже в случае подгоревшего ужина. Возможно, некоторые из наших читателей сочтут возможным вообще отказаться от оставления чаевых в подобных ситуациях и будут по своему правы. Тем не менее, поскольку данный пример широко используется в литературе для демонстрации возможностей системы MATLAB, он приводится здесь без изменения в своем оригинальном виде.

В качестве входных параметров системы нечеткого вывода будем рассматривать 2 нечеткие лингвистические переменные: "*качество обслуживания*" и "*качество приготовления заказанных блюд*" (или сокращенно — "*качество ужина*"), а в качестве выходных параметров — нечеткую лингвистическую переменную "*величина чаевых*".

В качестве терм-множества первой лингвистической переменной "*качество обслуживания*" будем использовать множество T_1={"*плохое*", "*хорошее*", "*отличное*"}, а в качестве терм-множества второй лингвистической переменной "*качество ужина*" будем использовать множество T_2={"*подгоревший*", "*превосходный*"}. В качестве терм-множества выходной лингвистической переменной "*величина чаевых*" будем использовать множество T_3={"*малые*", "*средние*", "*щедрые*"}. При этом каждый из термов первой и второй входной переменной (качество обслуживания и приготовления заказанных блюд) будем оценивать по 10-балльной порядковой шкале, при которой цифре 0 соответствует наихудшая оценка, а цифре 10 — наилучшая оценка. Что касается термов выходной переменной, то будем предполагать, что малые чаевые составляют около 5% от стоимости заказанных блюд, средние чаевые — около 15%, а щедрые чаевые — около 25%.

С учетом сделанных уточнений, рассмотренная субъективная информация о величине чаевых может быть представлена в форме 3-х правил нечетких продукций следующего вида (система нечеткого вывода типа Мамдани):

ПРАВИЛО_1: ЕСЛИ "*качество обслуживания плохое*" ИЛИ "*ужин подгоревший*"

ТО "*величина чаевых малая*"

ПРАВИЛО_2: ЕСЛИ "*качество обслуживания хорошее*"

ТО "*величина чаевых средняя*"

ПРАВИЛО_3: ЕСЛИ "*качество обслуживания отличное*" ИЛИ "*ужин превосходный*"

ТО "*величина чаевых щедрая*"

Примечание

Заканчивая содержательное описание демонстрационного примера "*Чаевые в ресторане*", следует отметить, что соответствующая ему система нечеткого вывода хранится во внешнем файле с именем tipper.fis в папке C:\MATLAB6p1\toolbox\fuzzy\fuzdemos, если система MATLAB установлена на диске C:.

Процесс разработки системы нечеткого вывода в интерактивном режиме для рассмотренного выше примера "*Чаевые в ресторане*" состоит в выполнении следующей последовательности действий:

1. Вызвать редактор систем нечеткого вывода FIS, для чего в окне команд набрать имя соответствующей функции `fuzzy`. После выполнения этой команды на экране появится графический интерфейс редактора FIS с именем системы нечеткого вывода Untitled и типом системы нечеткого вывода (Мамдани), предложенными по умолчанию (см. рис. 12.1).
2. Поскольку в примере 12.1 рассматривается система нечеткого вывода с двумя входами, необходимо добавить в разрабатываемую систему FIS еще одну входную переменную. Для этого следует выполнить команду меню **Edit>Add Variable... >Input**. В результате выполнения этой команды на диаграмме системы нечеткого вывода появится новый желтый прямоугольник с именем второй входной переменной: `input2` (рис. 12.7).

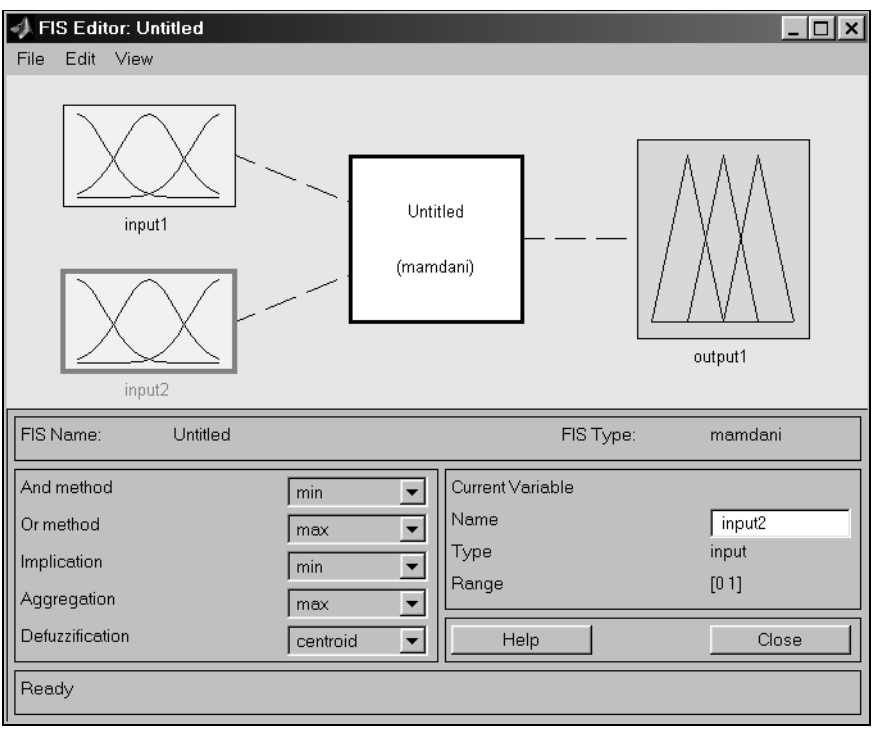


Рис. 12.7. Вид редактора FIS после добавления второй входной переменной

3. Изменим имена входных и выходных переменных, предложенных системой MATLAB по умолчанию. Для этого необходимо выделить прямоугольник с именем соответствующей переменной, выполнив щелчок на его изображении

на диаграмме (стороны выделенного прямоугольника имеют красный цвет). После чего следует набрать новое имя переменной в поле ввода **Name** в правой части редактора FIS. Результат изменения имен переменных системы нечеткого вывода изображен на рис. 12.8.

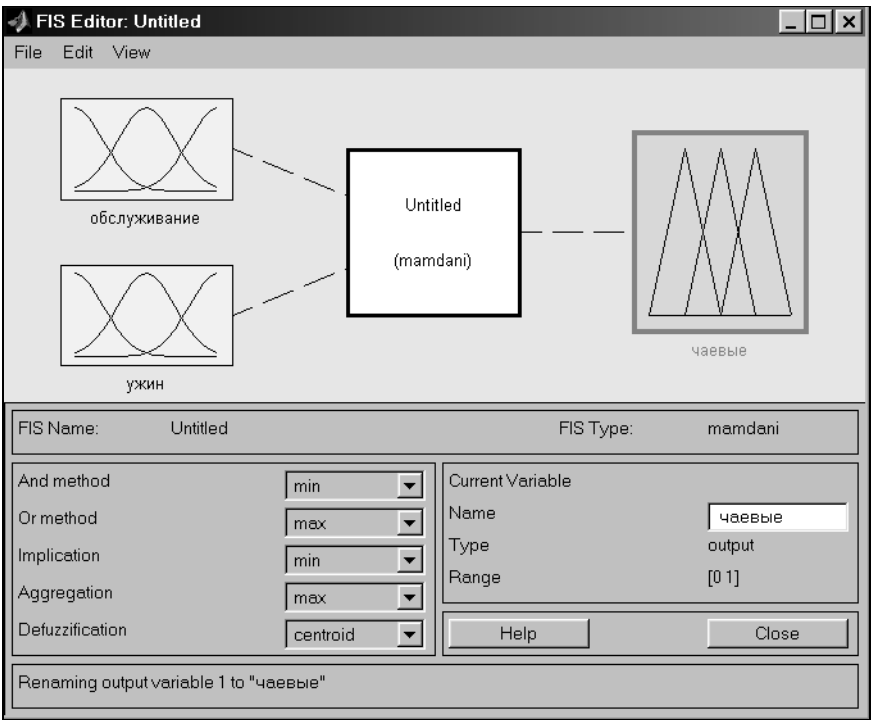


Рис. 12.8. Вид редактора FIS после изменения имен переменных, предложенных системой MATLAB по умолчанию

Примечание

Чтобы избежать проблем с корректным отображением символов кириллицы, следует давать такие имена переменным, которые состоят из одного слова без дополнительных служебных символов. Более того, в версиях MATLAB 6.0/6.1 имеется ряд проблем с кириллицей в названиях переменных и термов (символы "с" и "я"). Как это ни парадоксально, эти проблемы отсутствуют в предыдущей версии MATLAB 5.3.1. Именно по этой причине вторая переменная в примере 12.1 получила название "ужин".

4. Изменим имя системы нечеткого вывода (Untitled), предложенное по умолчанию. Для этого сохраним создаваемую структуру FIS во внешнем файле с именем `mytip.fis`, выполнив команду меню **File>Export>To Disk...**. При этом

будет вызвано стандартное диалоговое окно сохранения файла, в котором пользователю предлагается ввести имя соответствующего файла (расширение файла приписывается автоматически). Оставим без изменения предложенные системой MATLAB по умолчанию: метод нечеткого логического И **(And method)** — значение "min", метод нечеткого логического ИЛИ **(Or method)** — значение "max", метод импликации **(Implication)** — значение "min", метод агрегирования **(Aggregation)** — значение "max" и метод дефаззификации **(Defuzzification)** — значение "centroid". Очевидно, эти значения могут быть изменены пользователем.

5. Теперь необходимо определить термы и их функции принадлежности для входных и выходных переменных нашей системы нечеткого вывода. Для этой цели следует воспользоваться редактором функций принадлежности, который может быть вызван одним из следующих способов:

   - двойным щелчком на значке прямоугольника с именем соответствующей переменной;

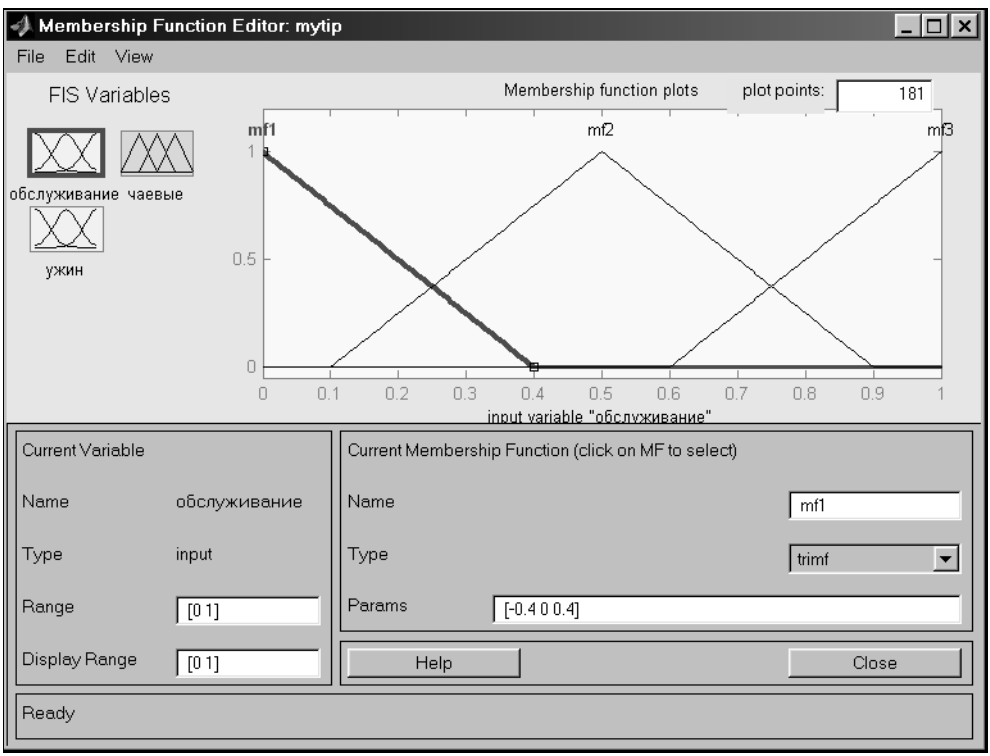


Рис. 12.9. Вид редактора функций принадлежности после его вызова с функциями принадлежности для термов переменной "*обслуживание*", предложенных системой MATLAB по умолчанию

- командой меню **Edit>Membership Functions...** (предварительно должен быть выделен прямоугольник с именем соответствующей переменной);
- нажатием клавиш <Ctrl>+<2> (предварительно также должен быть выделен прямоугольник с именем соответствующей переменной).

После вызова редактора функций принадлежности каждой из переменных по умолчанию предлагается 3 терма с треугольными функции принадлежности (рис. 12.9).

Вначале изменим диапазон определения значений входных переменных, для чего в полях ввода **Range** и **Display Range** изменим верхнее значение с 1 на 10 (баллов). Аналогично выполняются изменения соответствующих диапазонов для выходной переменной "чаевые", при этом верхнее значение 1 следует заменить на 30 (%). Изменения подтверждаются нажатием на клавишу <Enter> на клавиатуре.

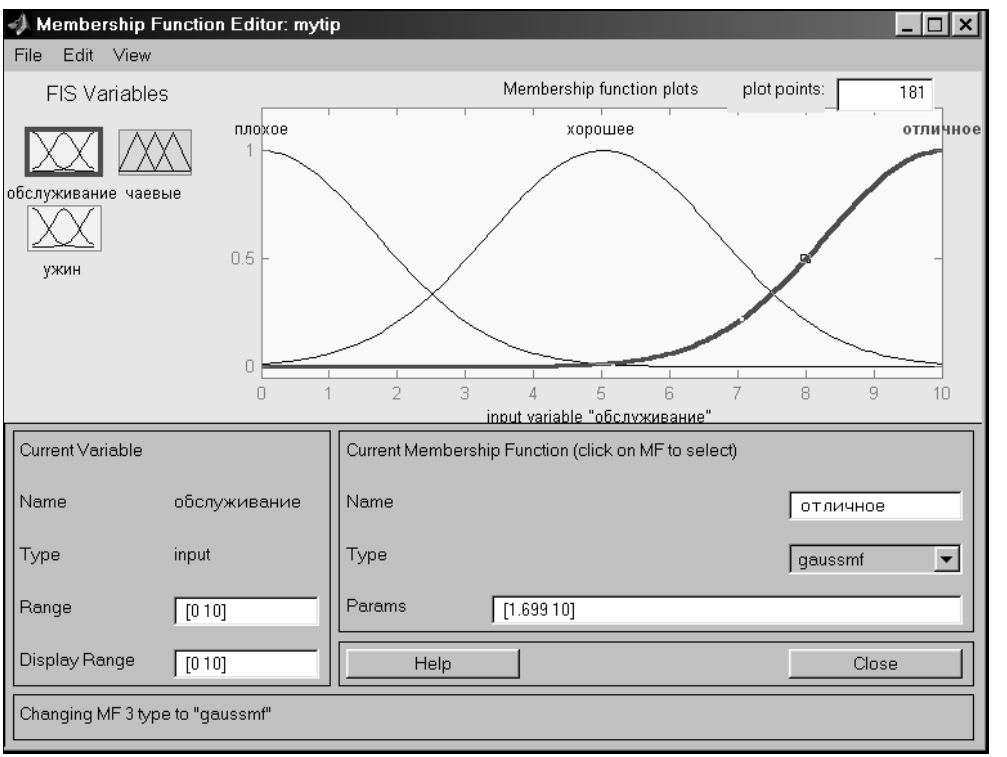


Рис. 12.10. Вид редактора функций принадлежности после изменения названия термов и типа их функций принадлежности для первой входной переменной "*обслуживание*"

Далее изменим названия термов первой входной переменной "*обслуживание*", предложенные системой MATLAB по умолчанию (`mf1`, `mf2`, `mf3`) на "*плохое*", "*хорошее*", "*отличное*" соответственно. После чего изменим тип функций принадлежности первой переменной, предложенный по умолчанию, на функции типа Гаусса (`gaussmf`), выбрав соответствующий пункт в поле **Type**. Параметры вновь заданных функций принадлежности оставим без изменения. Вид редактора функций принадлежности после внесенных изменений для первой из входных переменных изображен на рис. 12.10.

Аналогичным образом изменим названия термов второй входной переменной "*ужин*" и удалим один из термов с соответствующей функцией принадлежности. Для удаления терма следует выделить удаляемую функцию принадлежности и нажать клавишу <Delete> на клавиатуре. Переход к редактированию переменной осуществляется щелчком на изображении прямоугольника с именем необходимой переменной. Для переменной "*ужин*" изменим тип функций принадлежности ее термов на трапециевидные функции (`trapmf`) и их параметры следующим образом: для терма "*подгоревший*" зададим параметры [0 0 1 3], а для терма "*превосходный*" — [7 9 10 10].

Вид редактора функций принадлежности после внесенных изменений для второй входной переменной изображен на рис. 12.11.

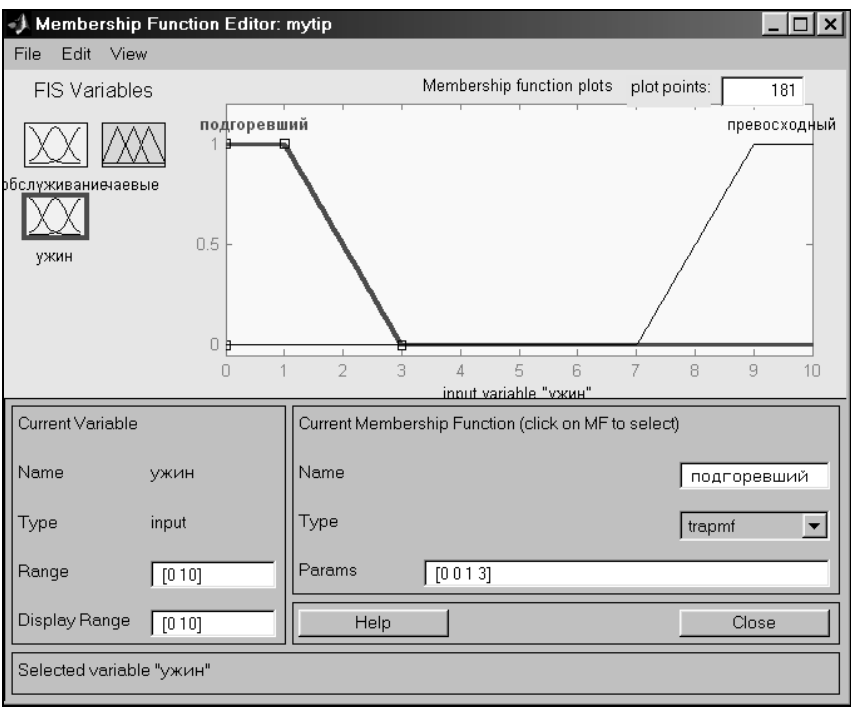


Рис. 12.11. Вид редактора функций принадлежности после изменения названия термов и типа их функций принадлежности для второй входной переменной "*ужин*"

Наконец, изменим названия термов и параметры функций принадлежности для выходной переменной "*чаевые*", оставив без изменения треугольный тип функций принадлежности, предложенный системой MATLAB. Для терма "*малые*" зададим параметры [0 5 10], а для терма "*средние*" — [10 15 20], для терма "*щедрые*" — [20 25 30]. Вид редактора функций принадлежности после сделанных изменений для выходной переменной "*чаевые*" изображен на рис. 12.12.

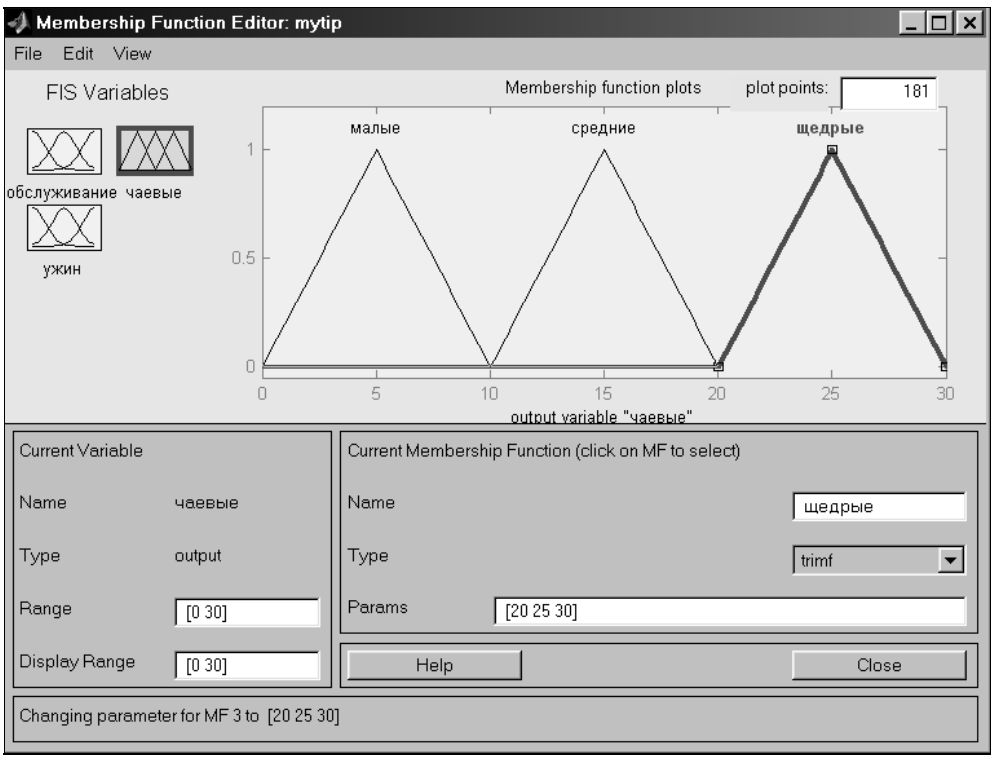


Рис. 12.12. Вид редактора функций принадлежности после изменения названия термов и типа их функций принадлежности для входной переменной "*обслуживание*"

6. Теперь настала очередь определить правила нечеткого вывода для разрабатываемой экспертной системы. Для этой цели следует воспользоваться редактором правил, который может быть вызван одним из следующих способов:

   - двойным щелчком на значке квадрата в центре с именем создаваемой системы нечеткого вывода (myfis);
   - командой меню **Edit>Rules...**;
   - нажатием клавиш <Ctrl>+<3>.

   Поскольку первоначально база правил нечеткого вывода пуста, то после вызова редактора правил центральное многострочное поле ввода не содержит

никаких правил. Для их определения следует использовать поля меню и переключатели в нижней части графического интерфейса редактора правил. Для задания первого правила следует оставить выделенные по умолчанию поле с именем терма "*плохое*" для первой входной переменной, поле с именем терма "*подгоревший*" для второй входной переменной и поле с именем терма "*малые*" для выходной переменной. Далее следует переключатель **Connection** поставить в положение **or** (логическое ИЛИ) и нажать на кнопку **Add rule**. После этого первое правило с символами кириллицы отобразится в верхнем окне.

Аналогичным образом задается второе правило, для которого следует выделить имена термов "*хорошее*", "*none*" и "*средние*", и третье правило с именами термов "*отличное*", "*превосходный*" и "*щедрые*" для соответствующих переменных. Вид редактора правил после их определения для разрабатываемой экспертной системы изображен на рис. 12.13.

Заметим, что в поле ввода **Weight** отображается вес каждого правила, который можно изменять в пределах интервала [0, 1] (оставим без изменения его значение по умолчанию, равное 1 для всех правил). Этот же вес правил записывается в круглых скобках в окне правил после каждого из правил нечеткого вывода.

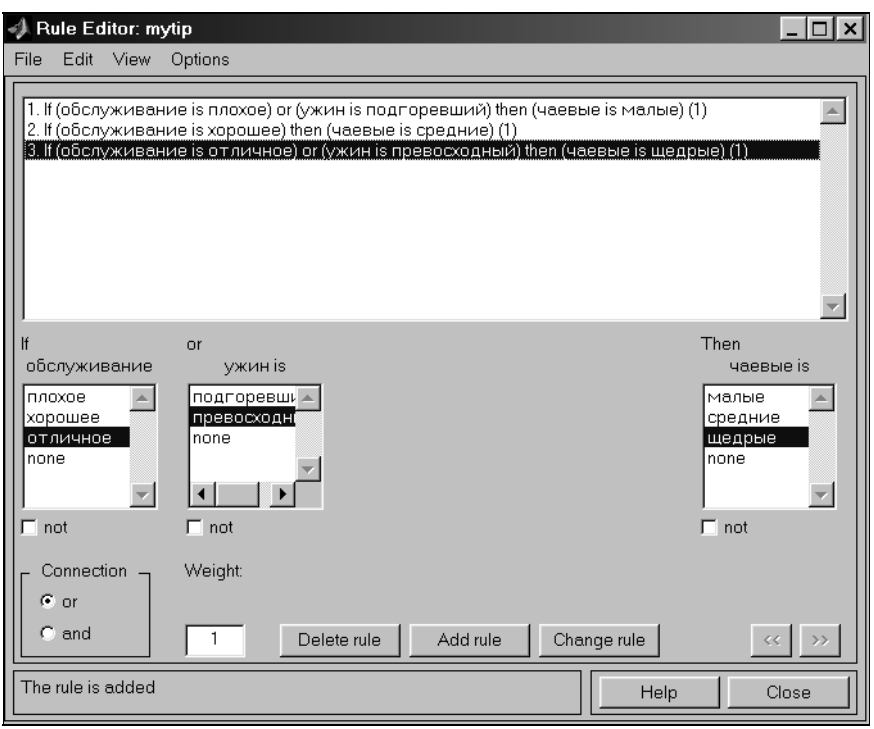


Рис. 12.13. Вид редактора правил нечеткого вывода после их определения

7. После задания правил нечеткого вывода оказывается возможным получить результат нечеткого вывода (значение выходной переменной) для конкретных значений входных переменных. С этой целью необходимо открыть программу просмотра правил одним из следующих способов:

   - командой меню **View>Rules** редактора FIS;
   - командой меню **View>Rules** редактора функций принадлежности;
   - командой меню **View>Rules** редактора правил;
   - нажатием клавиш <Ctrl>+<5>.

   После вызова программы просмотра правил для нашей системы нечеткого вывода по умолчанию для входных переменных предложены средние значения из интервала их допустимых значений (значения [5 5] в поле ввода **Input**). Это означает, что посетитель ресторана оценивает качество обслуживания в 5 баллов и качество ужина также в 5 баллов. Этим значениям входных переменных соответствует значение чаевых в 15%, которое отображается выше прямоугольников правил в правой части окна программы просмотра.

   Изменим значения входных переменных для другого случая, которому соответствует качество обслуживания в 0 баллов ("хуже некуда") и качество ужина в 10 баллов ("лучше не бывает"). Для этого курсор мыши переместим в поле ввода **Input** и введем соответствующие значения входных переменных: [0 10]. Система MATLAB оставит значение чаевых без изменения (15%), однако на диаграмме правил можно заметить результаты выполненных изменений (рис. 12.14).

   Поскольку процесс нечеткого моделирования предполагает анализ результатов нечеткого вывода при различных значениях входных переменных с целью установления адекватности разработанной нечеткой модели (в данном случае — экспертной системы), рассмотрим и другие случаи. Предположим, что качество обслуживания оценивается в 10 баллов ("лучше не бывает"), а качество ужина — в 2 балла ("бывает и хуже, но реже"). Введем соответствующие значения переменных аналогичным способом. В этом случае разработанная система нечеткого вывода рекомендует нам оставить чаевые в размере 16.4%.

   Если же предположить, что качество обслуживания по прежнему отличное (10 баллов), а качество ужина несколько улучшилось и оценивается в 3 балла, то величина чаевых существенно изменится и станет равной 24.7%. Более того, дальнейшее увеличение качества ужина не оказывает изменения величины чаевых. В частности, для значений входных переменных [10 10] величина чаевых составит по-прежнему 24.7%. Если некоторым из посетителей такая экспертная система покажется неадекватной (в частности, для случая значений входных переменных [10 10] можно бы оставить максимальные чаевые в 30%), то разработанная система нечеткого вывода потребует модификации. Данная модификация может потребовать изменения существующих правил или добавления новых, а также изменения параметров функций принадлежности

входных и выходной переменных. Более тонкая настройка модели может быть связана с увеличением количества термов для каждой из входных и выходных переменных, что, в свою очередь, приведет к увеличению количества правил в системе нечеткого вывода и общему усложнению модели. Все это предлагается выполнить читателям самостоятельно в качестве упражнения.

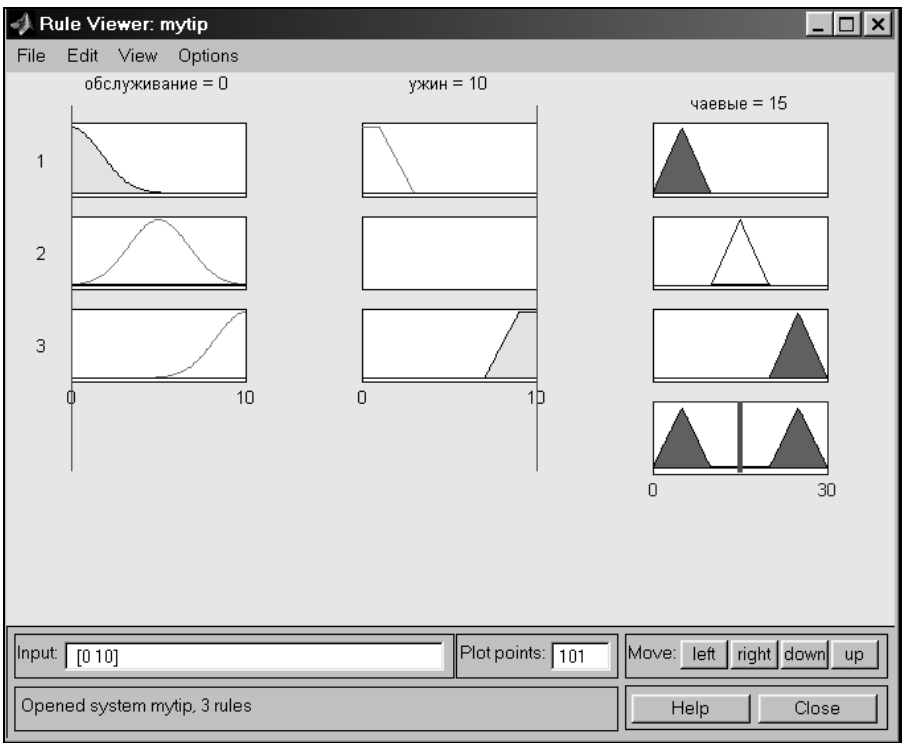


Рис. 12.14. Вид программы просмотра правил нечеткого вывода после изменения значений входных переменных на [0 10]

8. Для окончательного анализа разработанной нечеткой модели может оказаться полезной программа просмотра поверхности нечеткого вывода, которая может быть вызвана одним из следующих способов:

   - командой меню **View>Surface** редактора FIS;
   - командой меню **View>Surface** редактора функций принадлежности;
   - командой меню **View>Surface** редактора правил;
   - командой меню **View>Surface** программы просмотра правил;
   - нажатием клавиш <Ctrl>+<6>.

Графический интерфейс программы просмотра поверхности нечеткого вывода для разработанной нечеткой модели изображен на рис. 12.15.

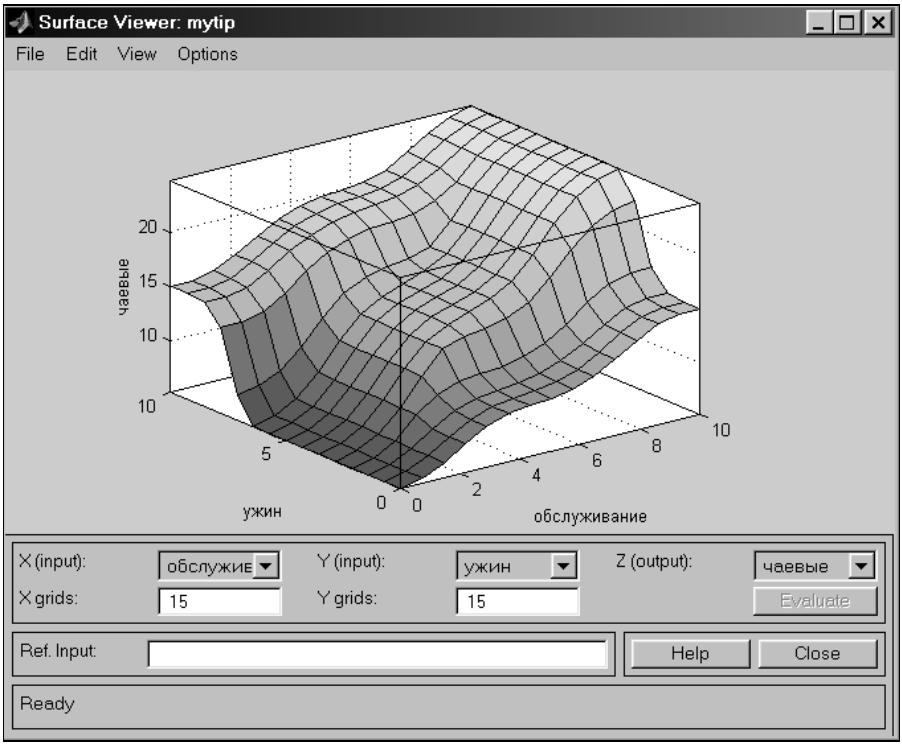


Рис. 12.15. Вид программы просмотра поверхности нечеткого вывода для разработанной нечеткой модели

Эта программа служит для общего анализа адекватности нечеткой модели, позволяя оценить влияние изменения значений входных нечетких переменных на значение одной из выходных нечетких переменных. В случае необходимости можно получить график зависимости выходной переменной от одной из входных переменных. Для этого необходимо выбрать нужную переменную в раскрывающемся списке **X (input)**, а в раскрывающемся списке **Y (input)** выбрать значение -none-. Полученный график зависимости изображен на рис. 12.16.

Полученный график зависимости соответствует среднему значению второй входной переменной *("качество ужина")* в 5 баллов. Это значение может быть изменено пользователем, для чего следует ввести нужное значение в поле ввода **Ref. Input:**. Заметим, что значение NaN для первой входной переменной соответствует ее изменению во всем интервале определения [0, 10].

Заканчивая рассмотрение процесса разработки простейшей системы нечеткого вывода в интерактивном режиме, следует заметить, что наиболее эффективным этот способ оказывается для сложных нечетких моделей с большим числом переменных и правил нечеткого вывода. В этом случае задание переменных и функций принадлежности их термов в графическом режиме, а также визуализация правил позволяют существенно уменьшить трудоемкость разработки нечеткой модели, снизить количество возможных ошибок и сократить общее время нечеткого моделирования.

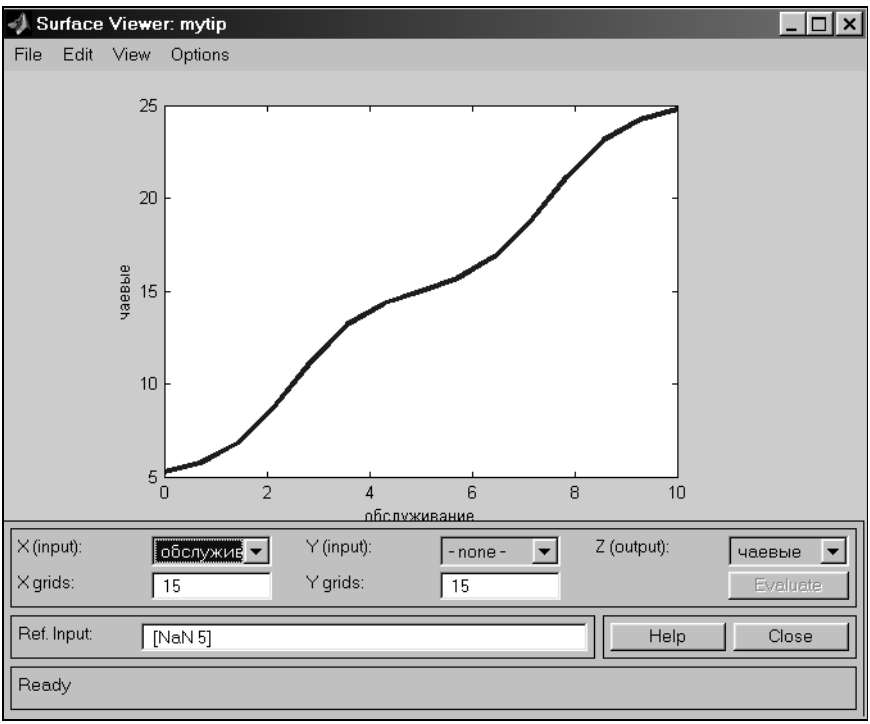


Рис. 12.16. График зависимости выходной переменной от первой из входных переменных для разработанной нечеткой модели

В то же время следует помнить, что количество переменных и правил в нечеткой модели, которые могут быть визуализированы, ограничено. В частности, если число входных переменных превышает 10, то их отображение в соответствующих графических редакторах происходит с искажениями.

Процесс разработки системы нечеткого вывода в режиме команд может дополнить, а в отдельных случаях, и заменить процесс разработки в интерактивном режиме, предоставляя пользователю полный контроль над всеми переменными рабочей области системы MATLAB.

12.3. Процесс разработки системы нечеткого вывода в режиме командной строки

Процесс разработки системы нечеткого вывода в режиме команд реализуется с помощью функций, входящих в состав пакета Fuzzy Logic Toolbox. В системе MATLAB реализованы следующие основные группы функций нечеткой логики:

- 11 встроенных функций принадлежности для термов нечетких переменных (табл. 12.2);
- функции управления структурой данных системы нечеткого вывода FIS (табл. 12.3);
- функции дополнительных методов и взаимодействия с пакетом Simulink (табл. 12.4).

Подробное описание функций пакета Fuzzy Logic Toolbox приводится в *приложении 3*.

Таблица 12.2. *Функции принадлежности пакета Fuzzy Logic Toolbox*

Функция	Назначение
`dsigmf`	Разность двух сигмоидальных функций принадлежности
`gauss2mf`	Функция принадлежности типа двухсторонней кривой Гаусса
`gaussmf`	Функции принадлежности типа кривой Гаусса
`gbellmf`	Обобщенная функция принадлежности типа колоколообразной кривой
`pimf`	*П*-образная функция принадлежности
`psigmf`	Произведение двух сигмоидальных функций принадлежности
`smf`	*S*-образная функция принадлежности
`sigmf`	Сигмоидальная функция принадлежности
`trapmf`	Трапециевидная функция принадлежности
`trimf`	Треугольная функция принадлежности
`zmf`	*Z*-образная функция принадлежности

Таблица 12.3. *Функции управления структурой данных системы нечеткого вывода*

Функция	Назначение
`addmf`	Добавляет функцию принадлежности в систему нечеткого вывода FIS
`addrule`	Добавляет правило в систему нечеткого вывода FIS

Таблица 12.3 *(окончание)*

Функция	Назначение
`addvar`	Добавляет переменную в систему нечеткого вывода FIS
`defuzz`	Дефаззификация функции принадлежности
`evalfis`	Выполняет нечеткий вывод в FIS
`evalmf`	Выполняет оценку функции принадлежности
`gensurf`	Генерирует поверхность вывода FIS
`getfis`	Выводит свойства нечеткой системы
`mf2mf`	Преобразует параметры двух функций принадлежности
`newfis`	Создает новую систему нечеткого вывода FIS
`parsrule`	Проверяет правильность правил вывода
`plotfis`	Отображает структуру входа/выхода системы нечеткого вывода FIS
`plotmf`	Изображает графики всех функций принадлежности соответствующей лингвистической переменной
`readfis`	Загружает систему нечеткого вывода FIS с диска
`rmmf`	Удаляет функцию принадлежности из системы нечеткого вывода FIS
`rmvar`	Удаляет переменную из системы нечеткого вывода FIS
`setfis`	Задает свойства нечеткой системы
`showfis`	Отображает обозначения FIS
`showrule`	Отображает правила системы нечеткого вывода FIS
`writefis`	Сохраняет систему нечеткого вывода FIS на диске

Таблица 12.4. *Функции дополнительных методов*

Функция	Назначение
`anfis`	Программа обучения системы нечеткого вывода FIS типа Сугено (только MEX)
`fcm`	Программа нахождения кластеров для алгоритма нечетких *c*-средних FCM
`fuzarith`	Выполнение операций нечеткой арифметики
`fuzblock`	Библиотека нечеткой логики пакета Simulink
`genfis1`	Генерирует матрицу системы нечеткого вывода FIS с использованием "жадного" алгоритма `grid`

Таблица 12.4 *(окончание)*

Функция	Назначение
`genfis2`	Генерирует матрицу системы нечеткого вывода FIS с использованием субтрактивной кластеризации (subtractive clustering)
`sffis`	*S*-функция нечеткого вывода для пакета Simulink
`subclust`	Программа нахождения кластеров для алгоритма субтрактивной кластеризации

При разработке системы нечеткого вывода в режиме команд необходимо знать, что нечеткая модель в рабочей области системы MATLAB представляется в форме так называемой *структуры*.

Примечание

Понятие структуры в общем случае является разновидностью типа данных языка программирования типа C/C++, для записи которого используется специальное ключевое слово `struct`.

В системе MATLAB структура представляет собой тип данных, которая состоит из *полей* и возможно других структур. В свою очередь, поле представляет собой простейший тип данных (число или строка). В рабочей области MATLAB структура представляется в форме массива и обозначается ключевым словом `struct array`.

Каждая система нечеткого вывода в MATLAB представляется в форме специальной структуры, которая может быть графически представлена с использованием нотации языка UML в форме диаграммы классов (рис. 12.17).

Для представления собственно правил в рабочей области MATLAB используется специальный цифровой формат. При этом все правила системы нечеткого вывода представляются в форме матрицы, содержащей одну или несколько строк. Каждой строке соответствует отдельное правило. Если система нечеткого вывода имеет m входных переменных и n выходных переменных, то соответствующая матрица должна иметь в точности $m + n + 2$ столбцов.

Первые m столбцов относятся к входным переменным системы. При этом номер столбца должен соответствовать номеру терма для конкретной входной переменной. Следующие n столбцов относятся к выходным переменным системы вывода. При этом каждый столбец также имеет номер, который должен соответствовать номеру функции принадлежности для выходной переменной.

Столбец с номером $m + n + 1$ содержит вес, с которым применяется данное правило. Вес может принимать любое значение между нулем и единицей. В общем случае целесообразно задавать вес равным 1 (задается по умолчанию).

Столбец с номером $m + n + 2$ содержит число 1, если для подусловий данного правила используется нечеткий оператор `AND` (нечеткое И). Этот столбец содер-

жит число 2, если для подусловий данного правила используется нечеткий оператор OR (нечеткое ИЛИ).

Так, например, для демонстрационного примера "*Чаевые в ресторане*", который хранится во внешнем файле с именем tipper.fis и может быть загружен в рабочую область командой `readfis('tipper')`, правила в рабочей области MATLAB будут представлены в форме следующей матрицы:

```
[ 1 1 1 1 2
  2 0 2 1 1
  3 2 3 1 2 ]
```

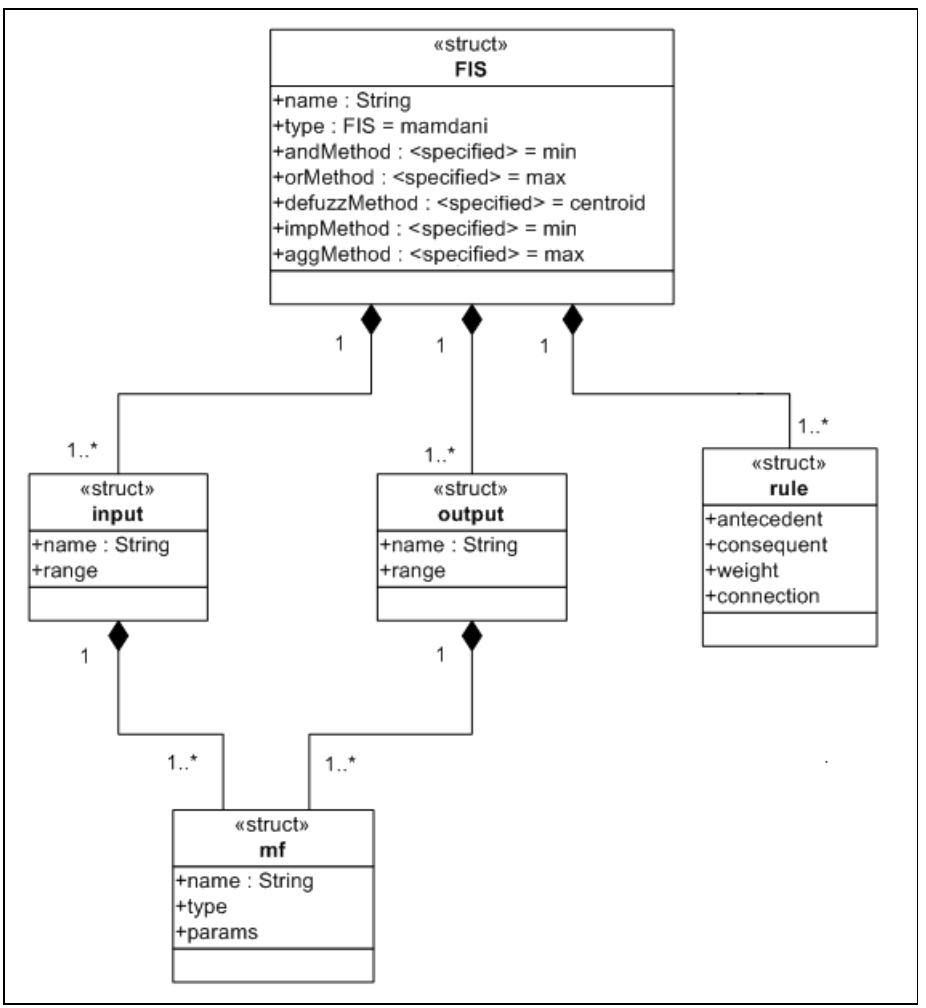


Рис. 12.17. Диаграмма классов структур системы нечеткого вывода MATLAB в нотации языка UML

Для данного примера первая строка этой матрицы может быть интерпретирована в форме следующего правила нечеткого вывода: "Если первая входная переменная принимает значение первого терма (функция принадлежности с номером 1) ИЛИ вторая входная переменная принимает значение первого терма (функция принадлежности с номером 1), то выходная переменная принимает значение первого терма (функция принадлежности с номером 1)". При этом вес данного правила равен 1. Если на месте некоторого терма расположена цифра 0, то это означает, что соответствующая входная переменная не используется в правиле нечеткого вывода с номером строки данной матрицы.

Таким образом, процесс разработки системы нечеткого вывода в командном режиме представляет собой последовательность функций, которые в фиксированном порядке определяют все элементы (поля и вложенные структуры) исходной структуры системы MATLAB. Ниже приводится последовательность команд, которые позволяют разработать систему нечеткого вывода для примера 12.1. При вводе соответствующих функций в окно команд результаты их выполнения можно контролировать либо с помощью окна просмотра рабочей области, либо непосредственно в окне команд, для чего имена всех функций следует набирать без завершающей точки с запятой. Напомним, что последний символ блокирует или запрещает отображение информации в окне команд после выполнения соответствующих функций.

Примечание

При разработке системы нечеткого вывода в режиме команд рекомендуется отказаться от символов кириллицы в записи имен переменных и их термов. В противном случае могут возникнуть проблемы с корректным функционированием системы MATLAB. Следует также заметить, что для задания полей исходной структуры с именем `a` и вложенных в нее структур в качестве разделителя используется принятый в некоторых языках программирования символ точки.

Листинг 12.1. Последовательность функций разработки системы нечеткого вывода для примера 12.1 "*Чаевые в ресторане*"

```
a=newfis('mytip');
a.input(1).name='service';
a.input(1).range=[0 10];
a.input(1).mf(1).name='poor';
a.input(1).mf(1).type='gaussmf';
a.input(1).mf(1).params=[1.5 0];
a.input(1).mf(2).name='good';
a.input(1).mf(2).type='gaussmf';
a.input(1).mf(2).params=[1.5 5];
a.input(1).mf(3).name='excellent';
a.input(1).mf(3).type='gaussmf';
```

```
a.input(1).mf(3).params=[1.5 10];
a.input(2).name='food';
a.input(2).range=[0 10];
a.input(2).mf(1).name='rancid';
a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-2 0 1 3];
a.input(2).mf(2).name='delicious';
a.input(2).mf(2).type='trapmf';
a.input(2).mf(2).params=[7 9 10 12];
a.output(1).name='tip';
a.output(1).range=[0 30];
a.output(1).mf(1).name='cheap';
a.output(1).mf(1).type='trimf';
a.output(1).mf(1).params=[0 5 10];
a.output(1).mf(2).name='average';
a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[10 15 20];
a.output(1).mf(3).name='generous';
a.output(1).mf(3).type='trimf';
a.output(1).mf(3).params=[20 25 30];
a.rule(1).antecedent=[1 1];
a.rule(1).consequent=[1];
a.rule(1).weight=1;
a.rule(1).connection=2;
a.rule(2).antecedent=[2 0];
a.rule(2).consequent=[2];
a.rule(2).weight=1;
a.rule(2).connection=1;
a.rule(3).antecedent=[3 2];
a.rule(3).consequent=[3];
a.rule(3).weight=1;
a.rule(3).connection=2;
showfis(a)
```

В результате выполнения этой последовательности функций в окне команд отобразится информация о созданной структуре нечеткого вывода (рис. 12.18).

Дальнейший анализ разработанной нечеткой модели можно выполнить либо с использованием графических средств программы просмотра правил описанным ранее способом, либо непосредственно с помощью функции командной строки

evalfis. Так, например, если после задания одним из рассмотренных выше способов структуры нечеткого вывода с именем a ввести команду evalfis([10 0], a), то в окне команд получим результат нечеткого вывода для значения первой входной переменной 10 и второй входной переменной 0 в форме: ans = 15.0000.

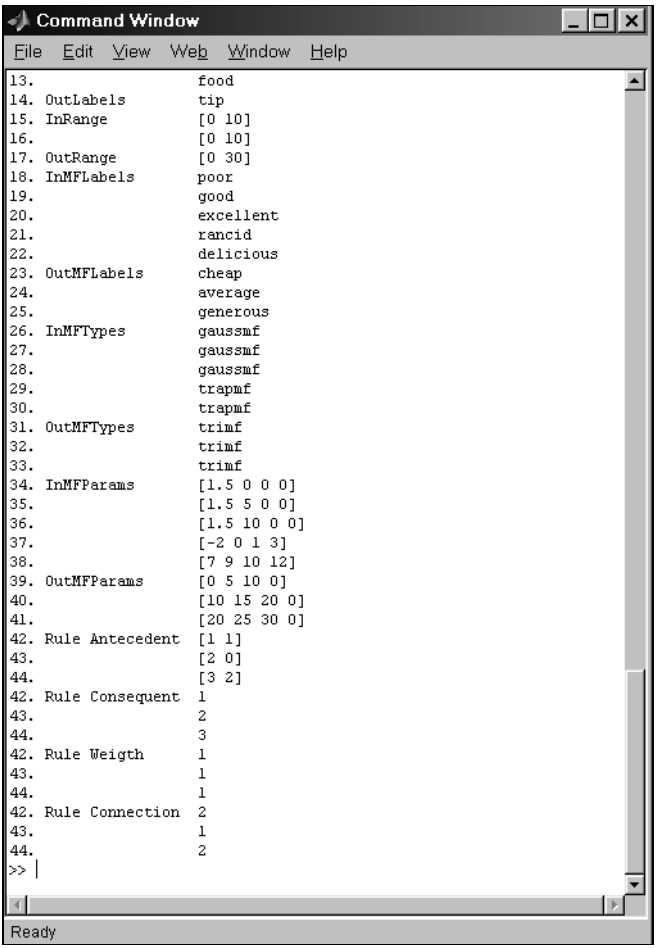


Рис. 12.18. Результат отображения созданной структуры нечеткого вывода в режиме команд

Примечание

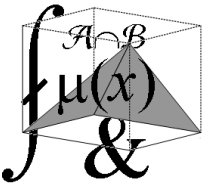
Чтобы исключить ввод каждой из функций рассмотренной выше последовательности команд, можно скопировать приведенный листинг в буфер обмена и сохранить его во внешнем файле в папке C:\MATLAB6p1\toolbox\fuzzy или в

папке C:\MATLAB6p1\work (если система MATLAB установлена по умолчанию на диске C:) с произвольным именем и расширением m. Для этой цели можно использовать встроенный редактор m-файлов системы MATLAB, вызываемый командой `edit`, либо любой другой ASCII-редактор. Заметим, что текстовый процессор MS Word из пакета MS Office не вполне подходит для этой цели. Если после сохранения созданного подобным образом m-файла на диске ввести его имя в окне команд, то получим результат, аналогичный последовательному вводу этих команд с клавиатуры. В данном случае мы познакомились с одним из простейших приемов программирования в системе MATLAB, особенности которого будут рассмотрены в *главе 14*.

В данной главе изложены лишь основные приемы разработки систем нечеткого вывода в среде MATLAB. В последующих главах рассматриваются другие особенности и дополнительные возможности, которые предоставляет пользователям система MATLAB. Что касается выбора наиболее удобного режима разработки и исследования нечетких моделей, то оставим этот вопрос на усмотрение читателей.

В заключение следует заметить, что режим команд служил основным в первых версиях системы MATLAB. В последних ее версиях все большее число задач может быть решено с помощью соответствующих графических средств, что представляется более удобным с точки зрения наглядности и трудоемкости.

Глава 13



Нечеткая кластеризация в Fuzzy Logic Toolbox

13.1. Общая характеристика задач кластерного анализа

Термином *кластерный анализ* принято обозначать совокупность методов, подходов и процедур, разработанных для решения проблемы формирования однородных классов в произвольной проблемной области.

Необходимость анализа больших объемов объективной и субъективной информации, связанных с неформализуемыми и плохо формализуемыми задачами различной физической природы, потребовала интенсивного развития новых научных направлений, среди которых важную роль играют прикладная статистика и методы анализа данных.

Методология применения методов *прикладной статистики* основывается на предположении о вероятностной интерпретации анализируемой информации и получении в результате применения этих методов закономерностей, имеющих стохастический характер.

Методы анализа данных, составной частью которых являются методы кластерного анализа, напротив, не используют априорных предположений о вероятностной природе исходной информации и руководствуются только эвристическими соображениями о характере и особенностях исследуемой совокупности объектов. Разработанные в рамках данного направления методы и алгоритмы приобретают в последнее время новое содержание в связи с исследованиями по теории возможностей. В основе данной теории лежит нечетко-возможностная интерпретация неопределенности, что в значительной степени согласуется с исходными установками методологии анализа данных.

Кластерный анализ (или автоматическая классификация, распознавание образов без учителя, численная таксономия, кластер-анализ) занимает одно из центральных мест среди методов анализа данных и представляет собой совокупность подходов, методов и алгоритмов, предназначенных для нахождения некоторого разбиения исследуемой совокупности объектов на подмножества относительно сходных, похожих между собой объектов. При этом исходным допущением для выделения таких подмножеств, получивших специальное название *кластеров*,

которые иногда называют также таксонами или просто классами, служит лишь неформальное предположение о том, что объекты, относимые к одному кластеру, должны иметь большее сходство между собой, чем с объектами из других кластеров.

Таким образом, выявление или нахождение кластеров в множестве данных исследуемой совокупности должно удовлетворять следующим требованиям:

- каждый кластер должен представлять собой концептуально однородную категорию и содержать похожие объекты с близкими значениями свойств или признаков;
- совокупность всех кластеров должна быть исчерпывающей, т. е. охватывать все объекты исследуемой совокупности;
- кластеры должны быть взаимно исключающими, т. е. ни один из объектов исследуемой совокупности не должен одновременно принадлежать двум различным кластерам.

Формально под задачей кластерного анализа заданного множества объектов понимается задача нахождения некоторого теоретико-множественного разбиения (группировки, покрытия) этого исходного множества объектов на непересекающиеся подмножества таким образом, чтобы элементы, относимые к одному подмножеству, отличались между собой в значительно меньшей степени, чем элементы из разных подмножеств. Обладающие подобным свойством подмножества также называются *кластерами*.

Возможность использования различных подходов к формальному определению кластеров послужила исходной причиной разработки большого многообразия методов и алгоритмов кластеризации. Методы и алгоритмы кластерного анализа как инструмент предварительного или разведочного анализа данных незаменимы при поиске закономерностей в больших наборах многомерных данных, таких как хранилища данных. При этом проблема кластер-анализа приобретает самостоятельное значение в контексте интеллектуального анализа данных (Data Mining).

13.2. Задача нечеткой кластеризации и алгоритм ее решения

Концептуальная взаимосвязь между кластерным анализом и теорией нечетких множеств основана на том обстоятельстве, что при решении задач структуризации сложных систем большинство формируемых классов объектов размыты по своей природе. Эта размытость состоит в том, что переход от принадлежности к непринадлежности элементов к данным классам скорее постепенен, чем скачкообразен. Поэтому наиболее адекватный ответ в подобного рода случаях следует искать не на вопрос: "Принадлежит ли рассматриваемый элемент тому или иному классу или нет?", а на вопрос: "В какой степени данный элемент принадлежит рассматриваемому классу?".

Требование нахождения однозначной кластеризации элементов исследуемой проблемной области является достаточно грубым и жестким, особенно при решении плохо или слабо структурируемых задач системного анализа. Методы нечеткой кластеризации ослабляют это требование. Ослабление требования осуществляется за счет введения в рассмотрение нечетких кластеров и соответствующих им функций принадлежности, принимающих значения из интервала [0, 1].

Таким образом, в общем случае задачей нечеткой кластеризации является нахождение нечеткого разбиения или нечеткого покрытия множества элементов исследуемой совокупности, которые образуют структуру нечетких кластеров, присутствующих в рассматриваемых данных. Эта задача сводится к нахождению степеней принадлежности элементов универсума искомым нечетким кластерам, которые в совокупности и определяют нечеткое разбиение или нечеткое покрытие исходного множества рассматриваемых элементов. Ниже приводится формальная постановка задачи нечеткой кластеризации и описание одного из наиболее конструктивных алгоритмов ее решения.

Общая формальная постановка задачи нечеткого кластерного анализа

Пусть исследуемая совокупность данных представляет собой конечное множество элементов $A=\{a_1, a_2,..., a_n\}$, которое получило название *множество объектов кластеризации*. В рассмотрение также вводится конечное множество признаков или атрибутов $P=\{p_1, p_2,..., p_q\}$, каждый из которых количественно представляет некоторое свойство или характеристику элементов рассматриваемой проблемной области. При этом натуральное n определяет общее количество объектов данных, а натуральное q — общее количество измеримых признаков объектов.

Далее предполагается, что для каждого из объектов кластеризации некоторым образом измерены все признаки множества P в некоторой количественной шкале. Тем самым каждому из элементов $a_i \in A$ поставлен в соответствие некоторый вектор $\boldsymbol{x_i} = (x_1^i, x_2^i,..., x_q^i)$, где x_j^i — количественное значение признака $p_j \in P$ для объекта данных $a_i \in A$. Для определенности будем полагать, что все x_j^i принимают некоторые действительные значения, т. е. $x_j^i \in \mathbb{R}$.

Вообще говоря, проблема измерения свойств или признаков у объектов данных является нетривиальной и имеет самостоятельное значение. В частности, процесс измерения свойств может быть реализован в различных шкалах, каждая из которых характеризуется допустимым преобразованием данных. В связи с этим для уточнения особенностей процесса измерений различают несколько *типов шкал измерений*. Ниже дается их краткая характеристика.

- *Шкала наименований* или номинальная шкала. Является наиболее простой из всех шкал измерений, поскольку может быть использована для установления отношения эквивалентности элементов относительно рассматриваемого признака. В данном случае в процессе измерения некоторого признака объекту ставится в соответствие некоторый символ или номер, который лишь отлича-

ет одно значение признака от другого. Допустимым преобразованием в шкалах наименований, которое не искажает результаты измерения признаков в этой шкале, является биективное или взаимно однозначное отображение между двумя множествами значений признаков. Бинарная шкала, которая состоит из двух элементов, обозначаемых произвольными символами, например: {0, 1} или {+, –}, удобно считать частным случаем шкалы наименований. Примеры признаков, измеряемых в шкалах наименований, — пол человека, марка автомобиля, название улиц, городов и других объектов.

- *Порядковая шкала* или шкала порядка. В дополнение к различию элементов по значениям признаков позволяет установить отношение порядка элементов относительно рассматриваемого признака. В этом случае в процессе измерения признака объекту ставится в соответствие, как правило, некоторое натуральное или целое число, которое может быть интерпретировано как значение признака в баллах. Допустимым преобразованием в шкалах порядка, которое не искажает результаты измерения признаков в этой шкале, является произвольное монотонно возрастающее отображение или функция между двумя множествами значений признаков. Примеры признаков, измеряемых в порядковых шкалах, — баллы или оценки на экзаменах, баллы за выступления в некоторых видах спорта.
- *Интервальная шкала* или шкала интервалов. В дополнение к порядку элементов по значениям признаков позволяет установить равенство интервалов значений рассматриваемого признака. В этом случае в процессе измерения признака объекту ставится в соответствие, как правило, некоторое действительное число, равное значению этого признака. Допустимым преобразованием в шкалах интервалов является произвольная линейная возрастающая функция между двумя множествами значений признаков. Характерным свойством этой шкалы является отсутствие абсолютного нуля. Пример признака, измеряемого в интервальной шкале, — температура в шкалах Цельсия и Фаренгейта.
- *Шкала отношений.* В дополнение к равенству интервалов элементов по значениям признаков позволяет установить равенство отношений значений рассматриваемого признака. В этом случае в процессе измерения признака объекту ставится в соответствие также некоторое действительное число, равное значению этого признака. Допустимым преобразованием в шкалах отношений является произвольная линейная возрастающая функция, проходящая через нуль. Характерным свойством этой шкалы является наличие абсолютного нуля. Примеры признаков, измеряемых в шкале отношений, — расстояние в метрах и футах, масса в килограммах и фунтах, скорость в км/ч и узлах.

Возвращаясь к измерению признаков у объектов кластеризации, само множество признаков следует выбирать таким образом, чтобы все x_j^i были измерены в шкале отношений или шкале интервалов. Именно в этом случае результаты нечеткой кластеризации имеют содержательную интерпретацию, адекватную проблеме нахождения нечетких кластеров.

Векторы значений признаков $\boldsymbol{x_i} = (x_1^i, x_2^i, \ldots, x_q^i)$ удобно представить в виде так называемой *матрицы данных* **D** размерности $(n \times q)$, каждая строка которой равна значению вектора $\boldsymbol{x_i}$.

Задача нечеткого кластерного анализа формулируется следующим образом: на основе исходных данных **D** определить такое нечеткое разбиение $\mathfrak{R}(\mathcal{A}) = \{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ (4.39)—(4.40) или нечеткое покрытие $\mathfrak{I}(\mathcal{A}) = \{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ (4.38) множества $\mathcal{A}=A$ на заданное число c нечетких кластеров $\mathcal{A}_k$ $(k \in \{2, \ldots, c\})$, которое доставляет экстремум некоторой целевой функции $f(\mathfrak{R}(\mathcal{A}))$ среди всех нечетких разбиений или экстремум целевой функции $f(\mathfrak{I}(\mathcal{A}))$ среди всех нечетких покрытий.

Сформулированная здесь задача нечеткого кластерного анализа является слишком общей. Для ее решения требуется дополнительно уточнить вид целевой функции и тип искомых нечетких кластеров (поиск нечеткого разбиения или покрытия).

Уточненная постановка задачи нечеткой кластеризации

Один из вариантов конкретизации задачи нечеткого кластерного анализа, для решения которой может быть использована специальная функция `fcm` системы MATLAB, основан на алгоритме ее решения методом нечетких c-средних.

Для уточнения вида целевой функции $f(\mathfrak{I}(\mathcal{A}))$ в рассмотрение вводятся некоторые дополнительные понятия. Прежде всего предполагается, что искомые нечеткие кластеры представляют собой нечеткие множества $\mathcal{A}_k$, образующие нечеткое покрытие исходного множества объектов кластеризации $\mathcal{A}=A$, для которого условие (4.38) принимает следующий вид:

$$\sum_{k=1}^{c} \mu_{A_k}(a_i) = 1 \quad (\forall a_i \in A), \tag{13.1}$$

где c — общее количество нечетких кластеров $\mathcal{A}_k$ $(k \in \{2, \ldots, c\})$, которое считается предварительно заданным $(c \in \mathbb{N}$ и $c > 1)$.

Примечание

Необходимость условия (13.1) обуславливается тем обстоятельством, что искомое нечеткое покрытие должно "покрывать" обычное не нечеткое множество объектов кластеризации A, которое в то же время является нечетким множеством $\mathcal{A}$, для которого значения функций принадлежности каждого из элементов равны 1. Иногда нечеткое покрытие, которое удовлетворяет условию (13.1), называют нечетким разбиением в смысле Заде.

Далее для каждого нечеткого кластера вводятся в рассмотрение так называемые *типичные представители* или *центры* v_k искомых нечетких кластеров $\mathcal{A}_k$

($k\in\{2,\ldots, c\}$), которые рассчитываются для каждого из нечетких кластеров и по каждому из признаков по следующей формуле:

$$v_j^k = \frac{\sum_{i=1}^{n}(\mu_{A_k}(a_i))^m \cdot x_j^i}{\sum_{i=1}^{n}(\mu_{A_k}(a_i))^m} \quad (\forall k\in\{2,\ldots, c\}, \forall p_j\in P), \tag{13.2}$$

где m — некоторый параметр, называемый *экспоненциальным весом* и равный некоторому действительному числу ($m>1$). Каждый из центров кластеров представляет собой вектор $\boldsymbol{v}_k=(v^k_1, v^k_2,\ldots, v^k_q)$ в некотором q-мерном нормированном пространстве, изоморфном $\mathbb{R}^q$, т. е. $v_j^k\in\mathbb{R}^q$, если все признаки измерены в шкале отношений.

Наконец, в качестве целевой функции будем рассматривать сумму квадратов взвешенных отклонений координат объектов кластеризации от центров искомых нечетких кластеров:

$$f(A_k, v_j^k) = \sum_{i=1}^{n}\sum_{k=1}^{c}(\mu_{A_k}(a_i))^m \sum_{j=1}^{q}(x_j^i - v_j^k)^2, \tag{13.3}$$

где m — экспоненциальный вес нечеткой кластеризации ($m\in\mathbb{R}$, $m>1$), значение которого задается в зависимости от количества элементов (мощности) множества A. Чем больше элементов содержит множество A, тем меньшее значение выбирается для m.

Задача нечеткой кластеризации. *Задача нечеткой кластеризации* может быть сформулирована следующим образом: для заданных матрицы данных **D**, количества нечетких кластеров c ($c\in\mathbb{N}$ и $c>1$), параметра m определить матрицу **U** значений функций принадлежности объектов кластеризации $a_i\in A$ нечетким кластерам $\mathcal{A}_k$ ($k\in\{2,\ldots, c\}$), которые доставляют *минимум* целевой функции (13.3) и удовлетворяют ограничениям (13.1), (13.2), а также дополнительным ограничениям (13.4) и (13.5):

$$\sum_{i=1}^{n}\mu_{A_k}(a_i) > 0 \quad (\forall k\in\{2,\ldots, c\}); \tag{13.4}$$

$$\mu_{A_k}(a_i) \geq 0 \quad (\forall k\in\{2,\ldots, c\}, \forall a_i\in A). \tag{13.5}$$

Условие (13.4) исключает появление пустых нечетких кластеров в искомой нечеткой кластеризации. Последнее условие (13.5) имеет чисто формальный характер, поскольку непосредственно следует из определения функции принадлежности нечетких множеств. В этом случае минимизация целевой функции (13.3) минимизирует отклонение всех объектов кластеризации от центров нечетких кластеров пропорционально значениям функций принадлежности этих объектов соответствующим нечетким кластерам.

Поскольку целевая функция (13.2) не является выпуклой, а ограничения (13.1), (13.2), (13.4), (13.5) в своей совокупности формируют невыпуклое множество допустимых альтернатив, то в общем случае задача нечеткой кластеризации относится к многоэкстремальным задачам нелинейного программирования.

Достоинством постановки задачи нечеткой кластеризации в виде (13.1)—(13.5) является естественная интерпретация как искомых нечетких кластеров, определяемых функциями принадлежности (13.5), так и их типичных представителей или центров (13.2), которые также определяются в результате решения поставленной задачи.

Недостатком данной постановки задачи нечеткой кластеризации является необходимость априорного задания общего числа нечетких кластеров c ($c \in \mathbb{N}$ и $c>1$), которое в отдельных случаях может быть неизвестно. Это обстоятельство может потребовать привлечения дополнительных процедур для его определения либо решения поставленной задачи для нескольких значений c с последующим выбором наиболее адекватного результата нечеткой кластеризации.

Алгоритм решения задачи нечеткой кластеризации методом нечетких *c*-средних

Основные идеи алгоритма для решения сформулированной задачи нечеткой кластеризации были предложены Дж. К. Данном (J. C. Dunn) в 1974 г. Этот алгоритм первоначально получил название нечеткого алгоритма ISODATA (fuzzy ISODATA или F-ISODATA). В 1980 г. Дж. К. Бездек (J. C. Bezdek) теоретически доказал сходимость этого алгоритма. Позже в 1981 г. Дж. Бездек обобщил алгоритм FISODATA на случай произвольных нечетких многообразий и предложил для этого алгоритма название нечетких c-средних (FCM, Fuzzy C-Means). Именно под таким названием алгоритм сейчас наиболее известен и реализован в системе MATLAB.

Алгоритм FCM для решения задачи нечеткой кластеризации в виде (13.1)—(13.5) имеет итеративный характер последовательного улучшения некоторого исходного нечеткого разбиения $\Re(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$, которое задается пользователем или формируется автоматически по некоторому эвристическому правилу. На каждой из итераций рекуррентно пересчитываются значения функций принадлежности нечетких кластеров и их типичные представители.

Алгоритм FCM закончит работу в случае, когда произойдет выполнение заданного априори некоторого конечного числа итераций, либо когда минимальная абсолютная разность между значениями функций принадлежности на двух последовательных итерациях не станет меньше некоторого априори заданного значения.

Формально алгоритм FCM определяется в форме итеративного выполнения следующей последовательности шагов:

1. Предварительно необходимо задать следующие значения: количество искомых нечетких кластеров c ($c \in \mathbb{N}$ и $c>1$), максимальное количество итераций алгорит-

ма s ($s \in \mathbb{N}$), параметр сходимости алгоритма ε ($\varepsilon \in \mathbb{R}_+$), а также экспоненциальный вес расчета целевой функции и центров кластеров m (как правило, $m=2$). В качестве *текущего* нечеткого разбиения на первой итерации алгоритма для матрицы данных $\mathbf{D}$ задать некоторое исходное нечеткое разбиение $\mathfrak{R}(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ на c непустых нечетких кластеров, которые описываются совокупностью функций принадлежности $\mu_k(a_i)$ ($\forall k \in \{2,\ldots, c\}$, $\forall a_i \in A$).

2. Для исходного текущего нечеткого разбиения $\mathfrak{R}(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ по формуле (13.2) рассчитать центры нечетких кластеров v_j^k ($\forall k \in \{2,\ldots, c\}$, $\forall p_j \in P$) и значение целевой функции $f(\mathcal{A}_k, v_j^k)$ по формуле (13.3). Количество выполненных итераций положить равным 1.

3. Сформировать *новое* нечеткое разбиение $\mathfrak{R}'(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ исходного множества объектов кластеризации A на c непустых нечетких кластеров, характеризуемых совокупностью функций принадлежности $\mu'_k(a_i)$ ($\forall k \in \{2,\ldots, c\}$, $\forall a_i \in A$), которые определяются по формуле:

$$\mu'_{A_k}(a_i) = \left(\sum_{l=1}^{c} \left(\frac{(\sum_{j=1}^{q} (x_j^i - v_j^k)^2)^{\frac{1}{2}}}{(\sum_{j=1}^{q} (x_j^i - v_j^l)^2)^{\frac{1}{2}}} \right)^{\frac{2}{m-1}} \right)^{-1} \quad (\forall k \in \{2,\ldots, c\}, \forall a_i \in A). \tag{13.6}$$

4. При этом если для некоторого $k \in \{2,\ldots, c\}$ и некоторого $a_i \in A$ значение $\sum_{j=1}^{q} (x_j^i - v_j^k)^2 = 0$, то для соответствующего нечеткого кластера $\mathcal{A}_k$ полагаем $\mu'_k(a_i)=1$, а для остальных $\mathcal{A}_l$ ($\forall l \in \{2,\ldots, c\}$, $l \neq k$) полагаем $\mu'_l(a_i)=0$. Если же таких $k \in \{2,\ldots, c\}$ для некоторого $a_i \in A$ окажется несколько, т. е. для них значение $\sum_{j=1}^{q} (x_j^i - v_j^k)^2 = 0$, то эвристически для меньшего из k полагаем $\mu'_k(a_i)=1$, а для остальных $l \in \{2,\ldots, c\}$, $l \neq k$ полагаем $\mu'_l(a_i)=0$.

5. Для нового нечеткого разбиения $\mathfrak{R}'(\mathcal{A})=\{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ по формуле (13.2) рассчитать центры нечетких кластеров v_j^k ($\forall k \in \{2,\ldots, c\}$, $\forall p_j \in P$) и значение целевой функции $f'(\mathcal{A}_k, v_j^k)$ по формуле (13.3).

6. Если количество выполненных итераций превышает заданное число s или же модуль разности $|f(\mathcal{A}_k, v_j^k) - f'(\mathcal{A}_k, v_j^k)| \leq \varepsilon$, т. е. не превышает значения параметра сходимости алгоритма ε, то в качестве искомого результата нечеткой кластеризации принять нечеткое разбиение $\mathfrak{R}'(\mathcal{A}) = \{\mathcal{A}_k \mid \mathcal{A}_k \subseteq \mathcal{A}\}$ и закончить выполнение алгоритма. В противном случае считать текущим нечетким разбиением $\mathfrak{R}(\mathcal{A}) = \mathfrak{R}'(\mathcal{A})$ и перейти на шаг 3 алгоритма, увеличив на 1 количество выполненных итераций.

Алгоритм FCM по своему характеру относится к приближенным алгоритмам поиска экстремума для целевой функции (13.3) при наличии ограничений: (13.1), (13.2), (13.4), (13.5). Поэтому в результате выполнения данного алгоритма определяется, вообще говоря, локально-оптимальное нечеткое разбиение $\Re^*(\mathcal{A})$, которое описывается совокупностью функций принадлежности $\mu_k(a_i)$ ($\forall k \in \{2,\ldots, c\}$, $\forall a_i \in A$), а также центры или типичные представители каждого из нечетких кластеров v_j^k ($\forall k \in \{2,\ldots, c\}$, $\forall p_j \in P$).

Примечание

Иногда возникает необходимость представить результаты нечеткой кластеризации в форме обычного не нечеткого разбиения. С этой целью можно использовать один из рассмотренных ранее методов дефаззификации *(см. главу 7)* для конечного множества значений функции принадлежности или любой другой метод приведения к четкости. Поскольку все функции принадлежности искомого нечеткого разбиения $\mu_k(a_i)$ ($\forall k \in \{2,\ldots, c\}$, $\forall a_i \in A$) нормированы условием (13.1), в качестве метода дефаззификации можно использовать метод максимального значения функции принадлежности.

Опыт решения прикладных задач нечеткой кластеризации показывает, что наиболее эффективный путь получения адекватных результатов заключается в многократном выполнении алгоритма FCM для различных исходных нечетких разбиений и, если не известно количество нечетких кластеров, для различных значений c ($c \in \mathbb{N}$ и $c>1$). Полученные результаты для одинаковых значений c сравниваются по значениям целевой функции полученных нечетких разбиений с целью принятия окончательного решения об искомой нечеткой кластеризации.

13.3. Средства решения задачи нечеткой кластеризации в пакете Fuzzy Logic Toolbox

В системе MATLAB для решения задачи нечеткой кластеризации на основе алгоритма FCM может быть использована специальная функция командной строки `fcm` или специальный графический интерфейс кластеризации, вызываемый функцией `findcluster`. Рассмотрим более подробно особенности применения этих средств.

Решение задачи нечеткой кластеризации в командном режиме

Функция командной строки `fcm` предназначена для решения задачи нечеткой кластеризации с использованием алгоритма FCM. Она может быть вызвана в одном из следующих форматов:

```
[center, U, obj_fcn] = fcm(data, cluster_n)
```

или

```
[center, U, obj_fcn] = fcm(data, cluster_n, options)
```

Входными аргументами этой функции являются:

- `data`: матрица исходных данных **D** кластеризации, *i*-строка которой представляет информацию об одном объекте нечеткой кластеризации $a_i \in A$ в форме вектора $\boldsymbol{x_i} = (x_1^i, x_2^i, \ldots, x_q^i)$, где x_j^i — количественное значение признака $p_j \in P$ для объекта данных $a_i \in A$;
- `cluster_n`: число искомых нечетких кластеров c ($c \in \mathbb{N}$ и $c>1$).

Выходными аргументами этой функции являются:

- `center`: матрица центров искомых нечетких кластеров v_j^k ($\forall k \in \{2,\ldots, c\}$, $\forall p_j \in P$), каждая строка которой представляет координаты центра одного из нечетких кластеров в форме вектора $\boldsymbol{v_k}$ ($\forall k \in \{2,\ldots, c\}$);
- `U`: матрица значений функций принадлежности искомого нечеткого разбиения $\mu_k(a_i)$ ($\forall k \in \{2,\ldots, c\}$, $\forall a_i \in A$);
- `obj_fcn`: значения целевой функции $f(\mathcal{A}_k, v_j^k)$ на каждой из итераций работы алгоритма.

Функция `fcm(data, cluster_n, options)` может быть вызвана с дополнительными аргументами `options`, которые предназначены для управления процессом нечеткой кластеризации, а также для изменения критерия остановки работы алгоритма и/или отображения информации на экране монитора.

Эти дополнительные аргументы имеют следующие значения:

- `options(1)`: экспоненциальный вес m для расчета матрицы нечеткого разбиения `U` (по умолчанию это значение равно $m=2$);
- `options(2)`: максимальное число итераций s (по умолчанию это значение равно $s=100$);
- `options(3)`: параметр сходимости алгоритма ε (по умолчанию это значение равно $\varepsilon=0.00001$);
- `options(4)`: информация о текущей итерации, отображаемая на экране монитора (по умолчанию это значение равно 1).

Если любое из значений дополнительных аргументов равно `NaN` (не число), то для этого аргумента используется значение по умолчанию. Функция `fcm` заканчивает свою работу, когда алгоритм FCM выполнит максимальное количество итераций s, или когда разность между значениями целевых функций на двух последовательных итерациях будет меньше заданного априори значения параметра сходимости алгоритма ε.

Примечание

Функция `fcm` реализована в виде m-файла и использует, в свою очередь, три другие функции: функцию `initfcm` для формирования матрицы исходного разбиения некоторым случайным образом, функцию `distfcm` для расчета матрицы расстояний между точками данных и центрами кластеров и функцию `stepfcm` для значений

целевой функции и функций принадлежности объектов нечетким кластерам на каждой итерации работы алгоритма FCM. Все эти функции также реализованы в виде m-файлов и находятся в папке C:\MATLAB6p1\toolbox\fuzzy\fuzzy.

П р и м е р 13.1. В качестве примера рассмотрим множество данных, содержащихся в системе MATLAB и использующихся в качестве тестовой совокупности объектов нечеткой кластеризации. Эти данные представляют собой матрицу данных **D** размерности (140×2), которые содержатся в файле fcmdata.dat, поставляемом вместе с системой MATLAB. В этом случае матрица данных **D** соответствует 140 объектам, для каждого из которых выполнены измерения по двум признакам, что является удобным для визуализации исходных данных и результатов нечеткой кластеризации в двумерном пространстве на плоскости. График координат точек на плоскости, соответствующих объектам нечеткой кластеризации, изображен на рис. 13.1.

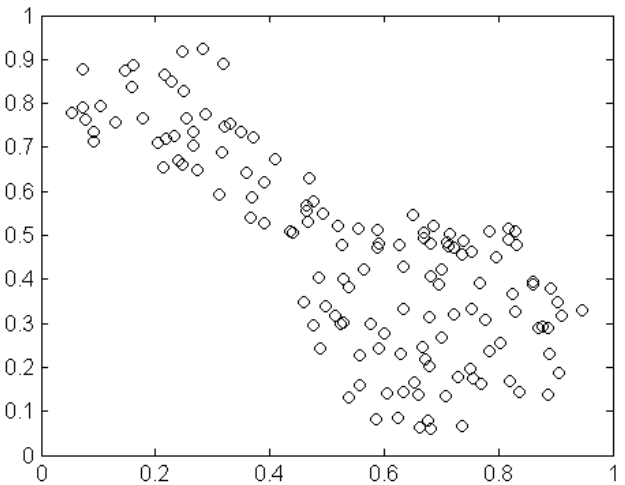


Рис. 13.1. Множество объектов нечеткой кластеризации, соответствующих матрице данных из файла fcmdata.dat

В листинге 13.1 приводится последовательность команд, которые обеспечивают решение задачи нечеткой кластеризации множества данных **D** и визуализацию полученных результатов. В этом примере используется первый формат записи функции `fcm`.

Листинг 13.1. Пример решения задачи нечеткой кластеризации с помощью функции `fcm` с использованием первого формата ее записи

```
load fcmdata.dat
plot(fcmdata(:,1),fcmdata(:,2),'o','color','k')
[center,U,obj_fcn] = fcm(fcmdata, 2);
```

```
maxU = max(U);
index1 = find(U(1,:)== maxU);
index2 = find(U(2, :)== maxU);
line(fcmdata(index1,1), fcmdata(index1, 2), 'linestyle', 'none',...
'marker', 'o','color', 'g');
line(fcmdata(index2,1), fcmdata(index2, 2), 'linestyle', 'none',...
'marker', 'x','color', 'r');
hold on
plot(center(1,1),center(1,2),'ko','markersize', 10, 'LineWidth', 2)
plot(center(2,1),center(2,2),'ko','markersize', 10, 'LineWidth', 2)
```

Результат решения задачи нечеткой кластеризации для 2-х нечетких кластеров с использованием указанной последовательности команд может быть визуализирован (рис. 13.2).

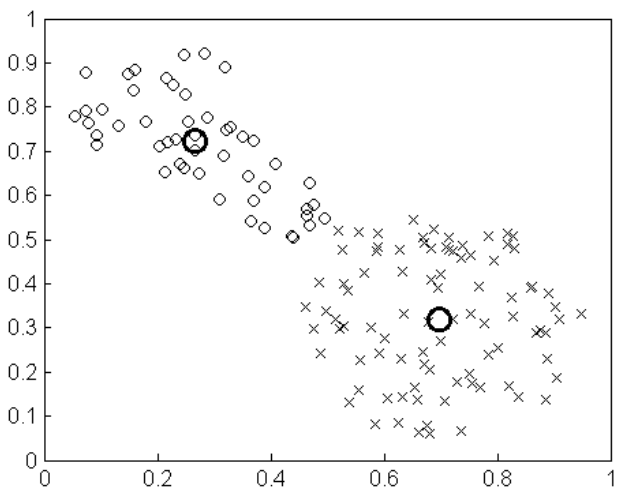


Рис. 13.2. Результат решения задачи нечеткой кластеризации для матрицы данных из файла fcmdata.dat

Если после записи функции `fcm` в третьей строке не указывать точку с запятой (`;`), то в окне команд будут отображены значения координат центров нечетких кластеров, значения функций принадлежности объектов нечетким кластерам и значения целевой функции на каждой из итераций работы алгоритма FCM (рис. 13.3).

Теперь выполним небольшой эксперимент и изменим параметры функции `fcm`, заданные по умолчанию. Для этого будем использовать второй формат ее записи с дополнительными значениями аргументов (листинг 13.2).

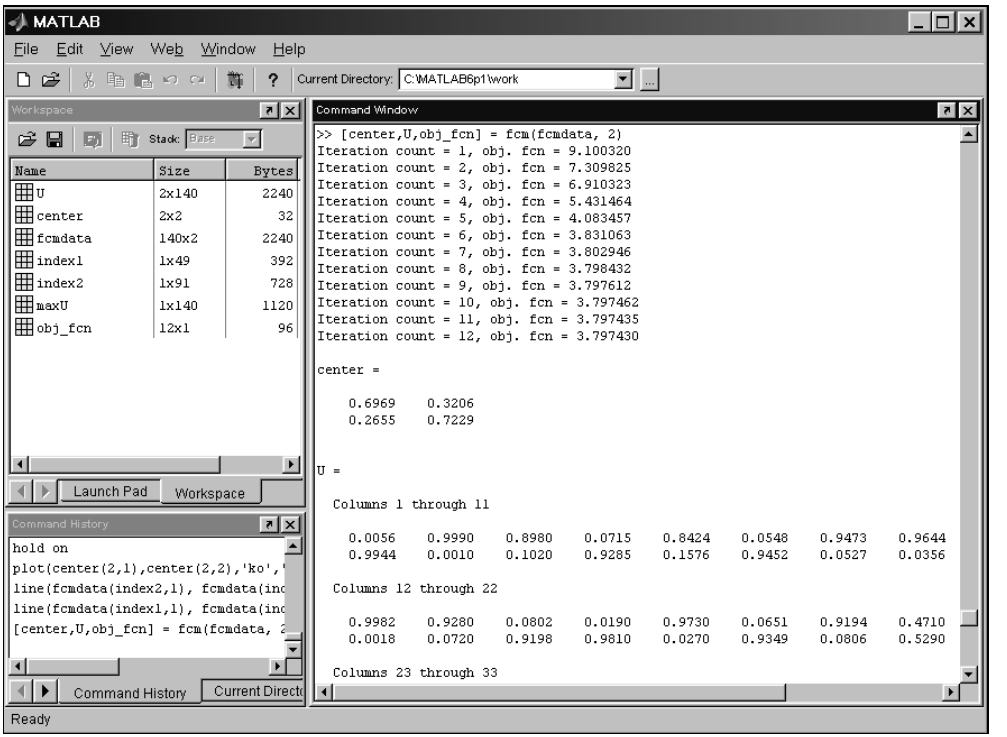


Рис. 13.3. Результаты решения задачи нечеткой кластеризации в окне команд системы MATLAB

Листинг 13.2. Пример решения задачи нечеткой кластеризации с помощью функции `fcm` с использованием второго формата ее записи

```
load fcmdata.dat
center,U,obj_fcn] = fcm(fcmdata, 2, [2.5 1000 0.0000001 1]);
maxU = max(U);
index1 = find(U(1,:)== maxU);
index2 = find(U(2, :)== maxU);
line(fcmdata(index1,1), fcmdata(index1, 2), 'linestyle', 'none',...
'marker', 'o','color', 'g');
line(fcmdata(index2,1), fcmdata(index2, 2), 'linestyle', 'none',...
'marker', 'x','color', 'r');
hold on
plot(center(1,1),center(1,2),'ko','markersize', 10, 'LineWidth', 2)
plot(center(2,1),center(2,2),'ko','markersize', 10, 'LineWidth', 2)
```

Результат решения задачи нечеткой кластеризации для 2-х нечетких кластеров с использованием указанной последовательности команд визуализирован (рис. 13.4). В этом случае экспоненциальный вес m=2.5, максимальное количество итераций s=1000, параметр сходимости алгоритма ε=0.0000001. Как можно увидеть, столь высокая точность расчетов практически не нужна, поскольку алгоритм заканчивает свою работу после выполнения 16 итераций.

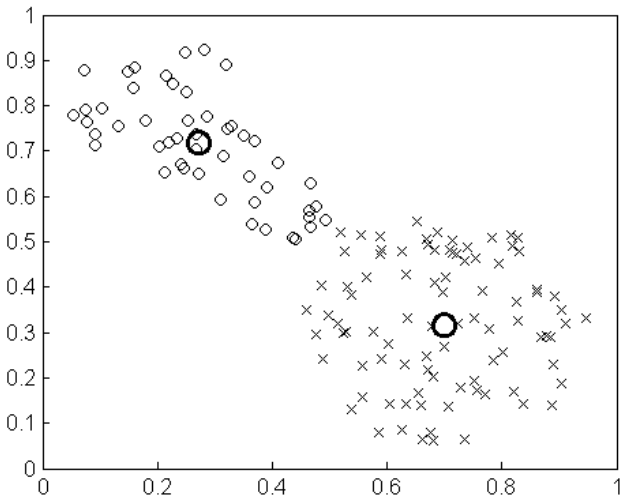


Рис. 13.4. Результат решения задачи нечеткой кластеризации с дополнительными параметрами функции `fcm`

Анализ графиков, изображенных рис. 13.2 и рис. 13.4, показывает их практическую идентичность, что позволяет сделать вывод о согласованности полученных результатов нечеткой кластеризации.

Решение задачи нечеткой кластеризации с использованием средств графического интерфейса

Функция `findcluster` предназначена для вызова графического интерфейса программы нечеткой кластеризации для метода нечетких c-средних и метода субтрактивной кластеризации (subtractive clustering). Она может быть вызвана в одном из следующих форматов: `findcluster` или `findcluster('file.dat')`.

В первом случае функция `findcluster` вызывает графический интерфейс GUI программы для выполнения нечеткой кластеризации алгоритмом FCM и/или нечеткой субтрактивной кластеризации. При этом необходимо загрузить в рабочую область исходные данные из внешнего файла с помощью кнопки **Load Data** (рис. 13.5).

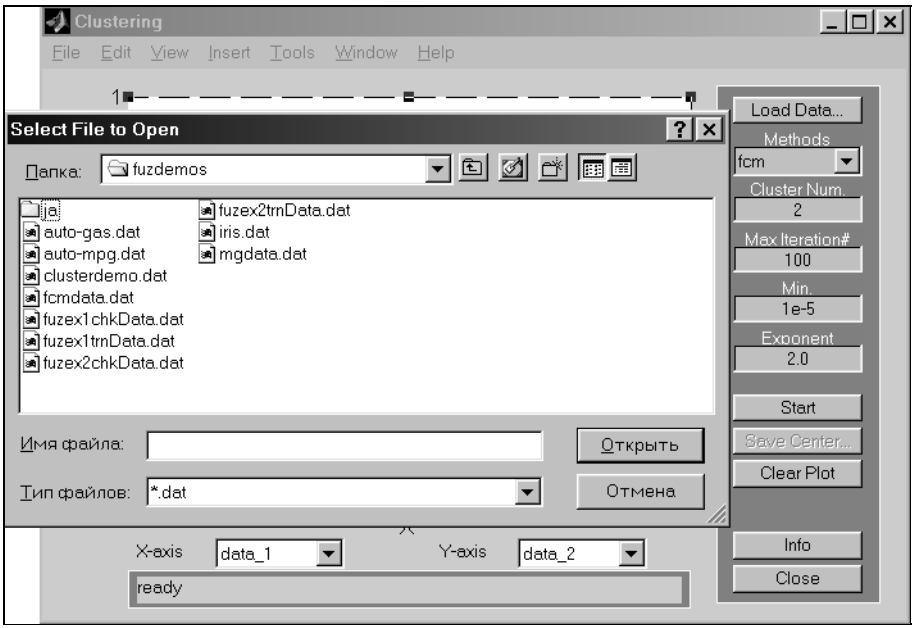


Рис. 13.5. Выбор внешнего файла
для загрузки исходных данных в рабочую область

Выбор метода нечеткой кластеризации осуществляется с помощью раскрывающегося списка **Methods**. Для каждого из методов нечеткой кластеризации в соответствующих строках ввода установлены значения параметров алгоритмов по умолчанию. Эти значения могут быть изменены пользователем. Для этого необходимо установить курсор ввода в соответствующее поле и набрать нужные цифры с учетом допустимых значений параметров. Назначение этих параметров для алгоритма FCM описано ранее при рассмотрении функции `fcm`. Назначение этих параметров для алгоритма субтрактивной кластеризации будет описано ниже при рассмотрении функции `subclust`.

Во втором случае функция `findcluster('file.dat')` вызывает графический интерфейс, а в рабочую обрасть автоматически загружаются данные кластеризации из внешнего файла file.dat. При этом на графике отображаются значения матрицы данных для первых двух признаков.

После нажатия на кнопку **Start** начинает работу соответствующий алгоритм нечеткой кластеризации с параметрами, установленными по умолчанию или измененными пользователем. Результаты работы алгоритма отображаются на графике (рис. 13.7). Найденные центры кластеров изображены черными кругами и их координаты можно сохранить во внешнем файле с целью последующего анализа.

Иногда число нечетких кластеров, необходимых для работы алгоритма FCM, априори является неизвестным. В этом случае целесообразно использовать реализованный в системе MATLAB так называемый алгоритм субтрактивной кластеризации.

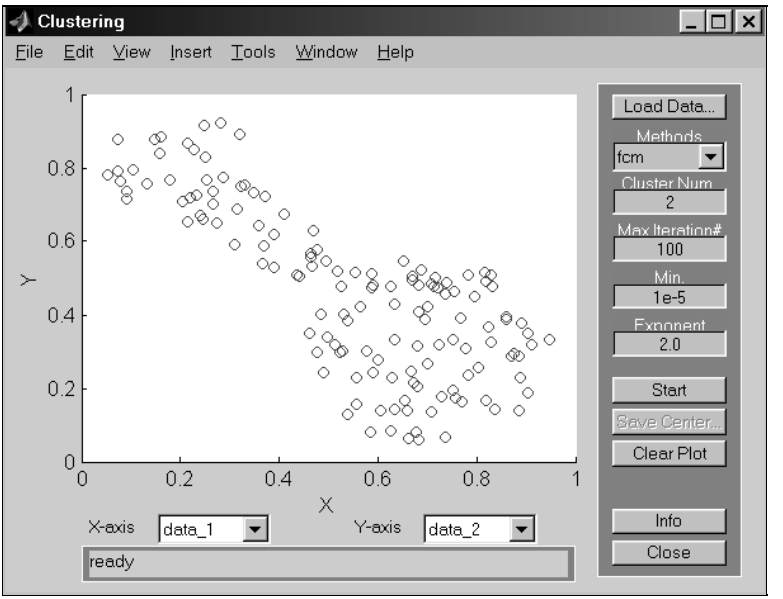


Рис. 13.6. Результат вызова графического интерфейса нечеткой кластеризации с уже загруженными исходными данными из внешнего файла fcmdata.dat

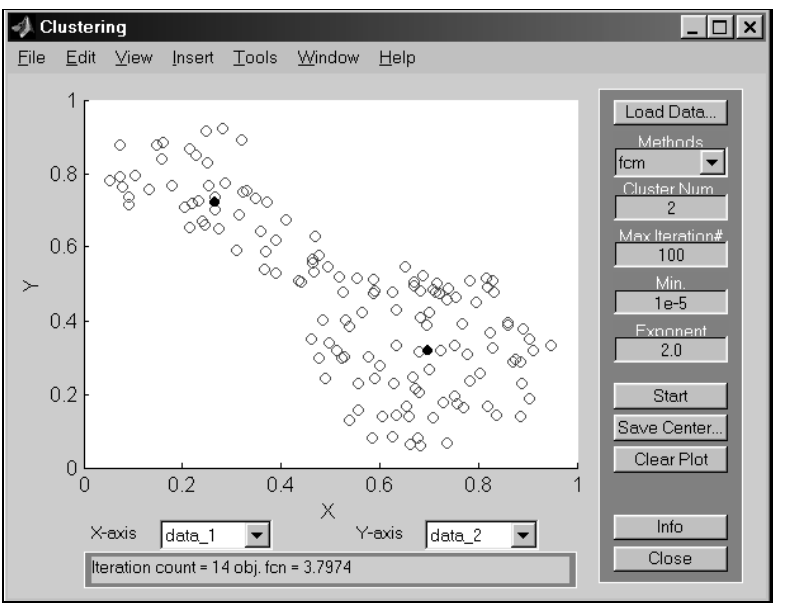


Рис. 13.7. Результат решения задачи нечеткой кластеризации алгоритмом FCM с помощью графического интерфейса для множества данных из внешнего файла fcmdata.dat

Решение задачи определения числа кластеров для нечеткой кластеризации в системе MATLAB

В случае отсутствия каких-либо априорных предположений относительно количества нечетких кластеров в системе MATLAB можно использовать функцию командной строки `subclust` или метод субтрактивной нечеткой кластеризации, реализованный в графическом интерфейсе кластеризации.

Идея метода субтрактивной кластеризации состоит в том, что каждая точка данных предполагается в качестве центра потенциального кластера, после чего следует вычислить некоторую меру способности каждой точки данных представлять центр кластера. Эта количественная мера основана на оценке плотности точек данных вокруг соответствующего центра кластера.

Данный алгоритм, который является обобщением метода кластеризации Р. Ягера (R. Yager), основан на выполнении следующих действий:

1. Выбрать точку данных с максимальным потенциалом для представления центра первого кластера.
2. Удалить все точки данных в окрестности центра первого кластера, величина которой задается параметром `radii`, чтобы определить следующий нечеткий кластер и координаты его центра.

Эти две процедуры повторяются до тех пор, пока все точки данных не окажутся внутри окрестностей радиуса `radii` искомых центров кластеров.

Функция командной строки `subclust` находит центры кластеров методом субтрактивной кластеризации. Она используется в следующем формате:

```
[C,S] = subclust(X, radii, xBounds, options)
```

При этом матрица `X` содержит данные кластеризации, каждая строка которой соответствует координатам отдельной точки данных. Параметр `radii` представляет собой вектор, компоненты которого принимают значения из интервала [0, 1] и задают диапазон расчета центров кластеров по каждому из признаков измерений. При этом делается предположение, что все данные содержатся в некотором единичном гиперкубе. В общем случае малые значения параметра `radii` приводят к нахождению малого числа больших по количеству точек кластеров. Наилучшие результаты получаются при значениях `radii` между 0.2 и 0.5.

Аргумент `xBounds` представляет собой матрицу размерности $(2\times q)$, которая определяет способ отображения матрицы данных `X` в некотором единичном гиперкубе. Здесь q — количество рассматриваемых признаков в множестве данных. Этот аргумент является необязательным, если матрица `X` уже нормализована. Первая строка этой матрицы содержит минимальные значения интервала измерения каждого из признаков, а вторая строка — максимальные значения измерения каждого из признаков.

Для изменения заданных по умолчанию параметров алгоритма кластеризации может быть использован дополнительный вектор `options`. Компоненты этого вектора могут принимать следующие значения:

- `options(1) = quashFactor` — параметр, используемый в качестве коэффициента для умножения значений radii, которые определяют окрестность центра кластера. Это осуществляется с целью уменьшения влияния потенциала граничных точек, рассматриваемых как часть нечеткого кластера (по умолчанию это значение равно 1.25);
- `options(2) = acceptRatio` — параметр, устанавливающий потенциал как часть потенциала центра первого кластера, выше которого другая точка данных может рассматриваться в качестве центра другого кластера (по умолчанию это значение равно 0.5);
- `options(3) = rejectRatio` — параметр, устанавливающий потенциал как часть потенциала центра первого кластера, ниже которого другая точка данных не может рассматриваться в качестве центра другого кластера (по умолчанию это значение равно 0.15);
- `options(4) = verbose` — если значение этого параметра не равно нулю, то на экран монитора выводится информация о выполнении процесса кластеризации (по умолчанию это значение равно 0).

Функция `subclust` возвращает матрицу `C` значений координат центров нечетких кластеров. При этом каждая строка этой матрицы содержит координаты одного центра кластера. Эта функция также возвращает вектор `S`, компоненты которого представляют σ-значения, которые определяют диапазон влияния центра кластера по каждому из рассматриваемых признаков. При этом все центры кластеров обладают одинаковым множеством σ-значений.

Пример 13.2. Рассмотрим решение задачи определения количества кластеров для множества исходных данных из примера 13.1. Напомним, что эти данные представляют собой матрицу данных **D** размерности (140×2), которые содержатся в файле fcmdata.dat. В этом случае вызов функции субтрактивной кластеризации может быть реализован следующим образом:

```
load fcmdata.dat
[C,S] = subclust(fcmdata,[0.5 0.5],[],[1.25 0.5 0.15 1])
```

Поскольку для множества данных из примера 13.1 используется 2 признака (матрица `datin` имеет 5 столбцов), то по каждому из признаков задаются радиусы окрестностей 0.5 и 0.5 соответственно. Коэффициенты шкалирования для отображения множества данных в единичный гиперкуб получаются на основе минимального и максимального значений данных.

Аргумент `squashFactor = 1.25` указывает на то, что необходимо определить кластеры, недалеко расположенные друг от друга. Аргумент `acceptRatio = 0.5` указывает на то, что для нахождения центров кластеров очень высокий потенциал не является необходимым. Аргумент `rejectRatio = 0.15` не исключает из рассмотрения точки данных, не обладающих высоким потенциалом. Наконец,

последний аргумент `verbose = 1` разрешает вывод информации о выполнении процесса кластеризации на экран монитора.

В результате выполнения этого фрагмента команд будут получены следующие значения матрицы центров кластеров и вектора σ-значений (рис. 13.8). Как можно заметить, для указанных значений аргументов рассматриваемая функция субтрактивной кластеризации находит три нечетких кластера и отображает координаты их центров в командном окне системы MATLAB.

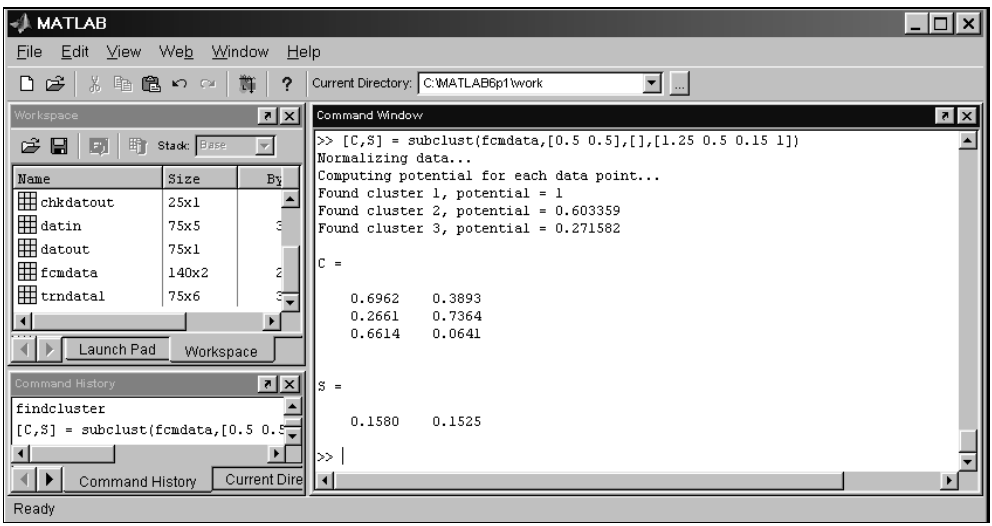


Рис. 13.8. Результат решения задачи нечеткой субтрактивной кластеризации с помощью функции командной строки для множества данных из внешнего файла fcmdata.dat

Для решения задачи субтрактивной кластеризации может быть также использован рассмотренный выше графический интерфейс нечеткой кластеризации, который вызывается функцией findcluster. Вызов этой функции может быть осуществлен в одном из следующих форматов: `findcluster` или `findcluster('file.dat')`, особенности которых были изложены ранее и иллюстрированы на рис. 13.5—13.7.

П р и м е р 13.3. Рассмотрим решение задачи определения количества кластеров для множества исходных данных из примера 13.1 с использованием графического интерфейса кластеризации. Для этого загрузим исходные данные из внешнего файла командой `findcluster('fcmdata.dat')`, выберем метод кластеризации **subtractive** в раскрывающемся списке **Methods** и нажмем кнопку **Start**. Остальные параметры оставим предложенными по умолчанию. Результат решения задачи субтрактивной кластеризации изображен на рис. 13.9 и также содержит 3 нечетких кластера.

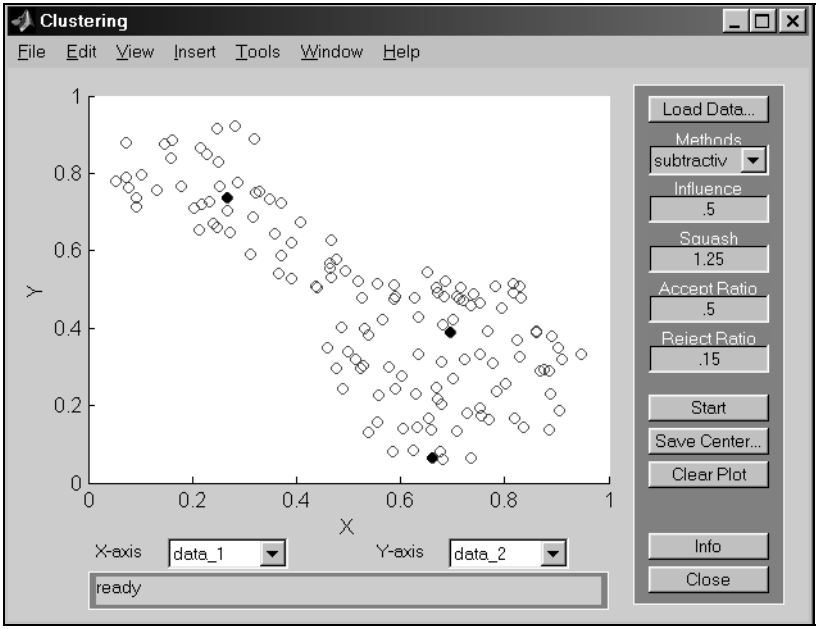
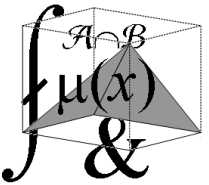


Рис. 13.9. Результат решения задачи нечеткой субтрактивной кластеризации с помощью графического интерфейса кластеризации для множества данных из внешнего файла fcmdata.dat

Таким образом, система MATLAB позволяет решать задачи нечеткой кластеризации двумя способами: с помощью функций командной строки и с помощью графического интерфейса кластеризации. Первый из них является более трудоемким, но обладает большей гибкостью и возможностью отображения в окне команд значений матриц центров нечетких кластеров, функций принадлежности и целевой функции. Второй способ представляется наиболее удобным для выполнения некоторой серии расчетов для различных значений входных параметров с целью визуального анализа полученных результатов.

В заключение этой главы следует отметить, что результаты нечеткой кластеризации имеют приближенный характер и могут служить лишь для предварительной структуризации информации, содержащейся в множестве исходных данных. Решая задачи нечеткой кластеризации, нужно помнить об особенностях и ограничениях процесса измерения признаков у совокупности объектов кластеризации. Поскольку нечеткие кластеры формируются на основе евклидовой метрики, соответствующее пространство признаков должно удовлетворять аксиомам метрического пространства. В то же время для поиска закономерностей в проблемной области, имеющих неметрический характер, необходимо использовать специальные средства и инструментарий, разработанные для интеллектуального анализа данных (Data Mining).

Глава 14



Основы программирования в среде MATLAB

Гибкость и популярность системы MATLAB во многом основана на возможности создания, редактирования и выполнения в ее среде программ, написанных пользователями. Эти программы могут расширять функциональные возможности системы или модифицировать отдельные ее функции желательным для пользователей образом. При этом используется собственный язык программирования MATLAB, который по форме хотя и напоминает отдельные конструкции языка C/C++, но имеет синтаксические отличия. Именно по этой причине ниже рассматриваются основные элементы языка программирования системы MATLAB.

14.1. Основы языка программирования системы MATLAB

Достоинством системы MATLAB является то обстоятельство, что практически все специальные функции реализованы в виде так называемых m-файлов, которые представляют собой обычные текстовые файлы, содержащие конструкции языка программирования MATLAB. При желании понять способ реализации той или иной функции можно всегда обратиться к исходному тексту соответствующего файла. Чтобы отметить этот факт, иногда говорят, что система MATLAB распространяется вместе с исходными текстами программ, имея в виду соответствующие m-файлы.

Примечание

Сравните этот факт с одним из наиболее часто употребляемых сторонниками Linux аргументов, когда, говоря о достоинствах этой ОС, специально и неоднократно подчеркивается, что она распространяется в своих исходных кодах.

M-файлы системы MATLAB могут быть двух видов: так называемые *скрипты*, которые содержат последовательность некоторых команд, и *функции*, которые предполагают использование некоторых аргументов и возвращают определенное значение. В обоих случаях используется один и тот же язык программирова-

ния MATLAB, который также применяется и при работе в окне команд. Эта унификация языка представляет собой исключительное удобство для пользователей, поскольку достаточно изучить только один язык программирования для всестороннего и глубокого использования возможностей системы MATLAB.

Как следствие этой унификации все операции и команды, рассмотренные ранее, могут быть использованы в тексте m-файлов. В системе MATLAB выполнение команд и операций реализовано в форме интерпретации, т. е. все команды m-файла выполняются в линейной последовательности без компиляции исходного текстового файла. Некоторые особенности выполнения действий с переменными были рассмотрены ранее при знакомстве с основными приемами работы в окне команд *(см. главу 11)*. В дополнение к основным алгебраическим операциям системы MATLAB (табл. 11.2) в логических выражениях могут быть использованы также операции сравнения (табл. 14.1) и логические операции (табл. 14.2).

Таблица 14.1. *Операции сравнения*

Символ операции	Имя функции	Название	Пример
<	lt	Меньше	i < 10
<=	le	Меньше или равно	5 <= pi^2
>	gt	Больше	pi > 1
>=	ge	Больше или равно	a >= b
= =	eq	Равно	'String'=='string'
~=	ne	Не равно	Result~=0

Таблица 14.2. *Логические операции*

Символ операции	Имя функции	Название	Пример
&	and	Логическое И	A&B
|	or	Логическое неисключающее ИЛИ	A|B
~	not	Логическое НЕ	~A
	xor	Логическое исключающее ИЛИ	xor(A, B)

Центральным понятием языка программирования системы MATLAB является математическое выражение, которое состоит из имен отдельных переменных, имен функций и операций. Имя переменной должно начинаться с буквы и может содержать произвольное количество букв, цифр и символов подчеркивания. В то

же время система MATLAB различает переменные только по первым 31 символам их имен, а также с учетом их регистра (большие и маленькие буквы различаются).

Как уже упоминалось ранее, каждая переменная имеет внутреннее представление в форме матрицы или массива. При этом скалярная переменная рассматривается как массив размерности 1×1. С другой стороны, каждая переменная имеет свой тип, который определяет допустимые операции над значениями этой переменной. В системе MATLAB предусмотрены 14 фундаментальных типов данных, которые представляют собой отдельные классы (рис. 14.1).

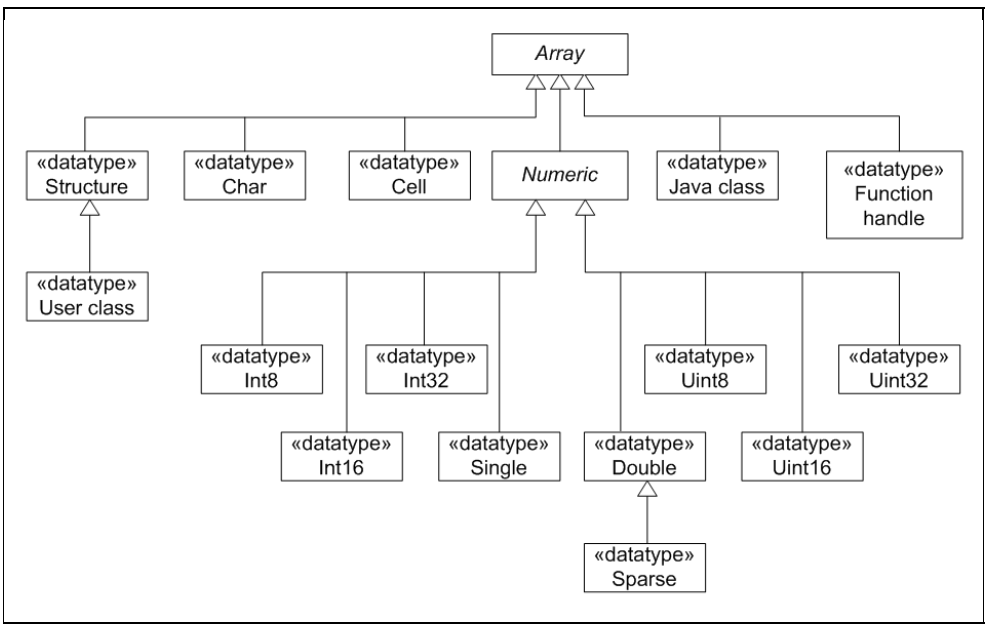


Рис. 14.1. Диаграмма классов фундаментальных типов данных системы MATLAB (в нотации языка UML)

Примечание

Тип данных `user class` определяется пользователем и не является формально определенным в языке программирования системы MATLAB. Классы *`Array`* и *`Numeric`* являются абстрактными, поэтому их имена записаны курсивом согласно нотации языка UML.

Краткая характеристика указанных фундаментальных типов данных и их примеры приводятся в табл. 14.3.

Таблица 14.3. *Характеристика фундаментальных типов данных MATLAB*

Тип данных	Пример	Описание
`int8`, `int16`, `int32`	`int(magic(3))`	Целое число со знаком 8, 16 32 бит длины соответственно. Используются для представления целых чисел в памяти с целью более эффективных преобразований. Этот тип данных непосредственно не может быть использован в математических вычислениях
`uint8`, `uint16`, `uint32`	`uint(magic(3))`	Целое число без знака 8, 16 32 бит длины соответственно. Также используются для представления целых чисел в памяти с целью более эффективных преобразований. Этот тип данных непосредственно не может быть использован в математических вычислениях
`double`	`3*10^300` `5+6i`	Произвольное число со знаком двойной точности. Основной тип данных во всех математических вычислениях и расчетах. Используется по умолчанию при задании произвольной числовой переменной
`sparce`	`speye(5)`	Разреженная числовая матрица размерности (2×2) с элементами двойной точности. Используется для более компактного представления в памяти матриц с большим количеством нулевых элементов. Для разреженных матриц определены специальные операции
`single`	`3*10^38`	Произвольное число со знаком одинарной точности. Используется для более компактного представления чисел в памяти. Этот тип данных непосредственно не может быть использован в математических вычислениях
`char`	`'Hello'`	Произвольный символ в кодировке Unicode. Массив данных типа `char` используется для представления строк произвольной длины. Для символьных строк определены специальные операции
`cell`	`{17 'hello' eye(2)}`	Ячейка с произвольным элементом. Массивы ячеек могут содержать элементы, представляющие различные типы данных, включая и другие массивы ячеек
`structure`	`a.day=12` `a.color='red'`	Структура представляет собой массив полей и, возможно, других структур. Обращение к отдельным полям структуры осуществляется обычным для языка программирования способом — с помощью оператора "*точка*"

Таблица 14.3 *(окончание)*

Тип данных	Пример	Описание
`function handle`	`@humps`	Дескриптор функции. Используется для подстановки одних функций в качестве аргументов других функций
`java class`	`java.awt.Frame`	Класс языка программирования Java. Применяется в случае использования классов, определенных в библиотеке Java, для программирования графических интерфейсов
`user class`	`inline('sin(x)')`	Тип данных, определяемый пользователем. Используется при написании программ в стиле объектно-ориентированного программирования

По умолчанию система MATLAB рассматривает каждую переменную как произвольное число со знаком (double), однако формат ее отображения в окне команд может различаться и зависит от настроек системы. При этом доступ ко всем переменным текущего сеанса работы осуществляется с помощью браузера рабочей области.

При разработке m-файлов следует различать два основных типа этих файлов:

- *m-файл-сценарий*, который может быть использован для определения значений переменных в рабочей области или их изменения. Этот тип m-файлов аналогичен записи команд в окне команд в текущем сеансе работы с программой. Например, последовательность функций для задания системы нечеткого вывода примера 12.1 "*Чаевые в ресторане*", сохраненная во внешнем текстовом файле с расширением m, представляет собой файл-сценарий;
- *m-файл-функция*, который реализует выполнение некоторой функции или процедуры и возвращает некоторое значение. Пример разработки файла-функции, который расширяет функциональность системы MATLAB, приводится далее в *разд. 14.3*.

Наибольший интерес в контексте программирования в системе MATLAB имеют m-файлы второго типа, которые записываются в специальном формате и могут содержать математические выражения, операторы управления последовательностью команд и текстовые комментарии.

Каждый m-файл-функция должен иметь специальную структуру. А именно — начинаться с ключевого слова `function`, после которого следует имя выходной переменной, имя функции и список ее входных параметров (аргументов). Завершаться текст m-файла может произвольным оператором, комментарием или выражением (ключевое слово `end` в этом случае не используется). При этом имя m-файла должно совпадать с именем функции.

Вызов реализованной в m-файле функции осуществляется аналогично обычной функции в окне команд. При этом указывается имя функции и список конкретных значений параметров (аргументов). В этом случае соответствующий m-файл

автоматически исполняется системой MATLAB, в результате чего реализованной в нем функцией будет возвращено конкретное значение.

Следует заметить, что все используемые в тексте m-файла переменные по своему характеру являются локальными, т. е. не оказывают побочного влияния на значения переменных с аналогичными именами из рабочей области. В тексте m-файла в случае необходимости могут быть использованы имена других функций системы MATLAB.

В общем случае последовательность выполнения команд и операторов m-файла является линейной, что типично для процедурного стиля написания программ. Для изменения линейной последовательности команд могут быть использованы специальные операторы управления, которые рассматриваются далее.

Операторы управления последовательностью выполнения команд

Операторами языка программирования MATLAB являются специальные синтаксические конструкции языка, предназначенные для выполнения определенных функциональных задач.

Условный оператор *if...elseif...else...end*

Условный оператор предназначен для реализации последовательного бинарного ветвления в программе на основе проверки определенных логических условий.

Полный синтаксис этого оператора:

```
if логическое_условие_1, выражения_1; elseif логическое_условие_2,
выражения_2; else выражения_3; end
```

Условный оператор выполняется следующим образом. Вначале проверяется *логическое условие_1*. Если оно принимает значение "*истина*", то последовательно выполняются *выражения_1*, после чего управление передается на конец условного оператора — `end`. Если *логическое условие_1* принимает значение "*ложь*", то *выражения_1* не выполняются и происходит переход к проверке *логического условия_2*. Если оно в свою очередь принимает значение "*истина*", то последовательно выполняются *выражения_2*, после чего управление передается на конец условного оператора — `end`. Если *логическое условие_2* принимает значение "*ложь*", то *выражения_2* не выполняются, а выполняются *выражения_3*, после чего выполнение условного оператора заканчивается.

Возможно использование более простого синтаксиса этого оператора в форме:

```
if логическое_условие_1, выражения_1; else выражения_2; end
```

или в простейшей форме:

```
if логическое_условие, выражения; end
```

Наряду с этим возможны и более сложные варианты этого оператора с вложенными условиями `elseif`. Для записи логических выражений могут быть использова-

ны операции сравнения (см. табл. 14.1) и логические операции (см. табл. 14.2). Важно помнить, что в системе MATLAB в случае истинности логическое выражение принимает значение 1, а в случае ложности — значение 0.

Листннг 14.1. Пример использования условного оператора со сложным логическим условием

```
n = 1;
s = -10:10;
if (s(15)>=0)&(n>0),
       n =s(16) + 1;
end
```

С примером использования условного оператора в полном варианте его синтаксиса можно познакомиться, просмотрев текст m-файла defuzz.m, который реализует различные методы дефаззификации в системе MATLAB.

Оператор выбора *switch...case...otherwise...end*

Оператор выбора предназначен для реализации параллельного ветвления в программе для выполнения различных вариантов выражений.

Полный синтаксис этого оператора:

```
switch исходное_выражение
       case значение_1, выражения_1;
       case значение_2, выражения_2;
       ...
       otherwise выражения_0;
end
```

Оператор выбора выполняется следующим образом. *Исходное_выражение* должно быть определено как некоторый скаляр (число или символ) или как строка текста. Если оно равно *значению_1*, то последовательно выполняются *выражения_1*, после чего управление передается на конец оператора выбора — `end`. Если *исходное_выражение* не равно *значению_1*, то *выражения_1* не выполняются, а управление передается на следующий по порядку вариант `case`. Если *исходное_выражение* равно *значению_2*, то последовательно выполняются *выражения_2*, после чего управление также передается на конец оператора выбора — `end`. Если *исходное_выражение* не равно *значению_2*, то *выражения_2* не выполняются, а управление передается на следующий по порядку вариант `case`. *Выражения_0* выполняются только в том случае, если *исходное_выражение* не равно ни одному из вариантов значений.

Возможно использование более простого синтаксиса этого оператора без секции `otherwise` *выражения_0*. Оператор выбора варианта целесообразно использо-

вать в качестве альтернативы условному оператору, чтобы избежать вложенности при проверке логических условий.

Листинг 14.2. Пример использования оператора выбора варианта

```
r = 1;
switch r
       case 5, s = 10;
       case 10, s = 15;
       otherwise s = 20;
end
```

Если окажется, что значение *исходного_выражения* равно нескольким значениям вариантов `case`, то в этом случае будет выполнен первый по порядку вариант последовательности выражений.

Оператор цикла *for...end*

Этот оператор предназначен для выполнения группы выражений языка фиксированное число раз, причем соответствующее число повторений заранее известно и задается до начала выполнения выражений цикла.

Синтаксис этого оператора:

`for` *переменная=начальное_значение* : *приращение* : *конечное_значение*,

выражения; `end`

Если *приращение* отсутствует, то по умолчанию оно принимается равным 1. Цикл типа `for` выполняется следующим образом. В начале цикла *переменной* цикла присваивается *начальное_значение*, и последовательно выполняются *выражения* тела цикла (итерация), после чего управление снова передается на его начало. Переменной цикла присваивается следующее значение, определяемое значением *приращения*. Если новое значение *переменной* цикла не превышает *конечного_значения* переменной, то выполняется очередная итерация цикла. Выполнения *выражений* цикла заканчивается, когда значение *переменной* цикла превысит ее *конечное_значение*. В этом случае *выражения* цикла не выполняются и происходит переход к выполнению следующего за `end` выражения.

Листинг 14.3. Пример использования оператора цикла типа `for` для подсчета суммы первых 100 натуральных чисел

```
s = 0;
for n = 1:100,
       s = s + n;
end
```

Циклы типа `for` могут быть вложенными. В качестве значений переменной цикла могут выступать элементы некоторого массива или матрицы. Ниже приводится вариант предыдущего примера, в котором переменная цикла принимает значения из заранее определенного массива.

Листинг 14.4. Вариант использования оператора цикла типа `for` для подсчета суммы первых 100 натуральных чисел

```
A = 1:100;
s = 0;
for n = A,
       s = s + n;
end
```

В случае использования в качестве значений переменной цикла элементов некоторой матрицы следует помнить, что элементы матрицы упорядочены по столбцам.

Оператор цикла *while...end*

Этот оператор предназначен для выполнения группы выражений языка несколько раз, причем соответствующее число повторений заранее неизвестно и определяется проверкой некоторого условия внутри цикла.

Синтаксис этого оператора:

`while` *логическое_условие*, *выражения*; `end`

Цикл типа `while` выполняется следующим образом. Вначале цикла проверяется *логическое_условие*. Если оно принимает значение "*истина*", то последовательно выполняются *выражения* тела цикла, после чего управление снова передается на его начало. Если *логическое_условие* принимает значение "*ложь*", то *выражения* не выполняются и происходит переход к выполнению следующего за `end` выражения.

Листинг 14.5. Пример использования оператора цикла типа `while` для подсчета суммы первых 100 натуральных чисел

```
n = 1;
s = 0;
while n<=100,
       s = s + n;
       n = n + 1;
end
```

Следует отметить, что невнимательное использование цикла типа `while` может служить источником ошибок и даже зависания системы MATLAB в случае выполнения бесконечного цикла.

Оператор *continue*

Этот оператор используется в теле операторов цикла `for` или `while`, чтобы закончить выполнение выражений текущей итерации. При вложенных циклах оператор `continue` передает управление на окончание итерации содержащего этот оператор цикла.

Листинг 14.6. Пример использования оператора `continue` в цикле типа `for` для подсчета суммы нечетных натуральных чисел в интервале от 1 до 100

```
s = 0;
for n = 1:100,
       if rem(n,2)==0,
              continue;
       end;
       s = s + n;
end
```

Заметим, что в приведенном фрагменте программы использована специальная функция `rem(x, y)`, которая возвращает остаток от деления x на y.

Оператор *break*

Этот оператор используется в теле операторов цикла `for` или `while`, чтобы закончить выполнение выражений всего цикла. После выполнения этого оператора управление передается следующему за `end` выражению. При вложенных циклах оператор `break` передает управление на окончание только того цикла, в котором находится этот оператор.

Листинг 14.7. Пример использования оператора `break` в цикле типа `for` для подсчета суммы первых 10 нечетных натуральных чисел в интервале от 1 до 100

```
s = 0;
k = 0;
for n = 1:100,
       if rem(n,2)==0,
              continue;
```

```
        end;
        s=s+n;
        k=k+1;
        if k == 10,
                break;
        end;
end
```

Оператор *return*

Этот оператор используется, как правило, в теле функций, чтобы закончить выполнение любых оставшихся выражений и возвратить управление в вызвавшую эту функцию программу или в окно команд текущего сеанса работы MATLAB.

Листинг 14.8. Пример использования оператора `return` в функции подсчета суммы первых *k* нечетных натуральных чисел в интервале от 1 до *n*

```
function out = sumodd(k,n)
if k>=n,
    error('k должно быть меньше чем n !');
end
out = 0;
j = 0;
for i = 1:n,
    if rem(i,2)==0,
              continue;
    end;
    out=out+i;
    j=j+1;
    if j==k,
        return;
    end;
end
```

Для того чтобы данную функцию можно было бы использовать в системе MATLAB, необходимо сохранить текст данной программы во внешнем файле с именем sumodd.m, а сам файл поместить в один из рабочих каталогов системы MATLAB, например, в каталог C:\MATLAB6p1\work. После этого созданная нами функция подсчета суммы первых *k* нечетных натуральных чисел в интервале от 1 до *n* становится доступной в любой программе и в сеансе работы

MATLAB. Для использования этой функции ее необходимо вызвать с заданными значениями двух ее аргументов. Например, если в окне команд набрать

```
>> x = sumodd(5,100)
```

то получим значение:

```
x =
        25
```

Рассмотренный пример иллюстрирует процесс написания программ в системе MATLAB и оформления их в форме функций. Далее в *разд. 14.3* приводится пример программы, расширяющей возможности пакета нечеткой логики системы MATLAB.

Защищенный блок *try...catch...end*

Защищенный блок предназначен для изменения последовательности выполнения выражений, если во время их выполнения обнаруживается некоторая ошибка.

Синтаксис защищенного блока следующий:

```
try, выражения_1;
catch, выражения_2;
end
```

Система MATLAB при исполнении защищенного блока делает попытку выполнить последовательность *выражений_1*. Если при выполнении хотя бы одного из выражений этой последовательности происходит ошибка, то выполнение всей последовательности *выражений_1* аннулируется. После этого происходит передача управления на выполнение *выражений_2*. Если же при выполнении *выражений_1* ошибки не происходит, то *выражения_1* не выполняются, и происходит передача управления на конец блока. В случае наличия ошибки при выполнении *выражений_2* система прерывает работу программы и сообщает о характере происшедшей ошибки.

Листинг 14.9. Пример использования защищенного блока (значение `renf` никак не определено)

```
try,
       k=0;
       s=renf;
catch,
       s=7;
       k=1;
end
```

При выполнении данного фрагмента программы в окно команд не будет выведено сообщение об ошибке "*??? Undefined function or variable 'renf'*", поскольку соответствующий оператор присваивания помещен в защищенный блок. После выполнения защищенного блока в рабочей области будут помещены следующие значения переменных: s=7 и k=1.

Следует заметить, что все операторы управления последовательностью выполнения команд должны заканчиваться словом `end`, которое указывает на конец соответствующего управляющего блока.

Текстовые комментарии

Текстовые комментарии используются в m-файлах для пояснения основных идей реализации процедур, а также отображения справочной информации при вызове справки по соответствующей функции m-файла. При этом текстовый комментарий может вставляться в любое место m-файла после символа начала комментария: `%`.

Для разработки и отладки m-файлов в системе MATLAB предназначен специальный редактор/отладчик. Ниже рассматривается назначение операций главного меню и панели инструментов редактора/отладчика, а также основные приемы работы с ним.

14.2. Основные приемы работы с редактором/отладчиком m-файлов

Для создания, редактирования и отладки m-файлов в системе MATLAB предназначен специальный отладчик m-файлов. Хотя m-файлы являются основными объектами программирования, редактор-отладчик позволяет открывать и редактировать любые типы файлов, включая обычные текстовые файлы (с расширением txt, dat и др.), файлы с текстами программ на различных языках программирования (с расширением c, cpp, h, html и другими), а также файлы моделей отдельных пакетов расширений системы MATLAB (с расширением mdl, fis, rtw и др.).

Открыть редактор-отладчик можно несколькими способами, например, нажав соответствующую кнопку (самую левую) на панели инструментов системы MATLAB. В результате откроется рабочее окно редактора-отладчика с предложенным системой именем нового m-файла по умолчанию (рис. 14.2).

Если необходимо редактировать существующий m-файл или любой другой текстовый файл, имеющийся на диске, то редактор-отладчик можно открыть, нажав соответствующую кнопку (вторую слева) на панели инструментов системы MATLAB. В этом случае будет вызвано стандартное диалоговое окно открытия файла, а после выбора файла — появится окно редактора-отладчика с загруженным в него файлом (рис. 14.3).

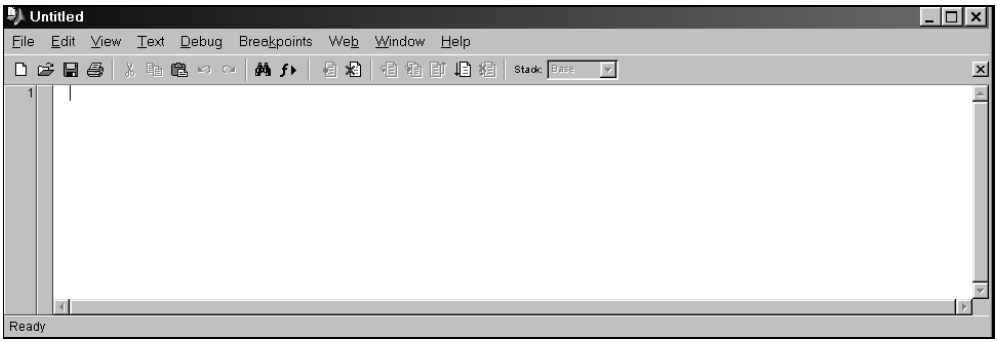


Рис. 14.2. Окно редактора-отладчика при создании нового m-файла

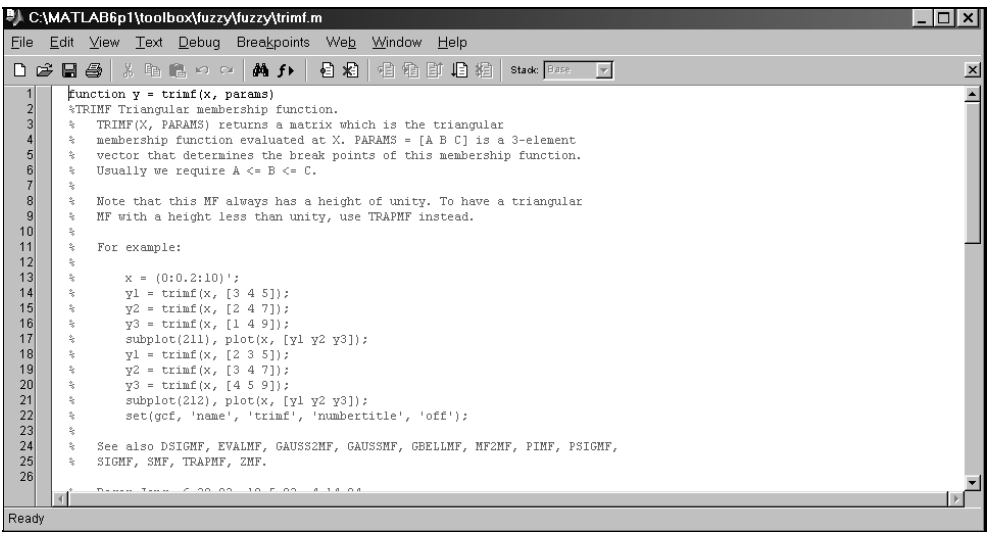


Рис. 14.3. Окно редактора-отладчика с открытым существующим m-файлом

Заметим, что открыть редактор-отладчик можно и без запуска системы MATLAB, для чего следует воспользоваться главным меню ОС MS Windows **(Пуск> Программы>MATLAB 6.1>M-file Editor)** или просто набрав в системном меню **Выполнить** команду `meditor`. Последний случай может быть использован для работы с редактором-отладчиком как обычным Windows-приложением. В этом случае в редакторе-отладчике доступен только режим редактирования файлов без возможности их отладки (соответствующие пункты меню будут недоступны).

Примечание

Открыть редактор-отладчик можно из окна команд, введя команду `edit` для создания нового m-файла или команду `edit <имя файла>` для открытия существующего файла.

Рабочий интерфейс редактора-отладчика представляет собой стандартное окно в стиле MS Windows, которое содержит собственное главное меню и панель инструментов. Основную область занимает окно редактирования файлов, в котором отображается текст создаваемого или выбранного файла. При этом каждая строка файла имеет собственный порядковый номер, а слова текста выделяются различным цветом. Служебные или ключевые слова программ выделяются синим цветом, комментарии в программах — зеленым, строки как элементы программ — красным, остальные элементы текста — черным.

Прежде всего, рассмотрим операции главного меню системы MATLAB и назначение отдельных кнопок панели инструментов. После этого остановимся на основных командах, которые могут оказаться необходимыми в процессе работы в режиме командной строки.

Назначение операций главного меню

Окно редактора-отладчика системы MATLAB имеет главное меню (рис. 14.4), которое позволяет пользователю выполнять отладку m-файлов, открывать и сохранять редактируемые файлы, изменять внешний вид элементов графического интерфейса, вызывать справочную информацию и т. д. Рассмотрим назначение отдельных пунктов главного меню редактора-отладчика.

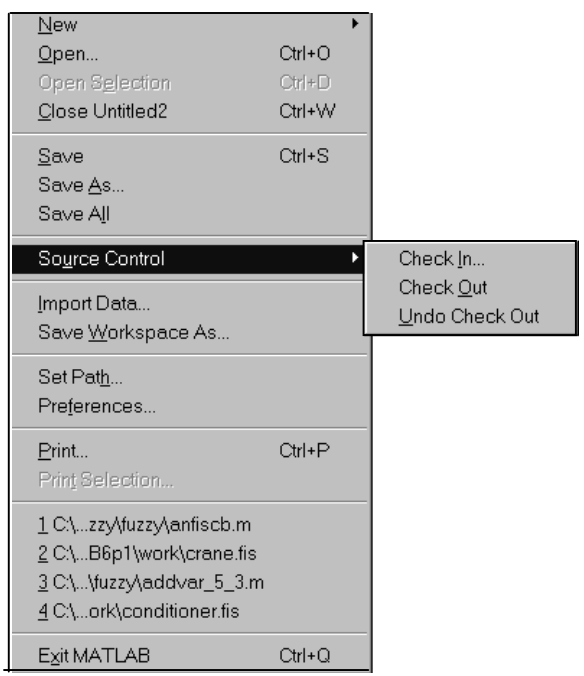


Рис. 14.4. Операции пункта меню **File** главного меню редактора-отладчика

- Пункт меню **File** (Файл) главного меню содержит следующие операции:
  - **New** — позволяет выбрать тип создаваемого нового объекта системы MATLAB и полностью совпадает с соответствующим пунктом меню рабочего интерфейса системы MATLAB, если редактор-отладчик вызван из сеанса работы с системой. Новый m-файл по умолчанию имеет имя Untitled. Если ранее в редакторе-отладчике был открыт некоторый m-файл, то при создании нового m-файла он не закрывается, а помещается в отдельное окно редактирования. Это окно можно восстановить с использованием соответствующих вкладок в нижней части рабочего интерфейса;
  - **Open...** — вызывает стандартное диалоговое окно открытия внешнего файла с диска. В этом окне можно выбрать нужный фильтр для отображения отдельных типов файлов системы MATLAB;
  - **Open Selection** — позволяет открыть внешний m-файл, предварительно выделив его имя в тексте редактируемого файла. Если файла с выбранным именем не существует, то будет выдано сообщение об ошибке;
  - **Close <имя файла>** — закрывает окно редактирования файла с соответствующим именем;
  - **Save** — сохраняет редактируемый файл под своим именем;
  - **Save As...** — вызывает диалоговое окно сохранения файла на диске с возможностью задать файлу новое имя;
  - **Save All** — сохраняет все открытые для редактирования файлы под своими именами;
  - **Source Control** — позволяет осуществлять контроль версий файлов при совместном использовании дополнительных средств конфигурационного управления (например, средства Rational ClearCase). Содержит дополнительное вложенное подменю с подпунктами: **Check In...** — установить контрольный комментарий в редактируемый файл, **Check Out...** — проверить последнюю версию файла, **Undo Check Out...** — отменить проверку последней версии файла.

  Остальные операции этого пункта меню совпадают с соответствующими операциями пункта меню **File** (Файл) рабочего интерфейса системы MATLAB (см. рис. 11.11), поэтому здесь не рассматриваются.
- Пункт меню **Edit** (Редактирование) редактора-отладчика расширен по сравнению с этим же пунктом главного меню системы MATLAB и содержит следующие операции (рис. 14.5):
  - **Undo** — отменяет выполнение последнего действия;
  - **Redo** — отменяет выполнение последней операции **Undo**;
  - **Cut** — вырезает выделенный фрагмент текста из редактируемого файла и помещает его в буфер обмена;
  - **Copy** — копирует выделенный фрагмент текста из редактируемого файла и помещает его в буфер обмена;

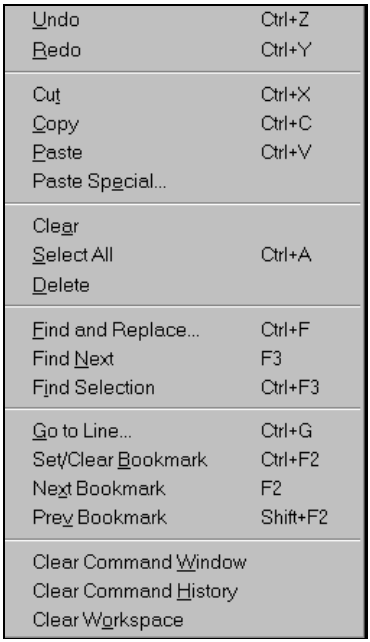


Рис. 14.5. Операции пункта меню **Edit** редактора-отладчика

- **Paste** — вставляет фрагмент текста из буфера обмена в текущую позицию редактируемого файла;
- **Paste Special...** — вызывает мастер импорта данных, который позволяет выполнить предварительный просмотр информации, хранящейся в буфере обмена;
- **Clear** — вырезает выделенный фрагмент текста из редактируемого файла без помещения его в буфер обмена;
- **Select All** — позволяет выделить всю информацию в окне редактирования;
- **Delete** — удаляет выделенный фрагмент текста из редактируемого файла;
- **Find and Replace** — позволяет найти фрагмент текста в редактируемом файле и заменить его на другой фрагмент текста;
- **Find Next** — осуществляет переход к следующему найденному фрагменту текста в редактируемом файле;
- **Find Selection** — находит вхождение последнего из заданных фрагментов текста в редактируемом файле;
- **Go to Line...** — осуществляет переход к редактированию строки с выбранным в диалоговом окне номером;

- **Set/Clear Bookmark** — позволяет задать/отменить закладку в тексте редактируемого файла для ее использования в следующих операциях;
- **Next Bookmark** — осуществляет переход к следующей по порядку закладке в редактируемом файле;
- **Prev Bookmark** — осуществляет переход к предыдущей по порядку закладке в редактируемом файле;
- **Clear Command Window** — очищает окно команд от информации текущего сеанса работы;
- **Clear Command History** — очищает окно истории команд от информации о введенных ранее командах;
- **Clear Workspace** — очищает рабочее пространство системы MATLAB от всей имеющейся в ней информации о переменных.

Пункты меню **View** (Вид), **Web** (Интернет), **Window** (Окно) и **Help** (Справка) практически идентичны соответствующим пунктам главного меню системы MATLAB, за исключением последнего, который содержит справочную информацию по разделу работы с редактором-отладчиком. Поскольку эти пункты меню были рассмотрены ранее в *главе 11*, далее описываются лишь новые пункты редактора-отладчика, отсутствующие в главном меню системы MATLAB.

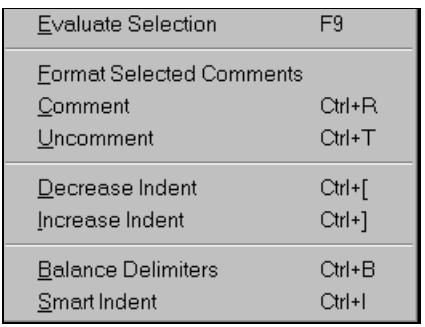


Рис. 14.6. Операции пункта меню **Text** редактора-отладчика

❒ Пункт меню **Text** (Текст) содержит следующие операции (рис. 14.6):

- **Evaluate Selection** — выполнение выделенного фрагмента текста программы;
- **Format Selected Comment** — автоматическое форматирование выделенного фрагмента или строки комментария;
- **Comment** — преобразование строки программы или выделенного фрагмента в комментарий посредством добавления слева символа `%`;
- **Uncomment** — преобразование строки или выделенного фрагмента комментариев в строку или фрагмент программы посредством удаления слева символа `%`;

- **Decrease Indent** — сдвиг строки программы или выделенного фрагмента влево на установленный интервал;
- **Increase Indent** — сдвиг строки программы или выделенного фрагмента вправо на установленный интервал;
- **Balance Delimiters** — выделение текста выражения между двумя парными разделителями (скобками);
- **Smart Indent —** автоматическое форматирование строки программы или выделенного фрагмента на установленный интервал отступа.

При работе с текстом редактируемого файла слева от каждой строки указывается ее порядковый номер. Пустой строке также присваивается отдельный номер. При отображении текста программы используется цветовое выделение различных по назначению слов. Ключевые слова, такие как операторы `if/elseif/else`, `for/while/end`, выделяются голубым цветом, комментарии — зеленым, имена переменных — черным, законченные значения строковых переменных — красным, точнее — каштановым (marron), а незаконченные, т. е. без завершающего знака "'" — пурпурным (purple).

При записи математических выражений также оказываются полезными следующие возможности редактора-отладчика. Если в тексте встречаются произвольные скобки, то для согласования количества открывающих и закрывающих скобок используется специальных маркер. Для этого достаточно установить курсор справа от одной из скобок. В этом случае будут выделены пара соответствующих скобок. В дополнение к этому двойной щелчок на произвольном символе выделяет весь текст внутри соответствующей пары скобок, что позволяет контролировать правильность задания сложных расчетных формул.

При наборе текста программы в редакторе-отладчике для визуализации циклов и условных управляющих конструкций автоматически используются отступы. В этом случае тело каждой соответствующей конструкции сдвигается вправо на установленный интервал.

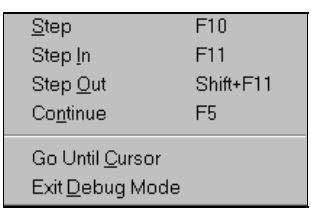


Рис. 14.7. Операции пункта меню **Debug** редактора-отладчика

❒ Пункт меню **Debug** (Отладка) содержит следующие операции (рис. 14.7):

- **Step** — выполнение одного шага построчной трассировки;
- **Step In** — пошаговая трассировка с заходом в вызываемые m-файлы;

- **Step Out** — пошаговая трассировка без захода в вызываемые m-файлы;
- **Continue** — продолжить выполнение до следующей точки останова;
- **Go Until Cursor** — продолжить выполнение до строки с установленным курсором;
- **Exit Debug Mode** — завершить отладку программы.

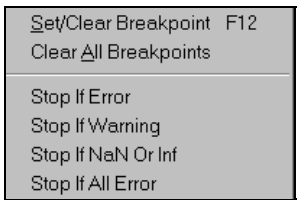


Рис. 14.8. Операции пункта меню **Breakpoints** редактора-отладчика

❐ Пункт меню **Breakpoints** (Точки останова) содержит следующие операции (рис. 14.8):

- **Set/Clear Breakpoint** — установить/отменить точку останова (прерывания);
- **Clear All Breakpoints** — отменить все точки останова;
- **Stop If Error** — прекратить отладку программы в случае обнаружения ошибки;
- **Stop If Warning** — прекратить отладку программы в случае обнаружения предупреждения;
- **Stop If NaN Or Inf** — прекратить отладку программы в случае обнаружения нечислового или бесконечного значения у какой-либо из переменных;
- **Stop If All Error** — прекратить отладку программы в случае обнаружения любой из перечисленных выше ситуаций.

Назначение операций панели инструментов

Панель инструментов редактора-отладчика содержит набор кнопок, которые дублируют наиболее часто выполняемые операции главного меню. Рассмотрим назначение отдельных кнопок панели инструментов редактора-отладчика (табл. 14.4).

Таблица 14.4. *Назначение кнопок панели инструментов редактора-отладчика*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	New M-File	Создает новую вкладку с пустым окном редактирования и именем файла Untitled, заданным по умолчанию
	Open File	Вызывает стандартное диалоговое окно открытия внешнего текстового файла с диска. При этом выбранный текстовый файл (например, m-файл), загружается в новую вкладку редактора-отладчика m-файлов
	Save	Сохраняет изменения в редактируемом файле
	Cut	Вырезает выделенный фрагмент из окна редактирования и помещает его в буфер обмена
	Copy	Копирует выделенный фрагмент из окна редактирования и помещает его в буфер обмена
	Paste	Вставляет фрагмент в позицию ввода окна редактирования из буфера обмена
	Undo	Отменяет выполнение последнего действия
	Redo	Отменяет выполнение последней операции **Undo**
	Print	Выводит на печать редактируемый файл
	Find text	Вызывает окно справочной системы MATLAB
	Show function	Позволяет установить текущий каталог (папку), содержимое которого будет отображаться в диалоговых окнах открытия и сохранения файлов на диске
	Set/clear breakpoint	Установить/отменить точку останова
	Clear all breakpoints	Отменить все точки останова
	Step	Выполнение одного шага трассировки
	Step in	Пошаговая трассировка с заходом в вызываемые m-файлы
	Step out	Пошаговая трассировка без захода в вызываемые m-файлы

Таблица 14.4 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Run	Начать отладку или выполнение программы
	Exit Debug Mode	Завершить отладку программы
	Stack	Стек функции

Кроме m-файлов, в системе MATLAB могут быть также использованы так называемые p-файлы и MEX-файлы. Первый тип файлов применяется для сокращения времени выполнения m-файлов посредством их предварительной компиляции в специальный бинарный формат. Второй тип файлов предназначен для компиляции внешних файлов программ пользователя, написанных на языках программирования C/C++ и Fortran. В обоих случаях применяются собственные средства системы MATLAB, которые здесь не рассматриваются.

14.3. Пример программы, расширяющей возможности пакета нечеткой логики Fuzzy Logic Toolbox

Пакет расширения Fuzzy Logic Toolbox содержит большое число функций, предназначенных для выполнения самых различных действий *(см. приложение 3)*. Анализ этих функций показывает, что среди них имеется специальная функция `fuzarith` для выполнения операций нечеткой арифметики с нечеткими числами и интервалами *(см. главу 5)*. В то же время отсутствует функция, которая позволяла бы выполнять базовые теоретико-множественные операции с нечеткими множествами *(см. главу 3)*.

Поэтому в качестве примера разработки m-файла рассмотрим следующий текст программы, в которой реализована функция `fuzoper`, предназначенная для выполнения теоретико-множественных операций с нечеткими множествами и расширяющая возможности системы MATLAB в этом контексте. Для работы с данной программой ее текст следует набрать в редакторе-отладчике и сохранить в рабочем каталоге пакета Fuzzy Logic Toolbox: C:\MATLAB6p1\toolbox\fuzzy\fuzzy (если система MATLAB установлена на диске C:) в файле с именем fuzoper.m.

Листинг 14.10. Пример программы, которая реализует выполнение операций с нечеткими множествами (публикуется с разрешения фирмы MathWorks Inc.)

```
function out = fuzoper(x, A, B, operator)
%  FUZOPER Нечеткие операции
%  C = FUZOPER(x, A, B, OPERATOR) возвращает нечеткое множество C
%  как результат применения операции OPERATOR для нечетких множеств A
%  и B на универсуме x.
%  A, B,и x должны быть векторами одинаковой размерности.
%  OPERATOR должен быть одной из следующих строк: 'miin', 'maun',
%  'alin', 'alun','boin', 'boun', 'drin', или  'drun'.
%  Возвращаемое нечеткое множество  C  является вектором той же
%  размерности, что  A и B.
%      Например:
%      x = 0:0.1:10;
%      A = trapmf(x, [1 2 3 5]);    % трапециевидное нечеткое множе-
%      ство  A
%      B = gaussmf(x, [2 5]);       % нечеткое множество с Гауссовой
%      функцией принадлежности  B
%      C1 = fuzoper(x, A, B, 'miin');
%      subplot(2,2,1);
%      plot(x, A, 'y--', x, B, 'm:', x, C1, 'c');
%      title('min-пересечение нечетких множеств A и B');
%      C2 = fuzoper(x, A, B, 'maun');
%      subplot(2,2,2);
%      plot(x, A, 'y--', x, B, 'm:', x, C2, 'c');
%      title('max-объединение нечетких множеств A и B');
%      C3 = fuzoper(x, A, B, 'alin');
%      subplot(2,2,3);
%      plot(x, A, 'y--', x, B, 'm:', x, C3, 'c');
%      title('алгебраическое пересечение нечетких множеств A и B');
%      C4 = fuzoper(x, A, B, 'alun');
%      subplot(2,2,4);
%      plot(x, A, 'y--', x, B, 'm:', x, C4, 'c');
%      title('алгебраическое объединение нечетких множеств A и B');
%          Copyright (c) A.B.Леоненков, 2003
%      Текст этого комментария адаптирует текст из комментария
%      файла                          fuzarith.m
%
```

```
xright=length(x);
if strcmp(operator, 'miin'),
                out = min(A,B);
   elseif strcmp(operator, 'maun'),
      out = max(A,B);
   elseif strcmp(operator, 'alin'),
      out = A.*B;
   elseif strcmp(operator, 'alun'),
      out = A+B-A.*B;
   elseif strcmp(operator, 'boin'),
      out = max(A+B-1,0);
   elseif strcmp(operator, 'boun'),
      out = min(A+B,1);
   elseif strcmp(operator, 'drin'),
      for i=1:xright
         if A(i)==1.0,
              out(i) = B(i);
          elseif B(i)==1.0,
                   out(i) = A(i);
              else out(i)=0.0;
         end
        end
   elseif strcmp(operator, 'drun'),
              for i=1:xright
         if A(i)==0.0,
              out(i) = B(i);
          elseif B(i)==0.0,
                   out(i) = A(i);
              else out(i)=1.0;
         end
        end
  else
        error('Это неизвестная операция!');
end
```

Приведем некоторые пояснения к тексту данной программы. Первая строка программы объявляет функцию `fuzoper(x, A, B, operator)`, которая должна быть вызвана для выполнения операций над нечеткими множествами. При этом переменная `x` предназначена для задания области определения соответствующих нечетких множеств `A` и `B`. Исходные нечеткие множества `A` и `B` могут быть опре-

делены произвольными функциями принадлежности, значения которых представляются в виде числовых массивов одинаковой размерности с `x`.

Для указания типа нечеткой операции предназначен параметр `operator`. При этом данный параметр может принимать одно из следующих строковых значений (сокращение от первых двух букв названия на английском):

- `'miin'` — пересечение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле min-intersection (3.3);
- `'maun'` — объединение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле max-union (3.4);
- `'alin'` — алгебраическое пересечение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле algebraic-intersection (3.19);
- `'alun'` — алгебраическое объединение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле algebraic-union (3.20);
- `'boin'` — граничное пересечение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле bound-intersection (3.31);
- `'boun'` — граничное объединение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле bound-union (3.32);
- `'drin'` — драстическое пересечение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле drastic-intersection (3.33);
- `'drun'` — драстическое объединение двух нечетких множеств $\mathcal{A}$ и $\mathcal{B}$ по формуле drastic-union (3.34).

Примечание

Если пользователь укажет другое значение параметра `operator`, отличное от перечисленных выше, то в окне команд будет выдано сообщение об ошибке: "*Unknown fuzzy set operator!*" (Неизвестный оператор для нечетких множеств). Для этой цели предназначен отдельный оператор (строка 66). Сообщение об ошибке будет выдано и в том случае, если одна из букв (или несколько) будут набраны в верхнем регистре.

Строки 2—35 являются комментарием к тексту программы и отображаются в окне команд при вызове справки по данной функции командой `help fuzoper` (до первой строки собственно тела программы).

Проверить работу этой функции можно следующим простым способом. Скопировать строки 11—31 в буфер обмена, открыть редактор-отладчик и вставить выделенный фрагмент во вновь открытом файле. После чего следует убрать символ комментария у всех строк и, сохранив данный текст во внешнем файле с расширением .m в одном из рабочих каталогов системы MATLAB, нажать кнопку **Run** (Выполнить) на панели инструментов редактора-отладчика. Вместо последней операции альтернативно можно ввести имя этого файла в окне команд.

Примечание

Данный способ тестирования функций может быть рекомендован для всех стандартных m-файлов, большинство из которых содержат примеры аналогично рассматриваемому примеру.

Результат выполнения данного тестового примера изображен на рис. 14.9. Следует заметить, что предварительно в рабочий каталог C:\MATLAB6p1\toolbox\fuzzy\ fuzzy должен быть помещен m-файл fuzoper.m с текстом рассматриваемой программы.

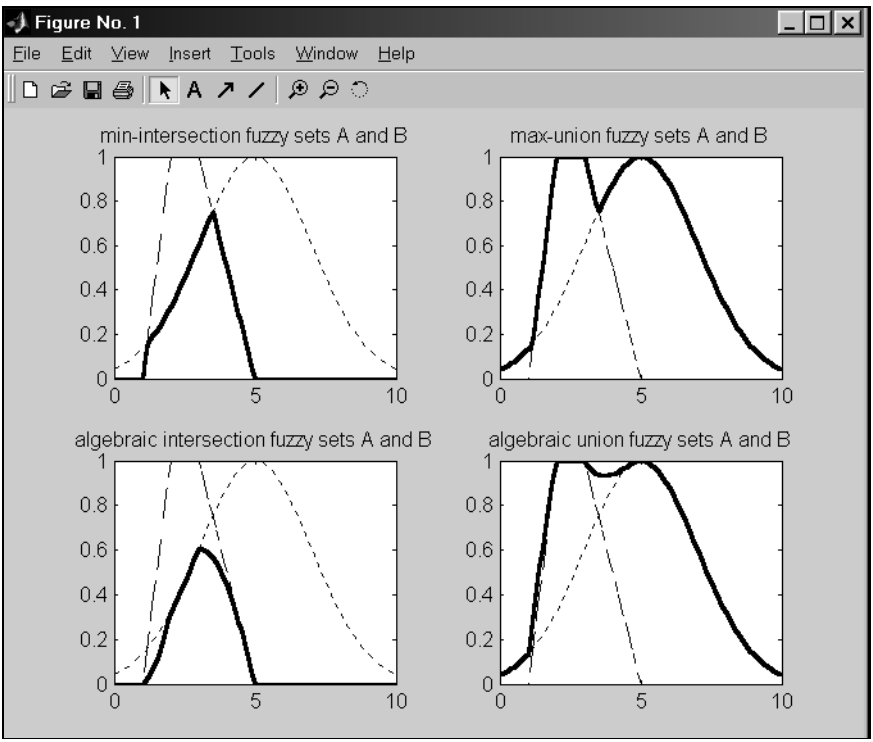


Рис. 14.9. Результат выполнения тестового примера рассматриваемой программой fuzoper.m

Собственно тело программы состоит из строк 36—69. При этом переменная с именем `xright` служит для указания общей размерности массива переменной `x`. Далее следует последовательная проверка условий на значения параметра `operator`, который определяет математическое выражение для вычисления результата соответствующей операции с нечеткими множествами. При этом для проверки совпадения двух строк используется функция `strcmp`, которая возвра-

щает значение "истина" или 1, если сравниваемые строки идентичны (с учетом регистра), и значение "ложь" или 0 — в противном случае.

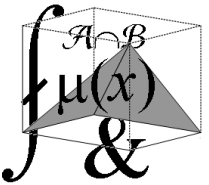
Примечание

В качестве упражнения можно рекомендовать дополнить указанную программу реализацией бинарных операций разности и симметрической разности нечетких множеств *(см. главу 3)*.

Таким образом, рассмотренный пример иллюстрирует одну из наиболее мощных и гибких возможностей системы MATLAB по расширению своей базовой функциональности за счет дополнительной разработки и включения в ее состав специальных m-файлов. Более того, представлением результатов научных и прикладных исследований в форме m-файлов достигается необходимая *унификация* в понимании этих результатов другими специалистами.

Именно по этой причине язык программирования m-файлов системы MATLAB приобретает растущую популярность в качестве независимого от вычислительной платформы, компиляторов и библиотек средства разработки и документирования вычислительных алгоритмов и процедур в различных областях теоретической и прикладной математики.

Глава 15



Основы нечетких нейронных сетей

Основным компонентом рассмотренных ранее средств системы MATLAB в рамках пакета Fuzzy Logic Toolbox является база правил нечетких продукций, которая занимает центральное место в процедурах нечеткого вывода. В то же время существуют целые классы прикладных задач, в которых выявление и построение правил нечетких продукций невозможно или связано с серьезными трудностями концептуального характера. К таким задачам относятся задачи распознавания образов, экстраполяции и интерполяции функциональных зависимостей, классификации и прогнозирования, нелинейного и ситуационного управления, а также интеллектуального анализа данных (Data Mining).

Общей особенностью подобных задач является существование некоторой зависимости или отношения, связывающего входные и выходные переменные модели системы, представляемой в форме так называемого "черного ящика" *(см. главу 1)*. При этом выявление и определение данной зависимости в явном теоретико-множественном или аналитическом виде не представляется возможным либо по причине недостатка информации о моделируемой проблемной области, либо сложности учета многообразия факторов, оказывающих влияние на характер данной взаимосвязи.

Для конструктивного решения подобных задач разработан специальный математический аппарат, получивший название *нейронных сетей*. Достоинством моделей, построенных на основе нейронных сетей, является возможность получения новой информации о проблемной области в форме некоторого прогноза. При этом построение и настройка нейронных сетей осуществляется посредством их обучения на основе имеющейся и доступной информации.

Недостатком нейронных сетей является представление знаний о проблемной области в специальном виде, которое может существенно отличаться от возможной содержательной интерпретации существующих взаимосвязей и отношений.

Нечеткие нейронные сети или гибридные сети по замыслу их разработчиков призваны объединить в себе достоинства нейронных сетей и систем нечеткого вывода. С одной стороны, они позволяют разрабатывать и представлять модели систем в форме правил нечетких продукций, которые обладают наглядностью и простотой содержательной интерпретации. С другой стороны, для построения правил нечетких продукций используются методы нейронных сетей, что является

более удобным и менее трудоемким процессом для системных аналитиков. В последнее время аппарат гибридных сетей повсеместно признается специалистами как один из наиболее перспективных для решения слабо или плохо структурированных задач прикладного системного анализа.

В настоящей главе рассматриваются основные понятия гибридных сетей, а также средства их разработки, предоставляемые системой MATLAB. В заключение приводятся примеры задач, решаемых с помощью моделей адаптивных систем нейро-нечеткого вывода (ANFIS, Adaptive Neuro-Fuzzy Inference System) в среде MATLAB.

15.1. Общая характеристика ANFIS-адаптивных систем нейро-нечеткого вывода

История исследований в области нейронных сетей берет свое начало с 1943 г., когда У. Маккалох (W. McCulloch) и У. Питтс (W. Pitts) предложили первую модель нейрона и сформулировали основные принципы функционирования головного мозга человека. Первые результаты в теории нейронных сетей связывают с появлением в период 1960-1970 гг. различных моделей искусственных нейронных сетей, которые были предложены в работах В. Уидроу (W. Widrow), К. Штайнбуха (K. Steinbuch), М. Минского (M. Minsky) и С. Пейперта (S. Papert). Однако началом современного этапа развития нейронных сетей принято считать 1982 г., когда американским математиком Дж. Хопфилдом (J. Hopfield) был разработан специальный класс нейронных сетей и предложены методы их обучения.

Рост популярности нейронных сетей связан с появлением первых коммерческих программных средств, позволяющих осуществлять построение соответствующих моделей для решения различных прикладных задач. Не является исключением и система MATLAB, в состав которой входит специальный пакет расширения Neural Network Toolbox. Этот пакет содержит средства, предназначенные для проектирования, моделирования, обучения и использования различных вариантов искусственных нейронных сетей, начиная с простейшей модели персептрона и заканчивая моделями ассоциативных и самоорганизующихся сетей.

Поскольку рассмотрение возможностей этого пакета является самостоятельной темой и выходит за рамки тематики настоящей книги, ниже описываются только особенности построения и использования гибридных сетей, реализованные в пакете Fuzzy Logic Toolbox.

Понятие нейронной сети и основные способы ее задания

К настоящему времени предложено и изучено большое количество вариантов и разновидностей нейронных сетей. Для рассмотрения моделей гибридных сетей необходимо дать некоторое общее определение нейронной сети (более точно —

искусственной нейронной сети), которое послужит основой для последующего определения модели нейро-нечеткого вывода. При этом с особенностями функционального представления биологических нейронов и общей проблематикой теории нейронных сетей можно более подробно ознакомиться в специальной литературе, которая приводится в конце книги.

Концептуальной основой и составной частью искусственных нейронных сетей является так называемый *искусственный нейрон*, который имеет определенную внутреннюю структуру (рис. 15.1) и правила преобразования сигналов.

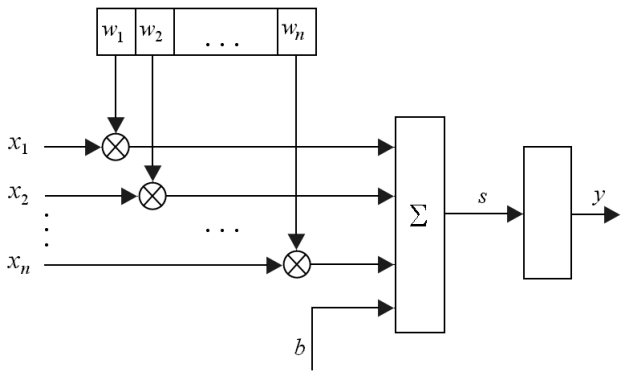


Рис. 15.1. Структура искусственного нейрона

Искусственный нейрон (далее просто — нейрон) состоит из умножителей (синапсов), сумматора и нелинейного преобразователя. *Синапсы*, изображаемые перечеркнутым кружком, предназначены для связи нейронов между собой и умножают входной сигнал x_i на некоторое постоянное число. Это число w_i, называемое *весом синапса*, характеризует силу этой связи. Сумматор выполняет сложение всех сигналов, поступающих на вход нейрона от других нейронов, и внешних входных сигналов. *Нелинейный преобразователь* предназначен для нелинейного изменения выходного значения сумматора согласно некоторой функции от одного аргумента. Эта функция называется *функцией активации* или *передаточной функцией* нейрона.

Правила преобразования сигналов определяются математической моделью нейрона, которая может быть записана в форме следующих аналитических выражений:

$$s = \sum_{i=1}^{n} w_i \cdot x_i + b; \tag{15.1}$$

$$y = f(s), \tag{15.2}$$

где w_i — вес синапса ($i \in \{1,2,\ldots, n\}$); b — значение смещения; s — результат суммирования; x_i — компонент вектора входа или входного сигнала ($i \in \{1,2,\ldots, n\}$); y — выходной сигнал нейрона; n — число входов нейрона; f — функция актива-

ции (передаточная функция) нейрона, представляющее собой некоторое нелинейное преобразование. В общем случае: w_i, x_i, $b \in \mathbb{R}$ ($i \in \{1,2,\ldots,n\}$).

Синаптические связи с положительными весами $w_i \in \mathbb{R}_+$ ($i \in \{1,2,\ldots,n\}$) называются *возбуждающими*, а с отрицательными весами $w_i \in \mathbb{R}_-$ ($i \in \{1,2,\ldots,n\}$) — *тормозящими*.

Таким образом, отдельно взятый искусственный нейрон полностью описывается своей структурой (см. рис. 15.1) и математической моделью (15.1)—(15.2). Получив вектор входного сигнала $\boldsymbol{x}$, нейрон выдает некоторое число y на своем выходе. В качестве функции активации нейрона могут быть использованы различные нелинейные преобразования (табл. 15.1).

Таблица 15.1. *Основные виды функций активации нейронов*

Название	Формула	Область значений
Линейная	$f(s) = k \cdot s$	$\mathbb{R}$
Полулинейная	$f(s) = \begin{cases} 0, & \text{если } s \le 0 \\ k \cdot s, & \text{если } s > 0 \end{cases}$	$\mathbb{R}_+$
Пороговая	$f(s) = \begin{cases} 0, & \text{если } s < T \\ 1, & \text{если } s \ge T \end{cases}$	{0, 1}
Модульная	$f(s) = \lvert s \rvert$	$\mathbb{R}_+$
Знаковая (сигнатурная)	$f(s) = \begin{cases} -1, & \text{если } s \le T \\ 1, & \text{если } s > T \end{cases}$	{–1, 1}
Квадратичная	$f(s) = s^2$	$\mathbb{R}_+$
Экспоненциальная	$f(s) = e^{-a \cdot s}$	$\mathbb{R}_+$
Синусоидальная	$f(s) = \sin(s)$	[–1, 1]
Логистическая (сигмоидальная)	$f(s) = \dfrac{1}{1+e^{-a \cdot s}}$	(0, 1)
Рациональная (сигмоидальная)	$f(s) = \dfrac{s}{a+\lvert s \rvert}$	(–1, 1)
Гиперболический тангенс (сигмоидальная)	$f(s) = \dfrac{e^{a \cdot s} - e^{-a \cdot s}}{e^{a \cdot s} + e^{-a \cdot s}}$	(–1, 1)

Таблица 15.1 *(окончание)*

Название	Формула	Область значений
Линейная с насыщением (шаговая)	$f(s)=\begin{cases}-1, & \text{если } s\le -1\\ s, & \text{если } -1<s<1\\ 1, & \text{если } s\ge 1\end{cases}$	[–1, 1]
Полулинейная с насыщением	$f(s)=\begin{cases}0, & \text{если } s\le 0\\ s, & \text{если } 0<s<1\\ 1, & \text{если } s\ge 1\end{cases}$	[0, 1]
Треугольная	$f(s)=\begin{cases}1-\lvert s\rvert, & \text{если } \lvert s\rvert\le 1\\ 0, & \text{если } \lvert s\rvert>1\end{cases}$	[0, 1]
Радиальная базисная (Гауссова)	$f(s)=e^{-s^2}$	(0, 1]

Нейронная сеть представляет собой совокупность отдельных нейронов, взаимосвязанных между собой некоторым фиксированным образом. При этом взаимосвязь нейронов определяется или задается структурой *(топологией)* нейронной сети. С точки зрения топологии нейронные сети могут быть полносвязными, многослойными и слабосвязными. В общем случае *структура* многослойной или многоуровневой нейронной сети может быть изображена следующим образом (рис. 15.2).

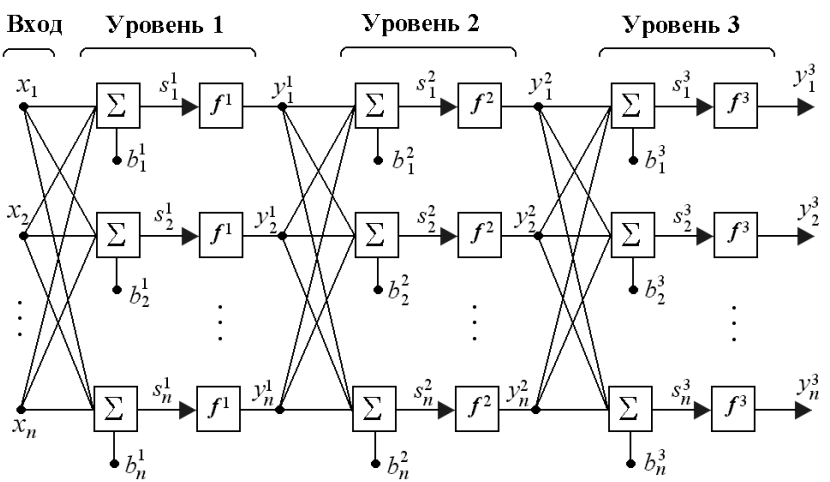


Рис. 15.2. Структура многоуровневой (трехуровневой) нейронной сети

Каждый из уровней нейронной сети называется ее *слоем*. При этом слой входного уровня называется *входным слоем*, слой уровня 1 и 2 — *скрытыми слоями*, а слой уровня 3 — *выходным слоем*.

В свою очередь многослойные нейронные сети могут быть следующих типов.

- Монотонные — каждый слой (кроме выходного) дополнительно разбивается на 2 блока: возбуждающий и тормозящий. Аналогично разбиваются и связи между блоками: на возбуждающие и тормозящие. При этом в качестве функций активации могут быть использованы только монотонные функции (см. табл. 15.1).
- Нейронные сети с обратными связями — информация с последующих слоев может передаваться на нейроны предыдущих слоев.
- Нейронные сети без обратных связей — информация с последующих слоев не может передаваться на нейроны предыдущих слоев. Классическим вариантом многослойной нейронной сети является полносвязная сеть прямого распространения (см. рис. 15.2).

Процесс построения и использования нейро-сетевых моделей состоит из следующих этапов:

1. Выбор типа и структуры нейронной сети для решения поставленной проблемы (синтез структуры нейронной сети).
2. Обучение нейронной сети (определение численных значений весов каждого из нейронов) на основе имеющейся информации о решении данной задачи экспертом или данных о решении задачи в прошлом.
3. Проверка нейронной сети на основе использования некоторого контрольного примера (необязательный этап).
4. Использование обученной нейронной сети для решения поставленной проблемы.

В настоящее время предложены различные схемы классификации нейронных сетей и соответствующие алгоритмы их обучения. Одним из самых распространенных алгоритмов обучения является так называемый алгоритм *обратного распространения ошибки* (back propagation). Этот алгоритм представляет собой итеративный градиентный алгоритм минимизации среднеквадратичного отклонения значений выхода от желаемых значений (минимизации ошибки) в многослойных нейронных сетях.

Выбор вида и структуры нейронной сети предопределяется спецификой решаемой задачи. При этом для решения отдельных типов практических задач разработаны оптимальные конфигурации нейронных сетей, которые наиболее адекватно отражают особенности соответствующей проблемной области. Дальнейшим развитием нейронных сетей являются так называемые гибридные сети, которые реализованы в пакете Fuzzy Logic Toolbox системы MATLAB.

Гибридная сеть как адаптивная система нейро-нечеткого вывода

Гибридная сеть представляет собой многослойную нейронную сеть специальной структуры без обратных связей, в которой используются обычные (не нечеткие) сигналы, веса и функции активации, а выполнение операции суммирования (15.1) основано на использовании фиксированной Т-нормы, Т-конормы или некоторой другой непрерывной операции. При этом значения входов, выходов и весов гибридной нейронной сети представляют собой вещественные числа из отрезка [0, 1].

Основная идея, положенная в основу модели гибридных сетей, заключается в том, чтобы использовать существующую выборку данных для определения параметров функций принадлежности, которые лучше всего соответствуют некоторой системе нечеткого вывода. При этом для нахождения параметров функций принадлежности используются известные процедуры обучения нейронных сетей.

В пакете Fuzzy Logic Toolbox системы MATLAB гибридные сети реализованы в форме так называемой адаптивной системы нейро-нечеткого вывода ANFIS. С одной стороны, гибридная сеть ANFIS представляет собой нейронную сеть с единственным выходом и несколькими входами, которые представляют собой нечеткие лингвистические переменные. При этом термы входных лингвистических переменных описываются стандартными для системы MATLAB функциями принадлежности, а термы выходной переменной представляются линейной или постоянной функцией принадлежности.

С другой стороны, гибридная сеть ANFIS представляет собой систему нечеткого вывода FIS типа Сугено нулевого или первого порядка, в которой каждое из правил нечетких продукций имеет постоянный вес, равный 1. В системе MATLAB пользователь имеет возможность редактировать и настраивать гибридные сети ANFIS аналогично системам нечеткого вывода, используя все рассмотренные ранее средства пакета Fuzzy Logic Toolbox.

15.2. Реализация ANFIS в среде MATLAB

В пакете Fuzzy Logic Toolbox системы MATLAB гибридные сети реализованы в форме адаптивных систем нейро-нечеткого вывода ANFIS. При этом разработка и исследование гибридных сетей оказывается возможной:

- в интерактивном режиме с помощью специального графического редактора адаптивных сетей, получившего название редактора ANFIS;
- в режиме командной строки с помощью ввода имен соответствующих функций с необходимыми аргументами непосредственно в окно команд системы MATLAB. Для работы в режиме командной строки предназначены специальные функции *(см. приложение 3)*.

Редактор ANFIS позволяет создавать или загружать конкретную модель адаптивной системы нейро-нечеткого вывода, выполнять ее обучение, визуализиро-

вать ее структуру, изменять и настраивать ее параметры, а также использовать настроенную сеть для получение результатов нечеткого вывода.

Графический интерфейс редактора ANFIS вызывается функцией `anfisedit` из командной строки (рис. 15.3).

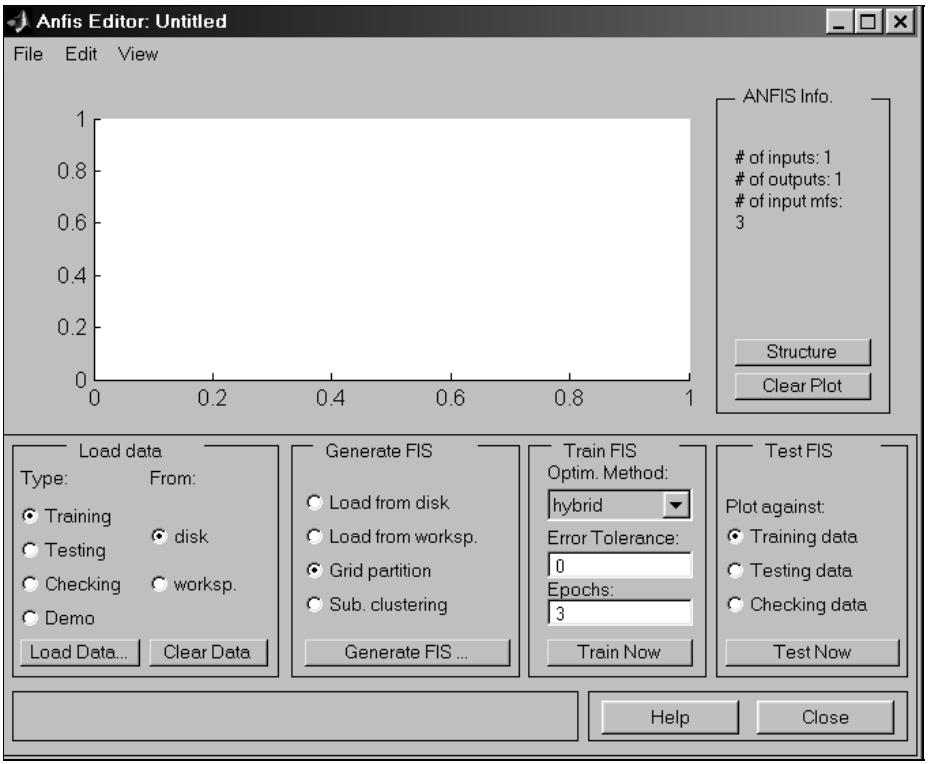


Рис. 15.3. Графический интерфейс редактора ANFIS, вызываемый функцией `anfisedit`

Главное меню редактора ANFIS достаточно простое и предназначено для работы с предварительно созданной системой нечеткого вывода. Основную часть графического интерфейса занимает окно визуализации данных, которое расположено ниже главного меню. Для вновь создаваемой гибридной сети это окно не содержит никаких данных.

Для создания гибридной сети необходимо загрузить данные. Для этой цели следует воспользоваться кнопкой **Load Data** в левой нижней части графического окна. При этом данные могут быть загружены из внешнего файла **(disk)** или из рабочей области **(worksp.)**. В первом случае необходимо предварительно создать файл с исходными данными (файл с расширением .dat), который представляет собой обычный текстовый файл. При этом исходные данные представляют со-

бой обычную числовую матрицу размерности $m \times (n+1)$, в которой количество строк m соответствует объему выборки, первые n столбцов — значениям входных переменных модели, а последний столбец — значению выходной переменной. Согласно правилам системы MATLAB отдельные значения матрицы отделяются пробелами, а каждая строка матрицы завершается символом "*перевод каретки*" (клавиша <Enter>).

Примечание

Хотя по количеству строк матрицы исходных данных не существует формальных рекомендаций, принято считать, что качество обучения гибридной сети, а, следовательно, и точность получаемых результатов пропорционально зависит от объема обучающей выборки. Что касается количества столбцов матрицы исходных данных, то следует отметить возможные проблемы с работоспособностью системы MATLAB, если количество входных переменных превышает 5—6.

Загружаемые исходные данные могут быть одного из следующих типов:

- обучающие данные **(Training)** — обязательные данные, которые используются для построения гибридной сети;
- тестовые данные **(Testing)** — необязательные данные, которые используются для тестирования построенной гибридной сети с целью проверки качества функционирования построенной гибридной сети;
- проверочные данные **(Checking)** — необязательные данные, которые используются для проверки построенной гибридной сети с целью выяснения факта переобучения сети;
- демонстрационные данные **(Demo)** — позволяют загрузить один из демонстрационных примеров гибридной сети.

Так, например, в папке C:\MATLAB6p1\toolbox\fuzzy\fuzdemos имеется два файла fuzex1trnData.dat и fuzex1chkData.dat, которые могут быть использованы для иллюстрации приемов разработки гибридных сетей с помощью редактора ANFIS. Первый из этих файлов содержит обучающие данные, для редактирования которых может быть использован любой текстовый редактор, например, блокнот (рис. 15.4) или редактор-отладчик m-файлов системы MATLAB.

После загрузки обучающих данных из файла fuzex1trnData.dat их структура будет отображена в рабочем окне редактора ANFIS (рис. 15.5).

При этом каждой строке данных соответствует отдельная точка графика, которая для обучающих данных изображается кружком. На горизонтальной оси указываются порядковый номер (индекс) отдельной строки данных, а вертикальная ось служит для указания значений выходной переменной. В случае рассматриваемого примера используется 25 точек обучающих данных.

Второй из файлов содержит проверочные данные, для редактирования которых также может быть использован любой текстовый редактор. Поскольку тестовые данные для данного примера отсутствуют, то в редактор ANFIS следует загрузить проверочные данные из файла fuzex1chkData.dat. Для этого необходимо

изменить тип загружаемых данных (**Checking**) и выбрать имя указанного файла при загрузке данных. В результате проверочные данные будут загружены в редактор ANFIS и изображены в рабочем окне (рис. 15.6).

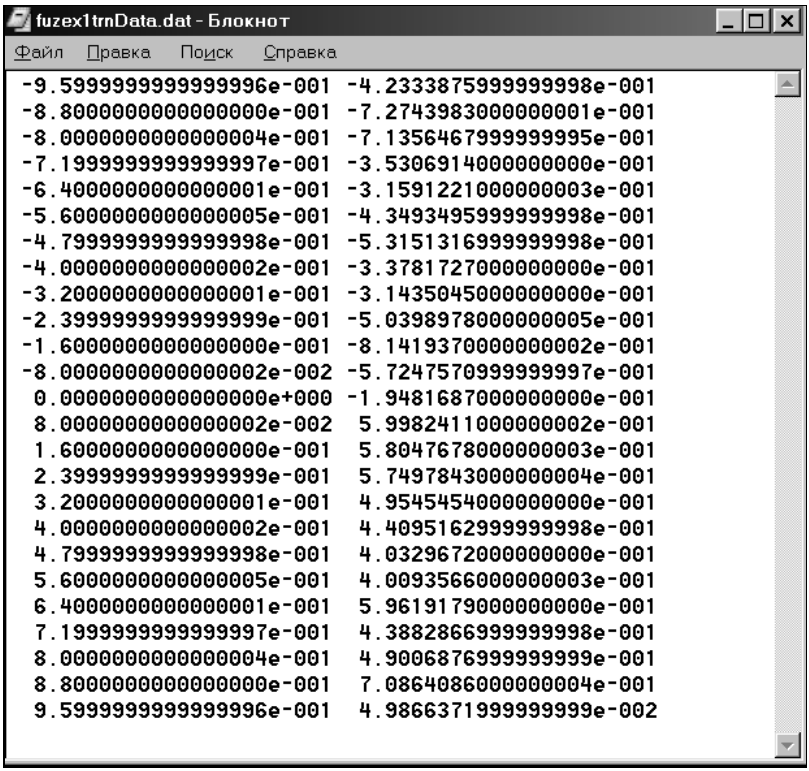


Рис. 15.4. Обучающие данные для демонстрации возможностей обучения гибридной сети ANFIS

В этом случае исходный график будет дополнен 26 точками проверочных данных, каждой строке которых также соответствует отдельная точка графика, изображаемая плюсом. На горизонтальной оси указываются порядковый номер (индекс) отдельной строки данных, а вертикальная ось служит для указания значений выходной переменной. В случае рассматриваемого примера используется 25 точек обучающих данных.

После подготовки и загрузки обучающих данных можно сгенерировать структуру системы нечеткого вывода FIS типа Сугено, которая является моделью гибридной сети в системе MATLAB. Для этой цели следует воспользоваться кнопкой **Generate FIS** в нижней части рабочего окна редактора. При этом две первые опции относятся к предварительно созданной структуре гибридной сети, а две последних — к форме разбиения входных переменных модели.

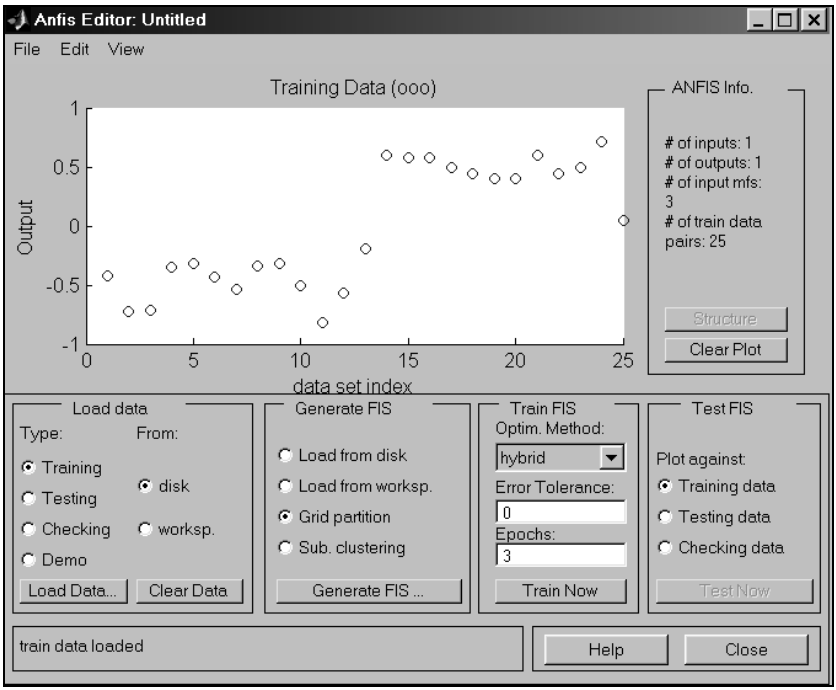


Рис. 15.5. Графический интерфейс редактора ANFIS после загрузки файла с обучающими данными fuzex1trnData.dat

Загрузить структуру уже созданной FIS можно либо с диска **(Load from disk)**, либо из рабочей области **(Load from worksp.)**. При создании структуры новой FIS можно независимо разбить все входные переменные на области их значений **(Grid partition)** или воспользоваться процедурой субтрактивной кластеризации для предварительного разбиения значений входных переменных на кластеры близких значений **(Sub. clustering)**.

После нажатия кнопки **Generate FIS** вызывается диалоговое окно с указанием числа и типа функций принадлежности для отдельных термов входных переменных и выходной переменной (рис. 15.7). В этом случае можно выбрать любой тип функций принадлежности из реализованных в системе MATLAB и рассмотренных ранее *(см. главу 12)*.

После генерации структуры гибридной сети можно визуализировать ее структуру, для чего следует нажать кнопку **Structure** в правой части графического окна. Структура полученной в результате системы нечеткого вывода FIS отображается в отдельном окне и достаточно тривиальна по своему виду (рис. 15.8).

Для рассматриваемого примера система нечеткого вывода содержит одну входную переменную с 4 термами, 4 правила нечетких продукций, одну выходную переменную с 4 термами. Компоненты системы FIS изображаются узлами соответствующего цвета. При этом узел с нормализующим коэффициентом для правил на рис. 15.8 не указан.

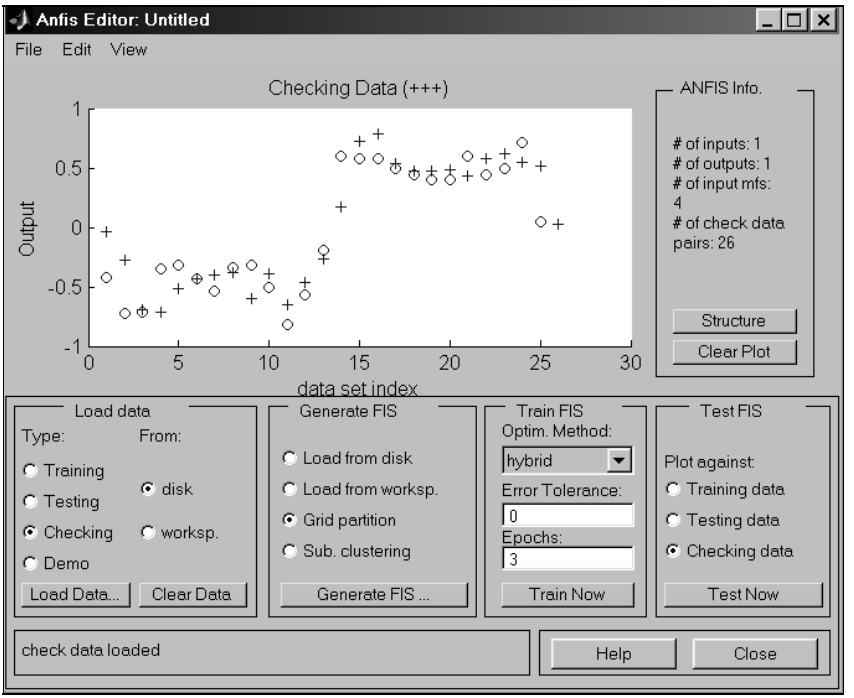


Рис. 15.6. Графический интерфейс редактора ANFIS после загрузки файла с проверочными данными fuzex1chkData.dat

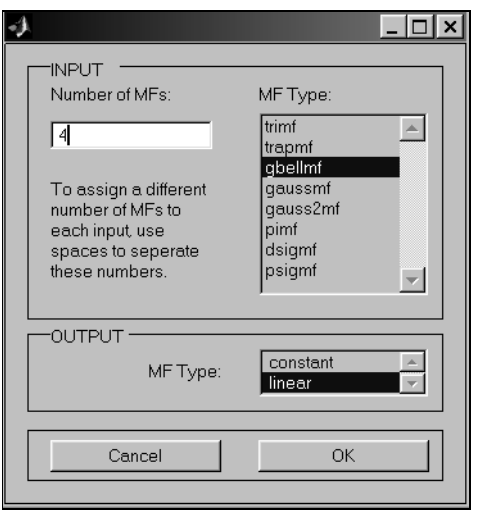


Рис. 15.7. Диалоговое окно для задания количества и типа функций принадлежности

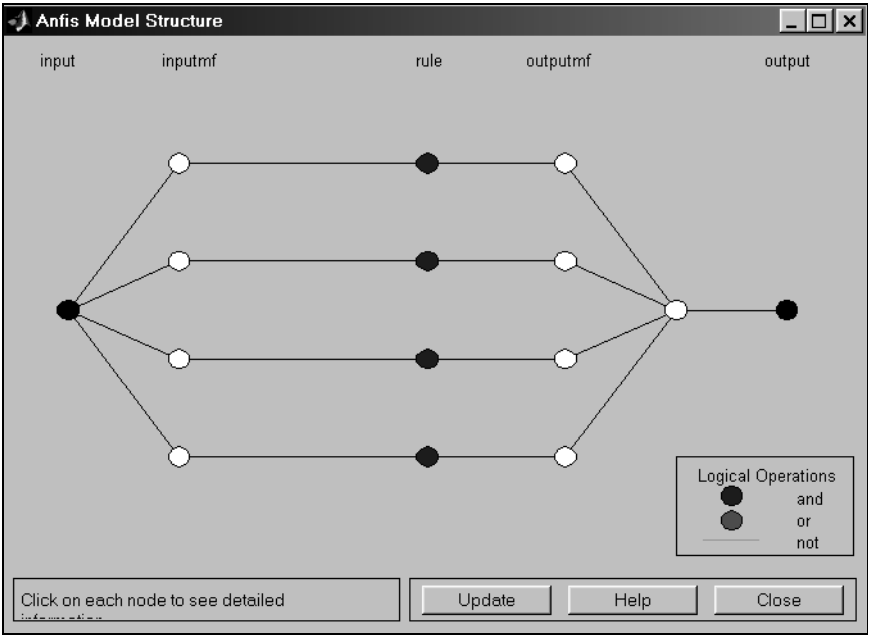


Рис. 15.8. Структура сгенерированной системы нечеткого вывода

Перед обучением гибридной сети необходимо задать параметры обучения, для чего следует воспользоваться следующей группой опций в правой нижней части рабочего окна:

1. Выбрать метод обучения гибридной сети — обратного распространения (**backpropo**) или гибридный (**hybrid**), представляющий собой комбинацию метода наименьших квадратов и метода убывания обратного градиента.
2. Установить уровень ошибки обучения (**Error Tolerance**) — по умолчанию значение 0 (изменять не рекомендуется).
3. Задать количество циклов обучения (Epochs) — по умолчанию значение 3 (рекомендуется увеличить и для рассматриваемого примера задать его значение равным 40).

Для обучения сети следует нажать кнопку **Train Now**. При этом ход процесса обучения иллюстрируется в окне визуализации в форме графика зависимости ошибки от количества циклов обучения (рис. 15.9).

В этом случае на верхнем графике изображена зависимость ошибки проверки от количества циклов обучения, а на нижнем графике — зависимость ошибки обучения от количества циклов обучения (знаком "*").

Аналогично могут быть выполнены дополнительные этапы тестирования и проверки гибридной сети, для которых необходимо предварительно загрузить соответствующие данные.

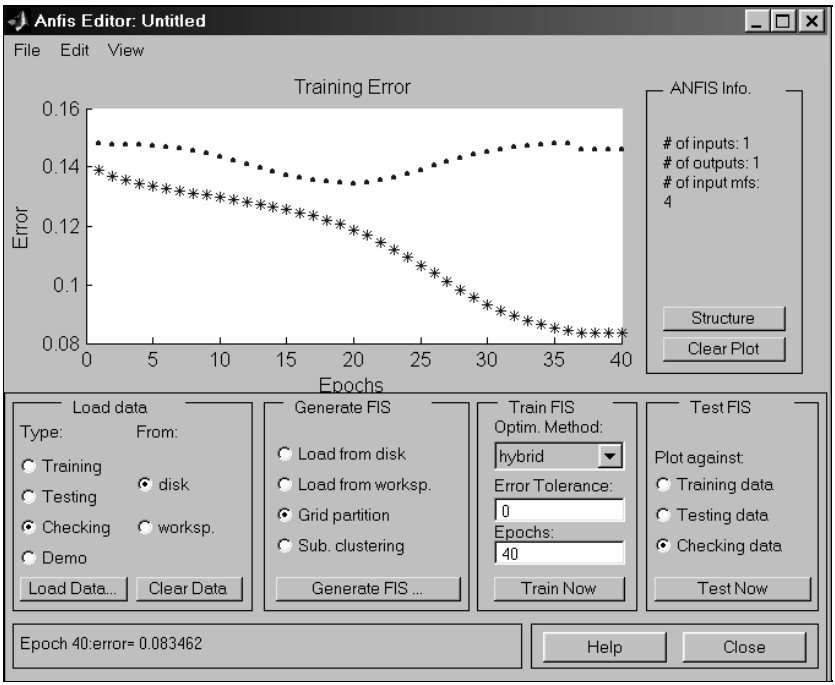


Рис. 15.9. График зависимости ошибок обучения и проверки от количества циклов обучения

Дальнейшая настройка параметров построенной и обученной гибридной сети может быть выполнена с помощью рассмотренных ранее стандартных графических средств пакета Fuzzy Logic Toolbox *(см. главу 12)*. Для этого рекомендуется сохранить созданную систему нечеткого вывода во внешнем файле с расширением fis, после чего следует загрузить этот файл в редактор систем нечеткого вывода FIS (рис. 15.10).

При этом также становятся доступными редактор функций принадлежности системы нечеткого вывода (Membership Function Editor), редактор правил системы нечеткого вывода (Rule Editor), программа просмотра правил системы нечеткого вывода (Rule Viewer) и программа просмотра поверхности системы нечеткого вывода (Surface Viewer).

Примечание

В последних версиях системы MATLAB вызов графических средств редактирования и просмотра моделей FIS возможен непосредственно из главного меню редактора ANFIS.

Для исследования построенной модели гибридной сети можно воспользоваться программой просмотра правил (Rule Viewer). Для получения интересующего значения выходной переменной необходимо задать конкретное значение входной пе-

ременной (например, 0.5) аналогично общим рекомендациям систем нечеткого вывода. При этом на графике функций принадлежности выходной переменной будет указано искомое значение выходной переменной — 0.451 (рис. 15.11).

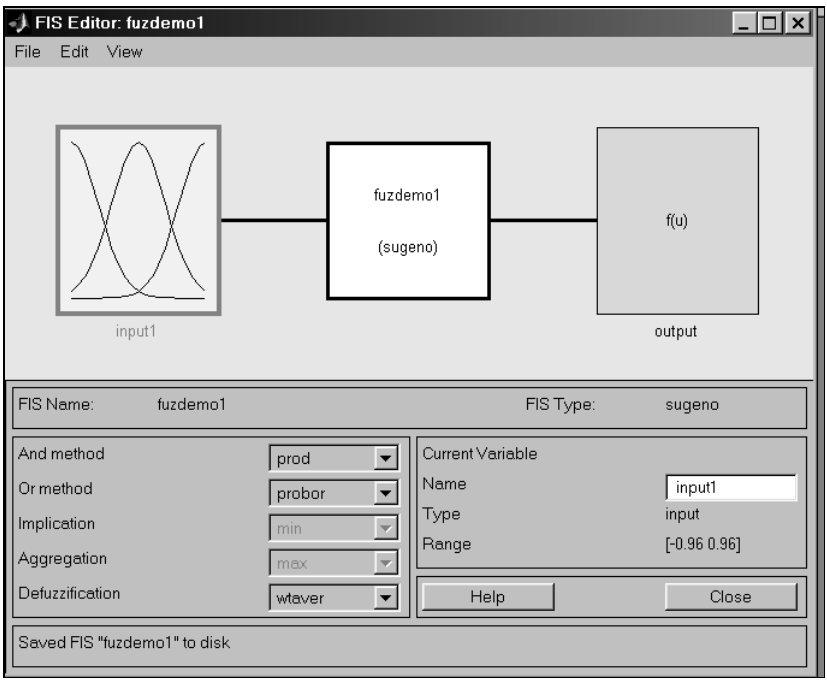


Рис. 15.10. Графический интерфейс редактора FIS для сгенерированной системы нечеткого вывода

В дополнение к этому можно выполнить визуальный анализ поверхности вывода для построенной гибридной сети, которая также позволяет оценить значения выходной переменной. Выполнить анализ обученной и настроенной гибридной сети можно посредством визуализации поверхности нечеткого вывода. Для этой цели следует воспользоваться программой просмотра поверхности системы нечеткого вывода (рис. 15.12).

Изображенная поверхность вывода может быть интерпретирована как график функциональной зависимости выходной переменной от входной для рассматриваемого примера табличного задания соответствующих пар значений. Как можно заключить из анализа данного графика, эта зависимость характеризует некоторый тренд.

Более подробно содержание отдельных этапов построения, обучения и настройки гибридных сетей ANFIS будет описано далее при рассмотрении примера построения адаптивной системы нейро-нечеткого вывода для решения задачи прогнозирования на финансовом рынке.

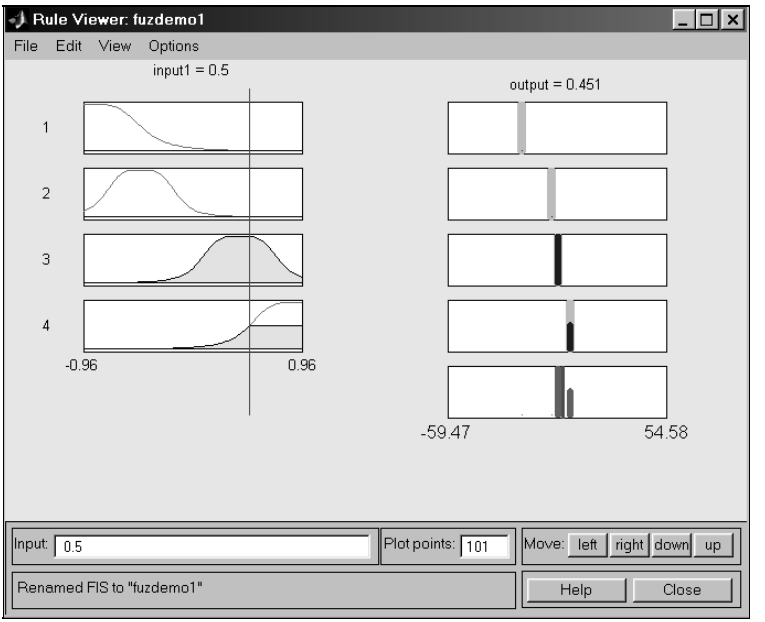


Рис. 15.11. Графический интерфейс просмотра правил сгенерированной системы нечеткого вывода

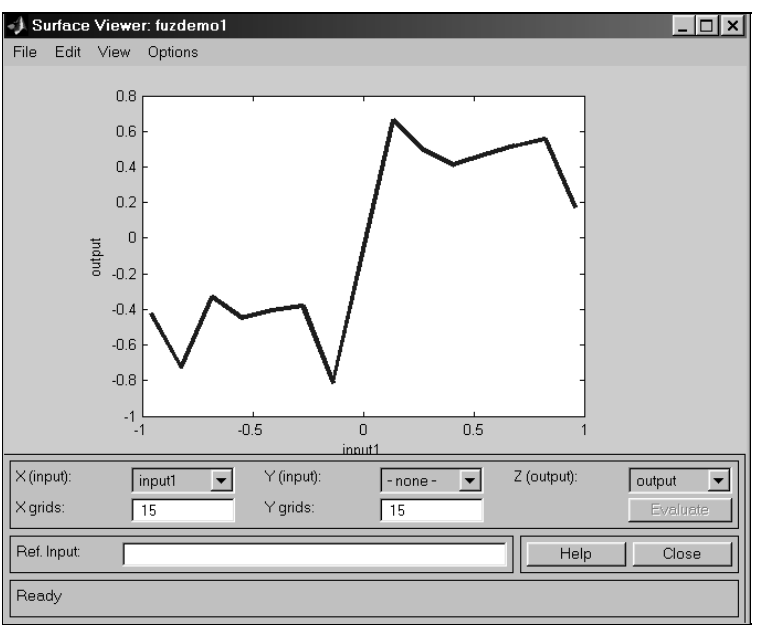


Рис. 15.12. Графический интерфейс просмотра поверхности сгенерированной системы нечеткого вывода

15.3. Пример решения задачи нейро-нечеткого вывода

Для иллюстрации процесса разработки гибридной сети в системе MATLAB рассмотрим задачу построения адаптивной системы нейро-нечеткого вывода для аппроксимации некоторой зависимости, которая описывается математической функцией $y=x^3$. Этот пример позволяет не только уточнить содержание и последовательность этапов разработки, но и оценить точность полученной нечеткой модели посредством сравнения прогнозируемых модельных значений с известными заранее значениями соответствующей функции.

Общая последовательность процесса разработки модели гибридной сети может быть представлена в следующем виде.

1. Для начала с помощью редактора-отладчика m-файлов подготовим обучающие данные, которые содержат 9 строк пар "значение входной переменной — значение выходной переменной" следующего вида (рис. 15.13). Сохраним обучающие данные во внешнем файле с именем function.dat.

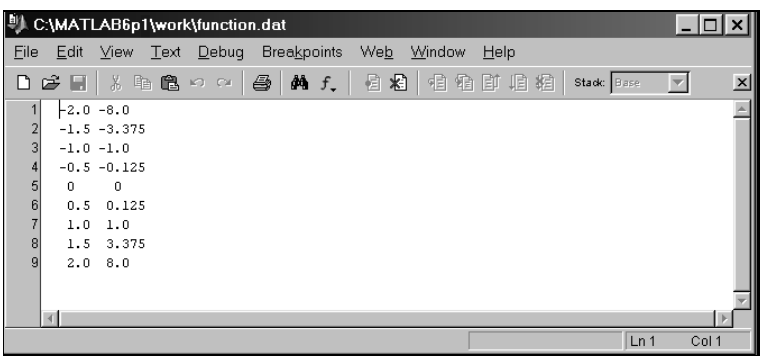


Рис. 15.13. Обучающие данные для примера построения гибридной сети ANFIS, представляющей функцию $y=x^3$

2. Далее загрузим этот файл с обучающими данными в редактор ANFIS (рис. 15.14). В рабочем окне редактора будет изображен график, форма которого аналогична исходной математической функции.
3. Поскольку в данном случае отсутствуют тестовые и проверочные данные, можно сразу приступить к генерации структуры системы нечеткого вывода FIS. Установив параметры генерации аналогично ранее рассмотренному демонстрационному примеру (рис. 15.7), получим структуру FIS, вид которой также совпадает с изображенной на рис. 15.8.
4. Теперь можно перейти к обучению сгенерированной системы нечеткого вывода. Для этого оставим без изменения предложенные системой MATLAB по умолчанию метод обучения (гибридный) и уровень ошибки (0), а количество циклов обучения изменим на 40. После обучения сети в рабочем окне редак-

тора ANFIS будет изображен график изменения ошибки в ходе выполнения отдельных циклов обучения (рис. 15.15).

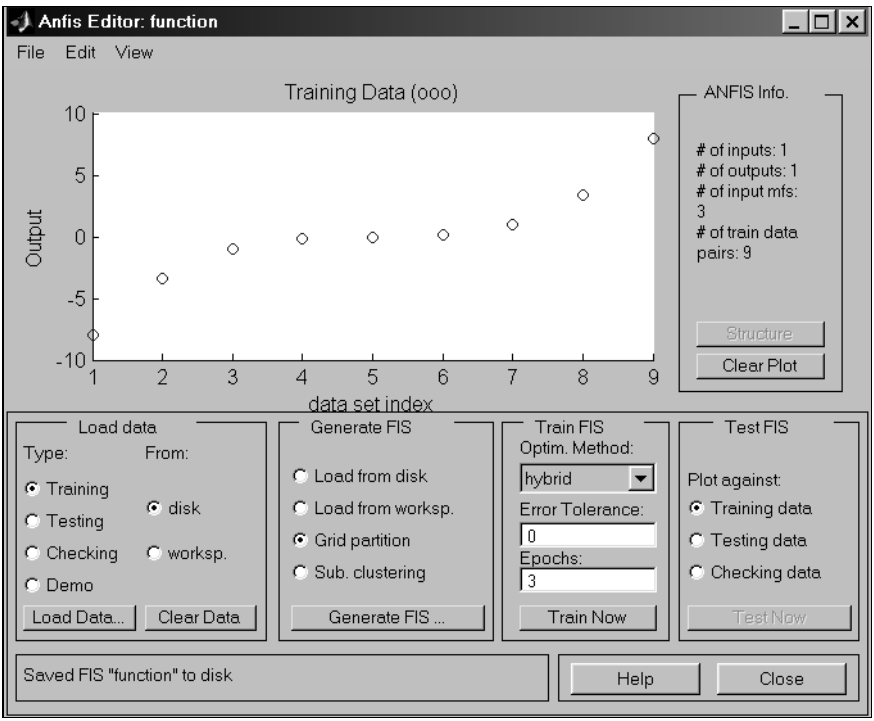


Рис. 15.14. Графический интерфейс редактора ANFIS после загрузки файла function.dat с обучающими данными

5. Выполнить анализ точности построенной нечеткой модели гибридной сети можно с помощью просмотра поверхности соответствующей системы нечеткого вывода (рис. 15.16).

Визуальный анализ изображенного графика с точным графиком функции $y=x^3$ позволяет судить о достаточно высокой степени их совпадения, что может свидетельствовать об адекватности построенной нечеткой модели гибридной сети.

Анализ адекватности построенной модели можно выполнить с помощью просмотра правил соответствующей системы нечеткого вывода (рис. 15.17).

Проверка построенной модели гибридной сети может быть выполнена для нескольких значений выходной переменной. С этой целью необходимо ввести конкретное значение в поле ввода **Input** (например, значение 1.1), после нажатия клавиши <Enter> с помощью построенной модели будет получено соответствующее значение выходной переменной (в данном случае значение 1.23). Сравнивая полученное значение с точным значением функции 1.331), получим относительную ошибку порядка 8%.

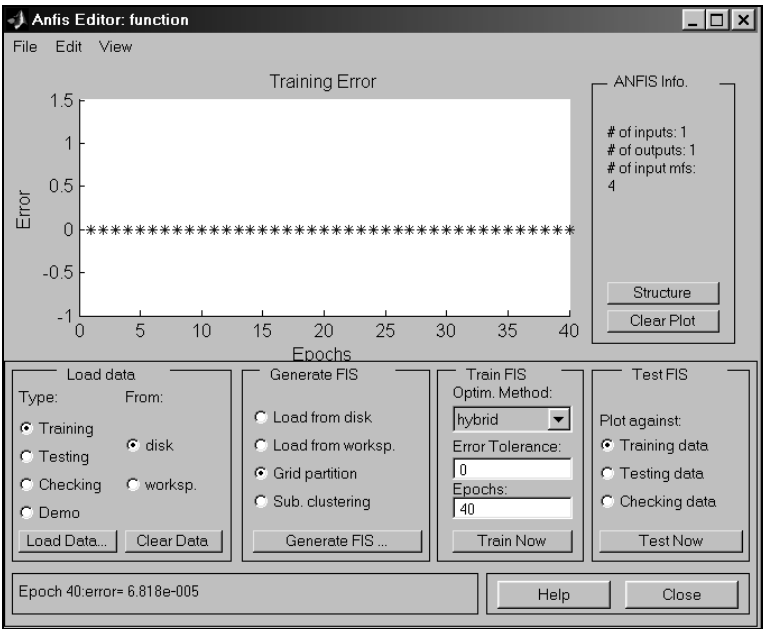


Рис. 15.15. График зависимости ошибки обучения от количества циклов обучения

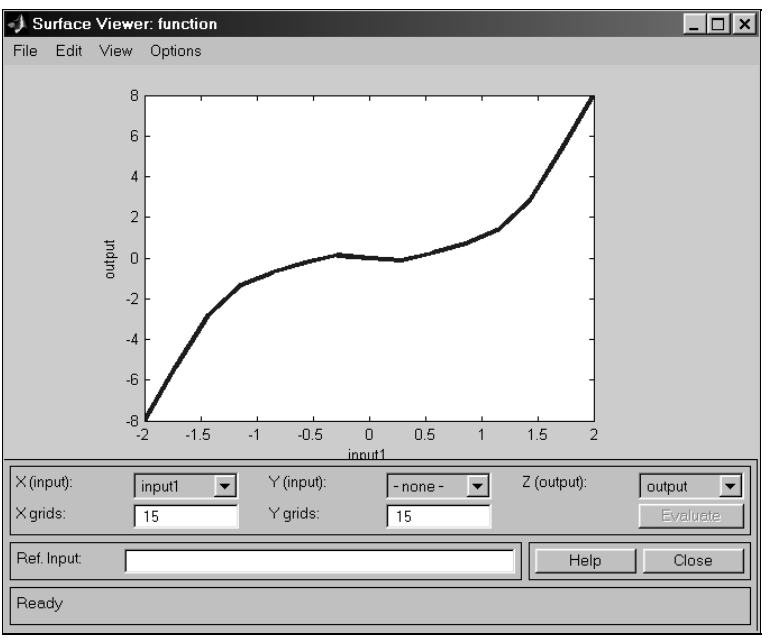


Рис. 15.16. Графический интерфейс просмотра поверхности сгенерированной системы нечеткого вывода

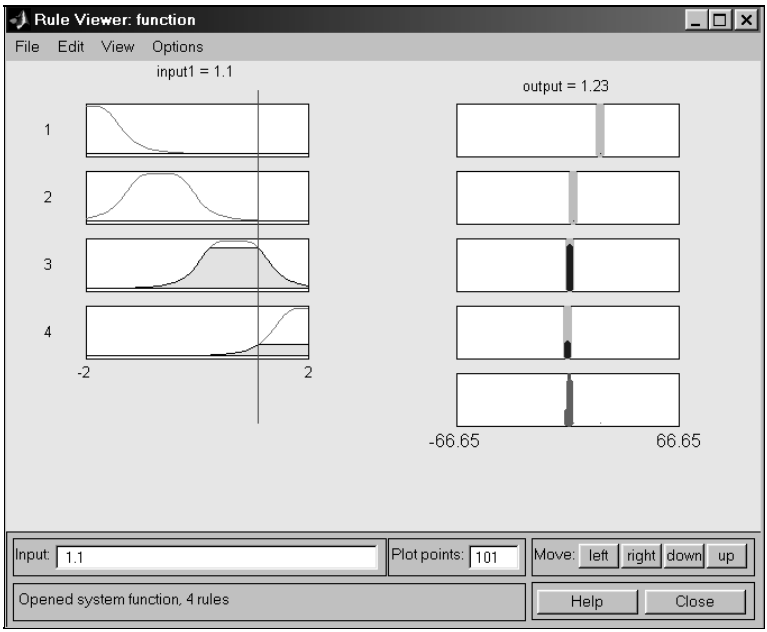


Рис. 15.17. Графический интерфейс просмотра правил сгенерированной системы нечеткого вывода

Менее удачной оказывается проверка для значения входной переменной 0.1, для которого построенная модель предлагает отрицательное значение –0.148. Очевидно, данный факт свидетельствует не в пользу адекватности построенной нечеткой модели и требует ее дополнительной настройки.

В общем случае дополнительная настройка модели может быть выполнена несколькими возможными способами. Наиболее приемлемыми из них представляются следующие.

1. Подготовка и загрузка большего по объему выборки файла с обучающими исходными данными.
2. Подготовка и загрузка дополнительного файла с проверочными исходными данными, сформированными для пар значений рассматриваемой математической функции, отсутствующих в выборке обучающих данных.
3. Редактирование типов и значений параметров функций принадлежности термов входной и выходной переменных с помощью редактора функций принадлежности системы MATLAB.

Проиллюстрируем третий способ дополнительной настройки построенной нечеткой модели гибридной сети. На первый взгляд он представляется наиболее естественным с точки зрения возможности визуального контроля выполняемых изменений параметров. С этой целью откроем редактор функций принадлежности и методом подбора изменим количественные значения параметров второй и третьей функции принадлежности входной переменной, поскольку именно они

"работают" при получении некорректного значения выходной переменной: –0.148 для значения входной переменной 0.2. (рис. 15.18).

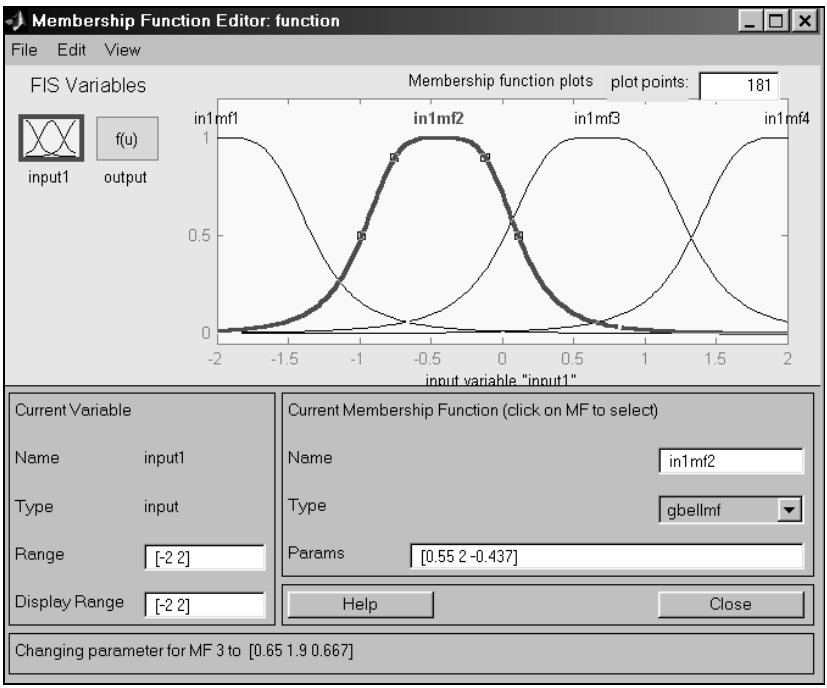


Рис. 15.18. Графический интерфейс редактора функций принадлежности построенной системы нечеткого вывода

После изменения параметров в поле **Params** для второй функции принадлежности на значения [0.55 2 –0.437] и для третьей функции принадлежности на значения [0.65 1.9 0.667] получим практически точное значение выходной переменной для исходного значения входной переменной 0.2 (рис. 15.19).

Примечание

Данный способ редактирования параметров функций принадлежности не всегда приводит к желаемому результату, поскольку оказывает влияние на другие, возможно правильные, значения выходной переменной. В этом можно убедиться, введя значение входной переменной, равное 0.1. Поэтому разработанная нечеткая модель не может быть признана адекватной и требует дополнительной модификации.

Более эффективным способом настройки, а точнее — модификации, данной нечеткой модели оказывается первый. С этой целью увеличим объем обучающей выборки до 17 пар значений, который сохраним в новом файле с именем function2.dat (рис. 15.20).

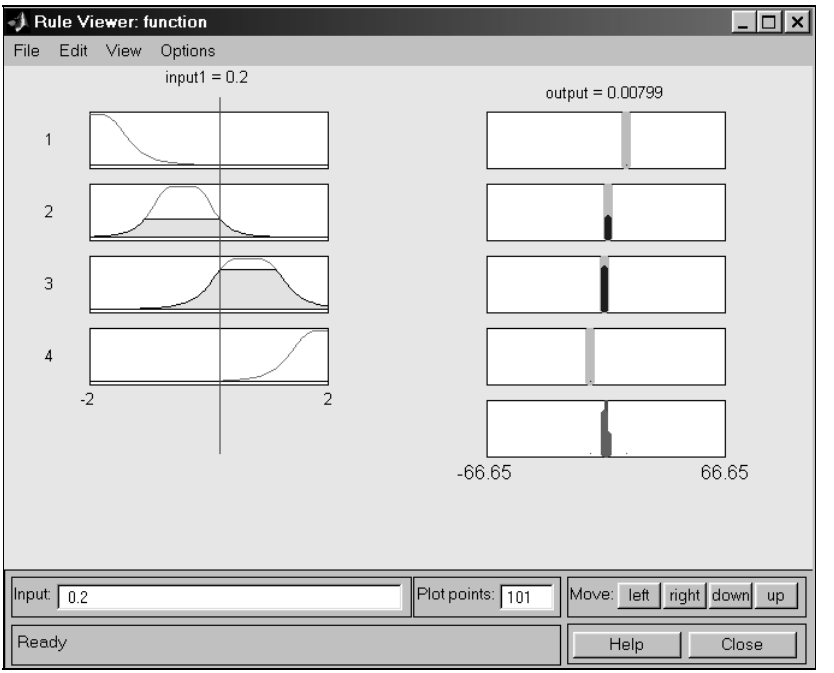


Рис. 15.19. Результат ручной настройки параметров функций принадлежности входной переменной для рассматриваемой системы нечеткого вывода

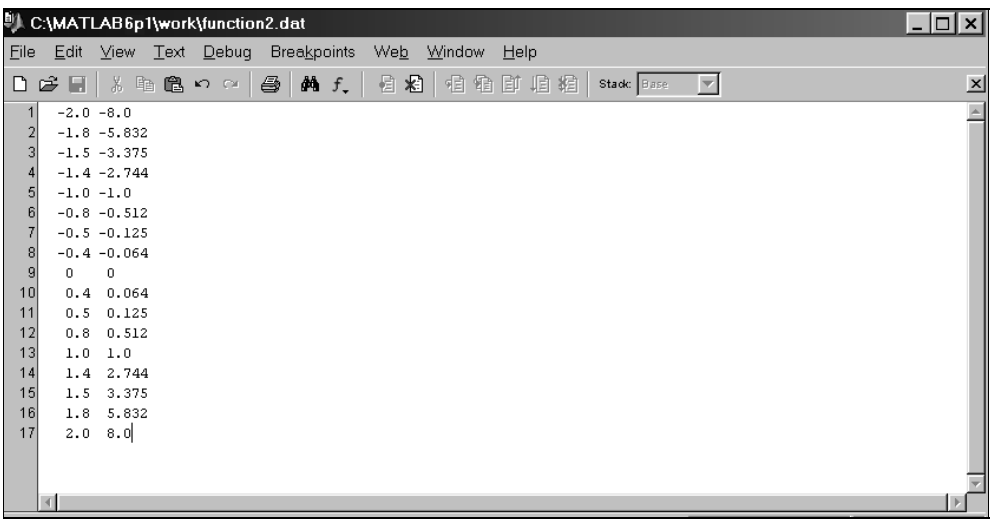


Рис. 15.20. Обучающие данные увеличенного объема для построения гибридной сети ANFIS, представляющей функцию $y=x^3$

После удаления всех ранее загруженных данных кнопкой **Clear Data** загрузим новый файл с обучающей выборкой. При генерации структуры новой FIS увеличим количество термов и, соответственно, количество функций принадлежности входной переменной до 5, оставив их тип без изменения (**gbellmf**). Процесс обучения выполним аналогично ранее рассмотренному. В результате будет получена новая система нечеткого вывода FIS, анализ которой показывает, что по сравнению с первым вариантом нечеткой модели она более точно описывает исходную математическую функцию (рис. 15.21).

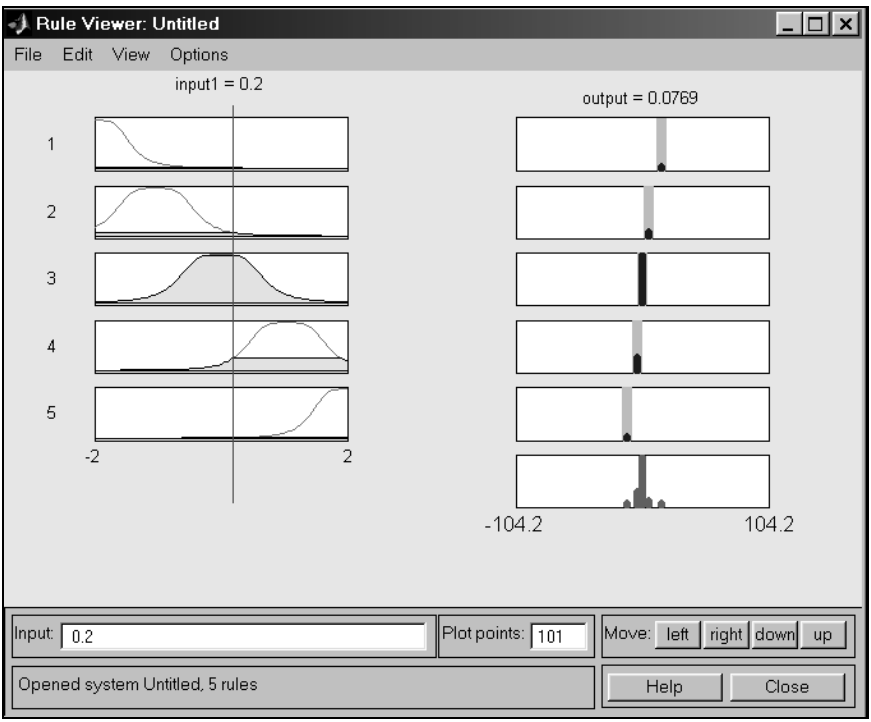


Рис. 15.21. Результат построения новой модели рассматриваемой системы нечеткого вывода

Действительно, для значений входной переменной 0.1, 0.2 и 0.3 с помощью данной модели будут получены значения выходной переменной, равные 0.0425, 0.0769 и 0.0956 соответственно. Для сравнения точные значения выходной переменной для указанных значений входной переменной равны 0.001, 0.008 и 0.027 соответственно. В некоторой степени улучшение качества вновь построенной нечеткой модели иллюстрируется графиком поверхности (рис. 15.22). который более точно по сравнению с ранее полученным графиком (рис. 15.16) соответствует исходной математической функции.

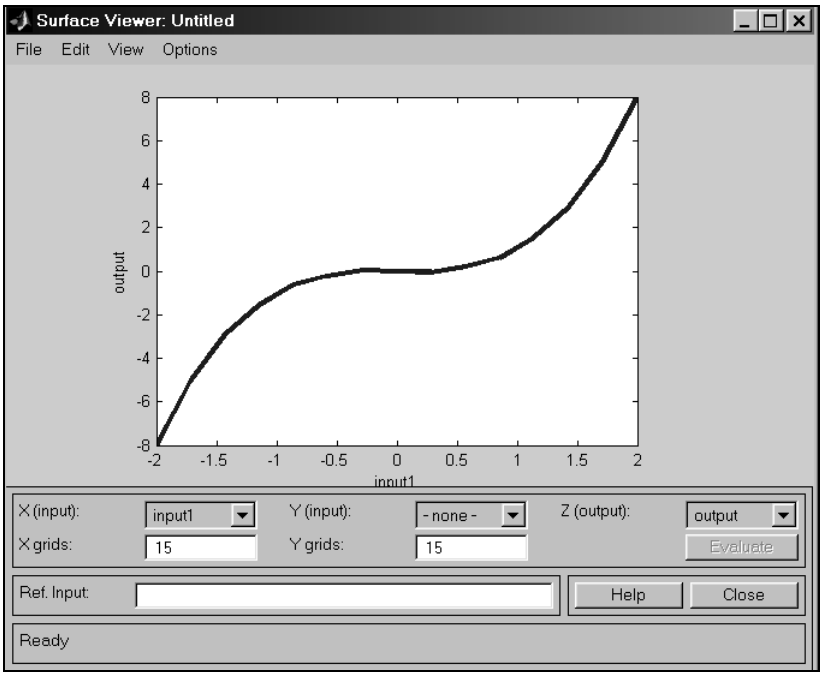


Рис. 15.22. График поверхности вновь сгенерированной и обученной системы нечеткого вывода

Хотя полученные значения все еще отличаются от точных значений исходной математической функции, выполненная модификация нечеткой модели позволила исключить явно ошибочные значения выходной переменной, связанные со знаком. Очевидно, проверка адекватности нечеткой модели для значений в интервалах (–0.4, 0) и (0, 0.4) заведомо обречена на возможные проблемы, поскольку в этих интервалах отсутствуют значения среди обучающих данных.

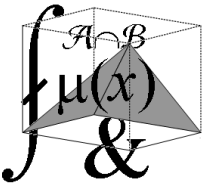
Исключение подобных пропусков в обучающих данных может служить основой для генерации новой нечеткой модели, которую предлагается читателям выполнить самостоятельно в качестве упражнения.

В заключение следует отметить, что даже простейшие рассмотренные примеры отражают творческий характер процесса построения и анализа моделей гибридных сетей. При этом выбор того или иного способа дополнительной настройки нечетких моделей зависит не только от специфики решаемой задачи, но и от объема доступной выборки обучающих и проверочных данных.

В случае недостаточной информации обучающих данных использование гибридных сетей может оказаться вообще нецелесообразным, поскольку получить адекватную нечеткую модель, а значит — и точный прогноз значений выходной переменной не представляется возможным.

Именно по этим причинам необходим предварительный анализ всех возможностей применяемых нечетких моделей для решения конкретных задач в той или иной проблемной области. Подобный анализ необходимо выполнять с системной точки зрения и с учетом всех складывающихся на данный момент обстоятельств. Только всесторонняя и полная оценка проблемной ситуации позволит разработать адекватную модель решения той или иной конкретной задачи нечеткого управления или принятия решений.

Глава 16



Примеры разработки нечетких моделей управления в среде MATLAB

Настоящая глава посвящена рассмотрению двух конкретных примеров систем нечеткого вывода в задачах управления и построению соответствующих нечетких моделей с использованием средств MATLAB. При этом в качестве первого примера используется задача управления кондиционером в помещении, а в качестве второго — задача управления контейнерным краном. Содержательные постановки этих задач были сформулированы ранее в *главе 7*.

16.1. Нечеткая модель управления кондиционером воздуха в помещении

Напомним, что эта задача связана с процессом управления температурой воздуха в помещении, в котором установлен бытовой кондиционер. Суть задачи состоит в том, чтобы сделать регулировку кондиционера автоматической, обеспечивая постоянную температуру воздуха в помещении.

Для решения этой задачи в *разд. 7.4* была построена база правил соответствующей системы нечеткого вывода, которая содержит 15 правил нечетких продукций следующего вида:

ПРАВИЛО_1: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* PS" ТО "β_3 *есть* NB"

ПРАВИЛО_2: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* NS" ТО "β_3 *есть* NS"

ПРАВИЛО_3: ЕСЛИ "β_1 *есть* PS" И "β_2 *есть* PS" ТО "β_3 *есть* NM"

ПРАВИЛО_4: ЕСЛИ "β_1 *есть* PS" И "β_2 *есть* NS" ТО "β_3 *есть* Z"

ПРАВИЛО_5: ЕСЛИ "β_1 *есть* NB" И "β_2 *есть* NS" ТО "β_3 *есть* PB"

ПРАВИЛО_6: ЕСЛИ "β_1 *есть* NB" И "β_2 *есть* PS" ТО "β_3 *есть* PS"

ПРАВИЛО_7: ЕСЛИ "β_1 *есть* NS" И "β_2 *есть* NS" ТО "β_3 *есть* PM"

ПРАВИЛО_8: ЕСЛИ "β_1 *есть* NS" И "β_2 *есть* PS" ТО "β_3 *есть* Z"

ПРАВИЛО_9: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* Z" ТО "β_3 *есть* NM"

ПРАВИЛО_10: ЕСЛИ "β_1 *есть* PS" И "β_2 *есть* Z" ТО "β_3 *есть* NS"

ПРАВИЛО_11: ЕСЛИ "β_1 *есть* NB" И "β_2 *есть* Z" ТО "β_3 *есть* PM"

ПРАВИЛО_12: ЕСЛИ "β_1 *есть* NS" И "β_2 *есть* Z" ТО "β_3 *есть* PS"

ПРАВИЛО_13: ЕСЛИ "β_1 *есть* Z" И "β_2 *есть* PS" ТО "β_3 *есть* NS"

ПРАВИЛО_14: ЕСЛИ "β_1 *есть* Z" И "β_2 *есть* NS" ТО "β_3 *есть* PS"

ПРАВИЛО_15: ЕСЛИ "β_1 *есть* Z" И "β_2 *есть* Z" ТО "β_3 *есть* Z"

Для сокращенной записи правил используются следующие обозначения: β_1 — первая входная лингвистическая переменная с именем "*температура воздуха*", β_2 — вторая входная лингвистическая переменная с именем "*скорость изменения температуры воздуха*", β_3 — выходная лингвистическая переменная с именем "*угол поворота регулятора*". Для сокращения записи правил использованы также символические обозначения значений отдельных термов переменных (см. табл. 7.1).

В качестве терм-множества первой лингвистической переменной используется множество T_1={"*очень холодная*", "*холодная*", "*в пределах нормы*", "*теплая*", "*очень теплая*"}, которое записывается в символическом виде: T_1={NB, NS, Z, PS, PB}. В качестве терм-множества второй лингвистической переменной используется множество T_2={"*отрицательная*", "*равна нулю*", "*положительная*"}, которое записывается в символическом виде: T_2={NS, Z, PS}. В качестве терм-множества выходной лингвистической переменной используется множество T_3={"*очень большой угол влево*", "*большой угол влево*", "*небольшой угол влево*", "*выключить кондиционер*", "*небольшой угол вправо*", "*большой угол вправо*", "*очень большой угол вправо*"}, которое записывается в символическом виде: T_3={NB, NM, NS, Z, PS, PM, PB}. При этом функции принадлежности термов из T_1 изображены на рис. 7.19, функции принадлежности термов из T_2 изображены на рис. 7.20, а функции принадлежности термов из T_3 изображены на рис. 7.21.

Разработку нечеткой модели (назовем ее conditioner) будем выполнять с использованием графических средств системы MATLAB. С этой целью откроем редактор FIS и определим 2 входные переменные с именами "*температура*" (β_1) и "*скорость*" (β_2) и одну выходную переменную с именем "*угол*" (β_3). Вид графического интерфейса редактора FIS для этих переменных изображен на рис. 16.1.

Поскольку мы используем систему нечеткого вывода типа Мамдани, оставим без изменения тип, предложенный системой MATLAB по умолчанию. Нет необходимости изменять и другие параметры разрабатываемой нечеткой модели, предложенные системой MATLAB по умолчанию, такие как логические операции (**min** — для нечеткого логического И, **max** — для нечеткого логического ИЛИ), методы импликации (**min**), агрегирования (**max**) и дефаззификации (**centroid**).

Далее следует определить функции принадлежности термов для каждой из переменных системы нечеткого вывода. Для этой цели воспользуемся редактором функций принадлежности системы MATLAB. Для первой входной переменной следует добавить два дополнительных терма к трем, заданным по умолчанию, и определить параметры соответствующих функций принадлежности. Численные

значения этих параметров можно взять из записи этой нечеткой модели на языке FCL *(см. главу 8)*. Вид графического интерфейса редактора функций принадлежности после задания первой входной переменной изображен на рис. 16.2.

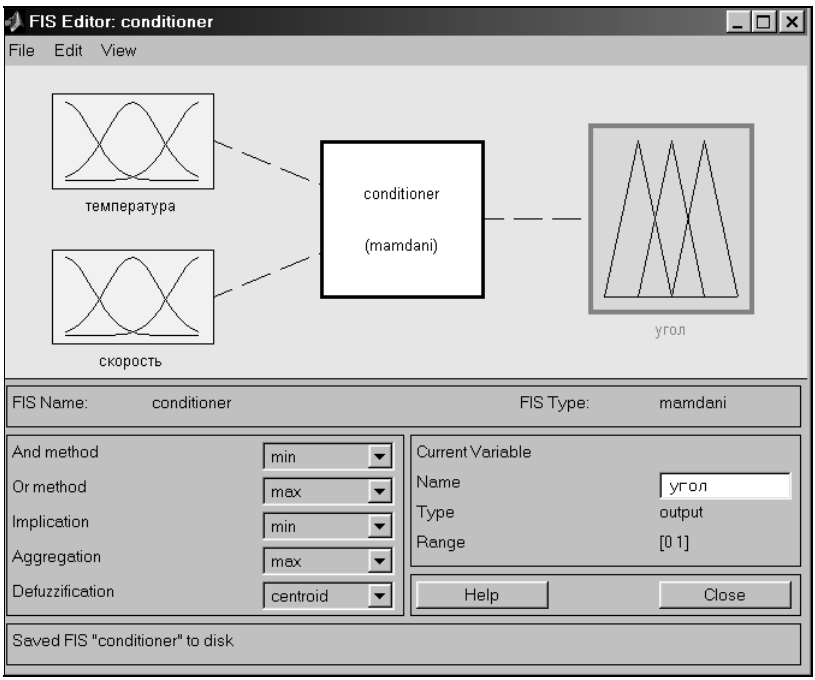


Рис. 16.1. Графический интерфейс редактора FIS после определения входных и выходных переменных разрабатываемой системы нечеткого вывода

Для второй входной переменной следует оставить 3 терма, заданные по умолчанию, и изменить только тип и параметры функций принадлежности. Для выходной переменной следует добавить 4 терма к трем, заданным по умолчанию, и задать параметры соответствующих функций принадлежности. Численные значения этих параметров также берутся из записи этой нечеткой модели на языке FCL *(см. главу 8)*. Вид графического интерфейса редактора функций принадлежности после задания выходной переменной изображен на рис. 16.3.

Примечание

Выполнение операции добавления термов переменных в редакторе функций принадлежности MATLAB версии 6.1 может привести к нестабильной работе системы. В этом случае для определения дополнительных функций принадлежности входных и выходных переменных следует воспользоваться функциями командной строки, которые были рассмотрены ранее в *главе 12* для примера разработки нечеткой модели в командном режиме.

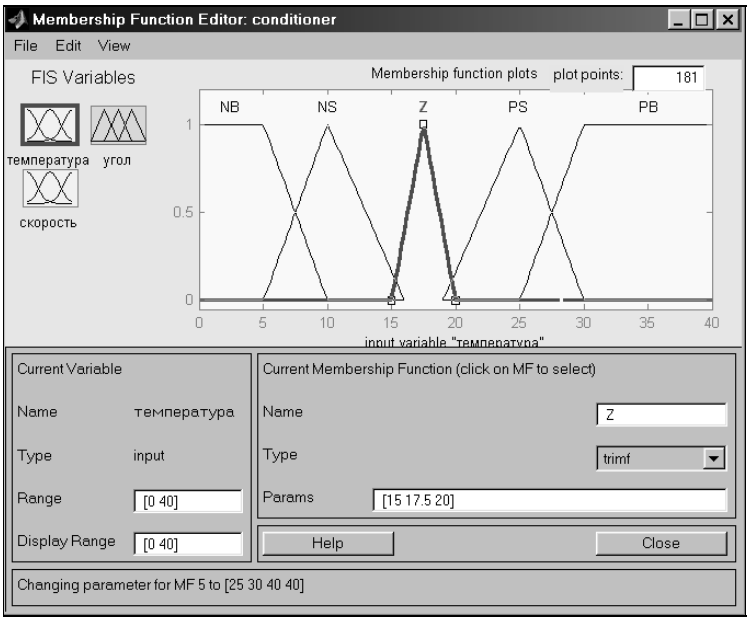


Рис. 16.2. Графический интерфейс редактора функций принадлежности после задания первой входной переменной "*температура*" для системы нечеткого вывода conditioner

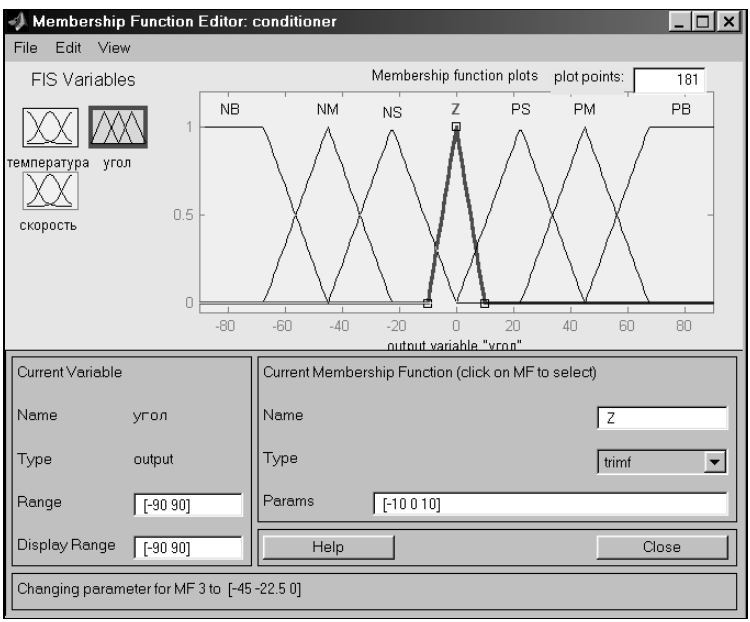


Рис. 16.3. Графический интерфейс редактора функций принадлежности после задания выходной переменной "*угол*" для системы нечеткого вывода conditioner

Теперь зададим 15 правил для разрабатываемой системы нечеткого вывода. Для этой цели воспользуемся редактором правил системы MATLAB. Вид графического интерфейса редактора правил после задания всех 15 правил нечеткого вывода изображен на рис. 16.4.

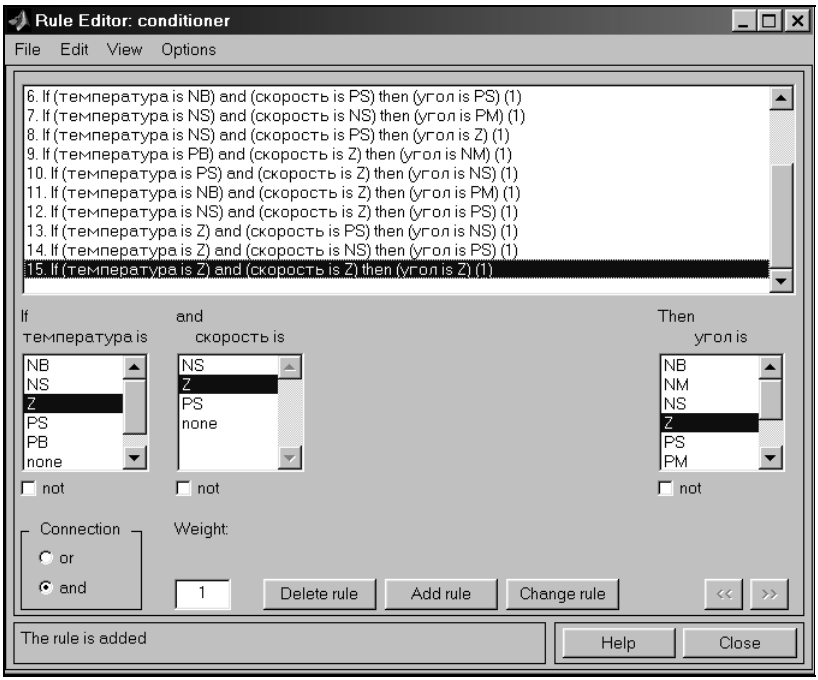


Рис. 16.4. Графический интерфейс редактора правил после задания базы правил для системы нечеткого вывода conditioner

Теперь можно выполнить оценку построенной системы нечеткого вывода для задачи автоматического управления кондиционером в помещении. С этой целью откроем программу просмотра правил системы MATLAB и введем значения входных переменных для частного случая, когда текущая температура воздуха в помещении равна 20 °С, а скорость ее изменения положительная и составляет 0.2 °С/мин. Процедура нечеткого вывода, выполненная системой MATLAB для разработанной нечеткой модели, выдает в результате значение выходной переменной "*угол*", равное –33.8° (рис. 16.5).

Данное значение соответствует включению режима "холод" кондиционера на треть своей мощности. Сравнение результатов нечеткого вывода для этих значений входных переменных, полученные на основе численных расчетов в *главе 7* (≈34° влево) и с помощью разработанной нечеткой модели MATLAB, показывает хорошую согласованность модели и подтверждает ее адекватность в рамках рассматриваемой модели.

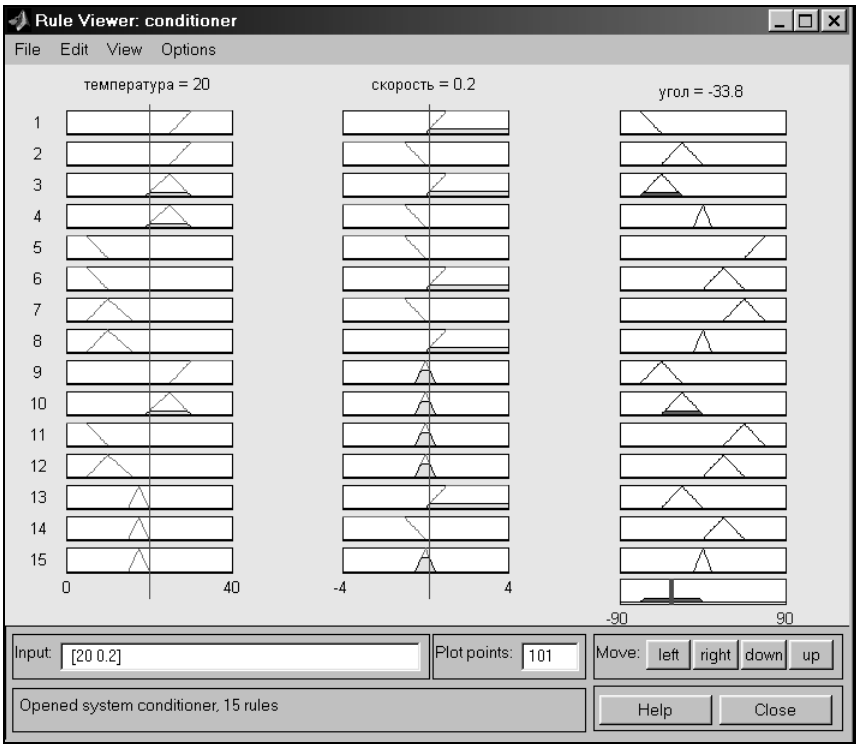


Рис. 16.5. Графический интерфейс программы просмотра правил после выполнения процедуры нечеткого вывода для значений входных переменных [20 0.2]

Процесс анализа и исследования построенной нечеткой модели включает в себя выполнение нечетких выводов для различных значений входных переменных и оценки полученных результатов с целью установления адекватности модели и внесения в нее необходимых изменений в случае несогласованности отдельных результатов. Проверка нечеткой модели для других значений входных переменных, например, 10 °С и 0.2 °С/мин приводит к результату 19.7°, что также подтверждает ее адекватность.

Примечание

Для более тонкой настройки построенной нечеткой модели необходимо знать технические характеристики конкретного кондиционера, установленного в помещении. В частности, значения параметров функций принадлежности отдельных термов могут зависеть от мощности кондиционера и зависимости мощности от угла поворота регулятора.

Для общего анализа разработанной нечеткой модели может оказаться полезной визуализация соответствующей поверхности нечеткого вывода (рис. 16.6).

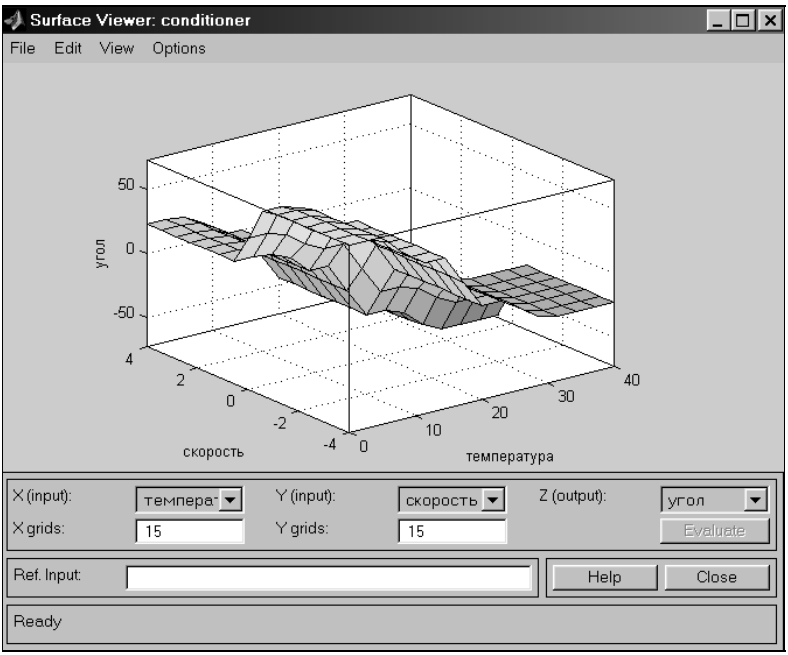


Рис. 16.6. Визуализация поверхности нечеткого вывода для системы нечеткого вывода conditioner

Данная поверхность нечеткого вывода позволяет установить зависимость значений выходной переменной от значений входных переменных нечеткой модели системы управления кондиционером. Эта зависимость может послужить основой для программирования контроллера или аппаратной реализации соответствующего нечеткого алгоритма управления в форме таблицы решений. В дополнение к этому установление данной зависимости является по сути решением задачи, известной в классической теории управления как задача синтеза управляющих воздействий. При этом для решения данной задачи были использованы средства нечеткой логики и теории нечетких множеств.

16.2. Нечеткая модель управления контейнерным краном

Напомним, что эта задача связана с процессом управления контейнерным краном, который используется для транспортировки моноблочных контейнеров при выполнении разгрузочных работ морских судов. Суть задачи состоит в том, чтобы разработать модель, позволяющую автоматически управлять процессом горизонтального перемещения контейнерного крана, исключив раскачивание контейнеров.

Для решения этой задачи в *разд. 7.4* была построена база правил соответствующей системы нечеткого вывода, которая содержит 6 правил нечетких продукций следующего вида:

ПРАВИЛО_1: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* Z" ТО "β_3 *есть* PM"

ПРАВИЛО_2: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* NS" ТО "β_3 *есть* PB"

ПРАВИЛО_3: ЕСЛИ "β_1 *есть* PB" И "β_2 *есть* NB" "β_3 *есть* PM"

ПРАВИЛО_4: ЕСЛИ "β_1 *есть* PM" И "β_2 *есть* NS" ТО "β_3 *есть* NM"

ПРАВИЛО_5: ЕСЛИ "β_1 *есть* PS" И "β_2 *есть* PS" ТО "β_3 *есть* PM"

ПРАВИЛО_6: ЕСЛИ "β_1 *есть* Z" И "β_2 *есть* Z" ТО "β_3 *есть* Z"

Для сокращенной записи правил используются следующие обозначения: β_1 — первая входная лингвистическая переменная с именем "*расстояние*", β_2 — вторая входная лингвистическая переменная с именем "*угол*", β_3 — выходная лингвистическая переменная с именем "*мощность*".

В качестве терм-множества первой лингвистической переменной используется множество T_1= {"*нуль*", "*близкое*", "*среднее*", "*далекое*"}, которое записывается в символическом виде: T_1={Z, PS, PM, PB}. В качестве терм-множества второй лингвистической переменной используется множество T_2= {"*отрицательный большой*", "*отрицательный малый*", "*нуль*", "*положительный малый*"}, которое записывается в символическом виде: T_2={NB, NS, Z, PS}. В качестве терм-множества выходной лингвистической переменной используется множество T_3= {"*отрицательная средняя*", "*нуль*", "*положительная средняя*", "*положительная большая*"}, которое записывается в символическом виде: T_3={NM, Z, PM, PB}. Графики функций принадлежности термов из T_1 изображены на рис. 7.24, функции принадлежности термов из T_2 изображены на рис. 7.25, а функции принадлежности термов из T_3 изображены на рис. 7.26.

Примечание

Рассматриваемая база правил нечеткой модели управления краном несколько отличается от построенной базы правил в *разд. 7.4*. В частности, для сокращения избыточности исключены не используемые в правилах нечеткого вывода термы, а их имена записаны в типовых обозначениях.

Разработку нечеткой модели (назовем ее crane) также будем выполнять с использованием графических средств системы MATLAB. С этой целью откроем редактор FIS и определим 2 входные переменные с именами "*расстояние*" (β_1) и "*угол*" (β_2) и одну выходную переменную с именем "*мощность*" (β_3). Вид графического интерфейса редактора FIS для этих переменных изображен на рис. 16.7.

Оставим без изменения тип системы нечеткого вывода, предложенный системой MATLAB по умолчанию, поскольку будем использовать систему нечеткого вывода типа Мамдани. Нет необходимости изменять и другие параметры разрабатываемой нечеткой модели, предложенные системой MATLAB по умолчанию, такие как логические операции (**min** — для нечеткого логического И, **max** — для

нечеткого логического ИЛИ), методы импликации **(min)**, агрегирования **(max)** и метода дефаззификации **(centroid)**.

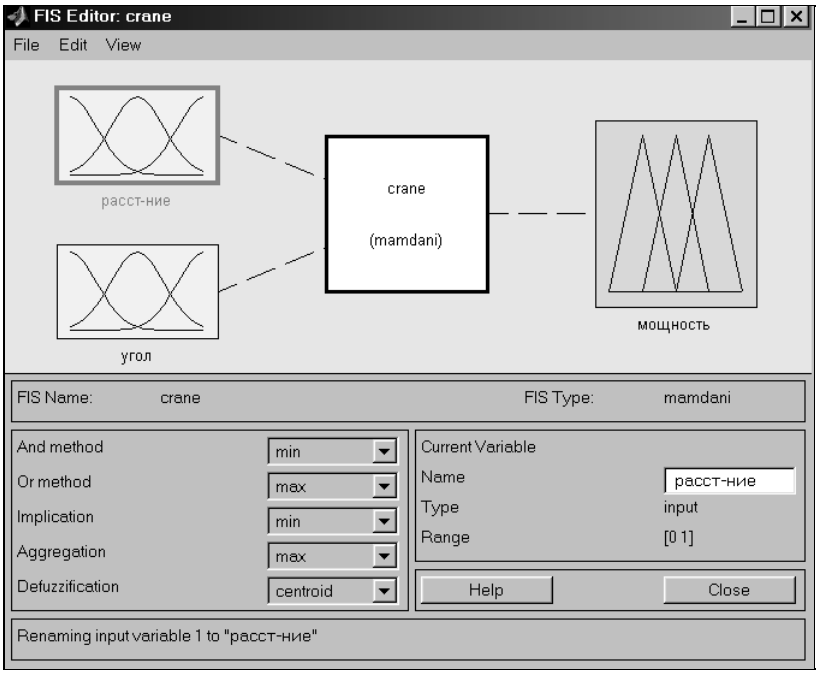


Рис. 16.7. Графический интерфейс редактора FIS после определения входных и выходных переменных для системы нечеткого вывода crane

Далее следует определить функции принадлежности термов для каждой из переменных системы нечеткого вывода. Для этой цели воспользуемся редактором функций принадлежности системы MATLAB. Для первой входной переменной следует добавить один дополнительный терм к трем, заданным по умолчанию, и определить параметры соответствующих функций принадлежности. Численные значения этих параметров можно взять из записи этой нечеткой модели на языке FCL *(см. главу 8)*. Вид графического интерфейса редактора функций принадлежности после задания первой входной переменной изображен на рис. 16.8.

Для второй входной переменной "*угол*" также следует определить 4 терма, изменив диапазон значений переменной и параметры треугольных функций принадлежности термов. Для выходной переменной "*мощность*" следует определить 4 терма, изменить диапазон значений переменной и определить соответствующие функции принадлежности термов.

Поскольку функции принадлежности термов выходной переменной равны постоянным значениям, как это определено в записи этой нечеткой модели на языке FCL *(см. главу 8)*, а среди встроенных функций принадлежности системы

MATLAB отсутствуют одноточечные функции принадлежности, то в качестве функций принадлежности термов возьмем треугольные функции принадлежности с подходящими значениями параметров. Вид графического интерфейса редактора функций принадлежности после задания выходной переменной изображен на рис. 16.9.

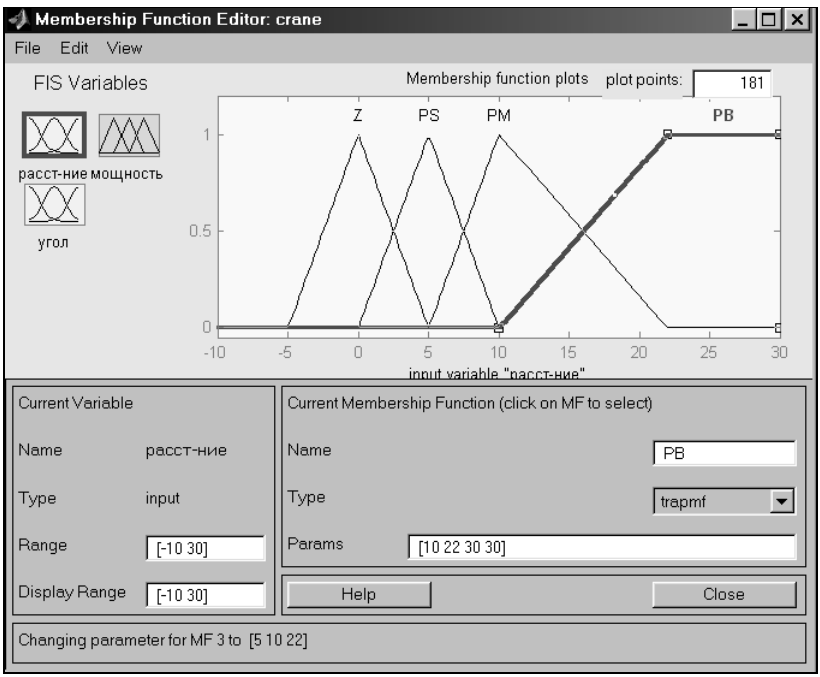


Рис. 16.8. Графический интерфейс редактора функций принадлежности после задания первой входной переменной "*расстояние*" для системы нечеткого вывода crane

Примечание

При задании треугольных функций принадлежности для термов выходной переменной исходная нечеткая модель для задачи управления краном несколько модифицируется с учетом возможностей среды реализации MATLAB. Если эту модель реализовывать в точности, то следует воспользоваться функциями командной строки, которые были рассмотрены ранее в *главе 12* для примера разработки нечеткой модели в командном режиме.

Теперь зададим 6 правил для разрабатываемой системы нечеткого вывода. Для этой цели воспользуемся редактором правил системы MATLAB. Вид графического интерфейса редактора правил после задания всех 6 правил нечеткого вывода изображен на рис. 16.10.

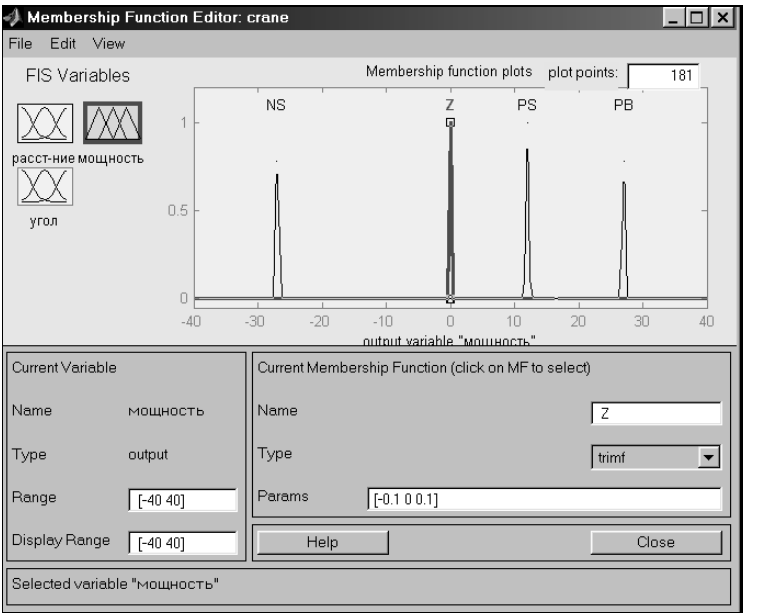


Рис. 16.9. Графический интерфейс редактора функций принадлежности после задания выходной переменной "*мощность*" для системы нечеткого вывода crane

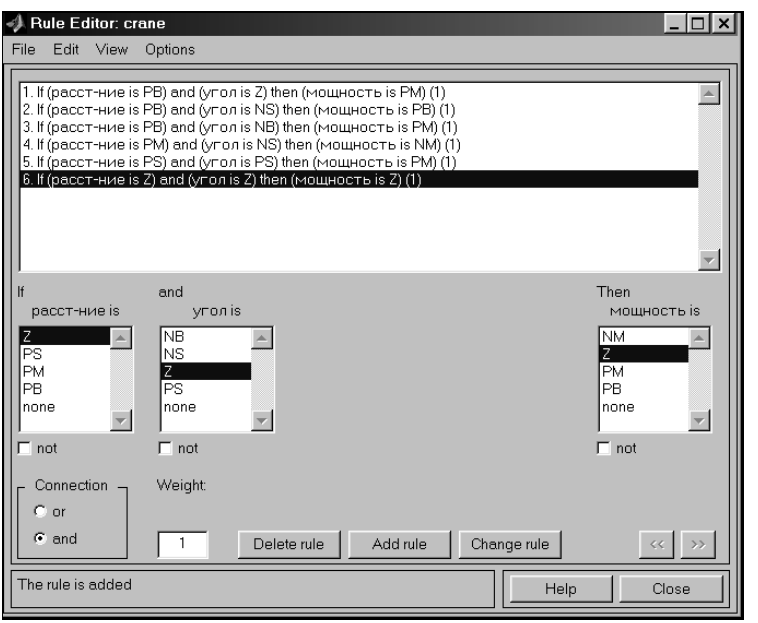


Рис. 16.10. Графический интерфейс редактора правил после задания базы правил для системы нечеткого вывода crane

Теперь можно выполнить оценку построенной системы нечеткого вывода для задачи автоматического управления портовым краном. С этой целью откроем программу просмотра правил системы MATLAB и введем значения входных переменных для частного случая, когда текущее расстояние до цели равно 7.5 м, а угол между контейнером и кабиной крана равен 10°. Процедура нечеткого вывода, выполненная системой MATLAB для разработанной нечеткой модели, выдает в результате значение выходной переменной "*мощность*", равное 12 киловатт (рис. 16.11).

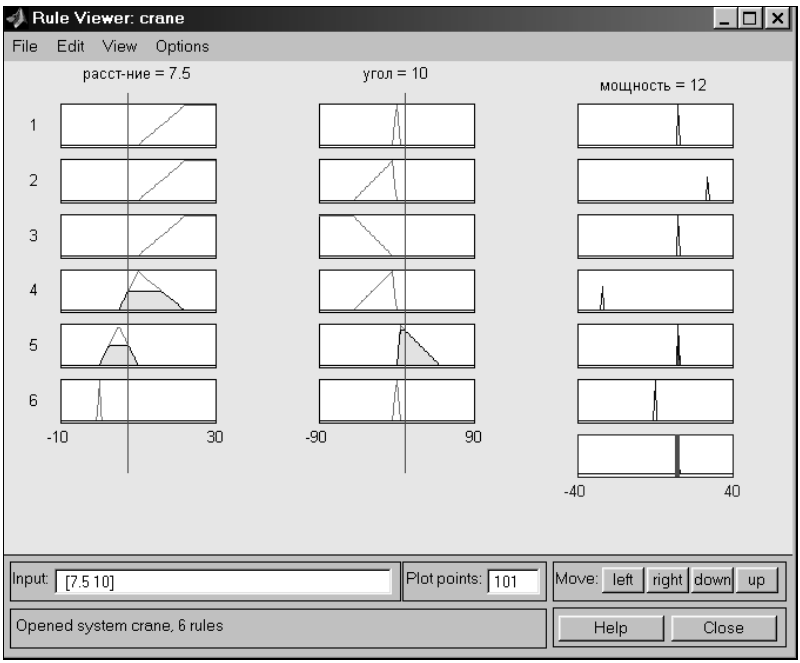


Рис. 16.11. Графический интерфейс программы просмотра правил после выполнения процедуры нечеткого вывода для значений входных переменных [7.5 10]

Данное значение соответствует включению двигателя крана для движения вправо (в положительном направлении), при этом сама мощность равна 12 киловатт. В данном случае сравнение результатов нечеткого вывода для этих значений входных переменных, полученных на основе численных расчетов в *главе 7* и с помощью разработанной нечеткой модели MATLAB, также показывает согласованность модели и подтверждает ее адекватность.

Примечание

Для более тонкой настройки построенной нечеткой модели также необходимо знать технические характеристики конкретного крана в том или ином в порту.

В частности, значения параметров функций принадлежности отдельных термов могут зависеть от мощности двигателей отдельных кранов, габаритов контейнеров и размещения целевых площадок для погрузки контейнеров. При этом может потребоваться увеличение количества правил для модифицированной нечеткой модели.

Для общего анализа разработанной нечеткой модели также может оказаться полезной визуализация соответствующей поверхности нечеткого вывода (рис. 16.12).

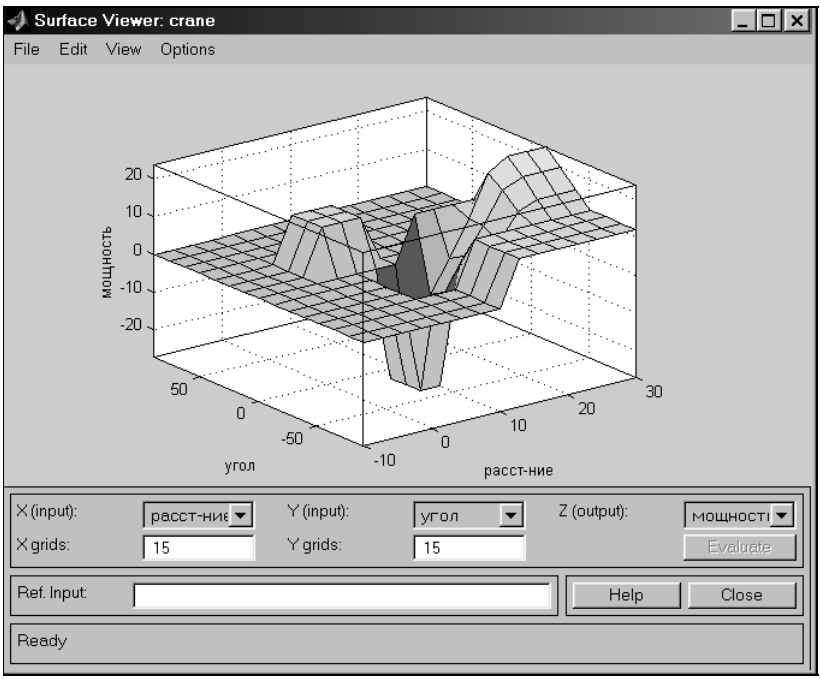
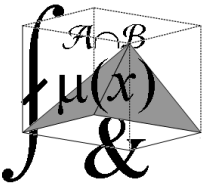


Рис. 16.12. Визуализация поверхности нечеткого вывода для системы нечеткого вывода crane

Данная поверхность нечеткого вывода позволяет установить зависимость значений выходной переменной от значений входных переменных нечеткой модели системы управления краном. Эта зависимость может послужить основой для программирования контроллера или аппаратной реализации соответствующего нечеткого алгоритма управления в форме соответствующей таблицы решений.

Рассмотренные примеры иллюстрируют все практические действия, которые необходимо выполнить для разработки и использования нечетких моделей в форме систем нечеткого вывода. При этом соответствующие графические средства системы MATLAB позволяют реализовать все этапы процесса нечеткого моделирования в удобной для пользователя форме.

Глава 17



Примеры разработки нечетких моделей принятия решений в среде MATLAB

В качестве иллюстрации практического применения системы MATLAB рассматриваются две практические задачи нечеткого моделирования. Первая из них возникает при оценивании финансовой состоятельности клиентов со стороны банков при выдаче долгосрочных кредитов (ссуд) на строительство недвижимости под залог. Вторая задача является типичной в практике анализа и прогнозирования цен на финансовом рынке валюты или ценных бумаг.

17.1 Оценивание финансовой состоятельности клиентов при предоставлении банковских кредитов

Данная задача была сформулирована и решена в рамках исследования, выполненного фирмой INFORM GmbH, являющейся разработчиком программного средства fuzzyTECH, которое рассматривается в *части III*. Этот пример позволяет не только познакомиться с особенностями разработки реальных приложений, но и может служить в качестве тестового при выполнении сравнительного анализа различных программных средств нечеткого моделирования.

Содержательная постановка задачи оценивания финансовой состоятельности клиентов

Суть рассматриваемой задачи заключается в следующем. При выдаче долгосрочных кредитов на строительство зданий или коттеджей под залог недвижимости для оценки состоятельности клиентов банками традиционно используется метод экспертных оценок. При этом целью банков является получение максимальной прибыли от заключенных сделок по предоставлению кредитов и исключение возможности финансовых потерь. Поэтому интересы банков сосредо-

точены, с одной стороны, на увеличении количества успешных сделок, а с другой стороны, на избежании неудачных сделок, когда клиент не возвращает выданный кредит или возвращает его не вовремя.

Традиционно основанием для принятия решений по предоставлению кредитов в будущем служит опыт успешных сделок, совершенных в прошлом. Руководство банка Home&Savings Bank, в интересах которого выполнялось соответствующее исследование, хотело бы обобщить правила предоставления кредитов с целью максимально полно использовать опыт экспертов. При этом необходимо исключить возможные ошибки субъективного характера со стороны отдельных менеджеров в случае неадекватного оценивания финансовой состоятельности клиентов.

Анализ стратегии предоставления кредитов на строительство зданий показывает, что для оценивания финансовой состоятельности клиентов могут быть использованы различные характеристики, такие как месторасположение строящегося здания, качество предполагаемого выполнения отделочных работ, оценка активов потенциального клиента, оценка дохода потенциального клиента за вычетом фиксированных расходов, величина подлежащих уплате процентов по кредиту. При этом собственно финансовая состоятельность клиента оценивается его кредитоспособностью.

Одной из первых формальных моделей, предложенных для решения данной задачи, являлась статистическая модель, основанная на вероятностной интерпретации количественной оценки положительного решения о предоставлении кредита. Однако более детальный анализ этой модели со временем показал ее неадекватность, связанную с недостаточным объемом статистической выборки и изменяющимися с течением времени условиями предоставления кредитов.

Именно по этой причине была предложена идея разработки нечеткой модели для оценивания финансовой состоятельности клиентов с целью принятия решений о предоставлении долгосрочных кредитов. При этом в качестве нечеткой модели используется система нечеткого вывода со следующими входными и выходными переменными.

Описание входных и выходных переменных рассматриваемой задачи

Содержательная интерпретация нечеткой модели предполагает выбор и спецификацию входных и выходных переменных соответствующей системы нечеткого вывода. При этом в нечеткой модели предполагается использовать 5 входных переменных и 1 выходную переменную.

В качестве первой входной переменной используется оценка месторасположения строящегося здания, которая непосредственно оценивает проект строящегося здания, принимая во внимание размещение здания в том или ином конкретном районе города или регионе пригорода. Очевидно, чем выше эта оценка, тем более ликвидным представляется проект в случае его реализации на рынке недвижимости.

В качестве второй входной переменной используется качество предполагаемого выполнения отделочных работ согласно архитектурному проекту строящегося здания. Эта переменная вносит дополнительный элемент в оценку стоимости строящегося здания.

В качестве третьей входной переменной используется оценка активов, которая используется для оценки имущества или авуаров в случае несостоятельности потенциального клиента при невозвращении им взятого кредита. Действительно, величина предоставляемого кредита должна основываться не только на учете стоимости строящегося здания, но и на собственной капитализации клиента.

В качестве четвертой входной переменной используется оценка дохода потенциального клиента за вычетом фиксированных расходов, которая также используется в случае несостоятельности потенциального клиента при невозвращении им взятого кредита. Чем выше значение этой переменной, тем более успешным представляется предоставление кредита клиенту.

В качестве пятой входной переменной используется величина подлежащих уплате процентов согласно предполагаемому плану выплат по взятому кредиту. Эта переменная связана со сроком предоставления кредита и его величиной, позволяя объединить в себе соответствующие характеристики кредита. Чем выше величина выплат по процентам, тем более высокими должны быть значения активов и доходов для положительного решения о предоставлении кредита потенциальному клиенту.

В качестве выходной переменной используется оценка кредитоспособности, которая является основой для принятия решения руководством банка по предоставлению кредита потенциальным клиентам. При этом решение о предоставлении кредита руководством банка принимается только в случае высокой оценки этой выходной переменной.

Анализ предоставления кредитов на строительство зданий показывает, что для анализа финансовой состоятельности потенциальных клиентов руководство банков применяет следующие эвристические правила:

1. Если величина дохода низкая и величина выплат средняя, то кредитоспособность очень низкая.
2. Если величина дохода низкая и величина выплат высокая, то кредитоспособность очень низкая.
3. Если величина дохода средняя и величина выплат высокая, то кредитоспособность очень низкая.
4. Если активы низкие и величина дохода низкая, то кредитоспособность очень низкая.
5. Если активы низкие и величина дохода средняя, то кредитоспособность очень низкая.
6. Если активы средние и величина дохода низкая, то кредитоспособность очень низкая.

7. Если качество отделки плохое, активы низкие и величина дохода высокая, то кредитоспособность очень низкая.
8. Если качество отделки плохое, активы средние и величина дохода средняя, то кредитоспособность очень низкая.
9. Если качество отделки плохое, активы высокие и величина дохода низкая, то кредитоспособность очень низкая.
10. Если качество отделки плохое, активы высокие и величина дохода средняя, то кредитоспособность очень низкая.
11. Если местоположение непрестижное, качество отделки хорошее, активы низкие и величина дохода высокая, то кредитоспособность средняя.
12. Если местоположение непрестижное, качество отделки прекрасное, активы низкие и величина дохода высокая, то кредитоспособность средняя.
13. Если местоположение престижное, качество отделки хорошее, активы низкие и величина дохода высокая, то кредитоспособность средняя.
14. Если местоположение очень престижное, качество отделки хорошее, активы низкие и величина дохода высокая, то кредитоспособность средняя.
15. Если местоположение непрестижное, качество отделки хорошее, активы средние и величина дохода средняя, то кредитоспособность средняя.
16. Если местоположение непрестижное, качество отделки прекрасное, активы средние и величина дохода средняя, то кредитоспособность средняя.
17. Если местоположение престижное, качество отделки хорошее, активы средние и величина дохода средняя, то кредитоспособность средняя.
18. Если местоположение очень престижное, качество отделки хорошее, активы средние и величина дохода средняя, то кредитоспособность средняя.
19. Если местоположение непрестижное, качество отделки хорошее, активы высокие и величина дохода низкая, то кредитоспособность средняя.
20. Если местоположение непрестижное, качество отделки прекрасное, активы высокие и величина дохода низкая, то кредитоспособность средняя.
21. Если местоположение престижное, качество отделки хорошее, активы высокие и величина дохода низкая, то кредитоспособность средняя.
22. Если местоположение очень престижное, качество отделки хорошее, активы высокие и величина дохода низкая, то кредитоспособность средняя.
23. Если местоположение непрестижное, качество отделки хорошее, активы высокие и величина дохода средняя, то кредитоспособность средняя.
24. Если местоположение непрестижное, качество отделки прекрасное, активы высокие и величина дохода средняя, то кредитоспособность средняя.
25. Если местоположение престижное, качество отделки хорошее, активы высокие и величина дохода средняя, то кредитоспособность средняя.
26. Если местоположение очень престижное, качество отделки хорошее, активы высокие и величина дохода средняя, то кредитоспособность средняя.

27. Если местоположение престижное, качество отделки прекрасное, активы средние и величина дохода высокая, то кредитоспособность очень высокая.
28. Если местоположение престижное, качество отделки прекрасное, активы высокие и величина дохода высокая, то кредитоспособность очень высокая.
29. Если местоположение очень престижное, качество отделки прекрасное, активы средние и величина дохода высокая, то кредитоспособность очень высокая.
30. Если местоположение очень престижное, качество отделки прекрасное, активы высокие и величина дохода высокая, то кредитоспособность очень высокая.
31. Если местоположение непрестижное, качество отделки хорошее, активы средние и величина дохода высокая, то кредитоспособность высокая.
32. Если местоположение непрестижное, качество отделки прекрасное, активы средние и величина дохода высокая, то кредитоспособность высокая.
33. Если местоположение престижное, качество отделки хорошее, активы средние и величина дохода высокая, то кредитоспособность высокая.
34. Если местоположение очень престижное, качество отделки хорошее, активы средние и величина дохода высокая, то кредитоспособность высокая.
35. Если местоположение непрестижное, качество отделки хорошее, активы высокие и величина дохода высокая, то кредитоспособность высокая.
36. Если местоположение непрестижное, качество отделки прекрасное, активы высокие и величина дохода высокая, то кредитоспособность высокая.
37. Если местоположение престижное, качество отделки хорошее, активы высокие и величина дохода высокая, то кредитоспособность высокая.
38. Если местоположение очень престижное, качество отделки хорошее, активы высокие и величина дохода высокая, то кредитоспособность высокая.
39. Если местоположение престижное, качество отделки прекрасное, то кредитоспособность высокая.
40. Если местоположение очень престижное, качество отделки прекрасное, то кредитоспособность высокая.

После рассмотрения содержательной постановки задачи можно приступить к построению ее нечеткой модели в форме соответствующей системы нечеткого вывода. Для этой цели воспользуемся рассмотренными ранее графическими средствами пакета Fuzzy Logic Toolbox системы MATLAB.

Нечеткая модель оценивания финансовой состоятельности клиентов

При построении нечеткой модели оценки финансовой состоятельности потенциальных клиентов было сделано предположение о том, что все рассматриваемые переменные измеряются в баллах в интервале действительных чисел от 0 до 10.

При этом самая низкая оценка значения каждой из переменных является 0, а самой высокой — 10. Этот подход аналогичен рассмотренному ранее примеру 12.1 "*Чаевые в ресторане*".

Фаззификация входных и выходных переменных

В качестве терм-множества первой входной переменной "*Местоположение*" (Location) будем использовать множество T_1={"*непрестижное*", "*престижное*", "*очень престижное*"} или в символическом виде T_1={PS, PM, PB} с функциями принадлежности термов, изображенными на рис. 17.1, *а*.

В качестве терм-множества второй входной переменной "*Отделка*" (Workmanship) будем использовать аналогичное множество T_2={"*плохая*", "*хорошая*", "*прекрасная*"} или в символическом виде T_2={PS, PM, PB} с функциями принадлежности термов, изображенными на рис. 17.1, *б*.

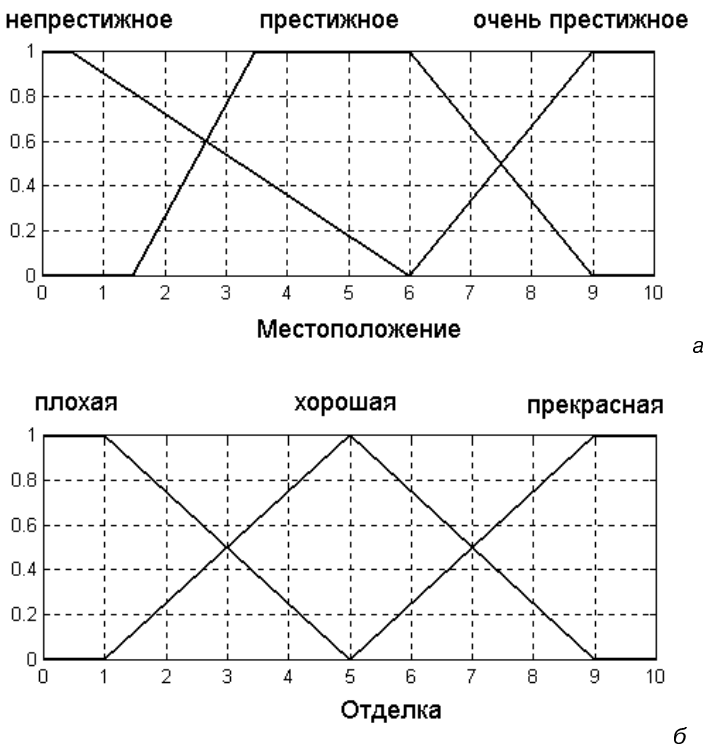


Рис. 17.1. Графики функций принадлежности для термов лингвистических переменных "*Местоположение*" и "*Отделка*"

В качестве терм-множества третьей лингвистической переменной "*Активы*" (Asset) будем использовать множество T_3={"*низкие*", "*средние*", "*высокие*"} или в символическом виде T_3={PS, PM, PB} с функциями принадлежности термов, изображенными на рис. 17.2, *а*.

В качестве терм-множества четвертой лингвистической переменной "*Доход*" (Income) будем использовать аналогичное множество T_4={"*низкий*", "*средний*", "*высокий*"} или в символическом виде T_4={PS, PM, PB} с функциями принадлежности термов, изображенными на рис. 17.2, *б*.

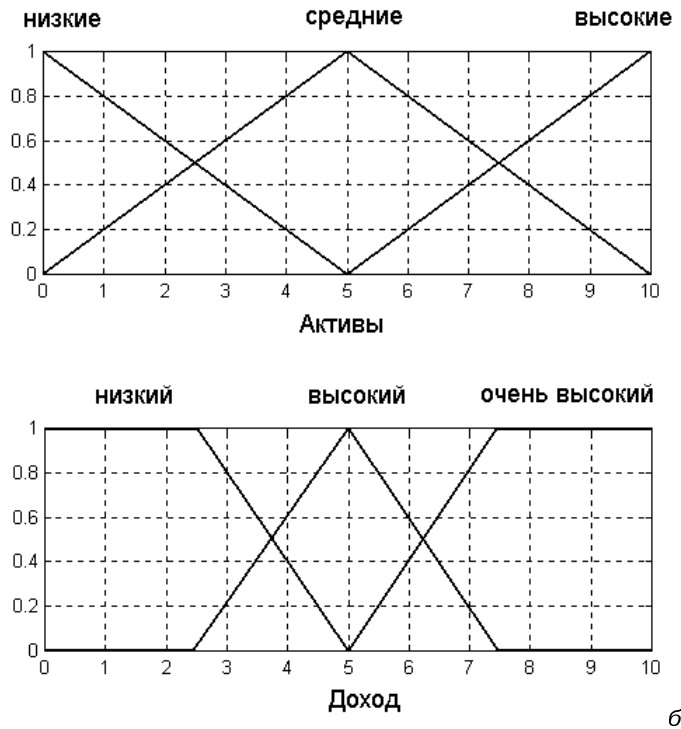


Рис. 17.2. Графики функций принадлежности для термов лингвистических переменных "*Активы*" и "*Доход*", измеряемых в баллах

В качестве терм-множества пятой лингвистической переменной "*Выплаты*" (Interest) будем использовать аналогичное множество T_5={"*низкие*", "*средние*", "*высокие*"} или в символическом виде T_5={PS, PM, PB} с функциями принадлежности термов, изображенными на рис. 17.3, *а*.

В качестве терм-множества выходной лингвистической переменной "*Кредитоспособность*" (Credit) будем использовать множество T_6={"*очень низкая*", "*низкая*", "*средняя*", "*высокая*", "*очень высокая*"} или в символическом виде

T_6={NB, NS, Z, PS, PB} с функциями принадлежности термов, изображенными на рис. 17.3, *б*.

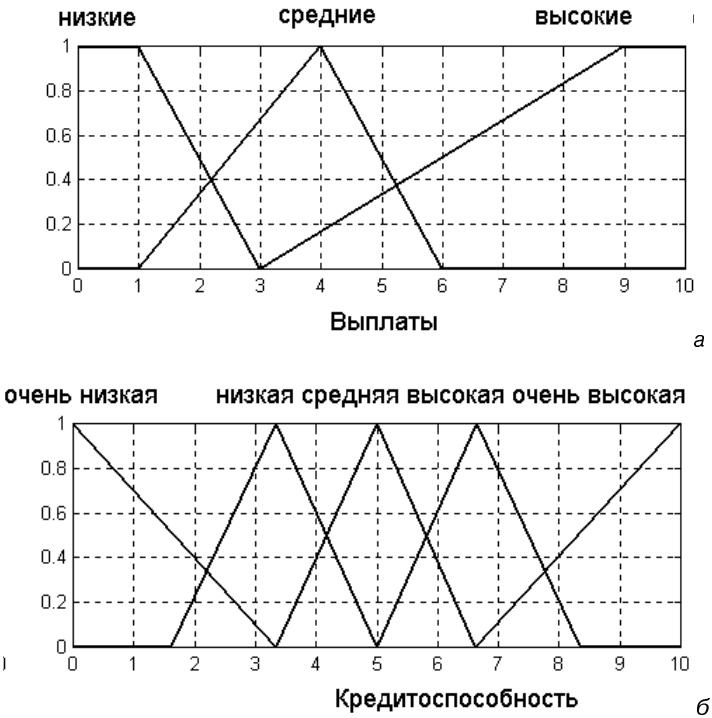


Рис. 17.3. Графики функций принадлежности для термов лингвистических переменных "*Выплаты*" и "*Кредитоспособность*", измеряемых в баллах

Формирование базы правил систем нечеткого вывода

Следующим этапом построения модели является построение базы правил. Для этой цели будем использовать 40 правил нечетких продукций, которые удобно представить в виде следующей таблицы (табл. 17.1).

Таблица 17.1. *Правила нечетких продукций для рассматриваемой системы нечеткого вывода*

Номер правила	Местополо-жение	Отделка	Активы	Доход	Выплаты	Кредито-способность
1				PS	PM	NB
2				PS	PB	NB

Таблица 17.1 *(продолжение)*

Номер правила	Местополо-жение	Отделка	Активы	Доход	Выплаты	Кредито-способность
3				PM	PB	NB
4			PS	PS		NB
5			PS	PM		NB
6			PM	PS		NB
7		PS	PS	PB		NB
8		PS	PM	PM		NB
9		PS	PB	PS		NB
10		PS	PB	PM		Z
11	PS	PM	PS	PB		Z
12	PS	PB	PS	PB		Z
13	PM	PM	PS	PB		Z
14	PB	PM	PS	PB		Z
15	PS	PM	PM	PM		Z
16	PS	PB	PM	PM		Z
17	PM	PM	PM	PM		Z
18	PB	PM	PM	PM		Z
19	PS	PM	PB	PS		Z
20	PS	PB	PB	PS		Z
21	PM	PM	PB	PS		Z
22	PB	PM	PB	PS		Z
23	PS	PM	PB	PM		Z
24	PS	PB	PB	PM		Z
25	PM	PM	PB	PM		Z
26	PB	PM	PB	PM		Z
27	PM	PB	PM	PB		PB
28	PM	PB	PB	PB		PB
29	PB	PB	PM	PB		PB
30	PB	PB	PB	PB		PB
31	PS	PM	PM	PB		PS

Таблица 17.1 *(окончание)*

Номер правила	Местополо-жение	Отделка	Активы	Доход	Выплаты	Кредито-способность
32	PS	PB	PM	PB		PS
33	PM	PM	PM	PB		PS
34	PB	PM	PM	PB		PS
35	PS	PM	PB	PB		PS
36	PS	PB	PB	PB		PS
37	PM	PM	PB	PB		PS
38	PB	PM	PB	PB		PS
39	PM	PB				PS
40	PB	PB				PS

В качестве схемы нечеткого вывода будем использовать метод Мамдани, поэтому методом активации будет MIN, который рассчитывается по формуле (7.6). Далее необходимо определить методы агрегирования подусловий. Поскольку во всех правилах 1—40 в качестве логической связки для подусловий применяется только нечеткая конъюнкция (операция "И"), то в качестве метода агрегирования будем использовать операцию min-конъюнкции. Для аккумуляции заключений правил будем использовать метод max-дизъюнкции, который также применяется в случае схемы нечеткого вывода методом Мамдани. Наконец, в качестве метода дефаззификации будем использовать метод центра тяжести (см. (7.9)).

Построение нечеткой модели средствами Fuzzy Logic Toolbox и анализ полученных результатов.

Разработку нечеткой модели (назовем ее mortgage) будем выполнять с использованием графических средств системы MATLAB. С этой целью в редакторе FIS определим 5 входных переменных с именами "*местоположение*" (β_1), "*отделка*" (β_2), "*активы*" (β_3), "*доход*" (β_4), "*выплаты*" (β_5) и одну выходную переменную с именем "*кредитоспособность*" (β_6). Вид графического интерфейса редактора FIS для этих переменных изображен на рис. 17.4.

Для решения поставленной задачи нечеткого моделирования будем использовать систему нечеткого вывода типа Мамдани. Оставим без изменения параметры разрабатываемой нечеткой модели, предложенные системой MATLAB по умолчанию, а именно, логические операции (**min** — для нечеткого логического И, **max** — для нечеткого логического ИЛИ), метод импликации (**min**), метод агрегирования (**max**) и метод дефаззификации (**centroid**).

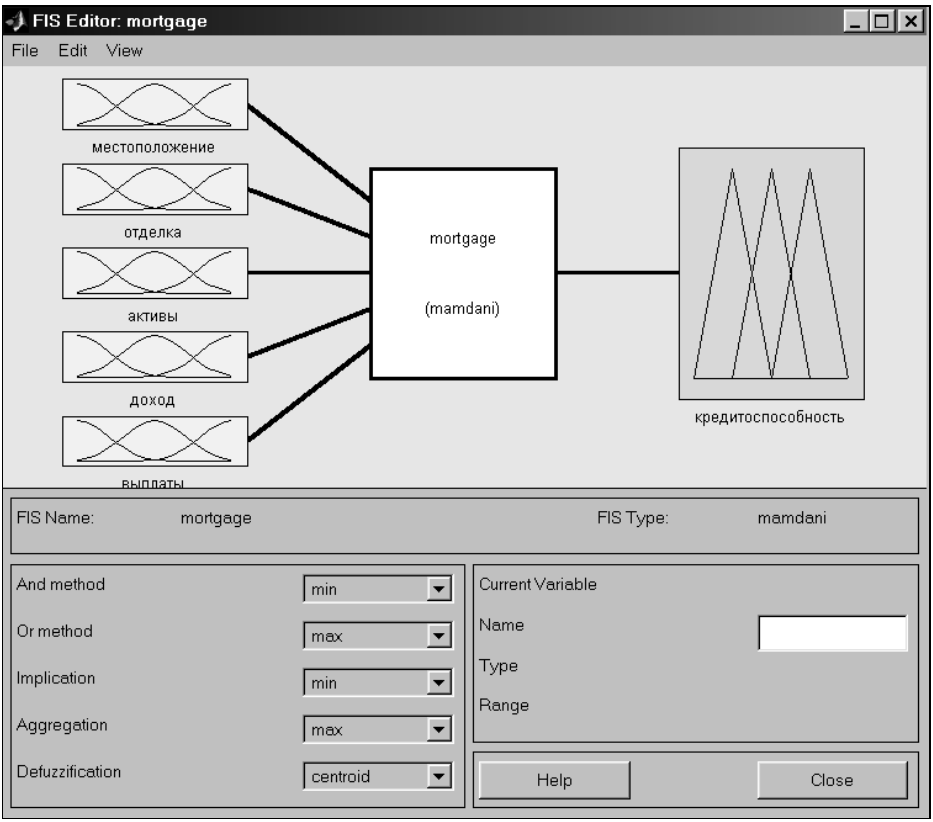


Рис. 17.4. Графический интерфейс редактора FIS после определения входных и выходной переменных системы нечеткого вывода mortgage

Далее следует определить функции принадлежности термов для каждой из 5 входных и единственной выходной переменных рассматриваемой системы нечеткого вывода. Для этой цели воспользуемся редактором функций принадлежности системы MATLAB. Будем использовать типы функций принадлежности и соответствующие численные значения их параметров, которые изображены на рис. 17.1—17.3. Графический интерфейс редактора функций принадлежности для выходной переменной "*кредитоспособность*" изображен на рис. 17.5.

Далее зададим 40 правил для разрабатываемой системы нечеткого вывода (табл. 17.1). Для этой цели воспользуемся редактором правил системы MATLAB. Вид графического интерфейса редактора правил после задания всех 40 правил нечеткого вывода изображен на рис. 17.6. Поскольку в рабочем окне отображаются не все переменные нечеткой модели, для управления режимом отображения переменных правил следует воспользоваться специальными кнопками **>>** и **<<**, расположенными в нижней правой части редактора правил.

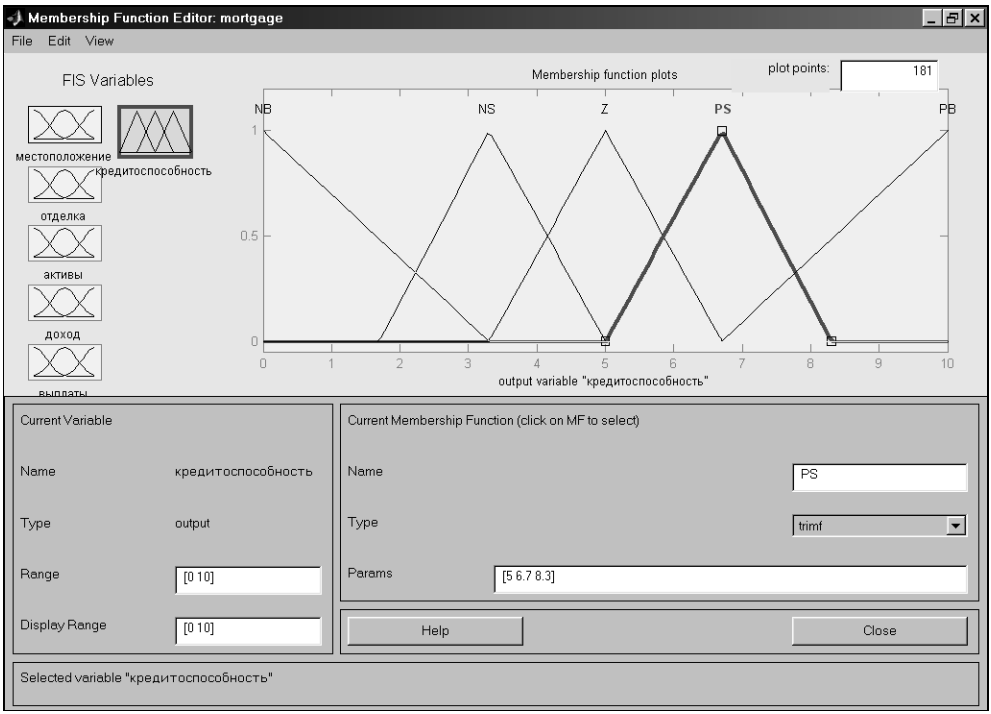


Рис. 17.5. Графический интерфейс редактора функций принадлежности для выходной переменной "*кредитоспособность*"

Теперь можно выполнить анализ построенной системы нечеткого вывода для рассматриваемой задачи оценки финансовой состоятельности клиентов. С этой целью откроем окно просмотра правил системы MATLAB и введем значения входных переменных для частного случая, когда значение входной переменной "*местоположение*" оценивается в 8 баллов, значение входной переменной "*отделка*" также оценивается в 8 баллов, значение входной переменной "*активы*" оценивается в 9 баллов, значение входной переменной "*доход*" оценивается в 9 баллов, и, наконец, значение входной переменной "*выплаты*" оценивается в 5 баллов. Это достаточно высокие оценки входных переменных, которые даже на интуитивном уровне свидетельствуют в пользу соответствующего клиента.

Процедура нечеткого вывода, выполненная системой MATLAB для разработанной нечеткой модели, выдает в результате значение выходной переменной "*кредитоспособность*", равное 7.75 балла (рис. 17.7). Это достаточно высокая оценка финансовой состоятельности потенциального клиента, которое может служить основанием для положительного решения со стороны банка о предоставлении кредита под залог. Как можно заключить, данный вывод полностью согласуется с ранее высказанными интуитивными соображениями.

Рис. 17.6. Графический интерфейс редактора правил после задания базы правил системы нечеткого вывода mortgage

Далее выполним анализ построенной системы нечеткого вывода для второго варианта исходных данных с более низкими оценками значений входных переменных. С этой целью изменим значения входных переменных: значение входной переменной "*местоположение*" оценим в 3 балла, значение входной переменной "*отделка*" оценим в 5 балла, значение входной переменной "*активы*" оценим в 4 балла, значение входной переменной "*доход*" оценим в 4 балла, и, наконец, значение входной переменной "*выплаты*" оценим в 2 балла.

Процедура нечеткого вывода, выполненная системой MATLAB, выдает в результате значение выходной переменной "*кредитоспособность*", равное 3.42 балла (рис. 17.8). Это достаточно низкая оценка финансовой состоятельности потенциального клиента, которое может служить основанием для отрицательного решения со стороны банка о предоставлении кредита под залог. Как можно заключить в этом случае, данный вывод также согласуется с интуитивными соображениями на этот счет.

Сравнение результатов нечеткого вывода для двух рассмотренных вариантов значений входных переменных показывает, что граничное значение выходной переменной "*кредитоспособность*", которое влияет на решение о предоставлении кредита, может быть выбрано в пределах 5 баллов.

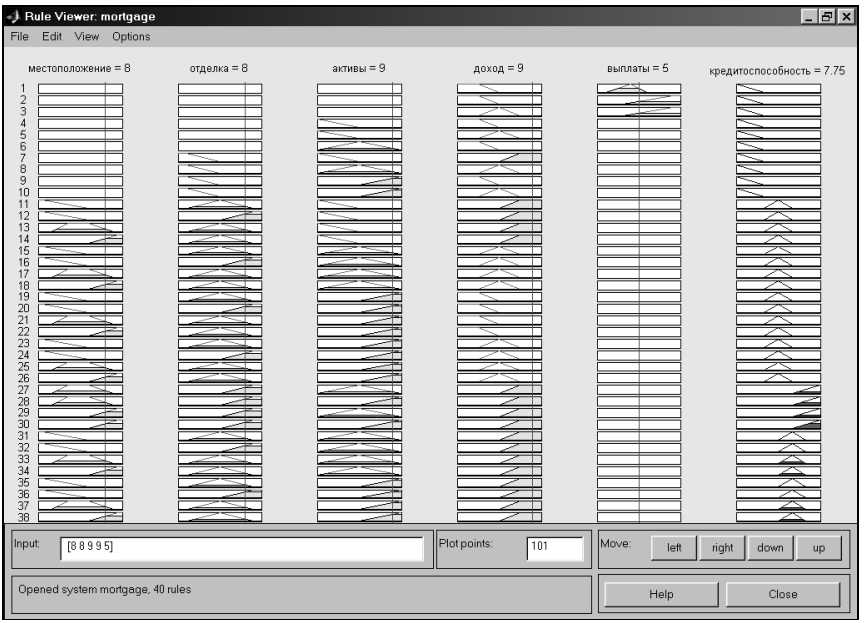


Рис. 17.7. Графический интерфейс программы просмотра правил после выполнения процедуры нечеткого вывода для первого варианта значений входных переменных

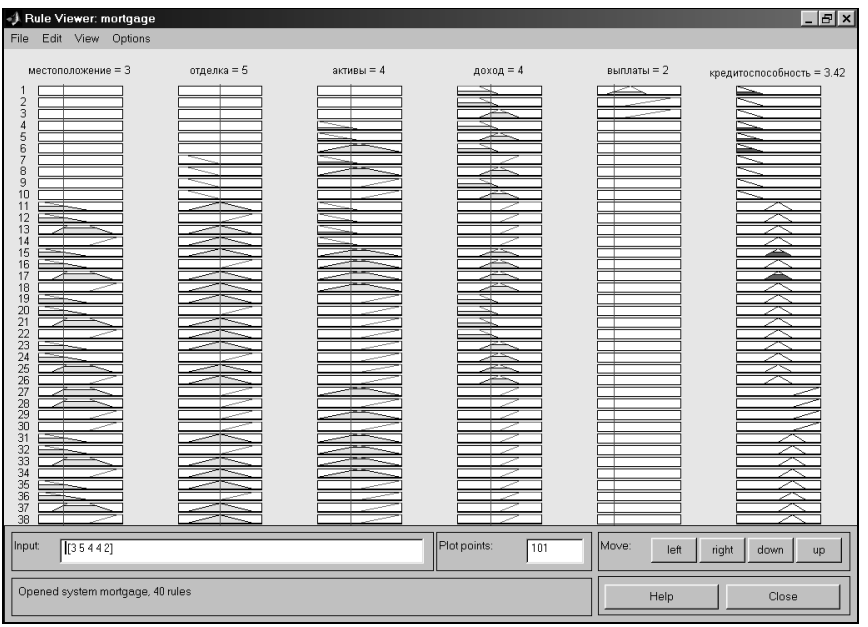


Рис. 17.8. Графический интерфейс программы просмотра правил после выполнения процедуры нечеткого вывода для второго варианта значений входных переменных

Примечание

Для более тонкой настройки построенной нечеткой модели необходимо дополнить ее конкретными методиками балльной оценки отдельных количественных значений входных и выходных лингвистических переменных. Поскольку такие методики в значительной степени зависят от рассматриваемой проблемной области, от сложившейся на данный момент рыночной конъюнктуры и имеют частный характер для конкретного банка, здесь они не рассматриваются.

Для общего анализа разработанной нечеткой модели также может оказаться полезной визуализация соответствующей поверхности нечеткого вывода (рис. 17.9).

Данная поверхность нечеткого вывода позволяет установить зависимость значений выходной переменной от значений отдельных входных переменных нечеткой модели. Анализ этих зависимостей может служить основанием для изменения функций принадлежности входных переменных или нечетких правил с целью повышения адекватности системы нечеткого вывода для конкретных стратегий банков.

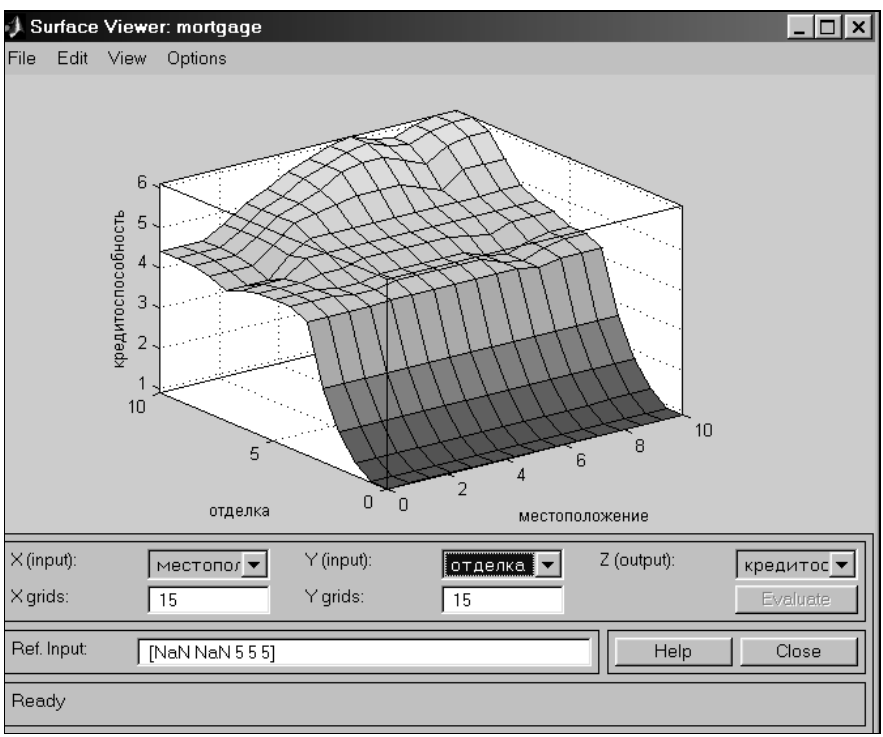


Рис. 17.9. Визуализация поверхности нечеткого вывода рассматриваемой модели для входных переменных "*местоположение*" и "*отделка*"

Следует отметить также то обстоятельство, что разработчики данной нечеткой модели отмечают ее несколько упрощенный характер по сравнению с реально используемой в процессе принятия решений руководством банка. В то же время рассмотренная нечеткая модель обладает достаточно высокой адекватностью, что обуславливает ее успешное применение в практике финансовых операций банка Home&Savings Bank.

В заключение этого примера следует отметить, что реализация данной нечеткой модели в среде MATLAB отличается от оригинальной модели в среде fuzzyTECH. Более подробно особенности построения соответствующей нечеткой модели средствами системы fuzzyTECH рассматриваются в *главе 20*.

17.2. Анализ и прогнозирование валютных цен на финансовом рынке

В качестве второго примера построения и использования адаптивной системы нейро-нечеткого вывода рассмотрим процесс разработки нечеткой модели гибридной сети для решения задачи прогнозирования валютных цен на финансовом рынке.

Суть данной задачи состоит в том, чтобы, зная динамику изменения курсовой стоимости продажи некоторой валюты за фиксированный интервал времени, предсказать значение ее курсовой стоимости на определенный момент времени в будущем. При этом характерной особенностью динамики изменения курса (тренда) является наличие двух основных тенденций в колебаниях соответствующих цен.

С одной стороны, наблюдается общее долгосрочное повышение курсовой стоимости, связанное с величиной инфляции. С другой стороны, наблюдается краткосрочное колебание цен, связанное с целым рядом случайных факторов, адекватное представление которых в той или иной формальной модели вряд ли возможно.

Традиционно для решения данной задачи применяются различные модели технического анализа, основанные на использовании различных индикаторов. В то же время наличие неявных тенденций в динамике изменения курсовой стоимости валют позволяет применить модель адаптивных нейро-нечетких сетей.

В качестве исходных данных можно воспользоваться информацией о динамике курса ЦБ РФ по валюте "Доллар США" (USD) за некоторый временной интервал, которая доступна в Интернете по адресу: **www.finmarket.ru**. Для конкретности возьмем значения курсовой стоимости USD за 1 единицу в период с 1 октября 2002 г. по 10 декабря 2002 г. Данную информацию для удобства дальнейшей работы представим в табличной форме (табл. 17.2).

Таблица 17.2. *Динамика курса иностранной валюты (USD) в период с 1.10.02 по 10.12.02*

Дата	Курс USD	Дата	Курс USD	Дата	Курс USD
01.10.02	31.6827	01.11.02	31.7701	03.12.02	31.8547
02.10.02	31.6919	02.11.02	31.7646	04.12.02	31.8584
03.10.02	31.6982	05.11.02	31.7744	05.12.02	31.8596
04.10.02	31.6809	06.11.02	31.7909	06.12.02	31.8578
05.10.02	31.68	07.11.02	31.7756	07.12.02	31.86
08.10.02	31.6795	11.11.02	31.7756	10.12.02	31.8597
09.10.02	31.6799	12.11.02	31.7756		
10.10.02	31.6803	13.11.02	31.8226		
11.10.02	31.6685	14.11.02	31.8157		
12.10.02	31.6703	15.11.02	31.8203		
15.10.02	31.6703	16.11.02	31.8225		
16.10.02	31.6762	19.11.02	31.8231		
17.10.02	31.6767	20.11.02	31.8224		
18.10.02	31.6761	21.11.02	31.823		
19.10.02	31.6727	22.11.02	31.8248		
22.10.02	31.6973	23.11.02	31.8222		
23.10.02	31.7272	26.11.02	31.8416		
24.10.02	31.7159	27.11.02	31.8382		
25.10.02	31.7109	28.11.02	31.84		
26.10.02	31.7314	29.11.02	31.84		
29.10.02	31.7411	30.11.02	31.8424		
30.10.02	31.6977				
31.10.02	31.7408				

Предположим, что нечеткая модель гибридной сети будет содержать 4 входных переменных. При этом первая входная переменная будет соответствовать курсу USD на текущий банковский день, вторая — курсу USD на предыдущий банковский день, т. е. на день (i–1), где через i обозначен текущий банковский день. Тогда третья входная переменная будет соответствовать курсу USD на (i–2) банковский день, а четвертая — курсу USD на (i–3) банковский день.

Соответствующие обучающие данные могут быть сведены в отдельную таблицу. Объем полученной таким образом обучающей выборки равен 40 (табл. 17.3), что

соответствует динамике курса USD в период с 4 октября 2002 г. по 29 ноября 2002 г. При этом данные за декабрь 2002 г. не вошли в состав обучающей выборки и могут быть использованы для проверки адекватности построенной нечеткой модели.

Таблица 17.3. *Обучающие данные для построения модели гибридной сети*

Первая входная переменная	Вторая входная переменная	Третья входная переменная	Четвертая входная переменная	Выходная переменная
31.6809	31.6982	31.6919	31.6827	31.68
31.68	31.6809	31.6982	31.6919	31.6795
31.6795	31.68	31.6809	31.6982	31.6799
31.6799	31.6795	31.68	31.6809	31.6803
31.6803	31.6799	31.6795	31.68	31.6685
31.6685	31.6803	31.6799	31.6795	31.6703
31.6703	31.6685	31.6803	31.6799	31.6703
31.6703	31.6703	31.6685	31.6803	31.6762
31.6762	31.6703	31.6703	31.6685	31.6767
31.6767	31.6762	31.6703	31.6703	31.6761
31.6761	31.6767	31.6762	31.6703	31.6727
31.6727	31.6761	31.6767	31.6762	31.6973
31.6973	31.6727	31.6761	31.6767	31.7272
31.7272	31.6973	31.6727	31.6761	31.7159
31.7159	31.7272	31.6973	31.6727	31.7109
31.7109	31.7159	31.7272	31.6973	31.7314
31.7314	31.7109	31.7159	31.7272	31.7411
31.7411	31.7314	31.7109	31.7159	31.6977
31.6977	31.7411	31.7314	31.7109	31.7408
31.7408	31.6977	31.7411	31.7314	31.7701
31.7701	31.7408	31.6977	31.7411	31.7646
31.7646	31.7701	31.7408	31.6977	31.7744
31.7744	31.7646	31.7701	31.7408	31.7909
31.7909	31.7744	31.7646	31.7701	31.7756
31.7756	31.7909	31.7744	31.7646	31.7756

Таблица 17.3 *(окончание)*

Первая входная переменная	Вторая входная переменная	Третья входная переменная	Четвертая входная переменная	Выходная переменная
31.7756	31.7756	31.7909	31.7744	31.7756
31.7756	31.7756	31.7756	31.7909	31.8226
31.8226	31.7756	31.7756	31.7756	31.8157
31.8157	31.8226	31.7756	31.7756	31.8203
31.8203	31.8157	31.8226	31.7756	31.8225
31.8225	31.8203	31.8157	31.8226	31.8231
31.8231	31.8225	31.8203	31.8157	31.8224
31.8224	31.8231	31.8225	31.8203	31.823
31.823	31.8224	31.8231	31.8225	31.8248
31.8248	31.823	31.8224	31.8231	31.8222
31.8222	31.8248	31.823	31.8224	31.8416
31.8416	31.8222	31.8248	31.823	31.8382
31.8382	31.8416	31.8222	31.8248	31.84
31.84	31.8382	31.8416	31.8222	31.84
31.84	31.84	31.8382	31.8416	31.8424

Примечание

Для большего удобства подготовки соответствующего файла с обучающими данными целесообразно воспользоваться редактором электронных таблиц MS Excel, обладающим возможностью копирования содержимого ряда ячеек.

Сохраним обучающую выборку во внешнем файле под именем priceUSD.dat. После этого откроем редактор ANFIS, в который загрузим этот файл с обучающими данными. Внешний вид редактора ANFIS с загруженными обучающими данными изображен на рис. 17.10.

Перед генерацией структуры системы нечеткого вывода типа Сугено после вызова диалогового окна свойств зададим для каждой из входных переменных по 3 лингвистических терма, а в качестве типа их функций принадлежности выберем треугольные функции (установленные системой MATLAB по умолчанию). В качестве типа функции принадлежности выходной переменной зададим линейную функцию (рис. 17.11).

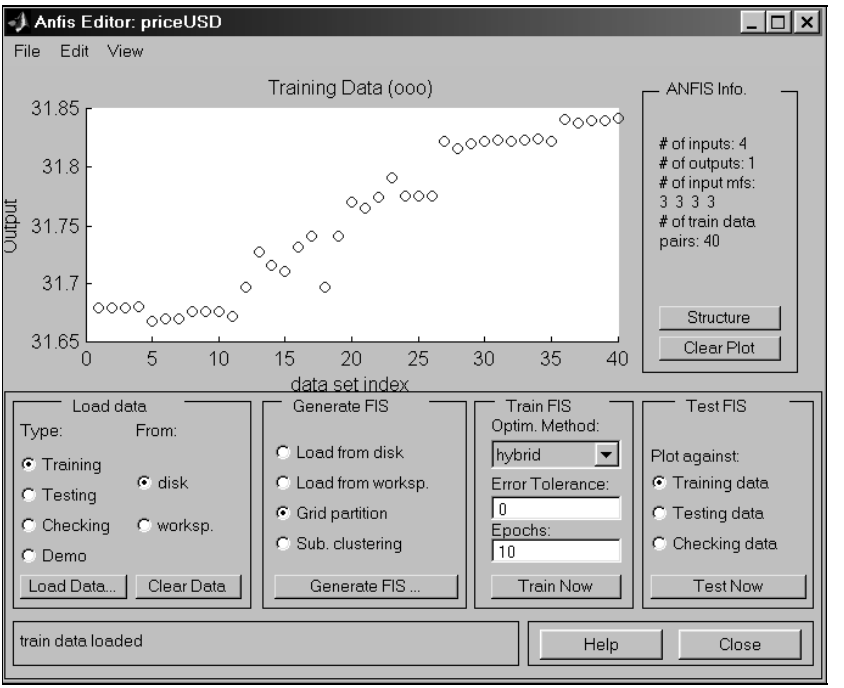


Рис. 17.10. Графический интерфейс редактора ANFIS после загрузки обучающих данных

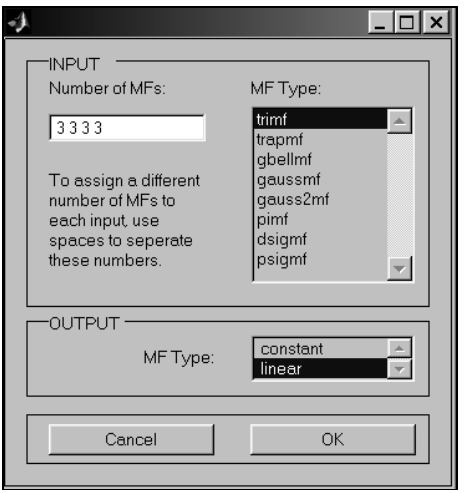


Рис. 17.11. Диалоговое окно для задания количества и типа функций принадлежности

Для обучения гибридной сети воспользуемся гибридным методом обучения с уровнем ошибки 0, а количество циклов обучения зададим равным 10. После окончания обучения данной гибридной сети может быть выполнен анализ графика ошибки обучения (рис. 17.12), который показывает, что обучение практически закончилось после 3-го цикла.

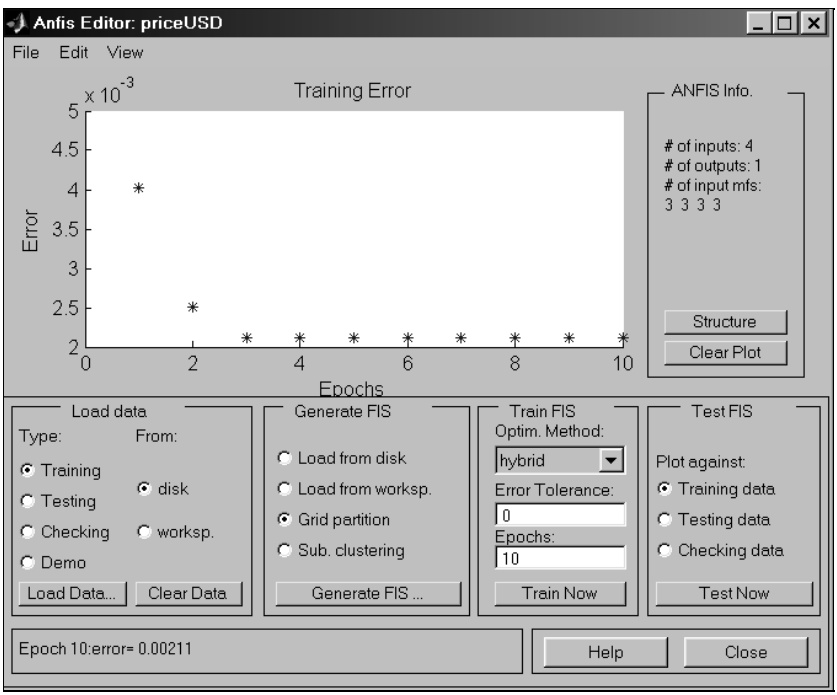


Рис. 17.12. График зависимости ошибки обучения от количества циклов обучения

После обучения гибридной сети можно визуально оценить структуру построенной нечеткой модели (рис. 17.13). Очевидно, графическая наглядность данной модели оставляет желать лучшего, поскольку общее количество правил в разработанной адаптивной системе нейро-нечеткого вывода равно 81, что затрудняет их визуальный контроль и оценку.

С помощью графических средств системы MATLAB можно выполнить контроль и настройку параметров функций принадлежности входных переменных и правил нечетких продукций. Для выполнения соответствующих операций можно воспользоваться редактором функций принадлежности (рис. 17.14). Однако до проверки адекватности построенной нечеткой модели оставим все параметры функций принадлежности без изменений.

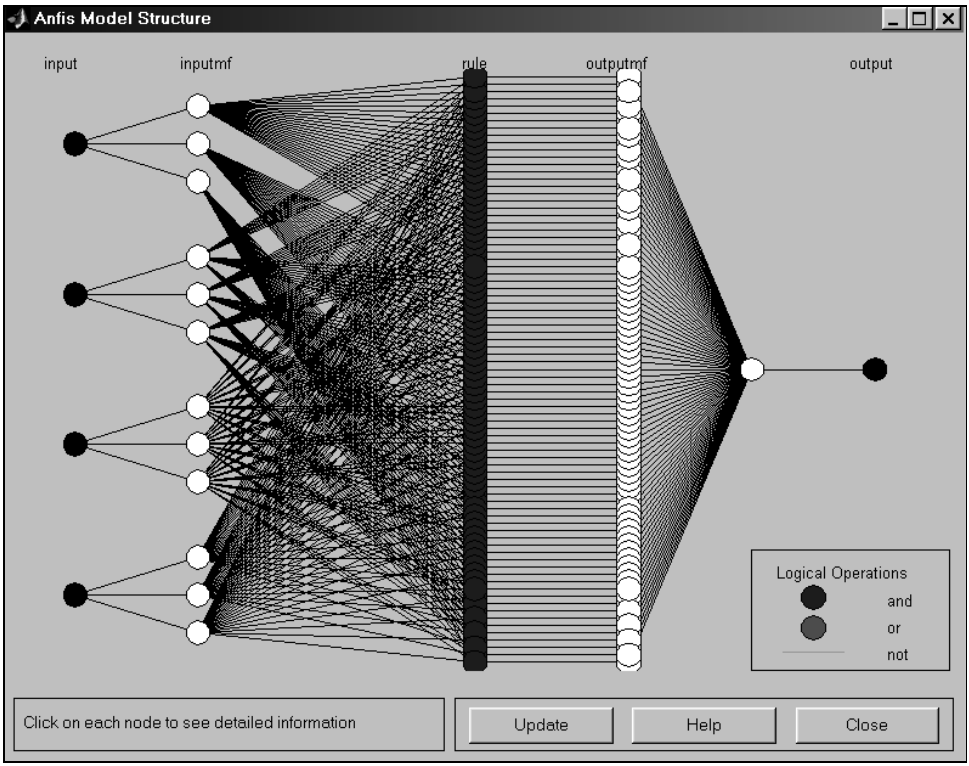


Рис. 17.13. Структура сгенерированной системы нечеткого вывода

Выполним проверку адекватности построенной нечеткой модели гибридной сети. Для этой цели сделаем ретроспективный прогноз значения курсовой стоимости USD на следующий банковский день, например, на 3 декабря 2002 г., считая для этого случая текущим банковским днем — 30 ноября 2002 г.

Поскольку точность количественных значений, обеспечиваемая графическими средствами пакета Fuzzy Logic Toolbox, является недостаточной для решения данной задачи, воспользуемся функцией командной строки `evalfis`. В качестве аргументов этой функции *(см. приложение 3)* укажем вектор значений курсовой стоимости USD на текущий и 3 предшествующих банковских дня. Полный формат вызова этой функции будет следующим:

```
out=evalfis([31.8424 31.84 31.84 31.8382], priceUSD)
```

где `out` — условное имя выходной переменной; `31.8424` — значение курсовой стоимости USD на 30.11.02; `31.84` — значение курсовой стоимости USD на 29.11.02; `31.84` — значение курсовой стоимости USD на 28.11.02; `31.8382` — значение курсовой стоимости USD на 27.11.02; `priceUSD` — имя структуры FIS, предварительно загруженной в рабочую область системы MATLAB.

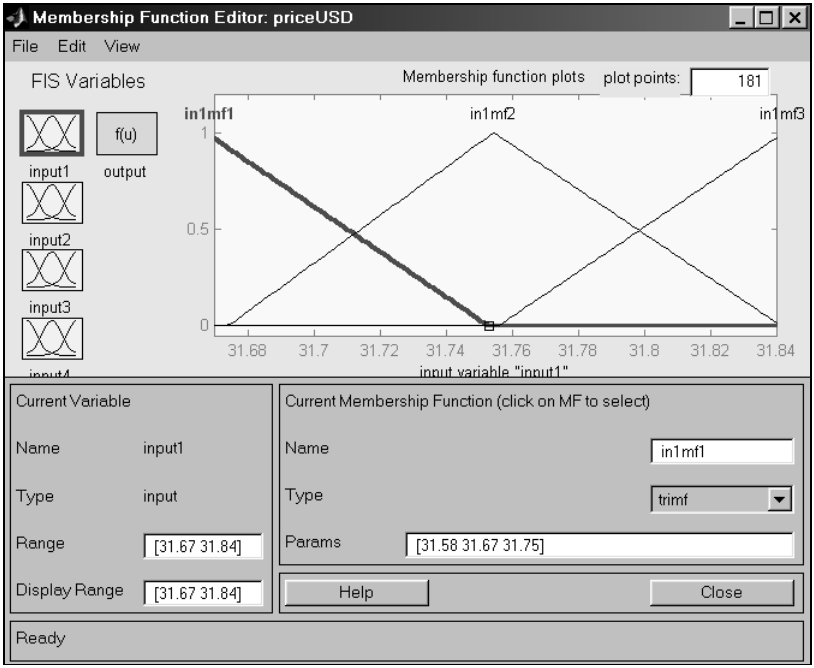


Рис. 17.14. Графический интерфейс редактора функций принадлежности построенной системы нечеткого вывода для проверки первой входной переменной

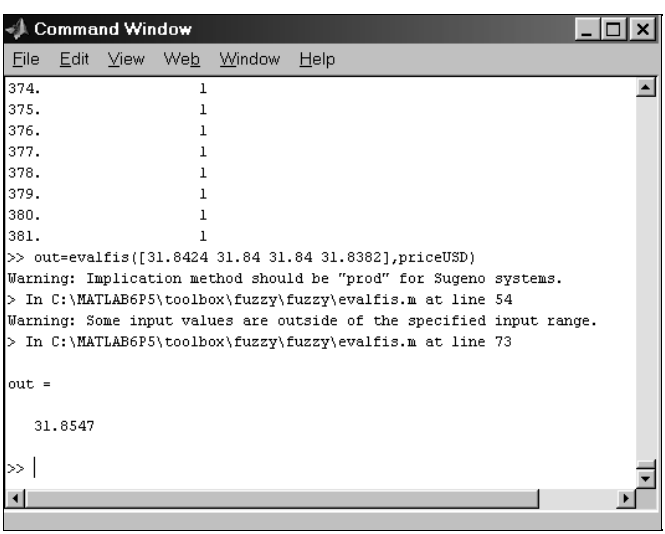
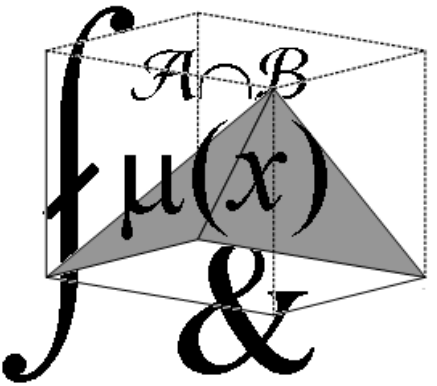


Рис. 17.15. Окно команд с отображением результата оценки построенной нечеткой модели гибридной сети

После выполнения этой команды с помощью разработанной нечеткой модели будет получено значение выходной переменной для 3.12.02, равное `31.8547` (рис. 17.15). Сравнивая полученное значение с соответствующим значением из табл. 17.2, можно констатировать *абсолютное совпадение* этих значений (!), несмотря на два предупреждения при выполнении команды нечеткого вывода системой MATLAB.

Таким образом, проверка построенной нечеткой модели гибридной сети показывает достаточно высокую степень ее адекватности реальным исходным данным, что позволяет сделать вывод о возможности ее практического использования для прогнозирования курсовой стоимости USD на финансовом рынке валют. В этом случае нечеткие модели адаптивных систем нейро-нечеткого вывода могут считаться новым и конструктивным инструментом технического анализа финансовых рынков.

Рассмотренный подход является перспективным направлением для построения и использования соответствующих нечетких моделей прогнозирования цен других финансовых инструментов, таких как курсы других валют, акций компаний, фьючерсов и опционов. Действительно, общим для всех этих инструментов с позиций технического анализа является отсутствие априорных предположений о динамике колебаний соответствующих курсов цен, что вполне согласуется с исходными предпосылками построения нечетких моделей адаптивных систем нейро-нечеткого вывода.



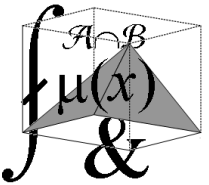
ЧАСТЬ III

НЕЧЕТКОЕ МОДЕЛИРОВАНИЕ В СРЕДЕ fuzzyTECH

Программа fuzzyTECH, разработанная и постоянно обновляемая компанией INFORM GmbH (Inform Software Corporation, Германия), предназначена для решения различных задач нечеткого моделирования. В отличие от системы MATLAB, программа fuzzyTECH является специализированным средством, которое позволяет разрабатывать и исследовать разнообразные нечеткие модели в графическом режиме, а также преобразовывать их в программный код на одном из языков программирования с возможностью последующей реализации в программируемых микроконтроллерах.

Программа fuzzyTECH обладает возможностью использования ее в качестве сервера или клиента при нечетком управлении удаленными объектами. Интересной особенностью программы fuzzyTECH, которую также следует обязательно отметить, является возможность автоматической генерации документации по нечетким моделям в виде текста с иллюстрациями в формате RTF.

Глава 18



Общая характеристика программы fuzzyTECH

В настоящей книге рассматривается демонстрационная версия программы fuzzyTECH 5.52 (далее — просто fuzzyTECH), которая доступна в Интернете по адресу: **www.fuzzytech.com**. Дистрибутив программы в форме самораспаковывающегося архивного файла занимает около 16 Мбайт. Основными ограничениями демонстрационной версии программы fuzzyTECH являются отсутствие возможности сохранения проекта и генерации программного кода на том или ином языке программирования.

Системные требования, которым должен удовлетворять компьютер пользователя для инсталляции программы fuzzyTECH, следующие.

- Процессор 486 и выше (для лучшей производительности рекомендуется Pentium I).
- Операционная система MS Windows 9*x*, MS Windows 4.0 или выше.
- Оперативная память (RAM) как минимум 16 Мбайт.
- Свободное место на жестком диске, необходимое для установки программы fuzzyTECH, как минимум 25 Мбайт.
- Графический адаптер как минимум VGA (256 цветов при разрешении 800×600) и монитор, поддерживающий не менее 256 цветов.
- Один из браузеров Интернета для просмотра документации в формате HTML и доступа к сайту компании INFORM GmbH по адресу: **www.fuzzytech.com**.
- Программа Adobe Acrobat Reader для просмотра документации в формате PDF.

Процесс инсталляции программы достаточно традиционен и предполагает ввод установочного пароля. Для получения установочного пароля на сайте компании INFORM GmbH необходимо заполнить регистрационную карту, после чего пароль высылается пользователю по e-mail.

18.1. Общая характеристика нечеткого проекта в среде fuzzyTECH

Каждая система нечеткого вывода в программе fuzzyTECH представляется в форме отдельного *проекта* (Project). При этом следует заметить, что в программе fuzzyTECH проекты называются *нечетко-логическими системами* или системами нечеткой логики (Fuzzy Logic Systems, сокращенно — FLS).

Примечание

С целью унификации терминологии и обозначений в *части III* нечетко-логические системы в контексте программы fuzzyTECH обозначаются общим для всех подобных систем термином — *системы нечеткого вывода*. В то же время сокращение FIS, принятое в системе MATLAB для систем нечеткого вывода, чтобы избежать возможной путаницы, здесь использоваться не будет, поскольку является специфичным для нечетких моделей пакета Fuzzy Logic Toolbox.

Хотя программа fuzzyTECH и система MATLAB используют единые принципы нечеткого моделирования, существует несколько принципиальных отличий в реализации систем нечеткого вывода FLS в программе fuzzyTECH от ранее рассмотренных систем FIS в пакете Fuzzy Logic Toolbox. Главные из них изложены ниже.

- Проект системы нечеткого вывода в fuzzyTECH может иметь несколько блоков правил (Rule Blocks) нечетких продукций, каждый из которых может содержать собственные входные и выходные лингвистические переменные. При этом отдельные блоки правил могут соединяться между собой последовательным или параллельным образом.
- Кроме входных (Inputs) и выходных (Outputs) лингвистических переменных проекты fuzzyTECH могут иметь так называемые *промежуточные* (Intermediates) лингвистические переменные. Эти переменные появляются в тех случаях, когда блоки правил соединяются последовательно, т. е. выход одного блока правил соединяется с входом другого блока правил.
- Все операции по разработке, редактированию, отладке и анализу проектов в программе fuzzyTECH выполняются в графическом интерактивном режиме, при этом для создания прототипов проектов и спецификации их отдельных компонентов могут быть использованы различные мастера (Wizards).
- На основе разработанного и отлаженного проекта программой fuzzyTECH может быть сгенерирован программный код реализации системы нечеткого вывода на одном из языков программирования (C, Java, MS Visual C++, MS Visual Basic, MS VBA, COBOL, Assembler, язык m-файлов системы MATLAB). В дальнейшем полученные подобным образом листинги программного кода могут быть откомпилированы для той или иной вычислительной платформы и использованы независимо от программы fuzzyTECH для реализации в нечетких микроконтроллерах.

Примечание

Хотя система MATLAB также позволяет в отдельных случаях преобразовывать m-файлы в программный код на языке C, однако программа fuzzyTECH специально ориентирована на более тесную интеграцию с аппаратной реализацией нечетких моделей управления в нечетких микроконтроллерах. Поэтому кроме рассматриваемой версии программы fuzzyTECH 5.5 Professional существуют также и другие варианты этой программы, а именно: fuzzyTECH Online IA-S5, Fuzzy-166, MCU-C, MCU-96, MCU-166, MCU-320, MCU-51, MCU-374, MCU-ST6, MCU-MP, MCU-HC11/12, MCU-HC05/08, которые специально ориентированы на тот или иной тип микропроцессора и позволяют генерировать код на языке Ассемблера. Так, например, релизы MCU-HC11/12 и MCU-HC05/08 предназначены для использования с нечеткими микроконтроллерами компании Motorola. Кроме этого, указанные релизы могут отличаться функциональностью модуля (RTRCD, Real Time Remote Control Debug).

- Разработчики программы fuzzyTECH явно указывают ограничения на размерность проектов систем нечеткого вывода, которые могут быть реализованы в ее среде. Рассматриваемая версия программы fuzzyTECH 5.5 Professional имеет следующие количественные ограничения на отдельные компоненты разрабатываемых проектов:
  - общее количество лингвистических переменных проекта не должно превышать 255, из них входных — 255 и выходных — 32;
  - каждая из лингвистических переменных может иметь не более 32 нечетких термов или не более 255 обычных (Categorical), т. е. не нечетких значений. При этом общее количество термов у всех переменных не должно превышать 65 535;
  - общее количество блоков правил нечетких продукций проекта не должно превышать 32. При этом у каждого блока правил может быть не более 11 входных и 11 выходных лингвистических переменных.
- Использование в программе fuzzyTECH технологии *динамического обмена данными* (Dynamic Data Exchange или сокращенно — DDE) позволяет совместно использовать разработанные нечеткие модели с другими программами и инструментами, такими как MS Access, MS Excel, MATLAB. При этом программа fuzzyTECH может выступать как в роли сервера, так и в роли клиента, что существенно расширяет диапазон возможных приложений разрабатываемых нечетких моделей. В последнем случае для нечеткого управления удаленными объектами необходимо использовать тот или иной протокол передачи данных (TCP/IP, IPX/SPX) и дополнительные интерфейсы (последовательный интерфейс RS232, SDI 5, SFS, FTOCC).
- Все проекты в среде fuzzyTECH сохраняются в отдельных файлах проектов формата FTL (Fuzzy Technology Language), которые имеют расширение ftl. Они представляют собой обычные текстовые файлы, которые можно просматривать и редактировать любым ASCII-редактором (например, MS Notebook). Формат FTL был специально разработан компаниями Intel Corp. и Inform Software Corp. в 1991 г. для представления систем нечеткого вывода

в форме структурируемого текста. Этот формат поддерживают все ведущие разработчики программных и аппаратных решений, основанных на использовании систем нечеткой логики, позволяя осуществить их перенос на различные платформы.

Примечание

Файлы формата FTL по своей структуре аналогичны записи системы нечеткого вывода на языке FCL *(см. главу 8)*. Последнее отнюдь не случайно, поскольку программа fuzzyTECH поддерживает стандарт IEC 1131-7. В то же время нельзя не заметить, что файлы проектов формата FTL также напоминают распространенные ранее файлы конфигурации оборудования, имевшие, как правило, расширение ini. Пример файла формата FTL с характеристикой его структуры приводится в *приложении 4*.

- Кроме формата FTL, файлы проектов могут быть представлены в формате FTR (fuzzyTECH Runtime), соответственно, с расширением ftr. Файлы формата FTR, в отличие от формата FTL, представляют проекты в бинарном виде. При этом форма представления зависит от используемой версии программы fuzzyTECH. Файл формата FTR содержит всю информацию о проекте и может быть использован для преобразования в соответствующий файл проекта формата FTL.
- В нечетких проектах fuzzyTECH могут быть использованы различные типы и формы функций принадлежности термов лингвистических переменных. Что касается типов функций принадлежности, то пользователь может выбрать один из вариантов, приведенных ниже.
  - *Стандартный* вариант функции принадлежности (Standard MBFs), который иногда называют "4-точечным" вариантом, поскольку основан на использовании 4 характеристических точек или параметров для задания соответствующей функции принадлежности.
  - *Произвольный* вариант функции принадлежности (Arbitrary MBFs), в рамках которого можно использовать до 16 характеристических точек или параметров для задания или аппроксимации соответствующей функции принадлежности.
  - *Инверсный* вариант функции принадлежности (Inverse MBFs), который может оказаться полезным при определении правил нечетких продукций с отрицанием существующих в проекте термов (inverse terms) для отдельных лингвистических переменных.
- Каждый из типов функции принадлежности может иметь одну из форм, приведенных ниже.
  - *Линейную* (L-shape), которая предполагает представление функции принадлежности в форме треугольной, трапециевидной функции или их некоторой комбинации.

- *S-образную* (S-shape), которая предполагает представление функции принадлежности в форме некоторой *S*-оообразной, *Z*-образной или *П*-образной кривой *(см. главу 2)*.

❒ В программе fuzzyTECH могут быть различные методы фаззификации входных переменных. При этом пользователь может выбрать один из следующих вариантов фаззификации.

- *Стандартный метод фаззификации* (Compute MBF) предполагает использование функций принадлежности стандартного типа — треугольных, трапециевидных и кусочно-линейных кривых.
- *Нечеткий вход* (Fuzzy Input) указывает, что все термы соответствующей лингвистической переменной представляются в форме вектора значений функции принадлежности. Этот вариант более эффективен при реализации систем нечеткого вывода на конкретной аппаратной платформе.
- *Нечеткий вход в форме таблицы* (Look-up-MBF) указывает, что нечеткие значения соответствующей лингвистической переменной вычисляются с помощью некоторого алгоритма и представляются в форме таблицы. Этот вариант может быть использован для увеличения скорости вычислений на некоторых типах микроконтроллеров и нечетких микропроцессоров.
- *Быстрый метод фаззификации* (Fast Computation of MBF) является разновидностью стандартного метода применительно к некоторым конкретным типам микроконтроллеров. Этот вариант обладает наибольшей эффективностью при использовании конкретной аппаратной платформы.
- *Обычный вход* (Categorical) указывает, что все термы соответствующей лингвистической переменной представляются в форме обычного вектора значений.

❒ В нечетких проектах fuzzyTECH могут быть также использованы различные методы агрегирования, композиции, аккумуляции и дефаззификации полученных результатов нечеткого вывода. Для агрегирования подусловий правил нечетких продукций разработчик может воспользоваться одним из следующих вариантов:

- min-max — для логического нечеткого И используется операция минимума (6.2), для логического нечеткого ИЛИ — операция максимума (6.6), для логического отрицания НЕ — операция разности (6.1). При этом может быть использован дополнительный параметр компенсации $\lambda \in [0, 1]$ для получения окончательного значения результата агрегирования подусловий в соответствии со следующей формулой:

$$b_k'' = (1-\lambda)\cdot \min_{i\in\{1,2,\ldots,m\}}\{\mu(a_i)\} + \lambda \cdot \max_{i\in\{1,2,\ldots,m\}}\{\mu(a_i)\}; \qquad (18.1)$$

- min-avg — для логического нечеткого И используется операция минимума (6.2), для логического нечеткого ИЛИ — арифметическое среднее, для логического отрицания НЕ — операция разности (6.1). При этом может быть использован дополнительный параметр компенсации $\lambda \in [0, 1]$ для получе-

ния окончательного значения результата агрегирования подусловий в соответствии со следующей формулой:

$$b_k'' = (1-\lambda)\cdot \min_{i\in\{1,2,\ldots,m\}}\{\mu(a_i)\} + \lambda\cdot\sum_{i=1}^{m}\frac{\mu(a_i)}{m}; \qquad (18.2)$$

- gamma — для агрегирования подусловий правил нечетких продукций используется следующая формула:

$$b_k'' = (\prod_{i=1}^{m}\mu(a_i))^{1-\gamma}\cdot(1-\prod_{i=1}^{m}(1-\mu(a_i)))^{\gamma}, \qquad (18.3)$$

где γ — количественный параметр компенсации, задаваемый разработчиком из некоторых эвристических соображений ($\gamma\in[0, 1]$).

❒ Для композиции заключений правил нечетких продукций пользователь может воспользоваться одним из следующих вариантов задания весовых коэффициентов правил (при этом используется метод prod-активизации, определяемый в соответствии с формулой (7.7)).

- *Стандартный* (Standard), при котором значение весового коэффициента для каждого правила предполагается неизменным и равным 1.

Примечание

Следует заметить, что хотя в программе fuzzyTECH весовой коэффициент правила нечеткой продукции получил название *степени поддержки* (Degree of Support или сокращенно — DoS), из соображений унификации терминологии для DoS будет использоваться более традиционный термин — весовой коэффициент.

- *Определяемый пользователем* (Fuzzy Associative Maps или сокращенно — FAM), при котором значение весового коэффициента DoS для каждого правила можно изменять в пределах интервала [0, 1].

❒ Для аккумуляции заключений правил нечетких продукций пользователь может воспользоваться одним из следующих методов:

- *обычным* (max), при котором результат нечеткого вывода в блоке правил нечетких продукций определяется как объединение нечетких множеств по формуле (6.6);
- *граничной суммы* (bsum), при котором результат нечеткого вывода в блоке правил нечетких продукций определяется как объединение нечетких множеств по формуле (6.8).

❒ В проектах fuzzyTECH могут быть использованы различные методы дефаззификации выходных переменных. При этом пользователю предоставляется возможность выбора одного из следующих методов дефаззификации.

- *Стандартный метод* (Center-of-Maximum или сокращенно — CoM), который, по мнению разработчиков, представляется методом наилучшего компромисса для получения окончательного значения выходных перемен-

ных. В программе fuzzyTECH метод дефаззификации CoM работает аналогично методу центра тяжести CoG, определяемому по формуле (7.9).

- *Метод центра площади* (Centre-of-Area или сокращенно — CoA), который определяется по формуле (7.11).
- *Метод среднего максимума* (Mean-of-Maximum или сокращенно — MoM), который, по мнению разработчиков, представляется наиболее гибким методом и определяется как арифметическое среднее между левым (7.12) и (7.13) правым модальными значениями, при этом в качестве моды нечеткого множества рассматривается максимальное значение ее функции принадлежности.
- *Метод гиперцентра максимума* (Hyper Center-of-Maximum или сокращенно — HyperCoM), который может быть использован только в дополнительных модулях программы fuzzyTECH. Этот метод позволяет учитывать положительные и отрицательные результаты нечетких выводов и в соответствии с этим формирует некоторое оптимальное значение для выходных переменных.

Примечание

Как можно заметить, используемые в программе fuzzyTECH сокращения CoM и MoM для методов дефаззификации отсутствуют в описании языка FCL *(см. главу 8)*. Этот факт представляется по меньшей мере странным, поскольку компания INFORM GmbH не только объявила о поддержке в программе fuzzyTECH стандарта IEC 1131-7, но, по всей видимости, была в числе разработчиков последнего.

18.2. Основные элементы рабочего интерфейса программы fuzzyTECH

После инсталляции в главном меню MS Windows появляется новая программная группа с именем: fuzzyTECH 5.5. Для запуска программы следует выбрать соответствующий пункт меню в этой программной группе **(Пуск>Программы> fuzzyTECH 5.5>fuzzyTECH 5.5 Professional Edition)**. После запуска программы fuzzyTECH на экране появится рабочий интерфейс (главное окно) программы с пустым проектом (рис. 18.1).

Если уже имеется разработанный ранее нечеткий проект, то его можно выбрать и загрузить в программу fuzzyTECH одним из следующих способов:

- через программную группу **Пуск>Программы>fuzzyTECH 5.5>Examples**;
- с использованием пункта главного меню **File>Open...** (Файл>Открыть);
- с использованием кнопки панели инструментов **Open** (Открыть).

После загрузки имеющегося проекта рабочий интерфейс программы fuzzyTECH изменится и будет отображать информацию о соответствующем проекте (рис. 18.2).

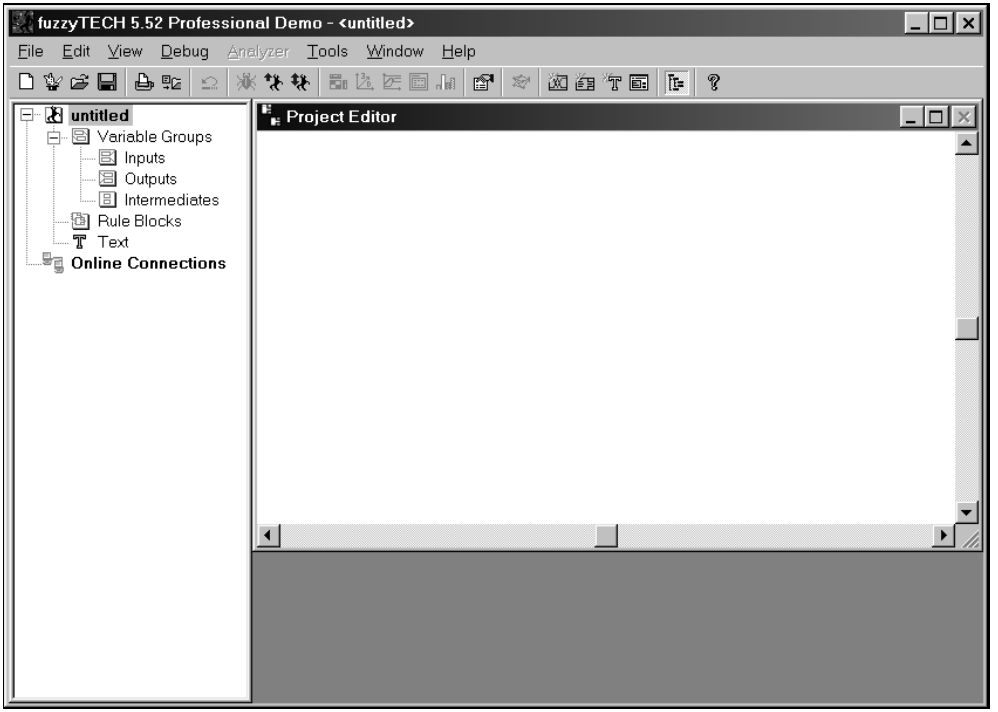


Рис. 18.1. Общий вид графического интерфейса программы fuzzyTECH 5.52 для нового проекта

Примечание

Следует обратить внимание на тот факт, что дистрибутив Demo-версии программы fuzzyTECH поставляется с примерами разработанных компанией INFORM GmbH проектов систем нечеткого вывода, которые удобно использовать для изучения особенностей процесса нечеткого моделирования в среде fuzzyTECH.

В верхней части графического интерфейса программы fuzzyTECH расположена строка заголовка, в которой указывается название и версия программы, а также имя разрабатываемого или загруженного проекта. В правой части строки заголовка находятся стандартные кнопки управления графическим окном программы.

Под строкой заголовка находится строка главного меню программы fuzzyTECH и строка панели инструментов (рис. 18.3). Выбор той или иной операции меню выполняется стандартным для всех приложений MS Windows образом. Панель инструментов содержит кнопки, позволяющие осуществить доступ к наиболее часто используемым командам и операциям. Назначение отдельных кнопок панели инструментов приводится в табл. 18.3.

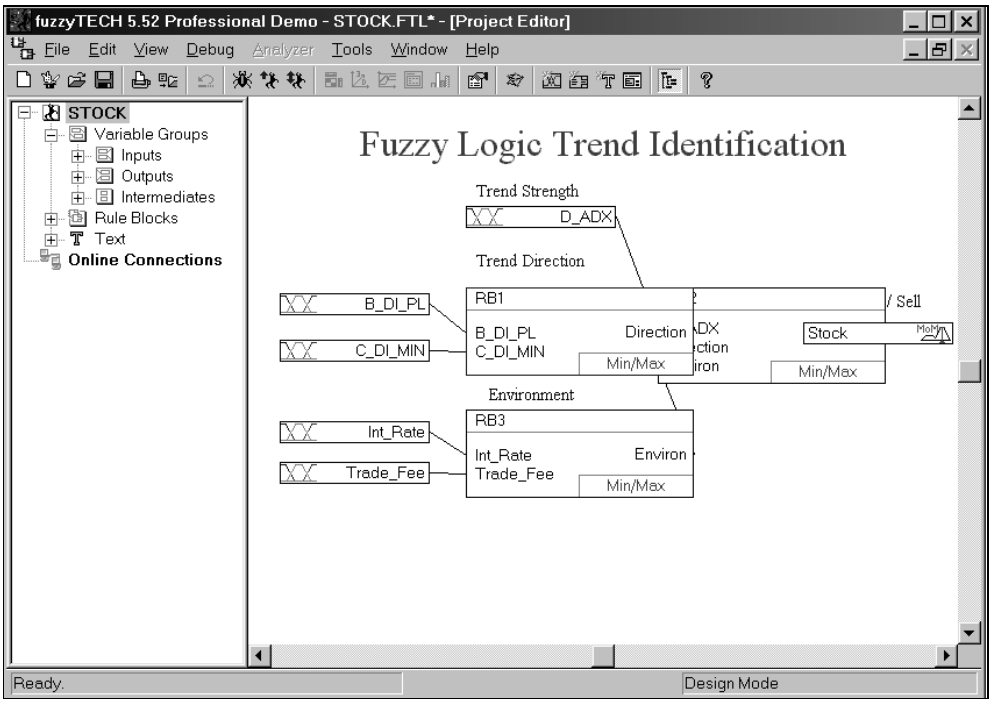


Рис. 18.2. Общий вид графического интерфейса программы fuzzyTECH 5.52 для разработанного проекта

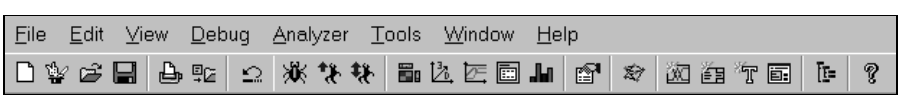


Рис. 18.3. Главное меню и панель инструментов программы fuzzyTECH

В центре графического интерфейса находится окно *редактора проекта* (Project Editor), в котором отображается структура системы нечеткого вывода в форме прямоугольников переменных и блоков правил (рис. 18.4). Редактор проекта является основным средством, которое используется для визуализации общей структуры и быстрого доступа к различным инструментам редактирования свойств систем нечеткого вывода в графическом режиме.

Редактор проекта программы fuzzyTECH позволяет визуализировать структуру всего проекта и графически представить отношения между компонентами проекта. Двойной щелчок на изображении того или иного прямоугольника позволяет открыть окно редактирования свойств соответствующего компонента системы нечеткого вывода.

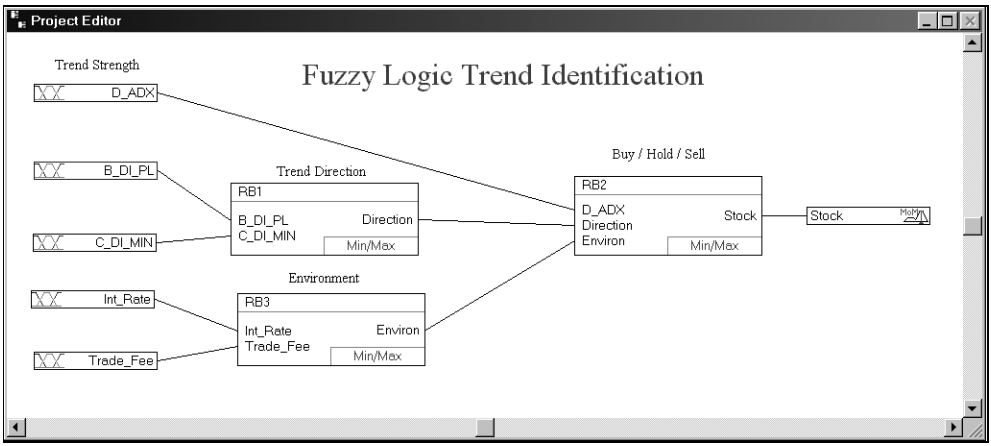


Рис. 18.4. Окно редактора проекта программы fuzzyTECH

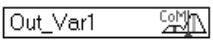
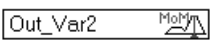
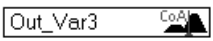
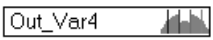
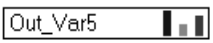
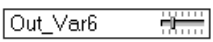
Отдельные компоненты системы нечеткого вывода в графическом окне редактора проекта представляются в форме прямоугольников или пиктограмм. При этом внешний вид используемых пиктограмм характеризует некоторые особенности или свойства отдельных компонентов проекта. Так, например, входные лингвистические переменные системы нечеткого вывода изображаются в виде прямоугольников, которые имеют дополнительный значок для визуализации метода фаззификации лингвистической переменной (табл. 18.1).

Таблица 18.1. *Графические пиктограммы методов фаззификации лингвистических переменных*

Графическое изображение пиктограммы	**Назначение дополнительного значка пиктограммы**
variable1	Стандартный метод фаззификации лингвистической переменной (Compute MBF)
Variable2	Нечеткий вход для лингвистической переменной (Fuzzy Input)
Variable3	Отсутствие фаззификации для лингвистической переменной (Categorical)
Variable4	Фаззификация лингвистической переменной с помощью дополнительного интерфейса, определяемая разработчиком

Выходные лингвистические переменные системы нечеткого вывода также изображаются в виде прямоугольников, которые имеют дополнительный значок для визуализации метода дефаззификации лингвистической переменной (табл. 18.2).

Таблица 18.2. *Графические пиктограммы методов дефаззификации лингвистических переменных*

Графическое изображение пиктограммы	Назначение дополнительного значка пиктограммы
	Метод центра максимума (Center-of-Maximum или сокращенно — CoM)
	Метод среднего максимума (Mean-of-Maximum или сокращенно — MoM)
	Метод центра площади (Centre-of-Area или сокращенно — CoA)
	Метод гиперцентра максимума (Hyper Center-of-Maximum или сокращенно — HyperCoM)
	Отсутствие дефаззификации выходной переменной (нечеткая лингвистическая переменная)
	Дефаззификация лингвистической переменной в окне анализа результатов нечеткого вывода

Каждый из компонентов системы нечеткого вывода может иметь свое имя, которое в разрабатываемом проекте специфицируется с помощью отдельного текстового объекта **(Text Object)** или блока текста. Для задания текстового объекта предназначено отдельное диалоговое окно, которое может быть вызвано с помощью команды меню **Edit>New Text…** или с помощью соответствующей кнопки панели инструментов (см. табл. 18.3). После вызова окна свойств текста можно не только определить собственно текстовый блок, но и свойства шрифта этого текста (рис. 18.5).

Примечание

В связи с заданием текста следует заметить, что в окне свойств текста доступны все шрифты, установленные в операционной системе MS Windows. Поэтому программа fuzzyTECH корректно отображает символы кириллицы в текстовых блоках в окне редактора разрабатываемого проекта.

Слева от окна редактора проекта расположено *окно просмотра структуры проекта* **(Treeview)**, которое представляет разрабатываемый проект в форме иерархической структуры или дерева (рис. 18.6). Окно просмотра структуры проекта содержит перечень всех компонентов проекта. Щелчок на изображении знака "+" группы компонентов позволяет раскрыть соответствующую вложенную структуру этой группы. В этом окне также можно получить быстрый доступ к инструментам редактирования отдельных компонентов или проекта в целом. А именно, двойной щелчок на имени выбранного компонента проекта позволяет открыть окно редактирования свойств данного компонента системы нечеткого вывода.

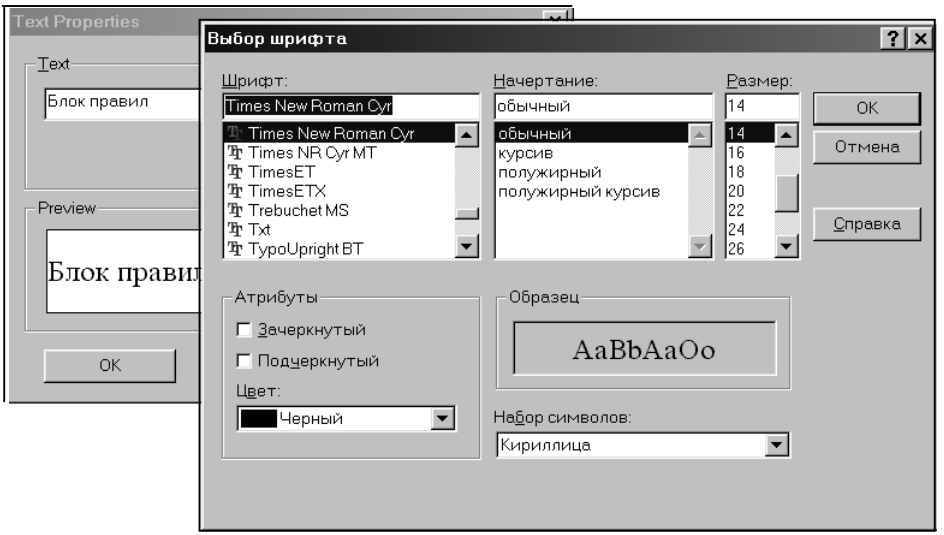


Рис. 18.5. Диалоговое окно задания текста и определения свойств его шрифта

Ни одно из этих двух окон не может быть закрыто в программе fuzzyTECH. Однако окно редактора проекта можно минимизировать с помощью соответствующей кнопки управления окном. Окно просмотра структуры проекта можно скрыть (сделать невидимым), нажав соответствующую кнопку на панели инструментов, которая расположена слева от кнопки **Справка** (см. табл. 18.3).

Строка главного меню программы fuzzyTECH содержит всплывающее меню с большим количеством различных операций некоторые из них содержат вложенные меню или вызовы диалоговых окон. При этом не все операции могут быть доступны в программе. Некоторые из них являются неактивными и требуют перехода в интерактивный режим отладки проекта.

Все пункты главного меню имеют комбинацию клавиш быстрого доступа, для чего необходимо одновременно нажать клавишу <Alt> и подчеркнутую букву соответствующего пункта меню. При этом в качестве рабочего языка в системе используется английский.

Примечание

Вообще говоря, программа fuzzyTECH в качестве рабочего языка может использовать как английский, так и немецкий. Что касается использования символов кириллицы, то программа позволяет вводить на русском лишь текстовые блоки и комментарии.

В нижней части графического интерфейса программы fuzzyTECH расположена строка состояния, в левой части которой дублируется название отдельных кнопок панелей инструментов, а в правой части указывается текущий режим работы программы.

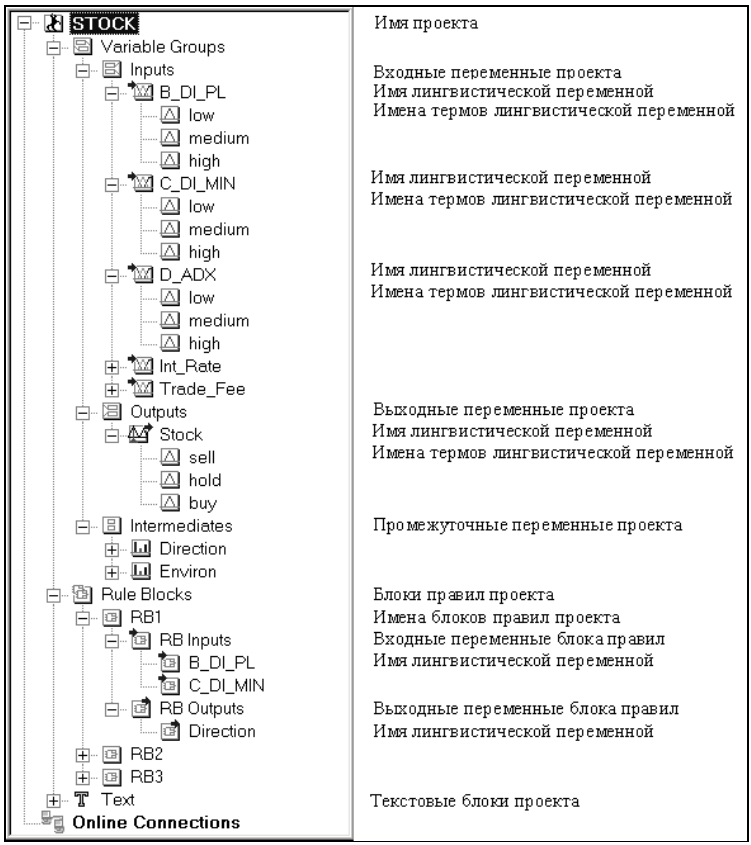


Рис. 18.6. Окно просмотра структуры проекта программы fuzzyTECH

В состав программы fuzzyTECH кроме демонстрационных примеров также входит встроенная справочная система и документация в формате PDF, которые содержат необходимую информацию по отдельным компонентам программы и процессу нечеткого моделирования в среде fuzzyTECH.

Встроенная справочная система программы fuzzyTECH

Для просмотра справочной информации в программе fuzzyTECH следует воспользоваться соответствующим пунктом главного меню **Help** (Справка). Далее можно выбрать один из вариантов справочной системы.

Первый из них — так называемый *быстрый старт* (**Help>Quick Help**). В результате выполнения этой операции на экране появится дополнительное окно с программой просмотра информации по изучению одного из демонстрационных

примеров системы нечеткого вывода в форме стандартной справки Microsoft (рис. 18.7). При этом все варианты справочной информации организованы в форме гипертекста с возможностью поиска раздела справки по ключевому слову.

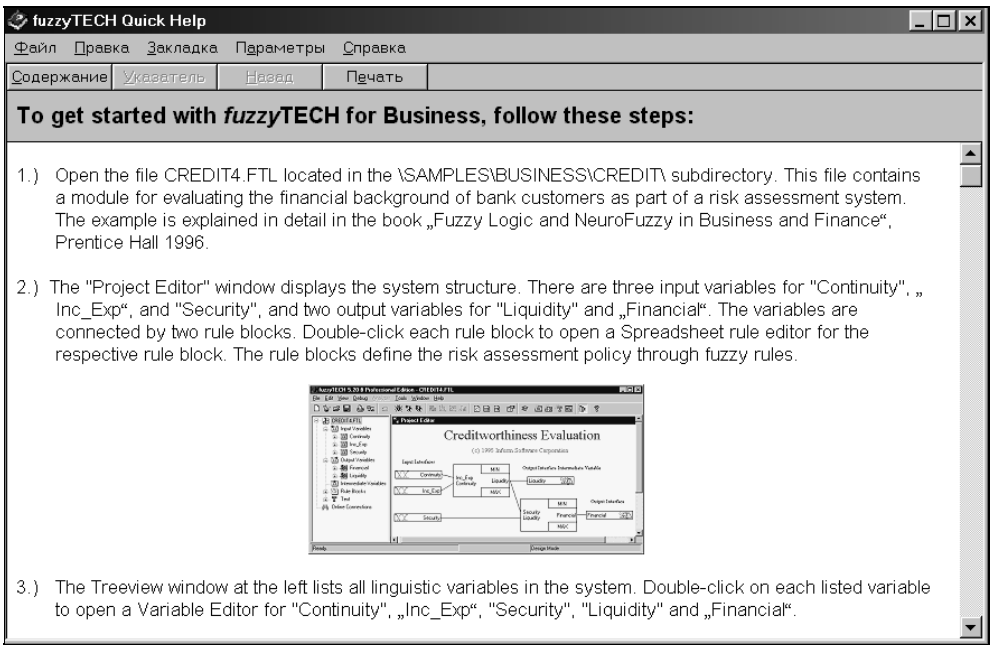


Рис. 18.7. Окно справочной системы fuzzyTECH с описанием одного из демонстрационных примеров

Второй вариант отображения справочной информации, предлагаемый программой fuzzyTECH, это справка по ее отдельным компонентам (**Help>fuzzyTECH Help** или F1). В результате выполнения этой операции на экране появится дополнительное окно с возможностью выбора информации по тому или иному компоненту программы fuzzyTECH. В этом случае информация также организована в форме стандартной справки Microsoft (рис. 18.8).

Третий вариант отображения справочной информации, предлагаемый программой fuzzyTECH, это также справка по ее отдельным компонентам, начиная с редактора проекта (**Help>Help on topic** или <Ctrl>+<F1>). В результате выполнения этой операции на экране также появится дополнительное окно со справочной информацией по редактору проектов (рис. 18.9).

Наконец, можно воспользоваться одним из разделов справочной системы, доступной через соответствующее меню в программной группе fuzzyTECH: **Пуск>Программы>fuzzyTECH 5.5>Help**. Хотя эти разделы повторяют справочную информацию основного меню программы, их можно просматривать независимо от программы fuzzyTECH без ее запуска.

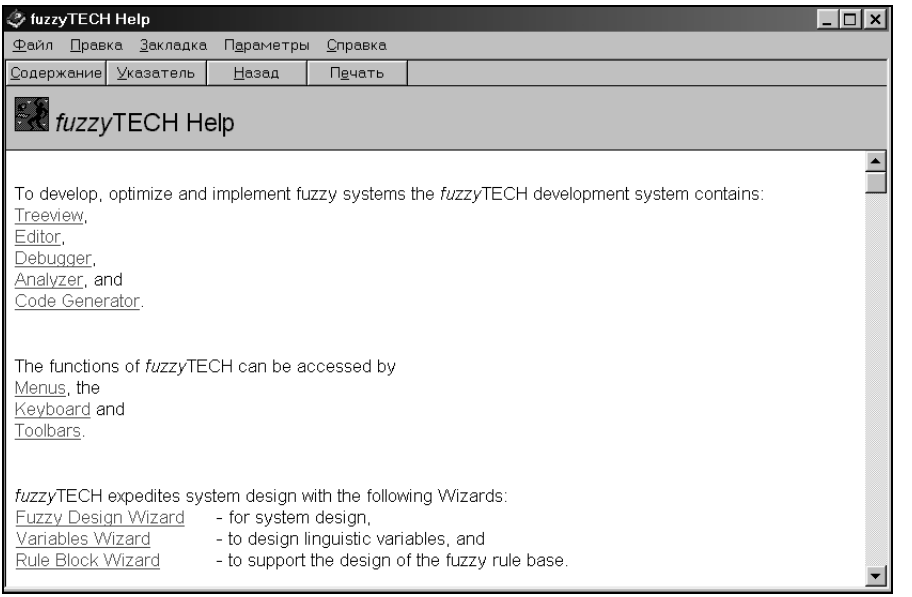


Рис. 18.8. Окно основной справочной системы fuzzyTECH

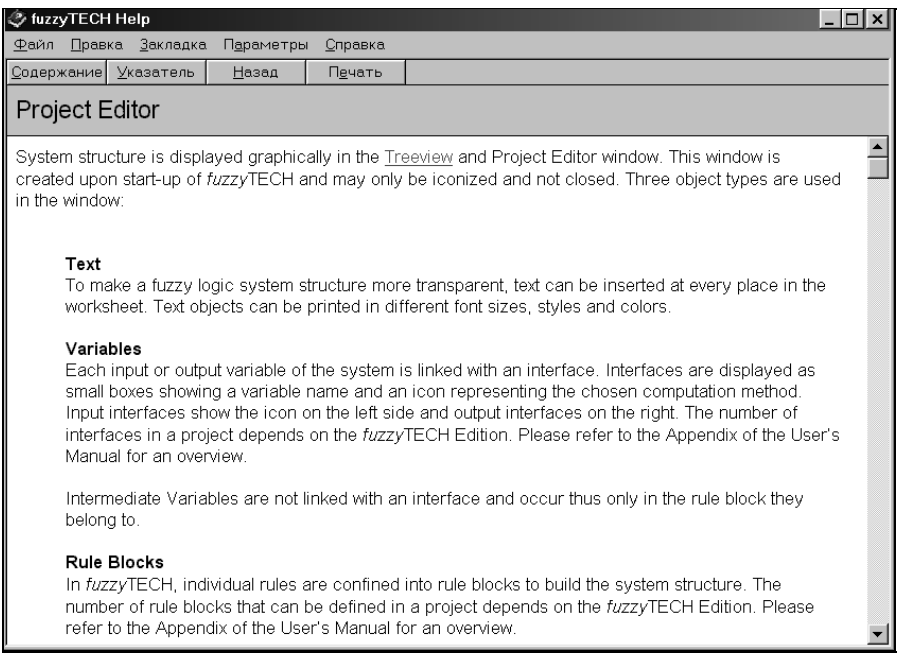


Рис. 18.9. Окно справочной системы fuzzyTECH с описанием возможностей редактора проектов

18.3. Назначение операций главного меню и панели инструментов программы fuzzyTECH

Программа fuzzyTECH имеет развитую систему операций главного меню с несколькими уровнями вложенности отдельных операций. Некоторые из наиболее часто используемых операций главного меню представлены в форме соответствующих кнопок на панели инструментов. Поэтому прежде чем приступить к изучению особенностей процесса нечеткого моделирования в среде fuzzyTECH, рассмотрим операции главного меню и назначение отдельных кнопок панели инструментов. После этого остановимся на некоторых особенностях нечетких проектов в среде fuzzyTECH.

Назначение операций главного меню

Окно программы fuzzyTECH имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с проектом, загружать и сохранять информацию во внешних файлах, изменять внешний вид элементов графического интерфейса, вызывать справочную информацию и т. д. Рассмотрим назначение отдельных операций главного меню программы fuzzyTECH.

❒ Пункт **File** (Файл) главного меню содержит следующие операции (рис. 18.10):

- **New** — вызывает диалоговое окно быстрого создания прототипа нового проекта (Generate Project), которое позволяет задать входные и выходные лингвистические переменные, их термы и блоки правил системы нечеткого вывода. При этом новый проект по умолчанию имеет имя Untitled;
- **Fuzzy Design Wizard...** — вызывает мастер нечеткого проекта (Fuzzy Design Wizard), который позволяет более детально задать не только структуру, но и характеристики прототипа нового проекта. Более подробно мастер нечеткого проекта рассматривается в *главе 19*;
- **Open...** — вызывает стандартное диалоговое окно открытия внешнего файла с диска. Если загружается файл проекта, то он должен иметь расширение ftl или ftr. При этом могут быть использованы следующие фильтры: *.FTL — формат файлов проектов в программе fuzzyTECH версии 3.*x* и более поздних, *.CFG — формат файлов конфигурации проектов, *.FTP — формат файлов проектов версии 2.*x* и более ранних, *.REV — формат файлов системы контроля версий (Revision Control System);
- **Close** — закрывает редактируемый проект системы нечеткого вывода. Следует отметить, что одновременно в программе fuzzyTECH может быть открыт для редактирования только один проект;

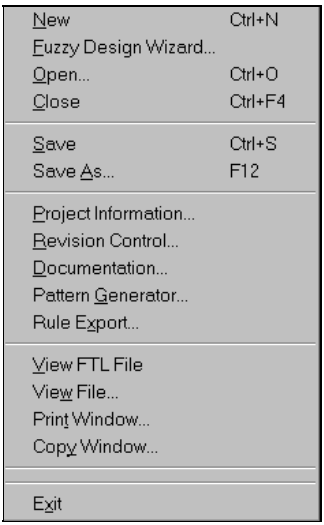


Рис. 18.10. Операции пункта **File** главного меню

- **Save** — позволяет сохранить редактируемый проект во внешнем файле на диске под текущим именем и с расширением ftl. Следует отметить, что в демонстрационной версии программы эта операция недоступна;
- **Save As...** — позволяет сохранить редактируемый проект во внешнем файле на диске и изменить при необходимости его имя. Следует также напомнить, что в демонстрационной версии программы эта операция недоступна;
- **Project Information...** — вызывает окно с информацией о редактируемом проекте, которое позволяет редактировать имя и автора проекта, а также комментарии к проекту;
- **Revision Control...** — вызывает окно настройки системы контроля версий программы fuzzyTECH, которое позволяет загрузить, сохранить или удалить файлы проектов согласно их текущему статусу. В рамках системы контроля версий файлы проектов сохраняются как бинарные файлы и могут быть защищены паролем. Система контроля версий позволяет сохранять все важные изменения, внесенные разработчиком в проект, включая дату и имя автора соответствующих изменений;
- **Documentation...** — вызывает диалоговое окно задания свойств для автоматической генерации документации по разработанному проекту системы нечеткого вывода в формате RTF на английском языке. При генерации документации она сохраняется во внешнем файле с расширением rtf. Имя файла документации может быть задано пользователем либо оставлено без изменения предложенное программой (по умолчанию — имя проекта). После генерации документации ее можно просмотреть и отредактировать в текстовом процессоре MS Word;

Примечание

При генерации документации по проекту вся информация о разрабатываемой системе нечеткого вывода по умолчанию группируется в следующие разделы: **Общая информация** (General Information), **Описание проекта** (Project Description), **Структура системы** (System Structure), **Лингвистические переменные** (Linguistic Variables), **Блоки правил** (Rule Blocks), **Настройки проекта** (Settings).

- **Pattern Generator… —** вызывает диалоговое окно генерации паттерна или шаблона проекта (Pattern Generator), которое позволяет автоматически сгенерировать некоторый шаблон входных данных разрабатываемой системы нечеткого вывода для анализа производительности соответствующей нечеткой модели. Сгенерированный шаблон сохраняется во внешнем файле с расширением csv;
- **Rule Export… —** вызывает диалоговое окно экспорта правил нечетких продукций (в демонстрационной версии программы эта операция недоступна);
- **View FTL File** — открывает файл текущего проекта в текстовом редакторе. По умолчанию файл с расширением ftl открывается в Блокноте (MS Notepad);
- **View File…** — вызывает стандартное диалоговое окно открытия файла проекта в текстовом редакторе. По умолчанию файлы с расширением ftl открываются в Блокноте (MS Notepad);
- **Print Window…** — позволяет распечатать на принтере одно из графических окон, открытых в программе fuzzyTECH. В этом случае вызывается диалоговое окно выбора объекта для печати на подключенном к данному компьютеру принтеру;
- **Copy Window…** — позволяет скопировать в буфер обмена MS Windows одно из графических окон, открытых в программе fuzzyTECH. В этом случае вызывается диалоговое окно выбора объекта для копирования в буфер обмена;
- секция с именами последних файлов проектов, с которыми осуществлялась работа в программе fuzzyTECH. Количество запоминаемых файлов проектов устанавливается в настройках программы fuzzyTECH;
- **Exit** — закрывает программу fuzzyTECH.

❒ Пункт меню **Edit** (Редактирование) содержит следующие операции (*рис.18.11*):

- **Undo** — отменяет выполнение последнего действия. Последовательно разработчик проекта может отменить 16 последних внесенных изменений. Аналогичным образом отменить выполненные действия можно последовательно нажимая клавиши: <Alt>+<Backspace>. В режиме отладки эта операция недоступна;
- **New Variable…** — вызывает мастер новой лингвистической переменной (Linguistic Variables Wizard). Более подробно мастер лингвистической переменной рассматривается в *главе 19*;

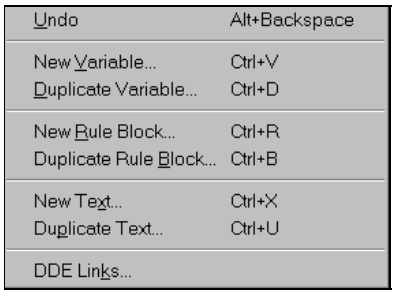


Рис. 18.11. Операции пункта **Edit** главного меню

- **Duplicate Variable...** — позволяет дублировать выделенную лингвистическую переменную проекта. При этом открывается мастер для новой лингвистической переменной, имя которой по умолчанию принимается как имя выделенной переменной с добавленным к нему справа символом подчеркивания "_";
- **New Rule Block...** — вызывает мастер нового блока правил (Rule Block Wizard). Более подробно мастер блока правил рассматривается в *главе 19*;
- **Duplicate Rule Block...** — позволяет дублировать выделенный блок правил проекта. При этом открывается мастер нового блока правил, имя которого должно отличаться от имен существующих блоков правил в проекте и должно быть задано разработчиком;
- **New Text...** — вызывает окно задания блока текста в редакторе проекта. Это окно позволяет не только ввести собственно текст, но и выбрать свойства шрифта;
- **Duplicate Text...** — позволяет дублировать выделенный блок текста в разрабатываемом проекте. При этом открывается окно свойств текста, в котором можно изменить не только текст, но и свойства шрифта;
- **DDE Links...** — вызывает диалоговое окно настройки связей DDE для конфигурации программы fuzzyTECH в качестве клиента.

❒ Пункт меню **View** (Вид) позволяет отображать на экране различные компоненты программы и разрабатываемого проекта и содержит следующие операции (рис. 18.12):

- **Toolbars** — позволяет настроить внешний вид графического интерфейса отдельных компонентов программы fuzzyTECH и содержит дополнительные подпункты: **Show all** — отображает панели инструментов всех графических окон программы fuzzyTECH; **Show none** — скрывает панели инструментов всех графических окон программы fuzzyTECH; **Main Window** — отображает/скрывает панель инструментов главного окна программы fuzzyTECH; **Variable Editor** — отображает/скрывает панель инструментов редактора лингвистических переменных; **Spreadsheet Rule Editor** — отображает/скрывает панель инструментов табличного редактора правил;

Matrix Rule Editor — отображает/скрывает панель инструментов матричного редактора правил; **Watch Window** — отображает/скрывает панель инструментов окна интерактивного режима отладки вывода; **Transfer Plot** — отображает/скрывает панель инструментов окна просмотра графиков на плоскости; **3D Plot** — отображает/скрывает панель инструментов окна просмотра трехмерных графиков; **Time Plot** — отображает/скрывает панель инструментов окна просмотра временных графиков; **Rule Analyzer** — отображает/скрывает панель инструментов анализатора правил;

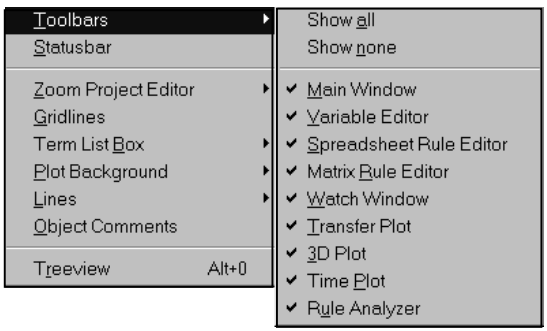


Рис. 18.12. Операции пункта **View** главного меню

- **Statusbar** — отображает/скрывает строку состояния программы fuzzyTECH;
- **Zoom Project Editor** — позволяет выбрать масштаб изображения системы нечеткого вывода в окне редактора проекта. Содержит дополнительные подпункты: **100%**, **75%** и **50%**, выбор которых изменяет размеры отображаемых компонентов проекта соответствующим образом;
- **Gridlines** — делает видимым/невидимым координатную сетку в редакторе лингвистических переменных и в окне просмотра временных графиков. Разрешение координатной сетки устанавливается автоматически;
- **Term List Box** — делает видимым/невидимым окно термов в редакторе лингвистической переменной. Более подробно редактор лингвистической переменной рассматривается в *главе 19*;
- **Plot Background** — устанавливает белый/черный фон в окнах просмотра трехмерных и временных графиков;
- **Lines** — позволяет изменять толщину кривых в редакторе лингвистической переменной при отображении функций принадлежности отдельных термов и в окне просмотра временных графиков;
- **Object Comments** — выбор этой опции позволяет отображать в форме всплывающей подсказки комментарий объекта при задержке на его изображении курсора мыши в редакторе проекта;
- **Treeview** — делает видимым/невидимым окно просмотра структуры проекта.

- Пункт меню **Debug** (Отладка) обеспечивает доступ к различным инструментам режима отладки разрабатываемого проекта и содержит следующие операции (рис. 18.13):
  - **Interactive** — включает режим интерактивной отладки проекта и вызывает соответствующее окно интерактивного режима отладки. Более подробно интерактивный режим отладки проекта рассматривается в *главе 19*;
  - **Serial Link** — включает режим отладки проекта совместно с другими приложениями с использованием некоторого последовательного интерфейса персонального компьютера. При использовании данного режима предполагается, что программа fuzzyTECH функционирует на компьютере 1, а в качестве компьютера 2 может использоваться любое аппаратное устройство или программа. При этом соединение компьютеров 1 и 2 осуществляется с помощью последовательного интерфейса;

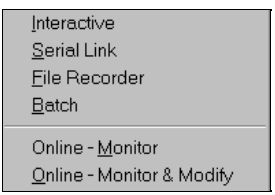


Рис. 18.13. Операции пункта **Debug** главного меню

  - **File Recorder** — включает режим отладки проекта с записью файлов. При использовании данного режима предполагается, что имеются предварительно созданные внешние файлы с некоторой выборкой данных или файл сгенерированного шаблона (паттерна);
  - **Batch** — включает режим отладки проекта с использованием внешнего текстового файла. При использовании данного режима предполагается, что имеется предварительно созданный текстовый файл в кодировке ASCII со значениями входных лингвистических переменных. После выполнения процедуры нечеткого вывода программой fuzzyTECH результаты записываются во внешний файл, который можно просмотреть и проанализировать в текстовом редакторе;
  - **Online-Monitor** — включает режим отладки проекта с использованием некоторой дополнительной программы реального времени (fuzzy runtime system). Применяется для реализации систем нечеткого вывода при управлении внешними объектами в реальном времени. В этом режиме используемая дополнительная программа реального времени защищается от возможных изменений;
  - **Online-Monitor & Modify** — включает режим отладки проекта с использованием некоторой дополнительной программы реального времени (fuzzy runtime system). В этом режиме разработчик имеет возможность вносить изменения в используемую дополнительную программу реального времени.

- Пункт меню **Analyzer** (Анализатор) позволяет отображать на экране различные инструменты анализа результатов нечеткого вывода для текущего проекта, становится доступным только при включении режима отладки проекта и содержит следующие операции (рис. 18.14):
  - **New Transfer Plot...** — создает новое окно просмотра графиков поверхности вывода на плоскости;
  - **New 3D Plot...** — создает новое окно просмотра трехмерных графиков поверхности вывода;
  - **New Time Plot...** — создает новое окно просмотра временных графиков системы нечеткого вывода;
  - **New Rule Analyzer...** — создает новое окно анализатора правил системы нечеткого вывода для первой выходной лингвистической переменной;
  - **Statistics** — включает/отключает в табличный редактор правил дополнительный столбец с информацией о количестве циклов активизации каждого из правил;
  - **Trace...** — включает режим записи значений входных лингвистических переменных при использовании внешних программ реального времени в некотором внешнем файле с расширением trc.

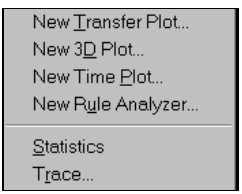


Рис. 18.14. Операции пункта **Analyzer** главного меню

- Пункт меню **Tools** (Средства) обеспечивает доступ к различным дополнительным инструментам программы fuzzyTECH, к ее настройкам, свойствам проекта и содержит следующие операции (рис. 18.15):
  - **Compile to** — позволяет сгенерировать для текущего проекта программный код на одном из языков программирования и для рассматриваемой версии программы fuzzyTECH содержит дополнительные подпункты: **C...** — вызывает окно задания имени файла с текстом исходного кода программы на языке C; **Java...** — вызывает окно задания имени файла с текстом исходного кода программы на языке Java; **M...** — вызывает окно задания имени m-файла с текстом исходного кода проекта для системы MATLAB; **FTR...** — вызывает окно задания имени бинарного файла формата FTR для текущего проекта; **FTR+Wrapper Class for Visual C++/MFC...** — кроме файла проекта в формате FTR позволяет сгенерировать описание класса для среды MS Visual C++ в форме приложения MFC; **FTR+Wrapper Class for**

Visual Basic... — кроме файла проекта в формате FTR позволяет сгенерировать модуль класса для среды MS Visual Basic в форме файла с расширением cls; **FTR+Wrapper Class for VBA...** — кроме файла проекта в формате FTR позволяет сгенерировать модуль класса для языка программирования MS Visual Basic for Applications в форме файла с расширением cls. Следует напомнить, что в демонстрационной версии программы эта операция главного меню недоступна;

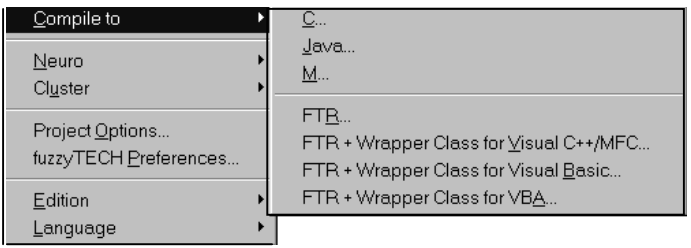


Рис. 18.15. Операции пункта **Tools** главного меню

- **Neuro** — позволяет использовать дополнительный модуль нейро-нечеткого вывода для улучшения и оптимизации характеристик разрабатываемого проекта и содержит два подпункта: **Configuration** — вызывает диалоговое окно задания свойств конфигурации модуля нейро-нечеткого вывода; **Learning** — вызывает диалоговое окно открытия файла с обучающей выборкой с целью последующего использования этого файла для обучения системы нейро-нечеткого вывода;
- **Cluster** — позволяет использовать дополнительный модуль кластеризации для оптимизации процесса обучения системы нейро-нечеткого вывода разрабатываемого проекта и содержит два подпункта: **IsodataCluster** — вызывает диалоговое окно открытия файла с исходной выборкой данных для последующей кластеризации методом ISODATA; **FuzzyCluster** — вызывает диалоговое окно открытия файла с исходной выборкой данных для последующей кластеризации методом нечеткой кластеризации;
- **Project Options...** — вызывает диалоговое окно с тремя вкладками для задания свойств разрабатываемого проекта, которые сохраняются в соответствующем файле проекта формата FTL;
- **FuzzyTECH Preferences...** — вызывает диалоговое окно с шестью вкладками для настройки рабочего интерфейса программы fuzzyTECH в соответствии с индивидуальными пожеланиями разработчика, при этом все настройки сохраняются;
- **Edition** — позволяет запустить другие версии программы fuzzyTECH, установленные на данном компьютере;
- **Language** — позволяет изменить язык меню и диалоговых окон программы fuzzyTECH.

- Пункт меню **Window** (Окно) позволяет отображать на экране графические окна различных компонентов программы fuzzyTECH и содержит следующие операции (рис. 18.16):
  - **Open all Variable Editors of Variable Group** — позволяет открыть дополнительные графические окна редакторов для отдельных лингвистических переменных текущего проекта системы нечеткого вывода и содержит следующие подпункты: **All Variables** — открывает окна редакторов для всех лингвистических переменных проекта; **Input** — открывает окна редакторов только входных лингвистических переменных проекта; **Output** — открывает окна редакторов только выходных лингвистических переменных проекта; **Intermediates** — открывает окна редакторов только промежуточных лингвистических переменных проекта;

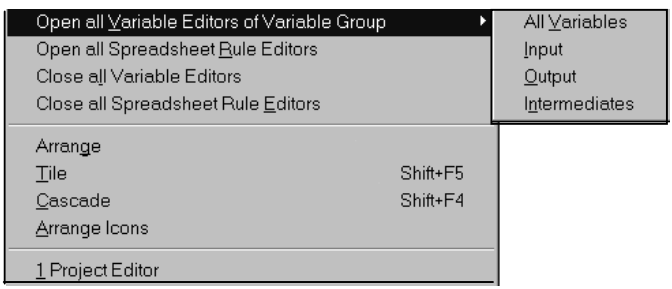


Рис. 18.16. Операции пункта **Window** главного меню

  - **Open all Spreadsheet Rule Editors** — позволяет открыть дополнительные графические окна табличных редакторов всех блоков правил разрабатываемого проекта;
  - **Close all Variable Editors** — позволяет закрыть все открытые ранее графические окна редакторов лингвистических переменных текущего проекта системы нечеткого вывода;
  - **Close all Spreadsheet Rule Editors** — позволяет закрыть все открытые ранее графические окна табличных редакторов блоков правил текущего проекта системы нечеткого вывода;
  - **Arrange** — позволяет отобразить в окне редактора проекта все открытые графические окна различных компонентов проекта, автоматически настраивая их геометрические размеры и выравнивая расположение;
  - **Tile** — позволяет отобразить в окне редактора проекта все открытые графические окна различных компонентов проекта, автоматически упорядочивая их по именам;
  - **Cascade** — позволяет отобразить в окне редактора проекта все открытые графические окна различных компонентов проекта в каскадной форме, автоматически выравнивая их геометрические размеры;

- **Arrange Icons** — позволяет упорядочить пиктограммы свернутых графических окон всех открытых компонентов текущего проекта в нижней части окна редактора проекта;
- секция с номерами и именами открытых графических окон в программе для быстрого переключения между ними.

❒ Пункт меню **Help** (Справка) содержит следующие операции (рис. 18.17):

- **Quick Help** — вызывает раздел быстрого старта справочной системы программы fuzzyTECH (рис. 18.7);
- **FuzzyTECH Help** — вызывает основной раздел справочной системы программы fuzzyTECH (рис. 18.8);
- **Help on topic** — вызывает раздел справочной системы программы fuzzyTECH, который соответствует текущему активному окну компонента разрабатываемого проекта, т. е. имеющему в текущий момент фокус редактирования;
- **Help on Neuro** — вызывает окно справочной системы с разделом справки о модуле нейро-нечеткого вывода программы fuzzyTECH;
- **Help on Cluster** — вызывает окно справочной системы с разделом справки о модуле кластеризации программы fuzzyTECH;
- **Fuzzy Primer** — вызывает окно справочной системы с разделом общей информации о нечеткой логике;
- **Shortkey Overview** — вызывает окно справочной системы с описанием клавиш быстрого доступа к различным компонентам программы fuzzyTECH;
- **FuzzyTECH on WWW** — вызывает браузер Интернета, установленный на компьютере разработчика по умолчанию, для доступа к дополнительной информации по программе fuzzyTECH, предоставляемой компанией INFORM GmbH;
- **About...** — отображает информацию о текущей рабочей версии программы fuzzyTECH и ее компонентах.

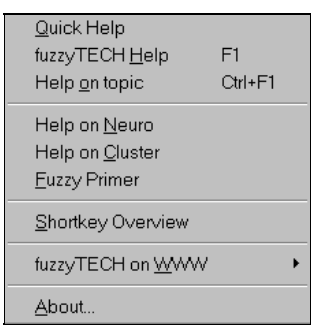


Рис. 18.17. Операции пункта **Help** главного меню

Назначение операций панели инструментов

Панель инструментов главного окна программы fuzzyTECH располагается ниже строки главного меню и содержит набор кнопок, которые дублируют наиболее часто выполняемые операции главного меню и обеспечивают быстрый доступ к выполнению соответствующих операций (табл. 18.3):

Таблица 18.3. *Назначение кнопок панели инструментов программы fuzzyTECH*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	New	Вызывает диалоговое окно генерации нового проекта (Generate Project)
	Design Wizard	Вызывает диалоговое окно мастера нечеткого проекта (Fuzzy Design Wizard)
	Open	Вызывает стандартное диалоговое окно открытия внешнего файла с расширениями ftl, cfg, ftp, rev с диска
	Save	Сохраняет текущий проект во внешнем файле с расширением ftl или cfg
	Documentation	Вызывает диалоговое окно задания свойств для автоматической генерации документации по разработанному проекту системы нечеткого вывода в формате RTF на английском языке
	Revision Control	Вызывает окно настройки системы контроля версий программы fuzzyTECH
	Undo	Отменяет выполнение последнего действия
	Interactive	Включает режим интерактивной отладки проекта и вызывает окно анализа результатов нечеткого вывода
	Monitor	Включает режим отладки проекта с использованием некоторой дополнительной программы реального времени без возможности внесения изменений в используемую дополнительную программу
	Monitor&Modify	Включает режим отладки проекта с использованием некоторой дополнительной программы реального времени с возможностью внесения изменений в используемую дополнительную программу

Таблица 18.3 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Transfer Plot	Создает новое окно просмотра графиков поверхности вывода на плоскости
	3D Plot	Создает новое окно просмотра трехмерных графиков поверхности вывода
	Time Plot	Создает новое окно просмотра временных графиков поверхности вывода
	Rule Analyzer	Открывает окно анализатора правил разрабатываемой системы нечеткого вывода
	Statistics	Включает/отключает в табличном редакторе правил дополнительный столбец с информацией о количестве циклов активизации каждого из правил
	Options	Вызывает диалоговое окно для задания свойств разрабатываемого проекта
	Neuro Learning	Вызывает диалоговое окно открытия файла с обучающей выборкой с целью его последующего применения для обучения системы нейронечеткого вывода
	Variable	Вызывает мастер новой лингвистической переменной (Linguistic Variables Wizard)
	Rule	Вызывает мастер нового блока правил (Rule Block Wizard)
	Text	Вызывает окно задания блока текста в редакторе проекта
	Properties	Вызывает диалоговое окно для индивидуальной настройки рабочего интерфейса программы fuzzyTECH
	Treeview	Отображает/скрывает окно просмотра структуры проекта (см. рис. 18.6)
	Help	Вызывает окно справочной системы программы fuzzyTECH

Указанные операции панели инструментов главного окна используются в дальнейшем при рассмотрении способов вызова различных графических инструментов программы fuzzyTECH.

18.4. Графические средства визуализации результатов нечеткого вывода в программе fuzzyTECH

Программа fuzzyTECH обладает встроенными средствами визуализации и графического представления результатов нечеткого вывода для разработанных проектов в форме кривых и поверхностей на плоскости и в 3-мерном пространстве. При этом могут быть использованы различные способы и стили их цветового представления, что обеспечивает наглядность получаемых изображений.

В процессе разработки и последующего анализа систем нечеткого вывода в интерактивном режиме могут быть использованы следующие специальные графические средства визуализации результатов нечеткого вывода, входящие в состав программы fuzzyTECH:

- графическое окно просмотра поверхности нечеткого вывода на плоскости;
- графическое окно просмотра трехмерной поверхности нечеткого вывода;
- графическое окно просмотра временных графиков значений лингвистических переменных.

Ниже рассматриваются особенности представления поверхностей нечеткого вывода с помощью указанных графических окон.

Графическое окно просмотра поверхности нечеткого вывода на плоскости

Графическое окно просмотра поверхности нечеткого вывода на плоскости (Transfer Plot Window) может быть открыто только в интерактивном режиме отладки проекта с помощью операции главного меню **Analyzer>New Transfer Plot…** или с помощью соответствующей кнопки панели инструментов главного окна (см. табл. 18.3).

После выполнения данной операции возникнет графическое окно с изображением поверхности нечеткого вывода на плоскости (рис. 18.18).

Графическое окно просмотра поверхности нечеткого вывода на плоскости ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов и вложенные списки с именами лингвистических переменных проекта. Отобразить или скрыть эти списки можно с помощью соответствующей кнопки панели инструментов (табл. 18.4).

Основную часть графического окна занимают секции визуализации поверхности по каждой из входных лингвистических переменных и секция визуализации собственно поверхности нечеткого вывода на плоскости. В первых двух секциях, расположенных выше и правее секции визуализации поверхности нечеткого вывода, изображаются вертикальные срезы или сечения поверхности вывода по текущему значению первой и, соответственно, второй входной лингвистической

переменной. Отобразить или скрыть эти секции можно с помощью соответствующей кнопки панели инструментов (см. табл. 18.4).

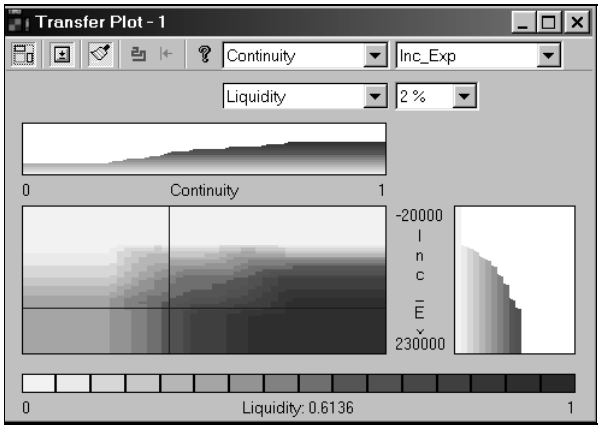


Рис. 18.18. Графическое окно просмотра поверхности нечеткого вывода на плоскости

Поверхность нечеткого вывода графически представляет функциональную зависимость выходной лингвистической переменной от двух входных лингвистических переменных в форме изменения оттенка и насыщенности некоторого цвета (по умолчанию — диапазона цветовой гаммы от желтого до коричневого).

Панель инструментов окна просмотра поверхности нечеткого вывода на плоскости содержит набор кнопок, которые позволяют изменять внешний вид данного графического окна и выполнять некоторые дополнительные операции. Назначение отдельных кнопок панели инструментов приводится в табл. 18.4.

Таблица 18.4. *Назначение кнопок панели инструментов окна просмотра поверхности нечеткого вывода на плоскости*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Cross Section View	Отображает/скрывает секции визуализации значений входных лингвистических переменных
	Variable Box	Отображает/скрывает секции задания имен входных и выходных лингвистических переменных
	Repaint	Перерисовывает график поверхности нечеткого вывода после внесенных изменений в проект
	Trace	Включает/выключает режим отображения значений входных лингвистических переменных при их записи с использованием внешних программ реального времени

Таблица 18.4 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Reset Trace	Обновляет отображение значений входных лингвистических переменных при их записи с использованием внешних программ реального времени
	Help	Вызывает раздел справочной системы, посвященный окну просмотра поверхности нечеткого вывода на плоскости

Выбор двух входных лингвистических переменных для построения соответствующей поверхности нечеткого вывода осуществляется с помощью раскрывающихся списков, расположенных справа от панели инструментов. Третий раскрывающийся список предназначен для выбора выходной лингвистической переменной проекта в том случае, когда в нечетком проекте используется несколько выходных переменных. Справа от списка выходных переменных находится небольшое окно со списком значений коэффициента аппроксимации **(Zoom Factor)** графика поверхности. Чем меньше значение данного коэффициента, тем более точно изображается поверхность нечеткого вывода.

Собственно поверхность нечеткого вывода изображается аналогично изображению рельефа земной поверхности на географической карте. При этом меньшим значениям выходной лингвистической переменной соответствуют более светлые оттенки цвета, а большим значениям — более темные оттенки цвета. Весь диапазон цветовых оттенков значений выходной лингвистической переменной изображен в нижней части графического окна. Здесь указывается также количественное значение выходной переменной как результат нечеткого вывода для текущих значений входных переменных.

Текущие значения входных лингвистических переменных визуализируются с помощью вертикальной (для первой) и горизонтальной (для второй) линий на графике поверхности. Щелкнув и удерживая левую кнопку мыши на этих линиях, посредством последующего перемещения курсора мыши в том или ином направлении можно изменить текущие значения входных лингвистических переменных. К аналогичному результату приводит однократный щелчок левой кнопки мыши в произвольной точке графика поверхности.

Анализ поверхности нечеткого вывода для разработанного проекта заключается в контроле или проверке адекватности влияния изменения текущих значений входных лингвистических переменных на выходную лингвистическую переменную. Для этого могут быть использованы различные значения этих переменных из всего диапазона возможных значений. Для более полного и всестороннего анализа нечеткого проекта следует воспользоваться дополнительными графическими окнами просмотра трехмерных поверхностей и временных графиков нечеткого вывода, которые рассматриваются ниже.

Закрыть активное графическое окно просмотра поверхности нечеткого вывода на плоскости можно с помощью соответствующей кнопки управления окном. При выходе из режима интерактивной отладки проекта все открытые ранее графические окна просмотра поверхности нечеткого вывода автоматически закрываются.

Графическое окно просмотра трехмерной поверхности нечеткого вывода

Графическое окно просмотра трехмерной поверхности нечеткого вывода **(3D Plot Window)** может быть открыто только в интерактивном режиме отладки проекта с помощью операции главного меню **Analyzer>New 3D Plot…** или с помощью соответствующей кнопки панели инструментов главного окна программы fuzzyTECH (табл. 18.3).

После выполнения данной операции появится графическое окно с изображением поверхности нечеткого вывода в пространстве трех измерений (рис. 18.19).

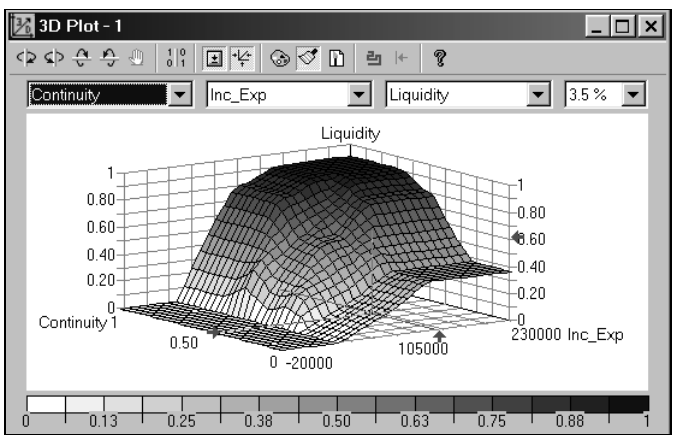


Рис. 18.19. Графическое окно просмотра трехмерной поверхности нечеткого вывода

Графическое окно просмотра трехмерной поверхности нечеткого вывода ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов и вложенные списки с именами лингвистических переменных проекта. Отобразить или скрыть эти списки можно с помощью соответствующей кнопки панели инструментов (табл. 18.5).

Основную часть графического окна занимает секция визуализации трехмерной поверхности нечеткого вывода. При этом поверхность нечеткого вывода графически представляется в трехмерном пространстве, в котором первым двум координатным осям соответствуют входные лингвистические переменные, а вертикальная ось координат соответствует выходной лингвистической переменной.

Панель инструментов окна просмотра трехмерной поверхности нечеткого вывода содержит набор кнопок, которые позволяют изменять внешний вид данного графического окна и выполнять некоторые дополнительные операции. Назначение отдельных кнопок панели инструментов приводится в табл. 18.5.

Таблица 18.5. *Назначение кнопок панели инструментов окна просмотра трехмерной поверхности нечеткого вывода*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Left	Поворачивает график трехмерной поверхности вывода на 10° влево
	Right	Поворачивает график трехмерной поверхности вывода на 10° вправо
	Up	Поворачивает график трехмерной поверхности вывода на 10° вверх
	Down	Поворачивает график трехмерной поверхности вывода на 10° вниз
	Stop	Останавливает непрерывное вращение графика трехмерной поверхности вывода, инициированное одним из способов поворота
	Vertical	Изменяет направление осей входных лингвистических переменных на противоположное
	Variable Box	Отображает/скрывает секции задания имен входных и выходных лингвистических переменных
	Cricp Values	Отображает/скрывает текущие значения лингвистических переменных, выделяемые с помощью красных стрелок
	Color Palette	Изменяет цвет графика трехмерной поверхности вывода
	Repaint	Включает режим отладки проекта с использованием некоторой дополнительной программы реального времени с возможностью внесения изменений в используемую дополнительную программу
	Hide Plot Drawing	Включает/выключает режим последовательного вычерчивания фрагментов графика трехмерной поверхности вывода
	Trace	Включает/выключает режим отображения значений входных лингвистических переменных при их записи с использованием внешних программ реального времени

Таблица 18.5 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Reset Trace	Обновляет отображение значений входных лингвистических переменных при их записи с использованием внешних программ реального времени
	Help	Вызывает раздел справочной системы, посвященный окну просмотра трехмерной поверхности нечеткого вывода

Выбор двух входных лингвистических переменных и выходной лингвистической переменной проекта для построения соответствующей трехмерной поверхности нечеткого вывода осуществляется с помощью вложенных списков, расположенных ниже панели инструментов. Справа от этих списков находится небольшое окно со списком значений коэффициента аппроксимации **(Zoom Factor)** трехмерного графика поверхности. Чем меньше значение данного коэффициента, тем более точно изображается поверхность нечеткого вывода.

Трехмерная поверхность нечеткого вывода изображается аналогично изображению графиков поверхностей в декартовой системе координат. Для более наглядного восприятия формы поверхности используются оттенки некоторого цвета, при этом меньшим значениям выходной лингвистической переменной соответствуют более светлые оттенки цвета, а большим значениям — более темные оттенки цвета. Изменить базовый цвет изображения поверхности можно с помощью соответствующей кнопки панели инструментов (см. табл. 18.5).

Текущие значения входных лингвистических переменных визуализируются с помощью красных стрелок на координатных осях графика поверхности в случае, если данный режим включен с помощью кнопки панели инструментов (см. табл. 18.5). Щелкнув и удерживая левую кнопку мыши на одной из этих линий, посредством последующего перемещения курсора мыши в том или ином направлении можно изменить текущее значение соответствующей входной лингвистической переменной.

Анализ поверхности нечеткого вывода для разработанного проекта заключается в визуальной проверке адекватности формы поверхности нечеткого вывода. Для этого можно использовать различные углы просмотра трехмерной поверхности и изменение направления координатных осей.

Закрыть активное графическое окно просмотра трехмерной поверхности нечеткого вывода можно с помощью соответствующей кнопки управления окном. При выходе из режима интерактивной отладки проекта все открытые ранее графические окна просмотра поверхности нечеткого вывода автоматически закрываются.

Графическое окно просмотра временных графиков значений лингвистических переменных

Графическое окно просмотра временных графиков значений лингвистических переменных нечеткого вывода (**Time Plot Window**) может быть открыто только в интерактивном режиме отладки проекта с помощью операции главного меню **Analyzer>New Time Plot...** или с помощью соответствующей кнопки панели инструментов главного окна программы fuzzyTECH (см. табл. 18.3).

После выполнения данной операции возникнет пустое графическое окно, которое после выполнения определенных настроек и выполнения серии изменений значений входных лингвистических переменных будет содержать временные графики соответствующих значений этих лингвистических переменных (рис. 18.20).

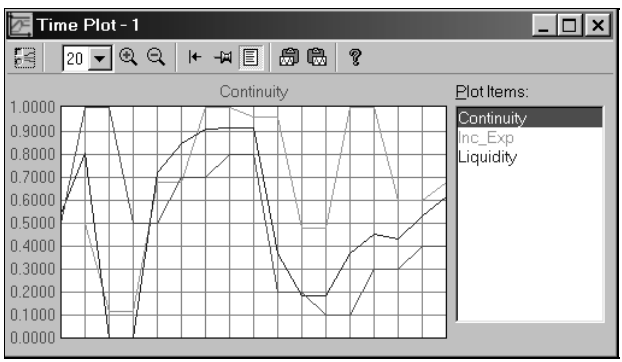


Рис. 18.20. Графическое окно просмотра временных графиков значений лингвистических переменных системы нечеткого вывода

Графическое окно просмотра временных графиков значений лингвистических переменных ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов. Основную часть графического окна занимает секция визуализации временных графиков значений лингвистических переменных, справа от которой находится секция имен лингвистических переменных. При этом временные графики значений лингвистических переменных графически представляются в двумерной декартовой системе координат, в которой горизонтальной координате соответствует ось времени, а вертикальной — значения отдельных лингвистических переменных.

Панель инструментов окна просмотра временных графиков значений лингвистических переменных содержит набор кнопок, которые позволяют изменять внешний вид данного графического окна и выполнять некоторые дополнительные операции. Назначение отдельных кнопок панели инструментов приводится в табл. 18.6.

Таблица 18.6. *Назначение кнопок панели инструментов окна просмотра временных графиков значений лингвистических переменных*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Configuration	Открывает диалоговое окно задания свойств конфигурации окна временных графиков, которое позволяет выбрать лингвистические переменные для построения соответствующих графиков
	Zoom Factor	Вложенный список из фиксированного набора значений для выбора коэффициента масштаба изображения графиков по горизонтальной оси времени
	Zoom In	Увеличивает значение коэффициента масштаба изображения графиков по горизонтальной оси на одно значение из вложенного списка времени
	Zoom Out	Уменьшает значение коэффициента масштаба изображения графиков по горизонтальной оси времени на одно значение из вложенного списка
	Reset	Очищает секцию изображения временных графиков
	Freeze	Останавливает/продолжает изображение временных графиков
	List Box	Отображает/скрывает секцию имен входных и выходных лингвистических переменных
	Add Input Variables	Добавляет все входные лингвистические переменные для построения соответствующих временных графиков
	Add Output Variables	Добавляет все выходные лингвистические переменные для построения соответствующих временных графиков
	Help	Вызывает раздел справочной системы, посвященный окну просмотра временных графиков нечеткого вывода

Выбор входных и выходных лингвистических переменных для построения соответствующих временных графиков нечеткого вывода можно выполнить либо при задании конфигурации этого графического окна, либо с помощью кнопок панели инструментов. На панели инструментов также имеется небольшое поле со списком значений коэффициента масштаба **(Zoom Factor)** для представления оси

времени. Чем меньше значение данного коэффициента, тем более мелкой представляется координатная сетка по горизонтальной оси времени.

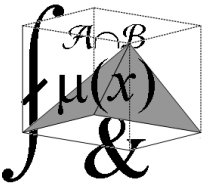
Временные графики нечеткого вывода служат для изображения всех последовательных изменений текущих значений входных и выходных лингвистических переменных в обычной декартовой системе координат. Для более наглядного восприятия графиков каждой из лингвистических переменных используются различные цвета. Изменить цвет изображения некоторого временного графика можно с помощью изменения цвета представления соответствующей лингвистической переменной, что удобно выполнять с помощью операции **Color...** (Цвет) контекстного меню.

Анализ временных графиков нечеткого вывода для разработанного проекта заключается в визуальной проверке адекватности последовательных результатов нечеткого вывода. Для этого можно использовать различный цвет фона (белый или черный) временных графиков.

Закрыть активное графическое окно просмотра временных графиков нечеткого вывода можно с помощью соответствующей кнопки управления окном. При выходе из режима интерактивной отладки проекта все открытые ранее графические окна просмотра временных графиков автоматически закрываются.

Примеры изображений, построенные с помощью различных средств графической визуализации результатов нечеткого вывода программы fuzzyTECH для некоторых задач управления и принятия решений, приводятся в *главе 20*.

Глава 19



Процесс нечеткого моделирования в среде fuzzyTECH

Процесс нечеткого моделирования в среде fuzzyTECH осуществляется в интерактивном режиме с помощью специальных графических средств, предназначенных для редактирования и визуализации компонентов систем нечеткого вывода. В дополнение к ним также могут быть использованы специальные средства или мастера вновь разрабатываемых или модифицируемых систем нечеткого вывода.

Ниже рассматриваются особенности разработки и редактирования систем нечеткого вывода с использованием соответствующих графических средств и даются рекомендации по применению мастеров. Примеры разработки нечетких моделей с помощью графических средств в среде fuzzyTECH для некоторых задач управления и принятия решений приводятся в *главе 20*.

19.1. Основные средства редактирования и анализа систем нечеткого вывода в fuzzyTECH

Для редактирования и последующего анализа систем нечеткого вывода в интерактивном режиме могут быть использованы следующие графические средства, входящие в состав программы fuzzyTECH.

- Графический редактор проекта системы нечеткого вывода (Project Editor).
- Графический редактор лингвистической переменной и функций принадлежности ее термов (Variable Editor).
- Графические редакторы правил системы нечеткого вывода (Spreadsheet Rule Editor, Matrix Rule Editor).
- Графические средства анализа результатов нечеткого вывода (Watch Window, Rule Analyzer).
- Графические средства просмотра поверхности системы нечеткого вывода (Transfer Plot Window, 3D Plot Window, Time Plot Window).

Вызов этих средств (кроме редактора проекта, который всегда открыт) возможен через операции главного меню, а также с помощью кнопок панели инструментов (см. табл. 18.3) или двойного щелчка мыши на графическом изображении отдельных компонентов системы нечеткого вывода в графическом окне редактора проекта.

Редактор проекта и графические средства просмотра поверхности системы нечеткого вывода были рассмотрены ранее в *главе 18*. В настоящей главе излагаются особенности графических средств, которые могут быть использованы для разработки и исследования широкого класса систем нечеткого вывода в среде fuzzyTECH.

Графический редактор лингвистической переменной и функций принадлежности их термов

Редактор лингвистической переменной и функций принадлежности ее термов (далее сокращенно — *редактор лингвистической переменной*), как следует из его названия, предназначен для спецификации термов отдельной лингвистической переменной проекта и редактирования ее функций принадлежности в графическом режиме.

Графическое окно редактора лингвистических переменных может быть открыто двойным щелчком мыши на изображении прямоугольника соответствующей лингвистической переменной в окне редактора проекта или двойным щелчком мыши на имени соответствующей лингвистической переменной в окне просмотра структуры проекта. Открыть окна редакторов для всех лингвистических переменных проекта можно с помощью соответствующей операции главного меню: **Window>Open all Variable Editors of variable Group>All Variables**.

Результат вызова редактора лингвистической переменной для некоторой переменной с именем Variable1, которая имеет три терма с именами: low, medium и high, изображен на рис. 19.1.

Графическое окно редактора лингвистической переменной ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов и две секции: секцию имен термов, расположенную слева, и секцию графиков функций принадлежности термов, расположенную справа. Внешний вид этих секций зависит от общего количества термов у редактируемой лингвистической переменной.

В секции графиков функций принадлежности изображаются две координатные оси, первая из них (горизонтальная) представляет весь диапазон значений редактируемой лингвистической переменной, а вторая (вертикальная) — интервал значений функции принадлежности, т. е. интервал [0, 1]. Ниже горизонтальной оси указываются единицы измерения значений редактируемой лингвистической переменной (по умолчанию — Units).

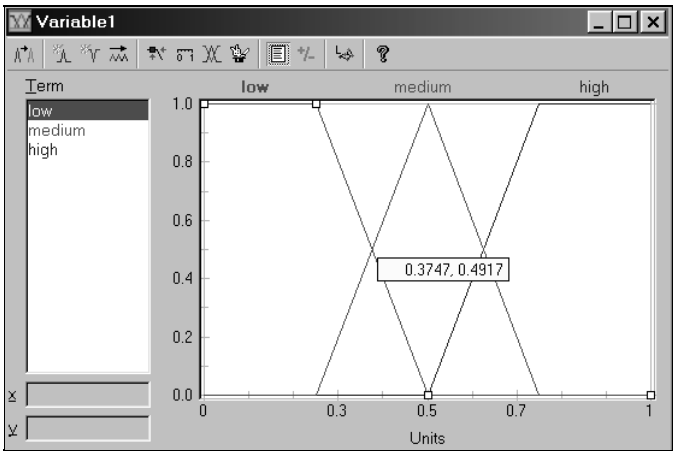


Рис. 19.1. Графическое окно редактора лингвистической переменной с именем Variable1

Результат вызова редактора лингвистической переменной для некоторой переменной с именем Variable 2, которая имеет 5 термов с именами: very_low, low, medium, high и very_high, изображен на рис. 19.2.

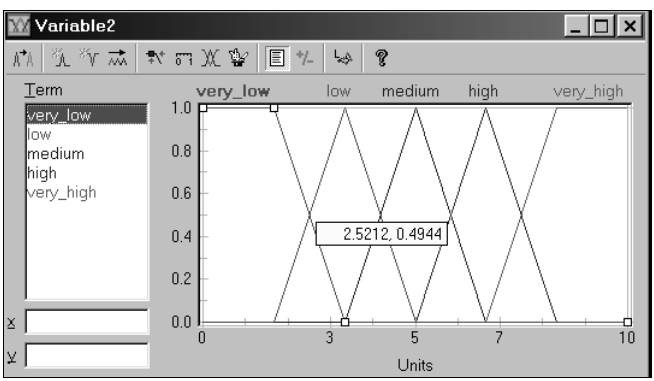


Рис. 19.2. Графическое окно редактора лингвистической переменной с именем Variable2

Для редактируемой лингвистической переменной один из термов в редакторе всегда выделен фоном цвета этого терма в секции имен термов. Для удобства визуального представления все термы редактируемой лингвистической переменной изображаются различными цветами. Изменить цвет отдельного терма можно с помощью операции редактирования цвета в окне свойств контекстного меню (рис. 19.3).

Выделенному терму в окне функций принадлежности соответствует график этого же цвета с характерными маленькими квадратами или маркерами в местах пересечения кривой с горизонтальными линиями, соответствующими значениям функции принадлежности 0 и 1. Указанные маркеры могут быть использованы разработчиком для изменения вида и значений функции принадлежности. Для этого следует позиционировать курсор мыши на нужном маркере, нажать левую кнопку мыши и, не отпуская ее, переместить маркер в нужном направлении. Изменение функции принадлежности можно контролировать визуально с помощью изменения соответствующего графика в секции функций принадлежности редактора.

При перемещении курсора в окне секции функций принадлежности появляется всплывающая подсказка с координатами указателя мыши на плоскости, при этом первое число соответствует горизонтальной координате (значению редактируемой лингвистической переменной), а второе число — вертикальной координате (значению функции принадлежности).

Выделить нужный терм в секции имен можно последовательным нажатием клавиш <↓> и <↑> на клавиатуре или щелчком левой кнопки мыши на имени терма в этой секции. При этом происходит выделение соответствующей кривой в секции функций принадлежности редактора.

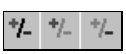
Выделенный терм редактируемой лингвистической переменной можно удалить, для чего достаточно нажать клавишу <Delete> на клавиатуре или воспользоваться соответствующей операцией контекстного меню (см. рис. 19.3). Для добавления новых термов к редактируемой лингвистической переменной следует воспользоваться соответствующей кнопкой панели инструментов редактора или операцией контекстного меню (см. рис. 19.3).

Панель инструментов редактора содержит набор кнопок, которые позволяют выполнить отдельные операции по редактированию лингвистической переменной. Назначение отдельных кнопок панели инструментов приводится в табл. 19.1.

Таблица 19.1. *Назначение кнопок панели инструментов редактора лингвистической переменной*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Next Term	Осуществляет переход к следующему терму редактируемой лингвистической переменной
	New Term	Открывает окно задания нового терма для редактируемой лингвистической переменной, при этом можно задать имя нового терма и форму его функции принадлежности
	Inverse Term	Создает новый терм для редактируемой лингвистической переменной, функция принадлежности которого симметрична функции принадлежности выделенного терма относительно горизонтальной линии на уровне 0.5

Таблица 19.1 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Sort Terms	Упорядочивает множество термов редактируемой лингвистической переменной по максимальным значениям их функций принадлежности, при этом вначале располагаются положительные термы, а затем отрицательные термы
	Grid	Открывает диалоговое окно свойств редактора лингвистической переменной с предложением изменить величину координатной сетки
	Base Variable	Открывает диалоговое окно свойств редактора лингвистической переменной с предложением изменить диапазон ее возможных значений
	Standard MBF	Преобразует функции принадлежности всех термов редактируемой лингвистической переменной к стандартной форме
	Variables Wizard	Вызывает третье по счету окно мастера лингвистической переменной (Linguistic Variables Wizard) с предложением изменить форму функций принадлежности отдельных термов
	Term List Box	Отображает/скрывает секцию с именами термов редактируемой лингвистической переменной
	**Hyperde-fuzzification**	Отображает/скрывает результаты дефаззификации (всех/положительных/отрицательных) значений выходной лингвистической переменной. Эта кнопка может изменять свой внешний вид, при этом она становится активной только в режиме отладки проекта и при выборе соответствующего метода дефаззификации — Hyper CoM
	Learn MBFs	Включает/отключает режим обучения для спецификации функций принадлежности термов редактируемой лингвистической переменной
	Help	Вызывает раздел справочной системы, посвященный редактору лингвистической переменной

Кроме панели инструментов, операции редактора лингвистической переменной доступны через контекстное меню, которое может быть вызвано с помощью нажатия правой кнопки мыши при позиционировании курсора в пределах графического окна редактора (рис. 19.3).

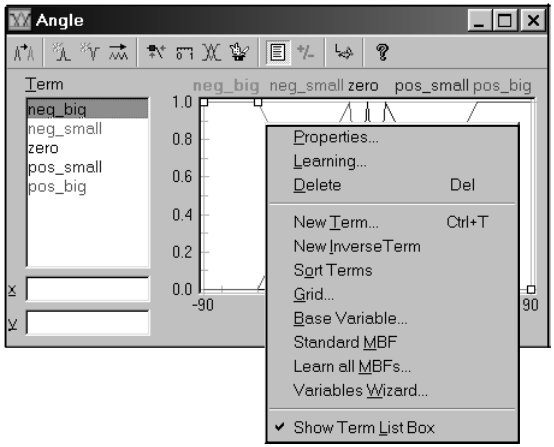


Рис. 19.3. Контекстное меню редактора лингвистической переменной

Контекстное меню редактора лингвистической переменной содержит следующие операции:

- **Properties...** (Свойства) — открывает окно свойств выделенного терма для их редактирования;
- **Learning...** (Обучение) — открывает окно свойств процедуры обучения функции принадлежности для выделенного терма;
- **Delete** (Удалить) — удаляет выделенный терм у редактируемой лингвистической переменной;
- **New Term...** (Новый терм) — открывает окно задания нового терма для редактируемой лингвистической переменной, при этом можно задать имя нового терма и форму его функции принадлежности;
- **New Inverse Term** (Новый инверсный терм) — создает новый терм для редактируемой лингвистической переменной, функция принадлежности которого симметрична функции принадлежности исходного выделенного терма относительно горизонтальной линии на уровне 0.5;
- **Sort Terms** (Сортировка термов) — упорядочивает множество термов редактируемой лингвистической переменной по максимальным значениям их функций принадлежности, при этом вначале располагаются положительные термы, а затем отрицательные термы;
- **Grid...** (Сетка) — открывает диалоговое окно свойств редактируемой лингвистической переменной с предложением изменить величину отображения координатной сетки;
- **Base Variable...** — открывает диалоговое окно свойств редактируемой лингвистической переменной с предложением изменить диапазон ее возможных значений;

- **Standard MBF** — преобразует функции принадлежности всех термов редактируемой лингвистической переменной к стандартной форме;
- **Learn all MBFs...** — включает/отключает режим обучения для спецификации функций принадлежности всех термов редактируемой лингвистической переменной;
- **Variables Wizard...** — вызывает третье по счету окно мастера лингвистической переменной (Linguistic Variables Wizard) с предложением изменить форму функций принадлежности отдельных термов;
- **Show Term List Box** — отображает/скрывает секцию с именами термов редактируемой лингвистической переменной.

Закрыть активный редактор лингвистической переменной можно с помощью соответствующей кнопки управления окном. Закрыть все открытые редакторы лингвистических переменных можно с помощью операции главного меню: **Window>Close all Variable Editors**.

При закрытии редактора лингвистических переменных все внесенные изменения автоматически сохраняются в текущем проекте (но не во внешнем файле проекта). При необходимости можно отменить внесенные изменения с помощью соответствующей кнопки панели инструментов программы fuzzyTECH (см. табл. 18.3) или операции главного меню.

Графические редакторы правил системы нечеткого вывода

Редакторы правил системы нечеткого вывода (далее сокращенно — *редакторы правил*), как следует из их названия, предназначены для редактирования правил нечетких продукций, используемых в различных блоках правил проекта. В программе fuzzyTECH редакторы правил могут иметь одну из графических форм, приведенных ниже.

- Табличный редактор правил (Spreadsheet Rule Editor).
- Матричный редактор правил (Matrix Rule Editor).

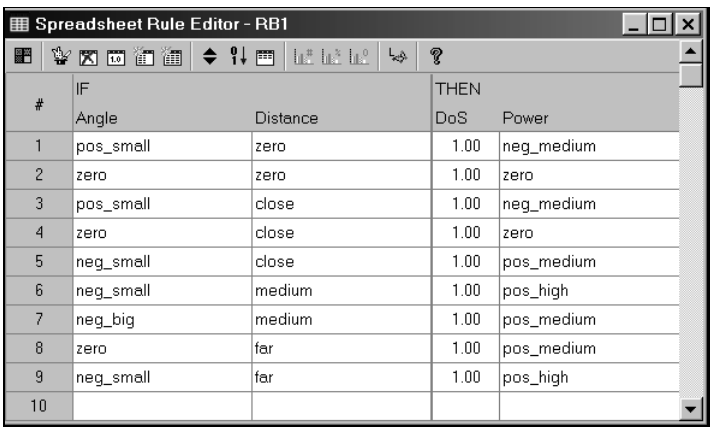
Графическое окно табличного редактора правил может быть открыто двойным щелчком мыши на изображении соответствующего блока правил в окне редактора проекта или двойным щелчком мыши на имени соответствующего блока правил в окне просмотра структуры проекта.

Примечание

Открытие табличного или матричного редактора правил по двойному щелчку мыши можно задать на вкладке **General** (Общие) диалогового окна командой операцию главного меню **Tools>FuzzyTECH Preferences...**. Если установить флажок **Open Matrix Rule Editor by Double-clicking Rule Block**, то по двойному щелчку мыши будет открываться матричный редактор правил, если этот флажок снять — то табличный редактор правил (установлен по умолчанию).

Открыть окна табличных редакторов для всех блоков правил проекта можно также с помощью соответствующей операции главного меню **Window>Open all Spreadsheet Rule Editors**.

Результат вызова табличного редактора правил для блока правил с именем RB1, который имеет 9 правил нечетких продукций, изображен на рис. 19.4.



#	IF		THEN	
	Angle	Distance	DoS	Power
1	pos_small	zero	1.00	neg_medium
2	zero	zero	1.00	zero
3	pos_small	close	1.00	neg_medium
4	zero	close	1.00	zero
5	neg_small	close	1.00	pos_medium
6	neg_small	medium	1.00	pos_high
7	neg_big	medium	1.00	pos_medium
8	zero	far	1.00	pos_medium
9	neg_small	far	1.00	pos_high
10				

Рис. 19.4. Графическое окно табличного редактора правил нечетких продукций

Графическое окно табличного редактора правил ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов и секцию правил нечетких продукций, которая занимает основную часть графического окна редактора. При этом правила в этой секции представляются в форме таблицы, что и предопределило название данного редактора.

Каждому правилу нечетких продукций редактируемого блока правил соответствует отдельная строка секции. Слева от правила указывается порядковый номер этого правила. Далее следуют имена термов входных лингвистических переменных, которые формируют отдельные подусловия правил нечетких продукций. После них указывается значение весового коэффициента правила (Degree of Support или сокращенно — DoS). Далее следуют имена термов выходных лингвистических переменных, которые формируют отдельные заключения правил нечетких продукций. Имена входных и выходных переменных редактируемого блока правил указываются вверху соответствующего столбца таблицы.

Панель инструментов табличного редактора правил содержит набор кнопок, которые позволяют выполнить некоторые операции по редактированию правил отдельного блока. Назначение кнопок панели инструментов приводится в табл. 19.2.

Таблица 19.2. *Назначение кнопок панели инструментов табличного редактора правил*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Matrix Editor	Переключает режим представления табличного редактора в режим представления матричного редактора для редактируемого блока правил
	Rule Block Wizard	Вызывает третье по счету окно мастера блока правил с предложением изменить степень влияния входных переменных на выходные переменные редактируемого блока правил
	Delete All Rules	Удаляет все правила у редактируемого блока правил
	Set DoS	Открывает окно задания значения весового коэффициента для правил редактируемого блока, при этом данному коэффициенту можно задать произвольное значение
	Partial Rule Block	Удаляет все правила редактируемого блока и формирует новую базу правил из всех возможных комбинаций термов входных лингвистических переменных. При этом столбцы значений выходных переменных остаются пустыми, а значения всех весовых коэффициентов правил (DoS) принимаются равными 1
	Full Rule Block	Удаляет все правила у редактируемого блока и формирует новую базу правил из всех возможных комбинаций термов как входных, так и выходных лингвистических переменных. При этом значения всех весовых коэффициентов правил (DoS) принимаются равными 1
	Fuzzy Operators	Открывает окно задания параметров нечетких операций для правил редактируемого блока (рис. 19.22)
	Sort Rules	Открывает окно задания параметров упорядочения правил нечетких продукций для редактируемого блока правил
	Compressed Rule Table	Открывает окно представления правил редактируемого блока в сокращенном (сжатом) формате
	Show Absolute Statistics	Добавляет к таблице правил редактируемого блока новый столбец с указанием абсолютного количества активизации правил в течение заданного интервала моделирования. Кнопка доступна только при включении анализатора статистики

Таблица 19.2 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Show Relative Statistics	Добавляет к таблице правил редактируемого блока новый столбец с указанием относительного (в %) количества активизации правил в течение заданного интервала моделирования. Кнопка доступна только при включении анализатора статистики
	Reset Statistics Counter	Сбрасывает (обнуляет) количественные значения активизации правил. Кнопка доступна только при включении анализатора статистики
	Learn DoS	Открывает диалоговое окно параметров обучения для определения значений весовых коэффициентов (DoS) правил редактируемого блока
	Help	Вызывает раздел справочной системы, посвященный табличному редактору правил нечетких продукций

Кроме панели инструментов, некоторые операции редактирования правил доступны через контекстное меню редактора, которое может быть вызвано с помощью нажатия правой кнопки мыши при позиционировании курсора в пределах графического окна редактора правил. Если контекстное меню открывается в области значения отдельного терма, то в контекстном меню появляются все возможные значения термов соответствующей лингвистической переменной, при этом выбор некоторого из этих значений изменяет подусловие редактируемого правила. Если контекстное меню открывается в области значения отдельного весового коэффициента, то в контекстном меню появляются все возможные значения весовых коэффициентов правил, при этом выбор некоторого из этих значений изменяет весовой коэффициент DoS редактируемого правила.

Графическое окно матричного редактора правил может быть открыто с помощью кнопки переключения окон редакторов (см. табл. 19.2). В дополнение к этому способу матричный редактор правил можно открыть двойным щелчком мыши на изображении соответствующего блока правил в окне редактора проекта или двойным щелчком мыши на имени соответствующего блока правил в окне просмотра структуры проекта, если в настройках программы fuzzyTECH установлен флажок у соответствующего свойства.

Результат вызова табличного редактора правил для блока правил с именем RB1, который имеет 9 правил нечетких продукций, изображен на рис. 19.5.

Графическое окно матричного редактора правил ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов. Основную часть окна редактора занимает секция правил нечетких продукций. Правила в этой секции представляются в форме матрицы, что и предопределило

название данного редактора. Внешний вид этой секции зависит от общего количества правил редактируемого блока и выбора режима цветового представления заключений правил.

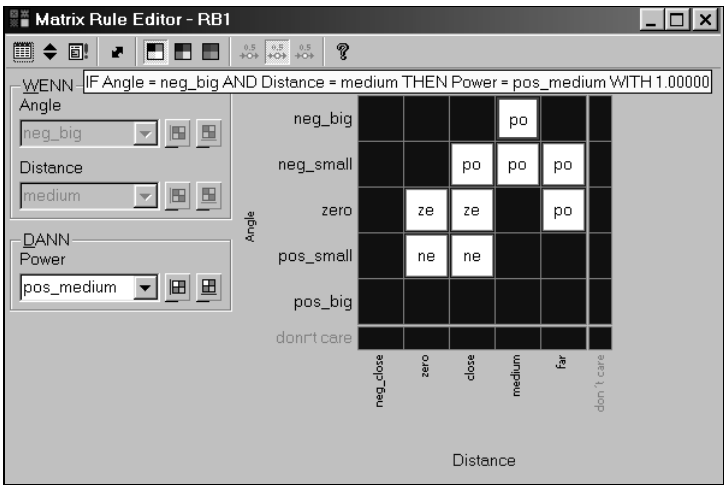


Рис. 19.5. Графическое окно матричного редактора правил нечетких продукций

Каждому правилу нечетких продукций редактируемого блока правил соответствует отдельная ячейка или клетка матрицы. Слева от матрицы правил указываются имена термов первой входной лингвистической переменной, внизу матрицы указываются имена термов второй входной лингвистической переменной. В соответствующей ячейке матрицы указывается сокращенное имя терма выходной лингвистической переменной, которое соответствует заключению отдельного правила нечетких продукций.

Значение весового коэффициента правила изображается оттенком цвета от белого (DoS = 1) до темно-серого (DoS = 0), если включен режим отображения в ячейках весовых коэффициентов правил. Имена входных переменных редактируемого блока правил указываются слева и снизу матрицы правил, а имя выходной лингвистической переменной указывается выше списка вложенного меню с именами значений ее термов.

При перемещении курсора в секции правил редактора появляется всплывающая подсказка с текстом соответствующего правила нечеткой продукции (рис. 19.5). Данная подсказка позволяет контролировать правильность записи отдельных правил в редактируемом блоке правил.

Панель инструментов матричного редактора правил содержит набор кнопок, которые позволяют выполнить отдельные операции по редактированию блока правил. Назначение отдельных кнопок панели инструментов приводится в табл. 19.3.

Таблица 19.3. *Назначение кнопок панели инструментов матричного редактора правил*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Spreadsheet Rule Editor	Переключает режим представления матричного редактора в режим представления табличного редактора для редактируемого блока правил
	Fuzzy Operators	Открывает окно задания параметров нечетких операций для правил редактируемого блока правил (см. рис. 19.22)
	FuzzyTECH Preferences... Matrix Rule Editor	Открывает диалоговое окно настроек рабочего интерфейса программы fuzzyTECH на вкладке свойств матричного редактора правил
	Swap Row and Column	Транспонирует матрицу правил редактируемого блока, меняя местами входные лингвистические переменные
	Display Gray Scale	Отображает значения весовых коэффициентов (DoS) в секции правил нечетких продукций в оттенках серого цвета
	Display Term Colors	Термы выходной лингвистической переменной в секции правил нечетких продукций отображаются в оттенках цвета этого терма, определенного в редакторе лингвистической переменной
	Display False Colors	Термы выходной лингвистической переменной в секции правил нечетких продукций отображаются в оттенках зеленого (для положительных значений терма) и красного цвета (для отрицательных значений терма)
	Display Input Aggregation	В ячейках матрицы секции правил отображаются промежуточные результаты вывода после агрегирования входных лингвистических переменных. Эта кнопка становится доступной только в режиме отладки проекта
	Display Degree of Support (DoS)	В ячейках матрицы секции правил отображаются значения весовых коэффициентов правил. Эта кнопка всегда нажата (по умолчанию) в режиме редактирования проекта
	Display Composition with Degree of Support (DoS)	В ячейках матрицы секции правил отображается промежуточный результат вывода после композиции выходных лингвистических переменных. Эта кнопка становится доступной только в режиме отладки проекта

Таблица 19.3 *(окончание)*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Help	Вызывает раздел справочной системы, посвященный матричному редактору правил нечетких продукций

Закрыть активный матричный редактор правил можно с помощью соответствующей кнопки управления окном. Закрыть все открытые табличные и матричные редакторы правил, как уже отмечалось ранее, можно с помощью операции главного меню **Window>Close all Spreadsheet Rule Editors**.

При закрытии редактора правил все внесенные изменения автоматически сохраняются в текущем проекте (но не во внешнем файле проекта). При необходимости можно отменить внесенные изменения с помощью соответствующей кнопки панели инструментов программы fuzzyTECH (см. табл. 18.3) или операции главного меню.

Графические средства анализа результатов нечеткого вывода

В качестве графических средств анализа результатов нечеткого вывода в программе fuzzyTECH используются специальные графические окна режима отладки проекта (Watch Window) и анализатора правил нечетких продукций (Rule Analyzer).

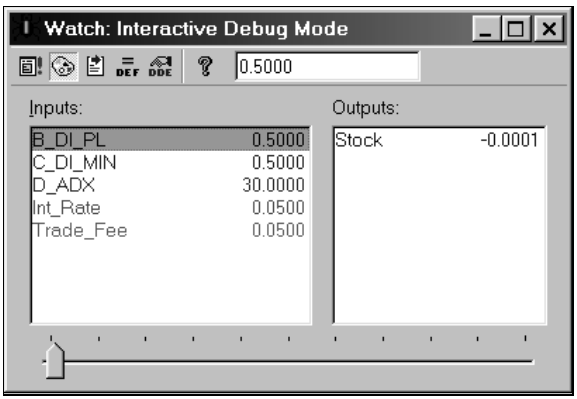
Режим отладки (Debug Mode) нечеткого проекта является основным способом получения информации о результатах нечеткого вывода в рамках разрабатываемой системы нечеткого вывода. Главное назначение этого режима — контроль значений выходных и промежуточных лингвистических переменных разработанной системы нечеткого вывода для различных значений входных лингвистических переменных. При этом основным средством контроля является графическое окно режима отладки проекта, а вспомогательным — графическое окно анализатора правил нечетких продукций.

Перейти из режима разработки проекта в интерактивный режим отладки можно с помощью операции главного меню **Debug>Interactive** или с помощью соответствующей кнопки панели инструментов программы fuzzyTECH (см. табл. 18.3).

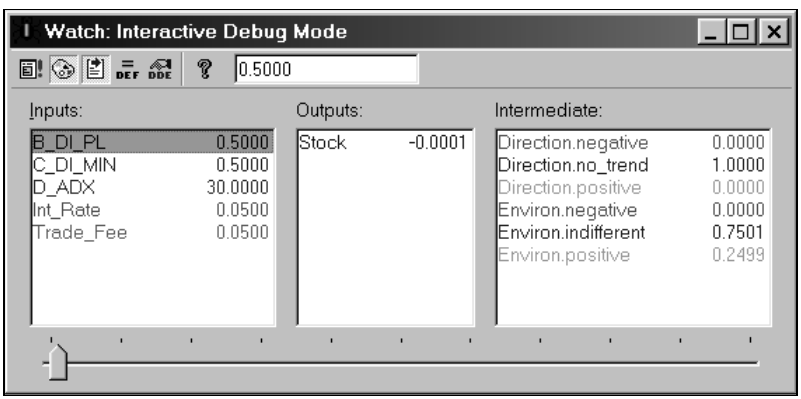
Графическое окно режима отладки проекта **(Watch Window)** открывается автоматически при переходе в режим отладки проекта и остается открытым постоянно до выхода из этого режима (рис. 19.6).

Графическое окно режима отладки проекта ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов и две (рис. 19.6, *а*) или три (рис. 19.6, *б*) секции. В первом случае — секцию имен и значений входных лингвистических переменных, расположенную слева, и секцию

имен и значений выходных лингвистических переменных, расположенную справа. Во втором случае в дополнение к указанным секциям справа добавляется секция имен и значений промежуточных лингвистических переменных, если они присутствуют в проекте. Внешний вид этих секций зависит от общего количества лингвистических переменных в текущем нечетком проекте. В нижней части графического окна режима отладки проекта расположен регулятор выбора значений входных лингвистических переменных.



а



б

Рис. 19.6. Графические окна режима отладки проекта системы нечеткого вывода

Для удобства графического восприятия каждая из лингвистических переменных проекта имеет собственный цвет, который служит для выделения входных переменных фоном соответствующего цвета в секции имен входных лингвистических переменных. При необходимости можно отменить режим цветного изображения переменных с помощью операции панели инструментов (табл. 19.4). Выделить нужную входную лингвистическую переменную в секции имен можно последовательным нажатием клавиш <↓> и <↑> на клавиатуре или щелчком левой кнопки мыши на имени лингвистической переменной в этой секции.

Выделенной входной лингвистической переменной всегда соответствует некоторое количественное значение, которое отображается в небольшом окне справа от панели инструментов. Изменить это значение можно непосредственным изменением значения в этом окне либо с помощью перемещения движка регулятора значений в ту или иную сторону. После изменения отдельных значений входных лингвистических переменных происходит соответствующее изменение значения выходной лингвистической переменной, которое отображается в секции имен выходных лингвистических переменных справа.

Панель инструментов графического окна режима отладки содержит набор кнопок, которые позволяют выполнить отдельные операции в режиме отладки. Назначение отдельных кнопок панели инструментов приводится в табл. 19.4.

Таблица 19.4. *Назначение кнопок панели инструментов окна режима отладки*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	FuzzyTECH Preferences - Watch Window	Открывает диалоговое окно настроек рабочего интерфейса программы fuzzyTECH на вкладке свойств окна режима отладки
	Colored	Включает/отключает режим цветового выделения лингвистических переменных проекта
	Intermediate Variables	Отображает/скрывает дополнительную секцию имен и значений промежуточных лингвистических переменных проекта. Кнопка недоступна в случае отсутствия в проекте промежуточных переменных
	Default Values	Устанавливает значения всех входных лингвистических переменных проекта, заданные по умолчанию
	Reinitialize DDE Links	Обновляет все связи динамического обмена данными, используемыми совместно с другими программами в рамках технологии DDE
	Help	Вызывает раздел справочной системы, посвященный графическому окну режима отладки проекта

Анализатор правил нечетких продукций (Rule Analyzer) можно вызвать с помощью операции главного меню **Analyzer>New Rule Analyzer…** или с помощью соответствующей кнопки панели инструментов программы fuzzyTECH (см. табл. 18.3). Напомним, что в последнем случае кнопка анализатора правил становится доступной только в режиме отладки проекта.

В результате вызова анализатора правил нечетких продукций будет открыто соответствующее графическое окно (рис. 19.7).

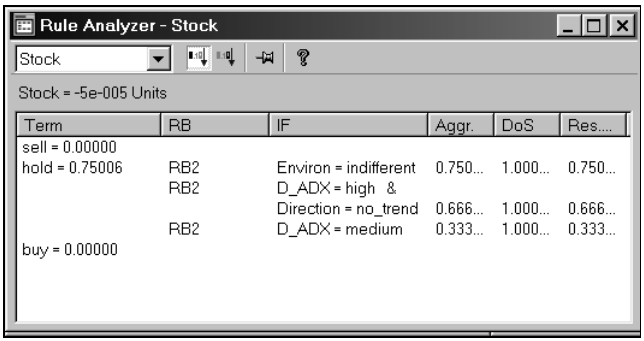


Рис. 19.7. Графическое окно анализатора правил нечетких продукций

Графическое окно анализатора правил ниже стандартной строки заголовка и кнопок управления окном имеет собственную панель инструментов. Центральную часть графического интерфейса анализатора правил занимает секция с именами и значениями термов выходной лингвистической переменной. Выбор выходной лингвистической переменной для анализа ее значений осуществляется с помощью поля списка, расположенного слева от панели инструментов

Панель инструментов окна анализатора правил режима отладки содержит всего лишь 4 кнопки, которые позволяют выполнить отдельные операции. Назначение кнопок панели инструментов приводится в табл. 19.5.

Таблица 19.5. *Назначение кнопок панели инструментов окна анализатора правил нечетких продукций*

Графическое изображение	Всплывающая подсказка	Назначение кнопки
	Sort Aggregation	Упорядочивает правила согласно значениям до агрегирования их подусловий
	Sort Result - Aggregation	Упорядочивает правила согласно значениям после агрегирования их подусловий
	Freeze	Останавливает процесс нечеткого вывода заключений для анализируемых правил проекта
	Help	Вызывает раздел справочной системы, посвященный графическому окну анализатора правил проекта

Закрыть графическое окно анализатора правил нечеткого вывода можно с помощью соответствующей кнопки управления окном либо автоматически при выходе из режима интерактивной отладки. Как указывалось ранее, для выхода из режима отладки проекта следует убрать отметку у операции главного меню **Debug>Interactive** или отжать ранее нажатую кнопку панели инструментов программы fuzzyTECH.

19.2. Основные средства разработки проектов и компонентов систем нечеткого вывода в fuzzyTECH

Для быстрой разработки прототипов проектов и задания отдельных компонентов систем нечеткого вывода в интерактивном режиме могут быть использованы специальные средства или мастера, входящие в состав программы fuzzyTECH, перечисленные далее.

- Мастер нечеткого проекта (FDW, Fuzzy Design Wizard).
- Мастер лингвистической переменной (Linguistic Variables Wizard).
- Мастер блока правил (Rule Block Wizard).
- Мастер нечетких систем реального времени (Online Wizard for Fuzzy Runtime System).

Далее в этой главе приводится характеристика этих мастеров и рассматриваются особенности создания новых проектов и компонентов систем нечеткого вывода с их помощью.

Мастер нечеткого проекта

Мастер нечеткого проекта предназначен для создания базовой структуры или прототипа системы нечеткого вывода и позволяет специфицировать все ее основные компоненты. Мастер нечеткого проекта реализован в форме последовательности диалоговых окон, каждое из которых служит для спецификации отдельных свойств компонентов создаваемого проекта.

Мастер нечеткого проекта может быть вызван с помощью операции главного меню **File>Fuzzy Design Wizard...** или с помощью соответствующей кнопки панели инструментов программы fuzzyTECH (см. табл. 18.3).

В результате вызова мастера нечеткого проекта будет открыто первое окно из последовательности диалоговых окон спецификации свойств отдельных компонентов создаваемого прототипа проекта (рис. 19.8).

В первом диалоговом окне разработчику предлагается согласиться с выбором варианта создания новой системы нечеткого вывода (**Create New System** — по умолчанию) или изменить его на вариант добавления системы нечеткого вывода к существующему проекту (**Append to Existing System**). При необходимости можно воспользоваться возможностью использования внешнего файла формата (CSV, Comma Separated Values) с некоторой выборкой данных для спецификации компонентов создаваемого проекта.

После этого с помощью нажатия кнопки **Next** можно перейти ко второму диалоговому окну мастера нечеткого проекта (рис. 19.9) для задания количества входных, выходных и промежуточных лингвистических переменных проекта и общего количества их термов.

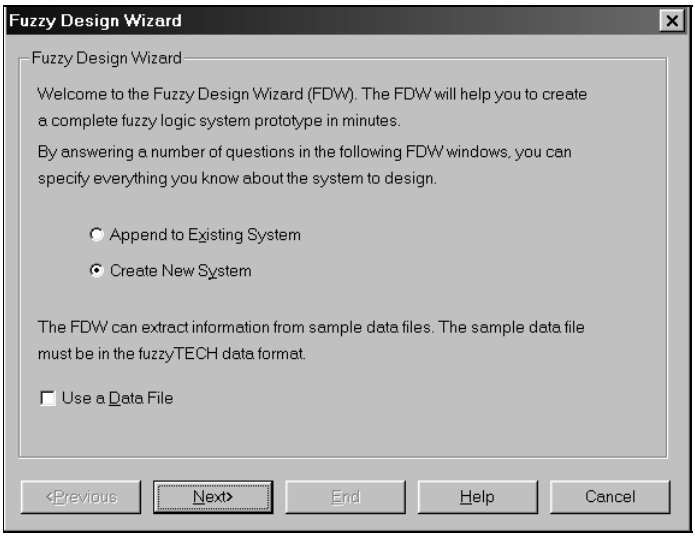


Рис. 19.8. Первое диалоговое окно мастера нечеткого проекта FDW

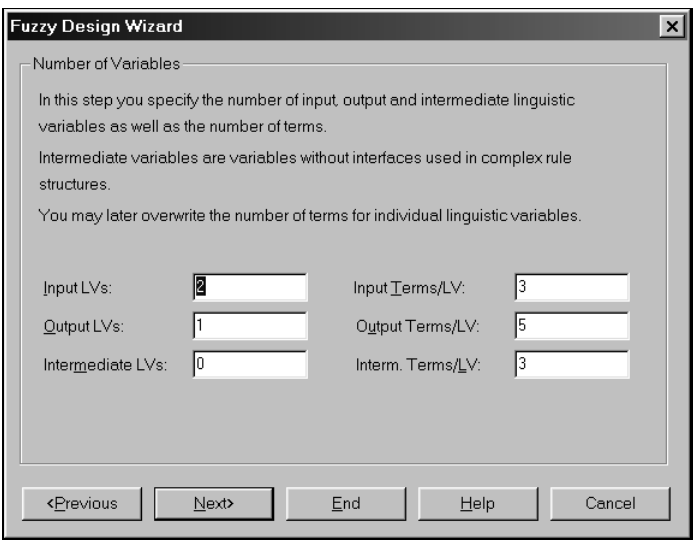


Рис. 19.9. Второе диалоговое окно мастера нечеткого проекта FDW

Во втором диалоговом окне мастера нечеткого проекта можно задать количество входных **(Input LVs)**, выходных **(Output LVs)** и промежуточных **(Intermediate LVs)** лингвистических переменных проекта, а также общее количество термов входных **(Input Terms/LV)**, выходных **(Output Terms/LV)** и промежуточных **(Interm. Terms/LV)** переменных.

После задания общего количества лингвистических переменных с помощью нажатия кнопки **Next** можно перейти к следующим диалоговым окнам мастера нечеткого проекта для спецификации параметров или свойств отдельных входных лингвистических переменных. При этом для каждой из входных переменных будет открыто отдельное диалоговое окно (рис. 19.10).

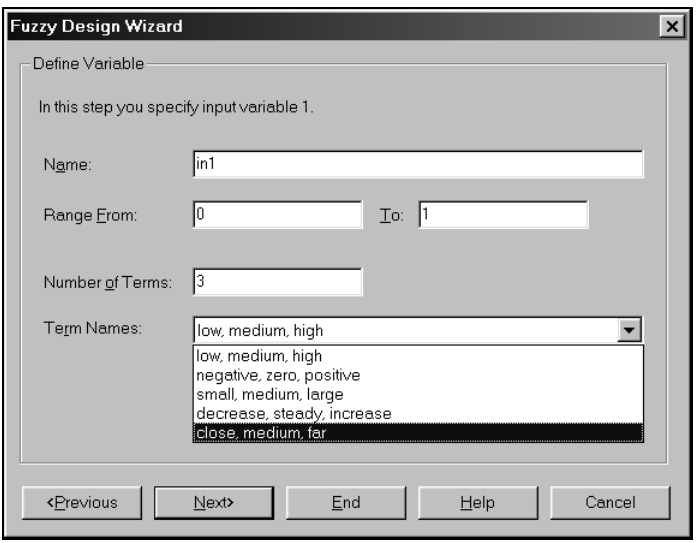


Рис. 19.10. Диалоговое окно мастера нечеткого проекта FDW для спецификации входных лингвистических переменных

В диалоговом окне спецификации свойств входных лингвистических переменных можно изменить следующие параметры, предложенные программой fuzzyTECH по умолчанию: имя лингвистической переменной **(Name — in1)**, диапазон изменения значений лингвистической переменной **(Range From)**, количество термов лингвистической переменной **(Number of Terms)**, а также выбрать имена этих термов **(Term Names)** из предложенного мастером списка.

Примечание

Следует заметить, что количество вариантов имен термов и сами эти имена в списке зависят от заданного общего количества термов у соответствующей лингвистической переменной.

После спецификации свойств отдельных входных лингвистических переменных с помощью нажатия кнопки **Next** можно перейти к аналогичным диалоговым окнам мастера нечеткого проекта для спецификации параметров или свойств отдельных выходных и промежуточных лингвистических переменных. При этом для каждой из выходных переменных также будет открыто отдельное диалоговое окно (рис. 19.11).

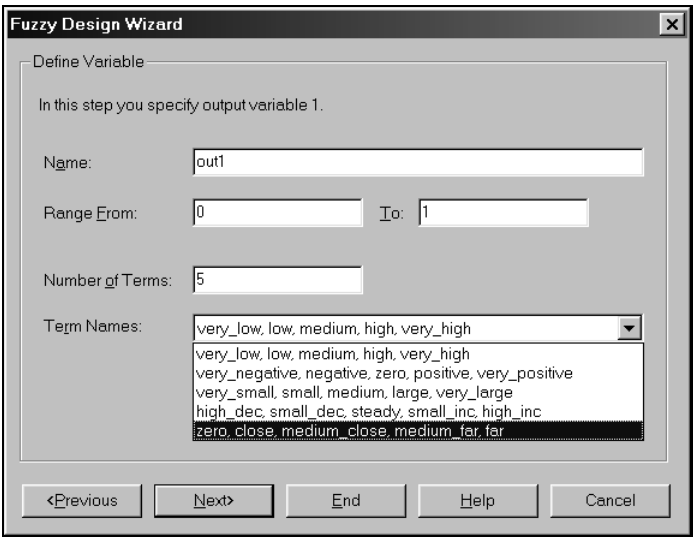


Рис. 19.11. Диалоговое окно мастера нечеткого проекта FDW для спецификации выходных лингвистических переменных

После спецификации свойств всех лингвистических переменных с помощью нажатия кнопки **Next** можно перейти к диалоговому окну для спецификации метода дефаззификации создаваемого прототипа системы нечеткого вывода (рис. 19.12).

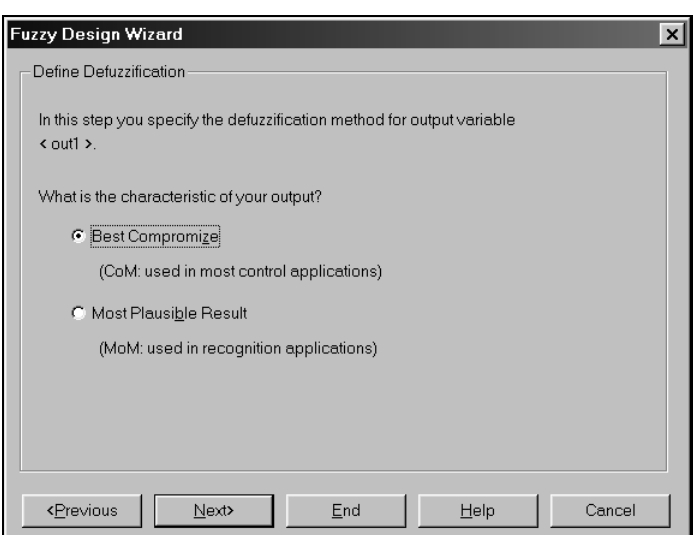


Рис. 19.12. Диалоговое окно мастера нечеткого проекта FDW для спецификации метода дефаззификации

В диалоговом окне спецификации метода дефаззификации можно либо согласиться с методом центра максимума (**Best Compromize (CoM: uesd in most control applications**)), предлагаемым мастером по умолчанию для большинства приложений нечеткого управления, либо изменить его на метод среднего максимума (**Most Plausible Result** (**MoM: used in recognition applications**)), который рекомендуется использовать в приложениях распознавания образов.

После выбора метода дефаззификации с помощью нажатия кнопки **Next** можно перейти к последнему диалоговому окну мастера нечеткого проекта для спецификации блоков правил нечетких продукций (рис. 19.13).

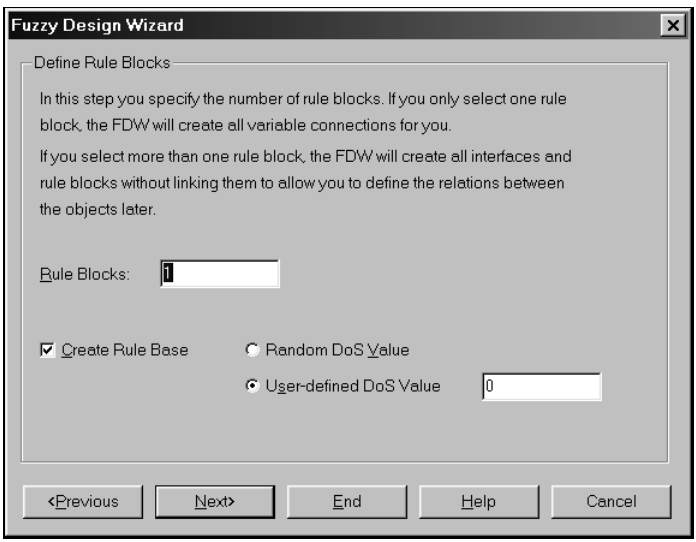


Рис. 19.13. Диалоговое окно мастера нечеткого проекта FDW для спецификации блока правил проекта

В диалоговом окне спецификации блоков правил можно задать общее количество блоков правил (**Rule Blocks**), сгенерировать полное множество правил на основе комбинации всех термов лингвистических переменных проекта (оставить флажок **Create Rule Base**). Здесь также можно задать значение весовых коэффициентов правил (**User-defined DoS Value**) либо выбрать вариант случайного задания их значений (**Random DoS Value**).

После спецификации блоков правил нечетких продукций создаваемого проекта с помощью нажатия кнопки **Next** и подтверждения будет создан прототип системы нечеткого вывода, структура которого будет изображена в окне редактора проекта (рис. 19.14).

Для быстрого создания стандартного прототипа нового проекта системы нечеткого вывода можно воспользоваться упрощенным вариантом мастера нечеткого проекта, который может быть вызван с помощью операции главного ме-

ню **File>New** или с помощью соответствующей кнопки панели инструментов программы fuzzyTECH (см. табл. 18.3).

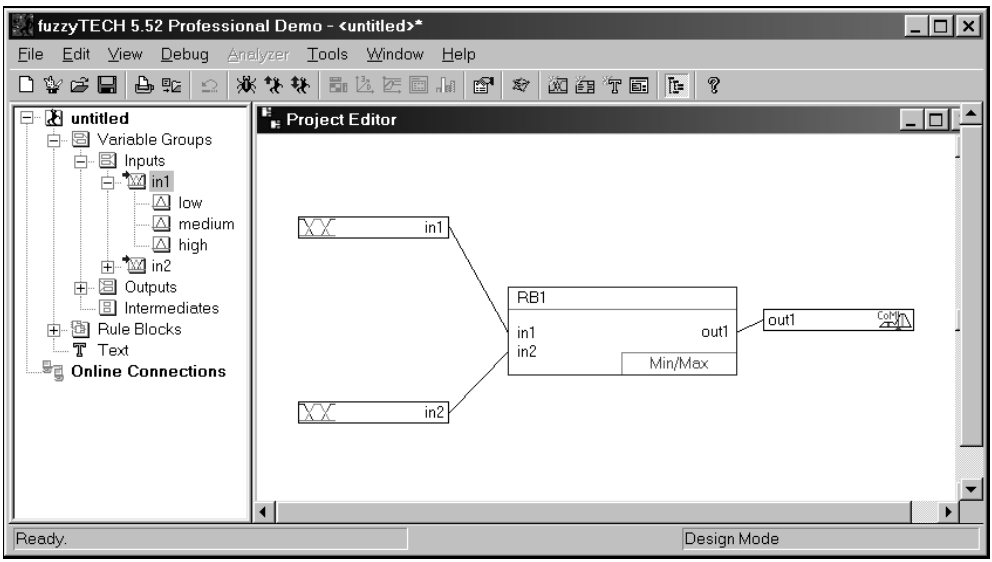


Рис. 19.14. Графический вид редактора проекта с новой системой нечеткого вывода, созданной с помощью мастера нечеткого проекта FDW

При вызове упрощенного мастера быстрого создания прототипа нечеткого проекта будет открыто единственное диалоговое окно спецификации общих свойств создаваемого нечеткого проекта (рис. 19.15).

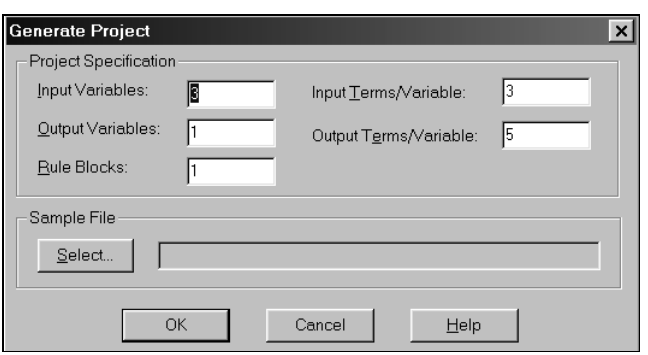


Рис. 19.15. Диалоговое окно быстрого создания стандартного прототипа нового нечеткого проекта

В диалоговом окне быстрого создания стандартного прототипа нечеткого проекта можно задать количество входных и выходных лингвистических переменных проекта, количество блоков правил, а также общее количество термов входных и выходных переменных. При необходимости можно использовать внешний файл с некоторой выборкой данных для спецификации функций принадлежности термов лингвистических переменных.

После спецификации общих свойств создаваемого нечеткого проекта и нажатия кнопки **OK** будет создан прототип системы нечеткого вывода, структура которой будет изображена в окне редактора проекта аналогично структуре, изображенной на рис. 19.14. После создания прототипов систем нечеткого вывода при необходимости редактирования свойств их отдельных компонентов можно воспользоваться рассмотренными ранее графическими средствами редактирования.

В завершение рассмотрения особенностей мастера нечеткого проекта следует заметить, что нажатие кнопки **End** в любом из диалоговых окон этого мастера закрывает данный мастер и завершает процесс создания прототипа нового нечеткого проекта. При этом все свойства на оставшихся не рассмотренными диалоговых окнах мастера для нового проекта принимаются равными предложенным программой fuzzyTECH по умолчанию.

В свою очередь нажатие кнопки **Previous** в любом из диалоговых окон (кроме первого) возвращает к предыдущему диалоговому окну данного мастера. Нажатие кнопки **Help** в любом из диалоговых окон открывает раздел справочной системы, посвященный соответствующему диалоговому окну, а нажатие кнопки **Cancel** закрывает мастер нечеткого проекта, отказываясь от создания соответствующего прототипа.

Мастер лингвистической переменной

Мастер лингвистической переменной предназначен для быстрого создания новой лингвистической переменной и позволяет специфицировать все ее основные свойства или параметры. Мастер лингвистической переменной реализован в форме последовательности диалоговых окон, каждое из которых служит для спецификации отдельных свойств вновь создаваемой лингвистической переменной.

Мастер лингвистической переменной может быть вызван с помощью операции главного меню **Edit>New Variable...**, а также с помощью соответствующей операции контекстного меню редактора проекта или одновременным нажатием клавиш <Ctrl>+<V> на клавиатуре.

В результате вызова мастера лингвистической переменной будет открыто первое окно из последовательности диалоговых окон спецификации свойств новой лингвистической переменной (рис. 19.16).

В первом диалоговом окне разработчику предлагается изменить имя вновь создаваемой лингвистической переменной, предложенное мастером по умолчанию (в поле **Name**) и выбрать тип этой переменной: входная (**Input Interface**), выходная (**Output Interface**) или промежуточная (**Intermediate**). Секция справа служит для спецификации метода фаззификации вновь создаваемой переменной. При

необходимости можно выбрать любой из 4 методов либо остановиться на стандартном **(Compute MBF)**, предложенное мастером по умолчанию. Кнопка **Color** служит для задания цвета отображения создаваемой переменной, который используется в графическом окне режима отладки.

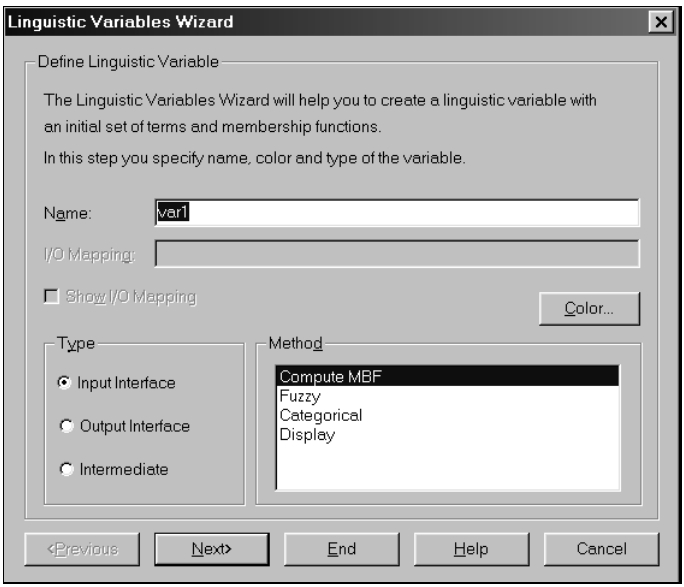


Рис. 19.16. Первое диалоговое окно мастера лингвистической переменной

После спецификации свойств лингвистической переменной в первом диалоговом окне мастера с помощью нажатия кнопки **Next** можно перейти ко второму диалоговому окну мастера лингвистической переменной (рис. 19.17).

Во втором диалоговом окне разработчику предлагается задать диапазон изменения значений создаваемой лингвистической переменной: ее минимальное значение **(Min)**, максимальное значение **(Max)**, и значение, принимаемое по умолчанию в режиме отладки **(Default)**. В поле **Unit** можно ввести наименование единиц измерения данной лингвистической переменной (только на английском).

После этого с помощью нажатия кнопки **Next** можно перейти к третьему диалоговому окну мастера лингвистической переменной (рис. 19.18) для спецификации общих свойств функций принадлежности ее термов.

В третьем диалоговом окне разработчику предлагается задать общее количество термов новой лингвистической переменная **(Number)** и выбрать их стандартные имена из вложенного списка **(Names)**. В этом окне разработчик может также уточнить общую форму функций принадлежности термов создаваемой лингвистической переменной. Для этого можно выбрать вариант задания функций при-

надлежности термов с равенством 1 **(High Shoulder** — рекомендуется для входных переменных) либо 0 **(Low Shoulder** — рекомендуется для выходных переменных) для всех наименьших и наибольших значений лингвистической переменной. При этом можно изменить положение мод отдельных функций принадлежности относительно ширины диапазона значений лингвистической переменной **(Width)**, для чего удобно воспользоваться расположенным ниже регулятором.

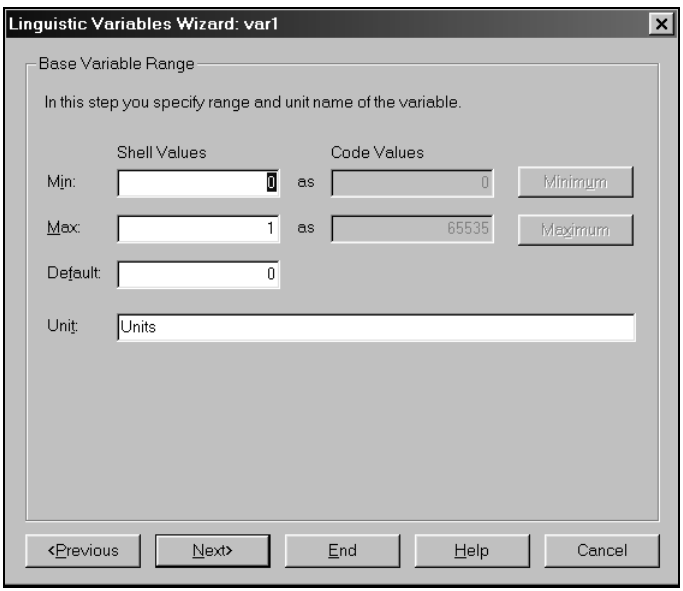


Рис. 19.17. Второе диалоговое окно мастера лингвистической переменной

В дополнение к этому можно изменить симметричность линейных функций принадлежности относительно их модальных значений **(Symmetric/Asymmetrical)**. При выборе первого варианта функции принадлежности термов являются симметричными относительно центра интервала значений лингвистической переменной. Дополнительно можно изменить положение мод отдельных функций принадлежности относительно ширины диапазона значений лингвистической переменной, задавая необходимое значение коэффициента **(Factor)** в интервале значений [0, 500]. При этом значению Factor = 0 соответствует симметричная форма кусочно-линейных кривых, а положительные значения Factor > 0 приближают моды функций принадлежности к оси симметрии.

Для случая выбора асимметричных функций принадлежности можно задать необходимое значение коэффициента **(Factor)** в интервале значений –500, 500]. При этом значению Factor = 0 по-прежнему соответствует симметричная форма кусочно-линейных кривых, положительные значения Factor > 0 смещают моды функций принадлежности вправо, а отрицательные значения Factor < 0 смещают

моды функций принадлежности вправо. Для изменения значений этого коэффициента также удобно воспользоваться расположенным ниже регулятором.

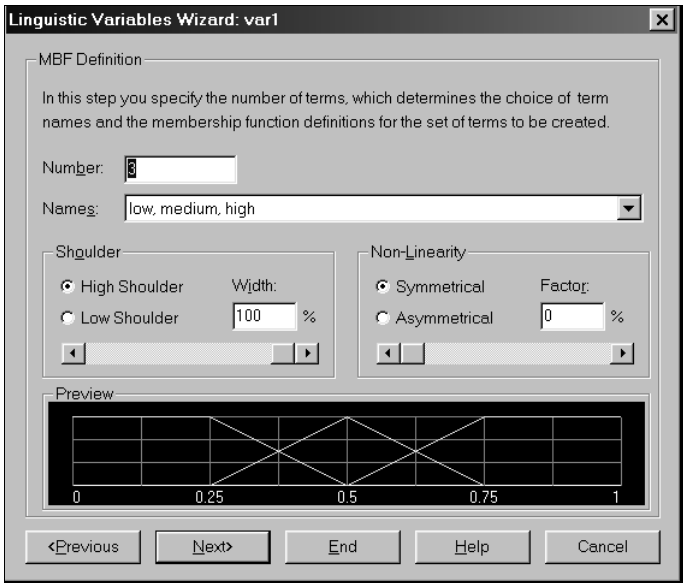


Рис. 19.18. Третье диалоговое окно мастера лингвистической переменной

Контроль настройки общей формы функций принадлежности создаваемой лингвистической переменной осуществляется с помощью графиков соответствующих кривых в окне предварительного просмотра **(Preview)**.

После задания общей формы функций принадлежности с помощью нажатия кнопки **Next** можно перейти к четвертому и последнему диалоговому окну мастера лингвистической переменной (рис. 19.19).

В этом диалоговом окне разработчику предлагается ввести текстовое описание имени переменной **(Variable Name Explanation)** и комментарий **(Comment)** вновь создаваемой лингвистической переменной. При этом возможен ввод символов кириллицы, которые корректно отображаются аналогично тексту на английском в форме всплывающей подсказки при выделении соответствующей лингвистической переменной в окне просмотра структуры проекта после добавления этой переменной к разрабатываемому проекту.

После спецификации блоков правил нечетких продукций создаваемого проекта с помощью нажатия кнопки **Next** или **End** будет создана новая лингвистическая переменная и добавлена к системе нечеткого вывода в окне редактора проекта (рис. 19.20). Для дальнейшего использования созданной лингвистической переменной следует воспользоваться мастером блока правил или окном свойств соответствующего блока правил.

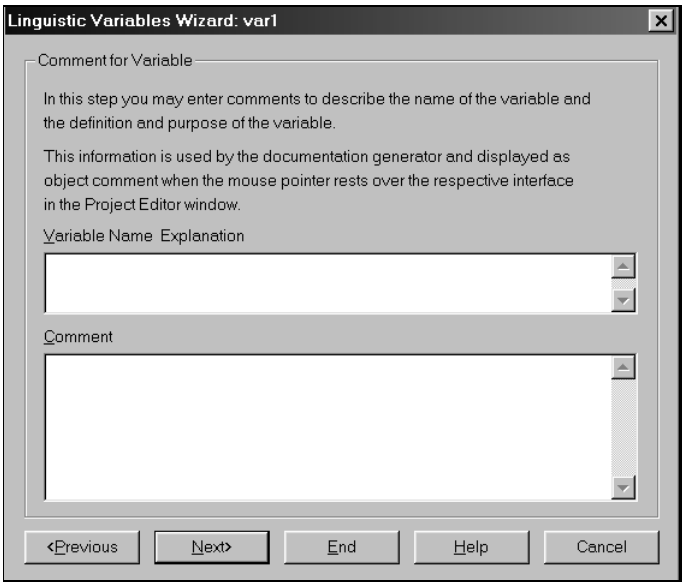


Рис. 19.19. Четвертое диалоговое окно мастера лингвистической переменной

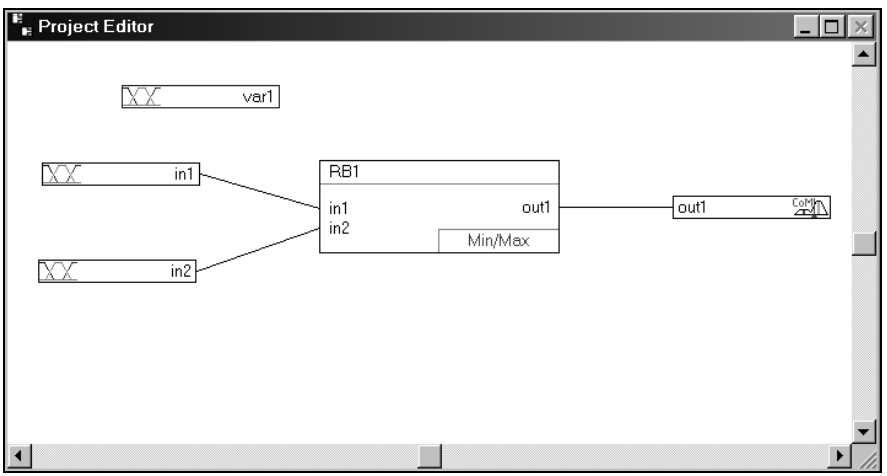


Рис. 19.20. Графический вид редактора проекта с добавленной к нему новой лингвистической переменной

В завершение рассмотрения особенностей мастера лингвистической переменной следует заметить, что нажатие кнопки **End** в любом из диалоговых окон этого мастера закрывает данный мастер и завершает процесс создания новой переменной. При этом все свойства на оставшихся не рассмотренными диалоговых ок-

нах мастера этой лингвистической переменной принимаются по умолчанию, предложенные программой fuzzyTECH.

В свою очередь нажатие кнопки **Previous** в любом из диалоговых окон (кроме первого) возвращает к предыдущему диалоговому окну данного мастера. Нажатие кнопки **Help** в любом из диалоговых окон открывает раздел справочной системы, посвященный соответствующему диалоговому окну, нажатие кнопки **Cancel** закрывает мастер лингвистической переменной.

Мастер блока правил

Мастер блока правил предназначен для создания нового блока правил и позволяет специфицировать все его основные свойства или параметры. Мастер блока правил также реализован в форме последовательности диалоговых окон, каждое из которых служит для спецификации отдельных свойств компонентов создаваемого блока правил.

Мастер блока правил может быть вызван с помощью операции главного меню **Edit>New Rule Block…**, а также с помощью соответствующей операции контекстного меню редактора проекта или одновременным нажатием клавиш <Ctrl>+<R> на клавиатуре.

В результате вызова мастера блока правил будет открыто первое окно из последовательности диалоговых окон спецификации свойств нового блока правил (рис. 19.21).

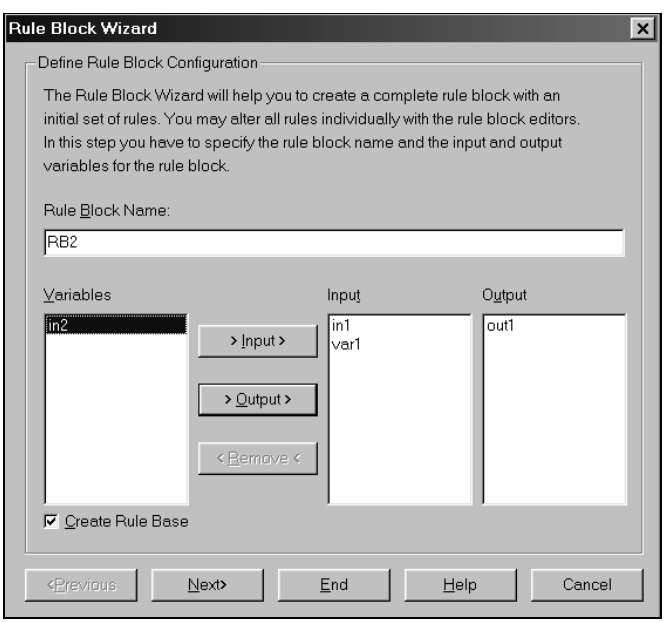


Рис. 19.21. Первое диалоговое окно мастера блока правил

В первом диалоговом окне разработчику предлагается определить в поле **Rule Block Name** имя вновь создаваемого блока правил или оставить без изменения предложенное мастером по умолчанию (RB2), а также определить входные и выходные лингвистические переменные создаваемого блока правил. Для этой цели следует выделить имя лингвистической переменной в секции слева и нажать кнопку **>Input>** для спецификации этой переменной в качестве входной или кнопку **>Output>** для спецификации этой переменной в качестве выходной. После нажатия одной из этих кнопок имя выделенной лингвистической переменной переносится в соответствующую секцию имен входных или выходных переменных справа.

Если оставить флажок **Create Rule Base** установленным, то мастер блока правил сгенерирует полное множество правил на основе комбинации всех термов лингвистических переменных этого блока правил. В противном случае для создаваемого блока правил будет определено пустое множество правил нечетких продукций.

После спецификации входных и выходных лингвистических переменных создаваемого блока правил в первом диалоговом окне мастера с помощью нажатия кнопки **Next** можно перейти ко второму диалоговому окну данного мастера (рис. 19.22).

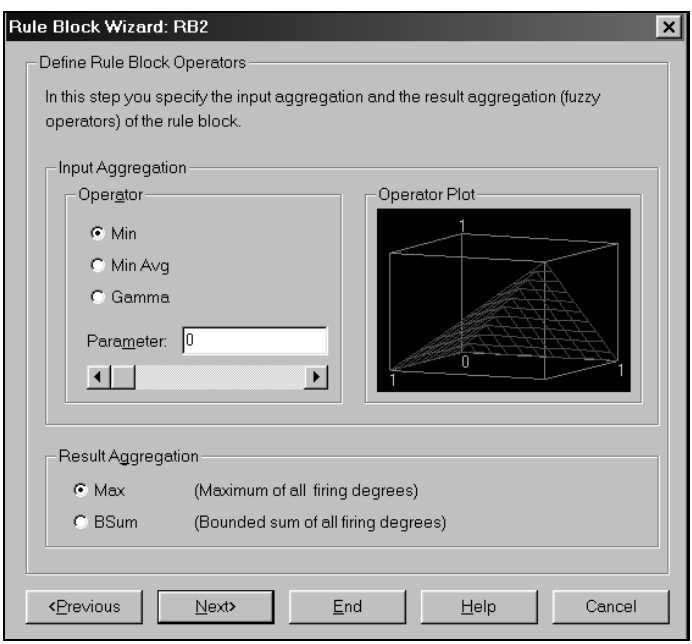


Рис. 19.22. Второе диалоговое окно мастера блока правил

Во втором диалоговом окне разработчику предлагается специфицировать операции агрегирования подусловий для всех правил создаваемого блока правил.

В первом случае **(Min)** для операции агрегирования используется формула (18.1). Если при этом параметр компенсации λ=0, то подусловия соединяются с помощью нечеткой операции И, если же параметр компенсации λ=1, то подусловия соединяются с помощью нечеткой операции ИЛИ.

Во втором случае **(Min Avg)** для операции агрегирования используется формула (18.2). Если при этом параметр компенсации λ=1, то для вычисления результата агрегирования подусловий используется операция арифметического среднего.

Наконец, в третьем случае **(Gamma)** для операции агрегирования используется формула (18.3). Если при этом параметр компенсации γ=0, то для вычисления результата агрегирования подусловий используется операция алгебраического произведения. При этом можно изменять значения параметров компенсации λ и γ, для чего удобно воспользоваться расположенным ниже регулятором.

Контроль настройки операций агрегирования создаваемого блока правил осуществляется с помощью графиков соответствующих кривых операций в окне просмотра **(Operator Plot)**.

Нижняя часть этого диалогового окна предназначена для спецификации результата агрегирования **(Result Aggregation)** или композиции правил для получения окончательных значений функции принадлежности термов заключений. В первом случае **(Max)** для операции композиции используется операция максимума, а во втором случае **(BSum)** — операция ограниченной суммы.

После этого с помощью нажатия кнопки **Next** можно перейти к третьему диалоговому окну мастера лингвистической переменной (рис. 19.23) для задания количественных значений степени влияния отдельных входных лингвистических переменных на выходные лингвистические переменные.

В диалоговом окне спецификации степени влияния каждой из входных лингвистических переменных на выходную переменную блока правил разработчику предлагается выбор из следующего множества значений: **Very Negative**, **Negative**, **Not at All**, **Positive**, **Very Positive**. При необходимости можно вовсе исключить влияние указанной входной лингвистической переменной на выходную, сбросив флажок **<in1> has an influence on <out1>**.

После спецификации степени влияния всех входных переменных на выходные переменные создаваемого блока правил с помощью нажатия кнопки **Next** можно перейти к последнему диалоговому окну мастера блока правил (рис. 19.24).

В этом диалоговом окне разработчику предлагается ввести текстовый комментарий **(Comment for Rule Block)** для вновь создаваемого блока правил. При этом также возможен ввод символов кириллицы, которые корректно отображаются аналогично тексту на английском в форме всплывающей подсказки при выделении соответствующего блока правил в окне просмотра структуры проекта после добавления этого блока правил к разрабатываемому проекту.

После спецификации всех свойств нового блока правил нечетких продукций с помощью нажатия кнопки **Next** или **End** будет создан новый блок правил и добавлен к системе нечеткого вывода в окне редактора проекта (рис. 19.25).

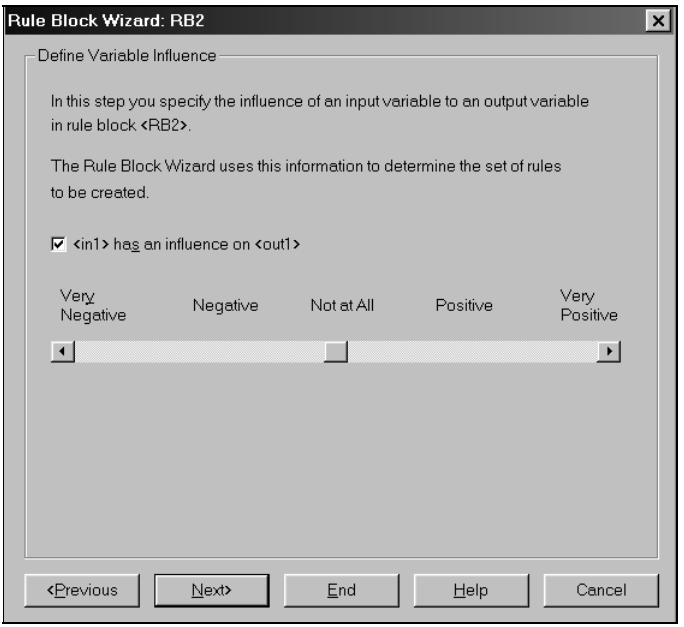


Рис. 19.23. Диалоговое окно спецификации степени влияния каждой из входных лингвистических переменных на выходную переменную блока правил

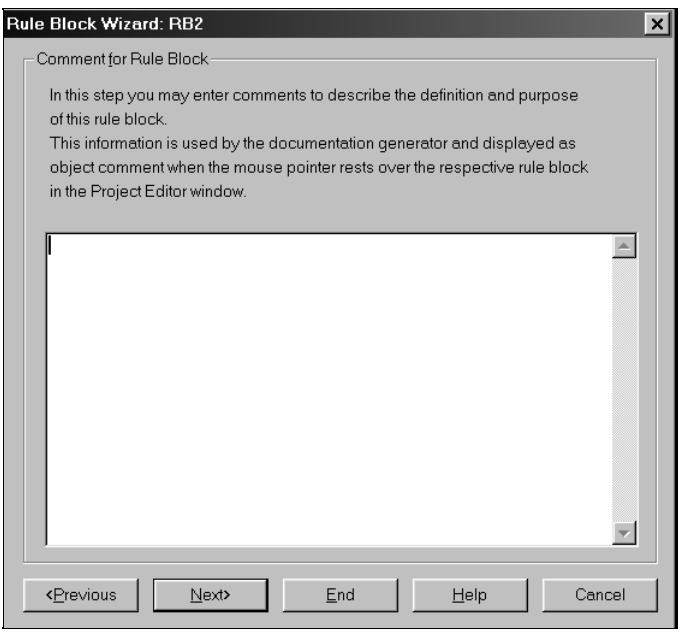


Рис. 19.24. Последнее диалоговое окно мастера блока правил

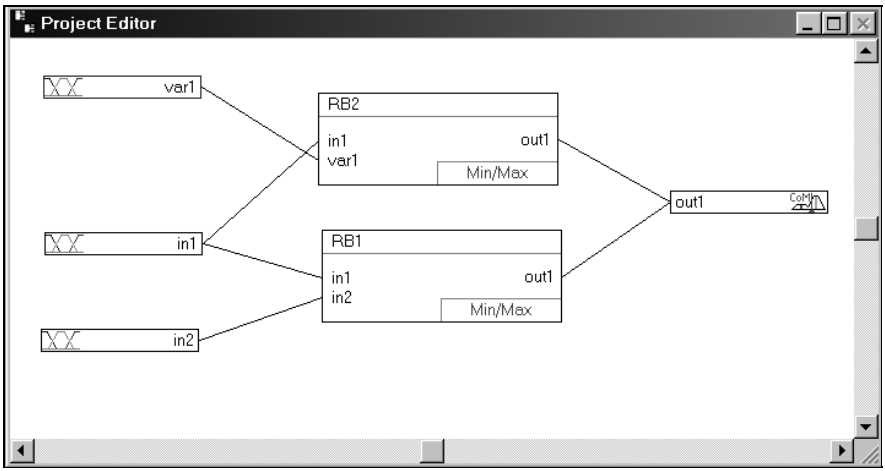


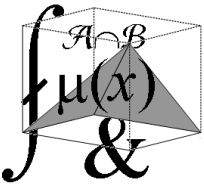
Рис. 19.25. Графический вид редактора проекта с добавленным к нему новым блоком правил

В завершение рассмотрения особенностей мастера блока правил следует заметить, что нажатие кнопки **End** в любом из диалоговых окон этого мастера закрывает данный мастер и завершает процесс создания нового блока правил. При этом все свойства на оставшихся не рассмотренными диалоговых окнах мастера для нового блока правил принимаются равными по умолчанию, предложенными программой fuzzyTECH.

В свою очередь нажатие кнопки **Previous** в любом из диалоговых окон (кроме первого) возвращает к предыдущему диалоговому окну данного мастера. Нажатие кнопки **Help** в любом из диалоговых окон открывает раздел справочной системы, посвященный соответствующему диалоговому окну, а нажатие кнопки **Cancel** закрывает мастер блока правил.

Примеры использования графических средств создания, редактирования и анализа разработанных проектов систем нечеткого вывода в среде fuzzyTECH для некоторых задач управления и принятия решений приводятся в *главе 20*.

Глава 20



Примеры разработки и анализа нечетких моделей в среде fuzzyTECH

В настоящей главе рассматриваются примеры нечетких моделей, которые иллюстрируют особенности практического применения различных инструментов программы fuzzyTECH.

Первый из примеров нечетких моделей, известный под именем "*Чаевые в ресторане*", стал уже в некоторой степени классическим, поскольку довольно часто используется в литературе для иллюстрации общих свойств систем нечеткого вывода. В дополнение к этому построенная средствами fuzzyTECH соответствующая нечеткая модель может быть использована для сравнительного анализа возможностей системы MATLAB и программы fuzzyTECH.

Второй прием также традиционно используется для демонстрации особенностей задач нечеткого управления. Речь идет о построении в среде fuzzyTECH нечеткой модели управления контейнерным краном. Соответствующая нечеткая модель также может быть использована для сравнительного анализа возможностей системы MATLAB и программы fuzzyTECH.

Наконец, третий пример представляет собой оригинальную нечеткую модель, разработанную компанией INFORM GmbH для решения проблемы оценивания финансовой состоятельности клиентов со стороны банков при выдаче долгосрочных кредитов на строительство недвижимости под залог. Содержательная постановка и нечеткая модель этой задачи для среды MATLAB были рассмотрены ранее.

20.1. Пример разработки системы нечеткого вывода для задачи "*Чаевые в ресторане*"

В качестве первого примера разработки системы нечеткого вывода в интерактивном режиме с помощью графических средств программы fuzzyTECH рас-

смотрим построение нечеткой модели для примера 12.1 "*Чаевые в ресторане*". Выбор примера сделан исключительно из методических соображений с целью сравнения возможностей программных сред нечеткого моделирования fuzzyTECH и MATLAB, поскольку соответствующая нечеткая модель входит в число демонстрационных примеров системы MATLAB. Содержательная постановка данной задачи и соответствующая нечеткая модель были рассмотрены в *главе 12*.

В качестве входных переменных системы нечеткого вывода будем рассматривать 2 нечеткие лингвистические переменные: "*Service*" (качество обслуживания) и "*Food*" (качество приготовления заказанных блюд или сокращенно — качество ужина), а в качестве выходной переменной — нечеткую лингвистическую переменную "*Tip*" (величина чаевых).

Примечание

Поскольку программа fuzzyTECH не позволяет задавать русскоязычные имена компонентам нечеткой модели, здесь используются соответствующие имена из оригинального демонстрационного примера системы MATLAB.

В качестве терм-множества первой лингвистической переменной "*Service*" используется множество T_1={"*poor*" (плохое), "*good*" (хорошее), "*excellent*" (отличное)}, а в качестве терм-множества второй лингвистической переменной "*Food*" используется множество T_2={"*rancid*" (подгоревший), "*delicious*" (превосходный)}. В качестве терм-множества выходной лингвистической переменной "*Tip*" используется множество T_3={"*cheap*" (малые), "*average*" (средние), "*generous*" (щедрые)}.

Как и ранее, каждый из термов первой и второй входной переменной (качество обслуживания и приготовления заказанных блюд) будем оценивать по 10-балльной порядковой шкале, при которой цифре 0 соответствует наихудшая оценка, а цифре 10 — наилучшая оценка. Что касается термов выходной переменной, то как и ранее будем предполагать, что малые чаевые составляют около 5% от стоимости заказанных блюд, средние чаевые — около 15%, а щедрые чаевые — около 25%.

Субъективная информация о величине чаевых представляется в форме следующих трех правил нечетких продукций (система нечеткого вывода типа Мамдани):

ПРАВИЛО_1: IF ("*Service* is *poor*") OR ("*Food* is *rancid*") THEN ("*Tip* is *cheap*")

ПРАВИЛО_2: IF ("*Service* is *good*") THEN ("*Tip* is *average*")

ПРАВИЛО_3: IF ("*Service* is *excellent*") OR ("*Food* is *delicious*") THEN ("*Tip* is *generous*")

Процесс разработки системы нечеткого вывода в среде fuzzyTECH для примера "*Чаевые в ресторане*" состоит в выполнении следующей последовательности действий:

1. Откроем мастер нечеткого проекта FDW (Fuzzy Design Wizard). Для этого можно воспользоваться специальной кнопкой на панели инструментов программы fuzzyTECH (см. табл. 18.3). В первом окне мастера нечеткого проекта

согласимся с предложенным по умолчанию вариантом создания новой системы нечеткого вывода **(Create New System)** и перейдем ко второму диалоговому окну мастера.

2. Во втором диалоговом окне мастера проекта зададим количество входных **(Input LVs** = 2), выходных **(Output LVs** = 1) и промежуточных **(Intermediate LVs** = 0) лингвистических переменных проекта, а также общее количество термов входных **(Input Terms/LV** = 3), выходных **(Output Terms/LV** = 3) и промежуточных **(Interm. Terms/LV** = 0) переменных (рис. 20.1). После этого перейдем к следующему диалоговому окну.

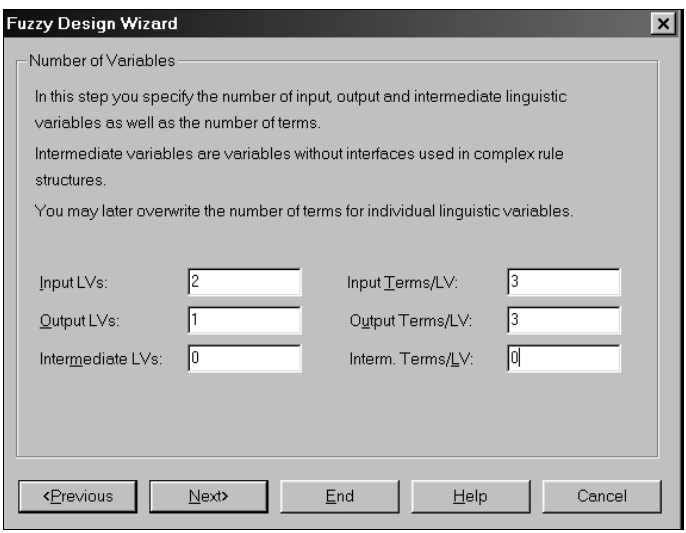


Рис. 20.1. Второе окно мастера нечеткого проекта

3. В следующих диалоговых окнах мастера нечеткого проекта изменим имена входных и выходной лингвистических переменных, предлагаемые программой fuzzyTECH по умолчанию, и диапазоны изменения их значений. Результат изменения имени первой входной лингвистической переменной и диапазона ее значений для создаваемого проекта изображен на рис. 20.2.

4. В последующих диалоговых окнах оставим без изменения свойства, предложенные программой fuzzyTECH по умолчанию: метод дефаззификации — центр максимума **(Best Compromize**, CoM) и количество блоков правил, равное 1. После окончания работы с мастером нечеткого проекта будет создан прототип системы нечеткого вывода. Структура системы нечеткого вывода с измененными значениями текстовых блоков изображена на рис. 20.3.

5. Далее необходимо более точно специфицировать термы и их функции принадлежности для входных и выходной лингвистических переменных разрабатываемой системы нечеткого вывода, для чего следует воспользоваться редактором лингвистической переменной. Для первой входной лингвистической

переменной изменим имена заданных ранее по умолчанию термов. Новые имена термов вводятся в окне свойств терма, которое может быть открыто с помощью соответствующей операции контекстного меню.

Вид редактора лингвистической переменной в процессе изменения имени первого терма для первой из входных переменных изображен на рис. 20.4.

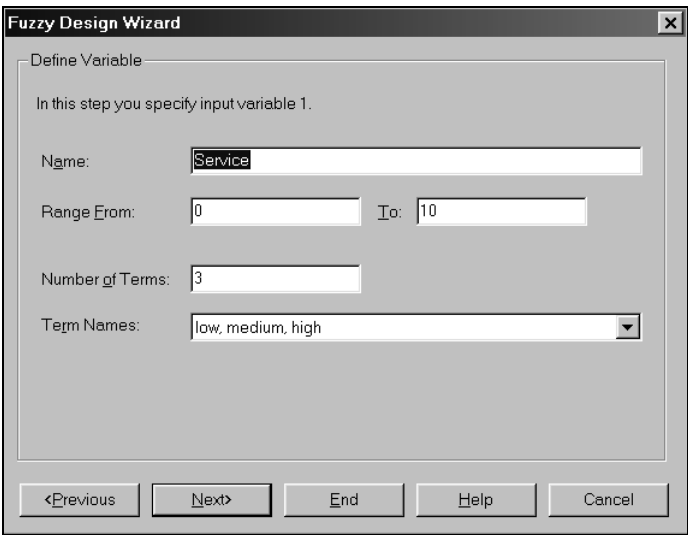


Рис. 20.2. Вид диалогового окна мастера нечеткого проекта для первой из входных лингвистических переменных

Рис. 20.3. Вид редактора проекта после создания прототипа системы нечеткого вывода с помощью мастера нечеткого проекта

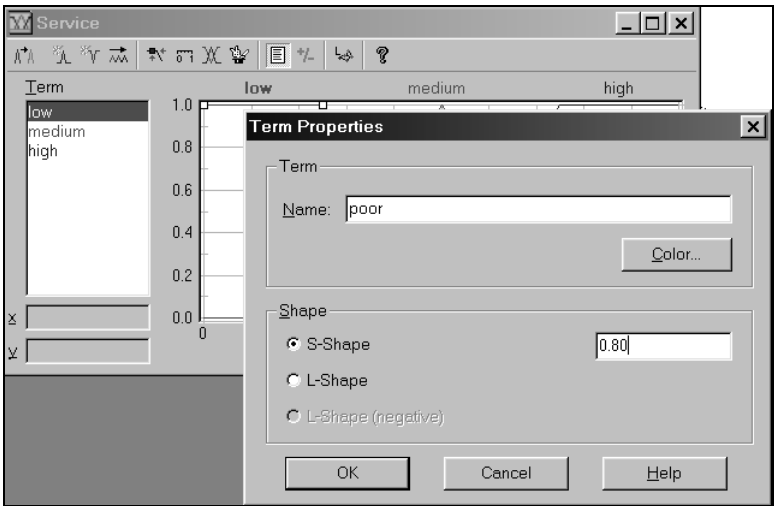


Рис. 20.4. Окно свойств термов первой из входных лингвистических переменных

6. Далее изменим форму и значения функций принадлежности термов первой входной лингвистической переменной. Для этого следует выбрать *S*-образную форму кривых в окне свойств каждого терма и задать коэффициент кривизны кривой равным 0.8.

   Вид редактора функций принадлежности после внесенных изменений для первой входной лингвистической переменной изображен на рис. 20.5.

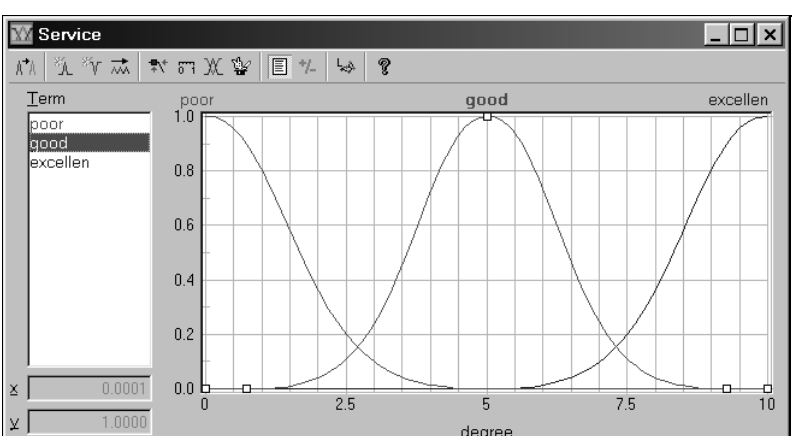


Рис. 20.5. Вид редактора лингвистической переменной после изменения названия термов и типа их функций принадлежности для первой входной переменной "*Service*"

7. Аналогичным образом изменим названия термов второй входной переменной "*Food*" и удалим один из термов с соответствующей функцией принадлежности. Для удаления терма следует выделить удаляемую функцию принадлежности и нажать клавишу <Delete> на клавиатуре. Вид редактора лингвистической переменной после внесенных изменений для второй входной лингвистической переменной изображен на рис. 20.6.

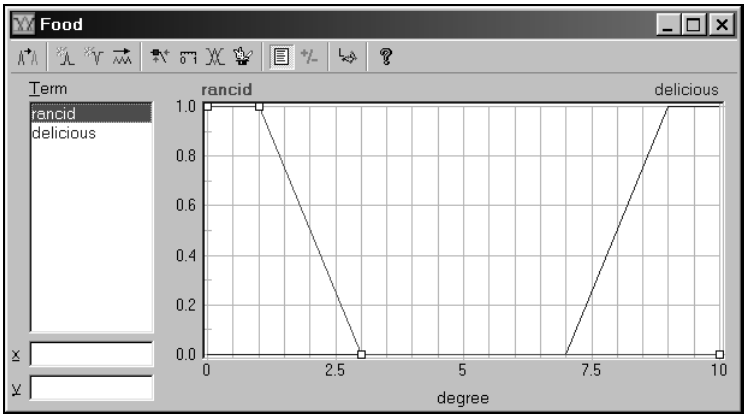


Рис. 20.6. Вид редактора лингвистических переменных после изменения названия термов и типа их функций принадлежности для второй входной переменной "*Food*"

8. Наконец, изменим названия термов и вид функций принадлежности для выходной переменной "*Tip*", оставив без изменения треугольный тип функций принадлежности, предложенный программой fuzzyTECH. Вид редактора лингвистической переменной после сделанных изменений для выходной переменной "*Tip*" изображен на рис. 20.7.
9. Теперь можно определить базу правил для разрабатываемой системы нечеткого вывода. Для этой цели можно воспользоваться табличным редактором блока правил. Поскольку первоначально база правил нечеткого вывода пуста, то после вызова табличного редактора правил центральное окно этого редактора не содержит никаких правил. Для их определения следует использовать кнопку **Full Rule Block** панели инструментов, после нажатия на которую будет сгенерирована база из 18 правил нечетких продукций. Удалим избыточные правила и один из термов подусловий для лингвистической переменной "*Food*".

   Вид табличного редактора блока правил после его определения для разрабатываемой системы нечеткого вывода изображен на рис. 20.8.
10. Поскольку процесс нечеткого моделирования предполагает анализ результатов нечеткого вывода при различных значениях входных переменных с целью установления адекватности разработанной нечеткой модели, перейдем в

интерактивный режим отладки. После появления графического окна режима отладки откроем также окна редакторов всех лингвистических переменных проекта и графическое окно анализатора правил.

Внешний вид рабочего интерфейса программы fuzzyTECH в режиме отладки проекта "*Чаевые в ресторане*" изображен на рис. 20.9.

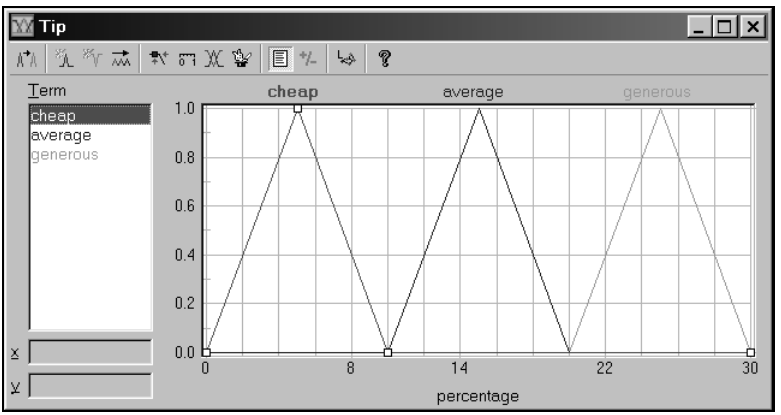


Рис. 20.7. Вид редактора лингвистической переменной после изменения названия термов и типа их функций принадлежности для выходной переменной "*Tip*"

Spreadsheet Rule Editor - RB1

#	IF		THEN	
	Food	Service	DoS	Tip
1	rancid	poor	1.00	cheap
2		good	1.00	average
3	delicious	excellent	1.00	generous
4				
5				
6				

Рис. 20.8. Вид редактора правил нечетких продукций после внесения изменений в блок правил проекта

Для окончательного анализа разработанной нечеткой модели можно воспользоваться графическим окном просмотра поверхности нечеткого вывода на плоскости (рис. 20.10, *а*) и графическим окном просмотра трехмерной поверхности нечеткого вывода (рис. 20.10, *б*).

Эти графические окна могут служить для общего анализа адекватности нечеткой модели, позволяя оценить влияние изменения значений входных нечетких переменных на значение одной из выходных нечетких переменных.

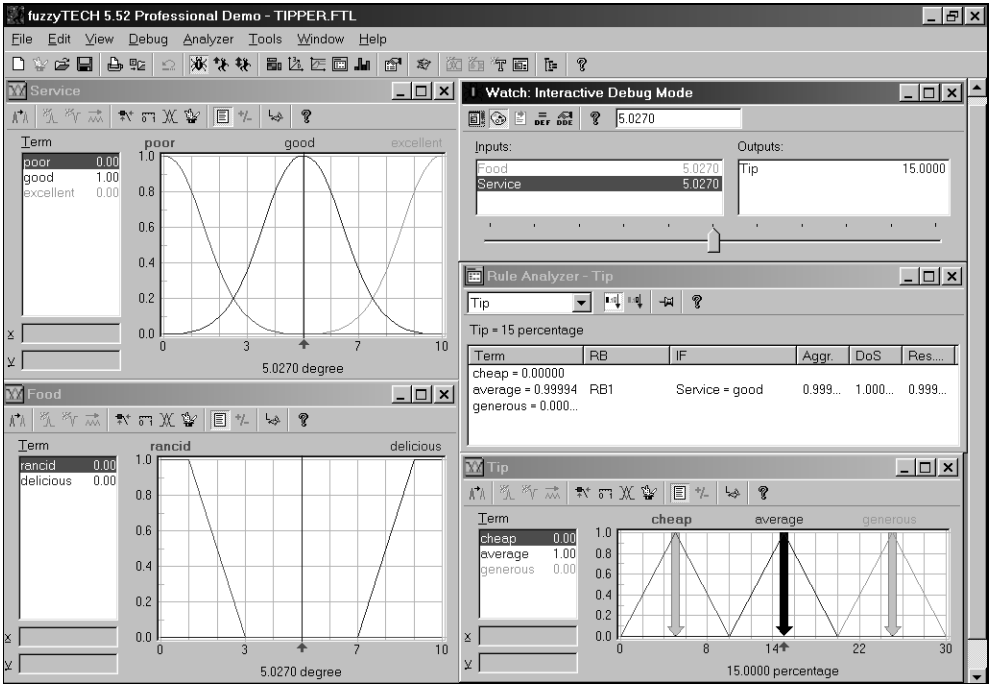


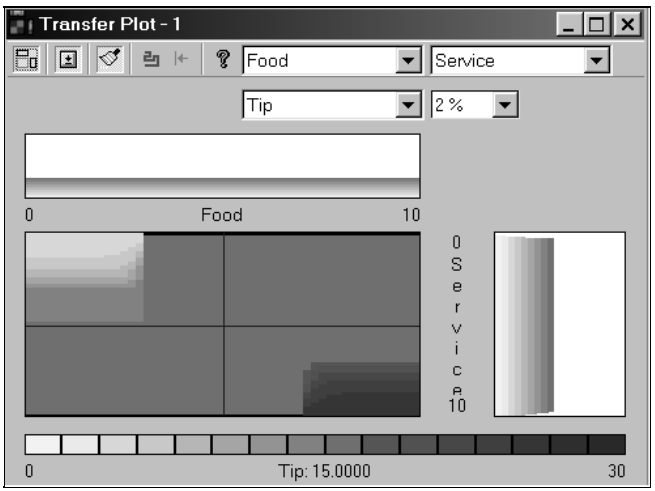
Рис. 20.9. Вид рабочего интерфейса программы fuzzyTECH в режиме отладки проекта "*Чаевые в ресторане*"

Изображенные графики поверхности нечеткого вывода и эксперименты с разработанной нечеткой моделью показывают, что ее точность оставляет желать лучшего. Действительно, существуют такие области значений входных лингвистических переменных, в которых малые изменения значений входных переменных модели приводят к большим (скачкообразным) изменениям значений выходной лингвистической переменной. Указанный факт свидетельствует о недостаточной устойчивости (робастности) разработанной нечеткой модели. Одним из путей повышения устойчивости и адекватности модели в целом может быть увеличение базы правил нечеткого вывода и уточнение функций принадлежности отдельных термов, что предлагается выполнить читателям самостоятельно в качестве упражнения.

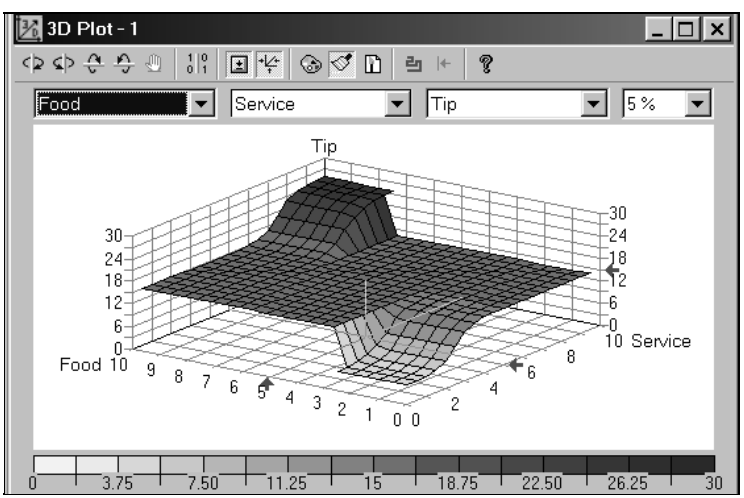
Заканчивая рассмотрение процесса разработки простейшей нечеткой модели для примера 12.1 "*Чаевые в ресторане*", следует отметить, что в демонстрационной версии невозможно сохранить разработанный проект во внешнем файле на диске. Однако тот факт, что программа fuzzyTECH сохраняет нечеткие проекты в обычных текстовых файлах формата FTL, подсказывает идею, как можно сохранить нечеткий проект, разработанный в среде fuzzyTECH.

Идея заключается в том, чтобы разработку прототипа системы нечеткого вывода проводить в форме создания соответствующего текстового файла нечеткого проекта. Для этого можно воспользоваться одним из текстовых ASCII-

редакторов. Что касается языка формата FTL, используемого для задания параметров нечеткого проекта в среде fuzzyTECH, то этот язык очень похож на язык нечеткого управления FCL *(см. главу 7)*.



а



б

Рис. 20.10. Вид поверхности нечеткого вывода на плоскости (*а*) трехмерной поверхности (*б*) для разработанной нечеткой модели в интерактивном режиме отладки

Примечание

В случае примера "*Чаевые в ресторане*" текст соответствующего файла проекта системы нечеткого вывода в формате FTL приводится в *приложении 4*.

Этот файл можно загрузить в демонстрационную версию программы fuzzyTECH и использовать для выполнения всех необходимых расчетов. Аналогичным образом можно поступить при решении других задач нечеткого моделирования, однако трудоемкость процесса разработки нечетких проектов в этом случае существенно возрастает.

20.2. Нечеткая модель управления контейнерным краном

Напомним, что эта задача связана с процессом управления контейнерным краном, который используется для транспортировки моноблочных контейнеров при выполнении разгрузочных работ морских судов в порту. Суть задачи состоит в том, чтобы разработать модель, позволяющую автоматически управлять процессом горизонтального перемещения контейнерного крана, исключив или добившись минимального раскачивания транспортируемых контейнеров при подобном перемещении.

Содержательная постановка этой задачи была рассмотрена в *разд. 7.4*, а нечеткая модель для среды MATLAB была построена в *разд. 16.2*.

Для анализа этой модели в среде fuzzyTECH целесообразно воспользоваться соответствующим примером, поставляемым вместе с демонстрационной версией программы. С этой целью в редактор проекта необходимо загрузить файл соответствующего проекта: Crane.ftl, который расположен в папке C:\Program Files\ fuzzyTECH 5.5\Dde\VB5 (если программа fuzzyTECH установлена на диске C:).

Вид графического окна редактора проекта для этой нечеткой модели изображен на рис. 20.11.

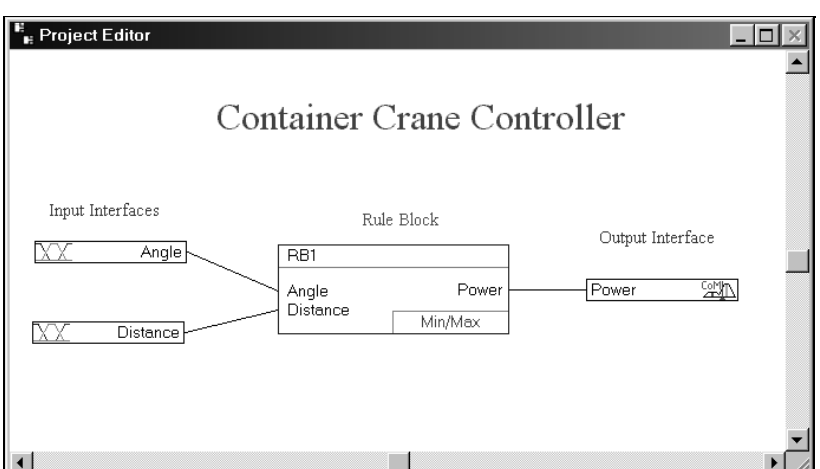


Рис. 20.11. Графическое окно редактора проекта для системы нечеткого вывода управления контейнерным краном в порту

В качестве первой входной лингвистической переменной данной нечеткой модели используется лингвистическая переменная "*Angle*" (угол), в качестве второй входной лингвистической переменной используется лингвистическая переменная "*Distance*" (расстояние). В качестве выходной лингвистической переменной данной нечеткой модели используется лингвистическая переменная "*Power*" (мощность).

В качестве терм-множества первой лингвистической переменной используется множество T_1= {"*neg_big*", "*neg_small*", "*zero*", "*pos_small*" , "*pos_big*"}. Графики функций принадлежности термов из T_1 изображены на рис. 20.12, *а*. В качестве терм-множества второй лингвистической переменной используется множество T_2= {"*neg_close*", "*zero*", "*close*", "*medium*" , "*far*"}. Графики функций принадлежности термов из T_2 изображены на рис. 20.12, *б*.

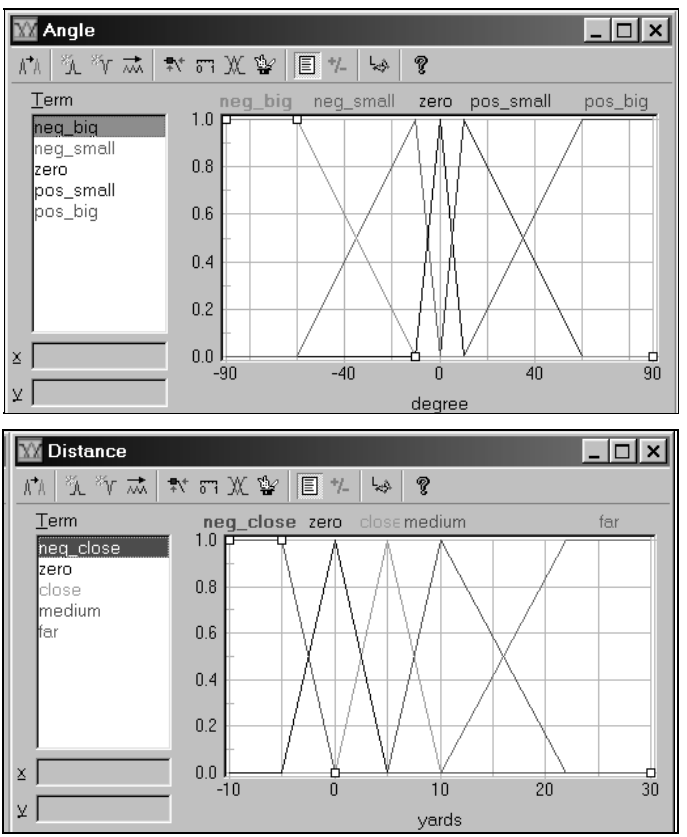


Рис. 20.12. Вид редактора лингвистических переменных для первой входной переменной "*Angle*" и второй входной переменной "*Distance*"

В качестве терм-множества выходной лингвистической переменной используется множество T_3={"*neg_high*", "*neg_medium*", "*zero*", "*pos_medium*" , "*pos_high*"}. Графики функций принадлежности термов из T_3 изображены на рис. 20.13.

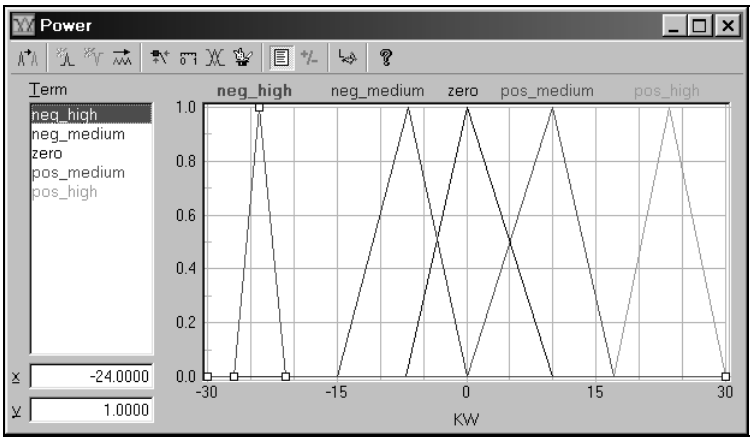


Рис. 20.13. Вид редактора лингвистических переменных для выходной переменной "*Power*"

Блок правил рассматриваемой системы нечеткого вывода содержит 9 правил нечетких продукций. Вид графического окна табличного редактора блока правил системы нечеткого вывода управления контейнерным краном изображен на рис. 20.14.

Spreadsheet Rule Editor - RB1

#	IF		THEN	
	Angle	Distance	DoS	Power
1	pos_small	zero	1.00	neg_medium
2	zero	zero	1.00	zero
3	pos_small	close	1.00	neg_medium
4	zero	close	1.00	zero
5	neg_small	close	1.00	pos_medium
6	neg_small	medium	1.00	pos_high
7	neg_big	medium	1.00	pos_medium
8	zero	far	1.00	pos_medium
9	neg_small	far	1.00	pos_high
10				

Рис. 20.14. Графический интерфейс редактора блока правил системы нечеткого вывода управления контейнерным краном

Теперь можно выполнить анализ данной системы нечеткого вывода для задачи управления контейнерным краном в порту. С этой целью перейдем в интерактивный режим отладки и дополнительно откроем окно анализатора правил. Внешний вид рабочего интерфейса программы fuzzyTECH в режиме отладки проекта "*Управление контейнерным краном*" изображен на рис. 20.15.

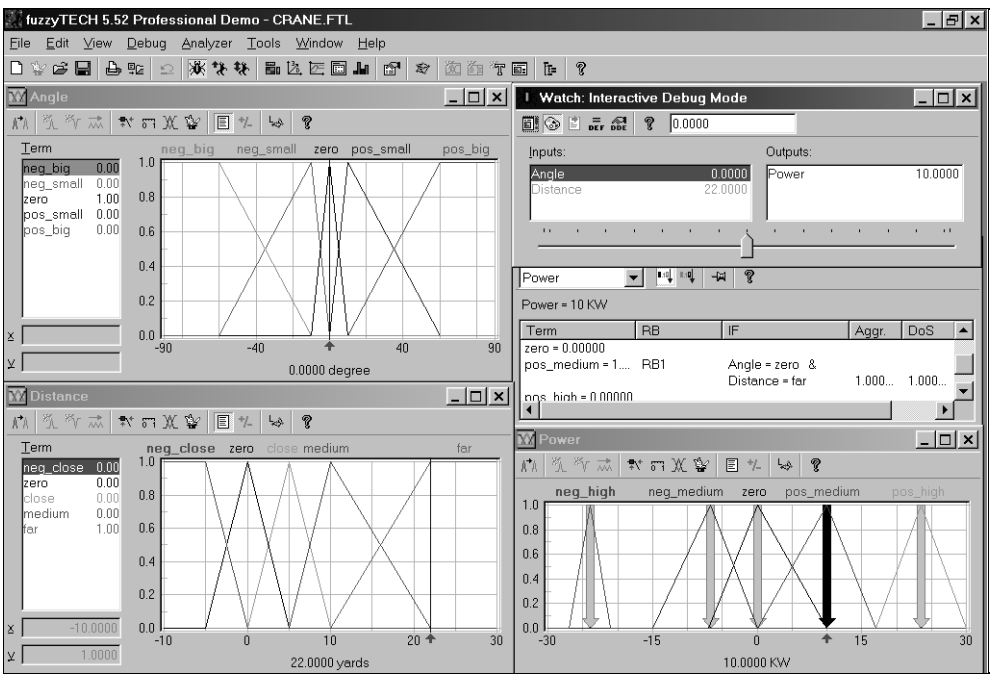


Рис. 20.15. Вид рабочего интерфейса программы fuzzyTECH в режиме отладки проекта "*Управление контейнерным краном*"

Для общего анализа рассматриваемой нечеткой модели можно воспользоваться графическим окном просмотра поверхности нечеткого вывода на плоскости (рис. 20.16, *а*) и графическим окном просмотра трехмерной поверхности нечеткого вывода (рис. 20.16, *б*).

Данная поверхность нечеткого вывода позволяет установить зависимость значений выходной переменной от значений входных переменных нечеткой модели системы управления краном. Эта зависимость может послужить основой для программирования контроллера или аппаратной реализации соответствующего нечеткого алгоритма управления в форме соответствующей таблицы решений.

Сравнительный анализ нечетких моделей управления контейнерным краном, построенных в средах MATLAB *(см. главу 16)* и fuzzyTECH, показывает, что увеличение правил нечетких продукций с 6 до 9 не приводит к заметному увели-

чению адекватности модели. С другой стороны, наличие разрывов на поверхности нечеткого вывода (рис. 20.16, *б*) может служить причиной неустойчивости данной нечеткой модели fuzzyTECH, на что необходимо обратить внимание пользователей и разработчиков соответствующих систем управления.

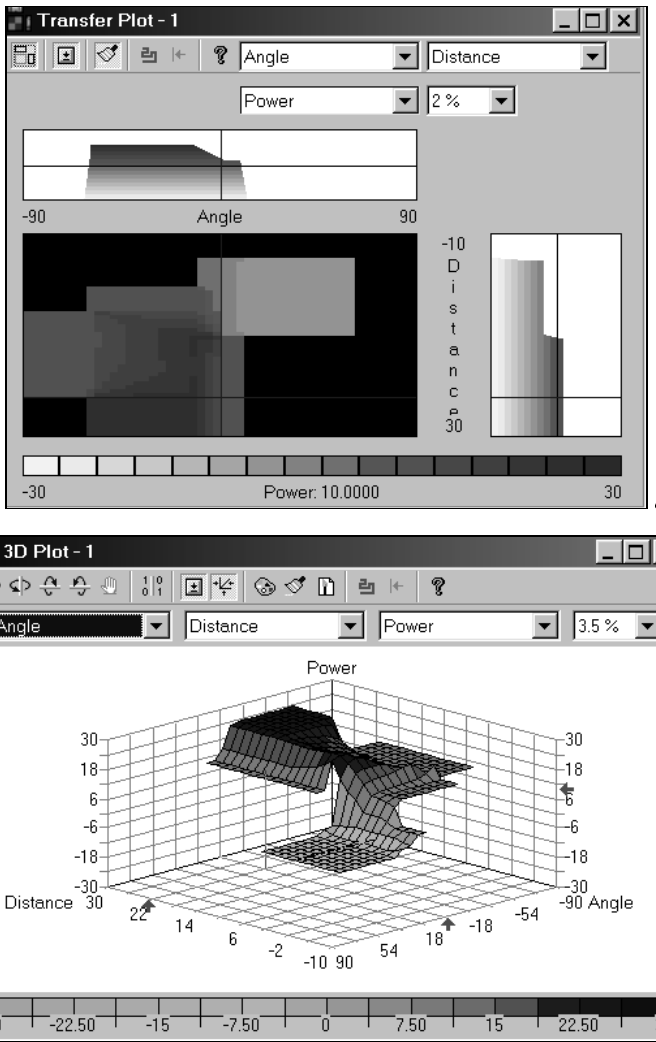


Рис. 20.16. Вид поверхности нечеткого вывода на плоскости (*а*) трехмерной поверхности (*б*) в интерактивном режиме отладки для разработанной нечеткой модели "*Управление контейнерным краном*"

20.3. Нечеткая модель оценивания финансовой состоятельности клиентов при предоставлении банковских кредитов

Данная задача была сформулирована и решена в рамках исследования, выполненного фирмой INFORM GmbH. Содержательная постановка этой задачи была рассмотрена ранее в *разд. 17.1*. Там же была построена модель соответствующей системы нечеткого вывода в среде MATLAB. Аналогичная нечеткая модель в среде fuzzyTECH может служить примером использования нескольких блоков правил и промежуточных лингвистических переменных. В то же время непосредственная реализация нечетких моделей с несколькими блоками правил в среде MATLAB без использования средств пакета моделирования Simulink не представляется возможной.

Для анализа этой модели в среде fuzzyTECH целесообразно воспользоваться соответствующим оригинальным примером, поставляемым вместе с демонстрационной версией программы. С этой целью в редактор проекта необходимо загрузить файл соответствующего проекта: Realest.ftl, который расположен в папке C:\Program Files\fuzzyTECH 5.5\SAMPLES\BUSINESS\REALEST (если программа fuzzyTECH установлена на диске C:).

Вид графического окна редактора проекта для этой нечеткой модели изображен на рис. 20.17.

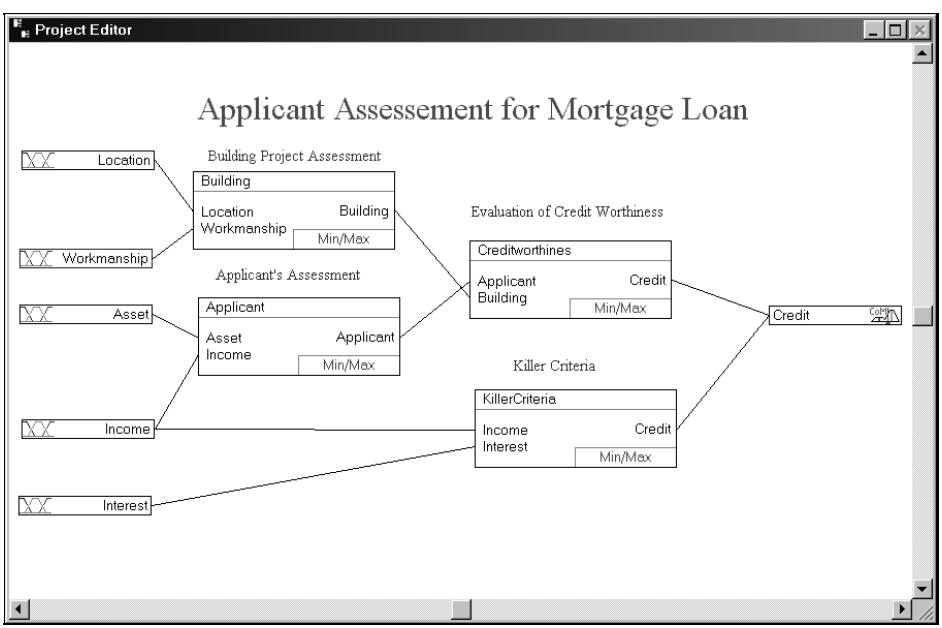


Рис. 20.17. Графическое окно редактора проекта для системы нечеткого вывода принятия решений по предоставлению кредитов

В качестве первой входной лингвистической переменной данной нечеткой модели используется лингвистическая переменная "*Location*" (местоположение), в качестве второй входной лингвистической переменной используется лингвистическая переменная "*Workmanship*" (отделка). В качестве третьей входной лингвистической переменной используется лингвистическая переменная "*Asset*" (активы), в качестве четвертой входной лингвистической переменной используется лингвистическая переменная "*Income*" (доход) и, наконец, в качестве пятой входной лингвистической переменной используется лингвистическая переменная "*Interest*" (выплаты).

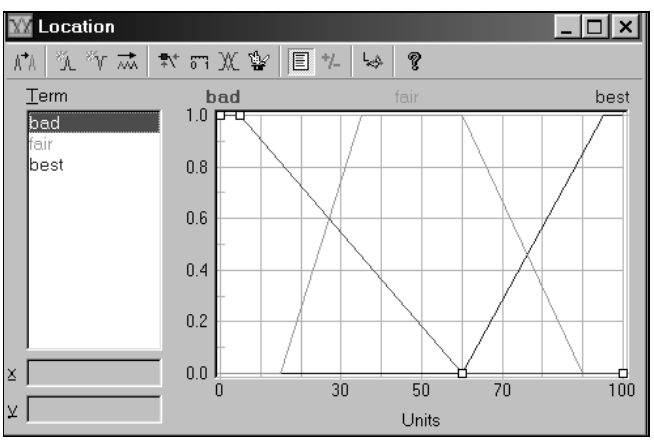
В качестве выходной лингвистической переменной данной нечеткой модели используется лингвистическая переменная "*Credit*" (кредитоспособность).

В качестве терм-множества первой входной лингвистической переменной "*Location*" используется множество T_1={"*bad*", "*fair*", "*best*"} с функциями принадлежности термов, изображенными на рис. 20.18, *а*.

В качестве терм-множества второй входной лингвистической переменной "*Workmanship*" используется множество T_2={"*lousy*", "*fair*", "*pretty_good*"} с функциями принадлежности термов, изображенными на рис. 20.18, *б*.

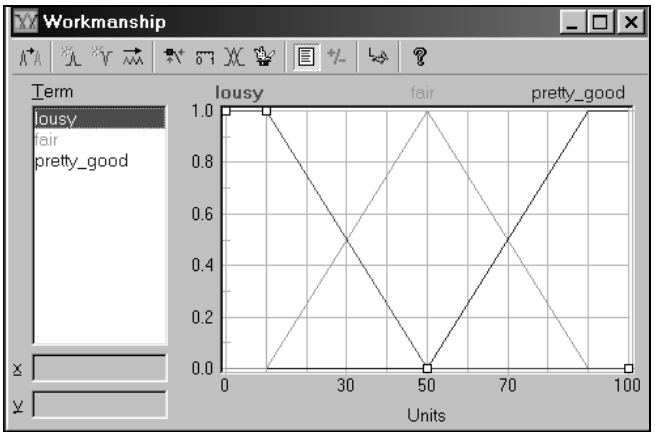
В качестве терм-множества третьей входной лингвистической переменной "*Asset*" используется множество T_3={"*low*", "*medium*", "*high*"} с функциями принадлежности термов, изображенными на рис. 20.19, *а*.

В качестве терм-множества четвертой входной лингвистической переменной "*Income*" используется множество T_4={"*low*", "*medium*", "*high*"} с функциями принадлежности термов, изображенными на рис. 20.19, *б*.



а

Рис. 20.18. Графики функций принадлежности для термов лингвистической переменной "*Location*" (*а*)



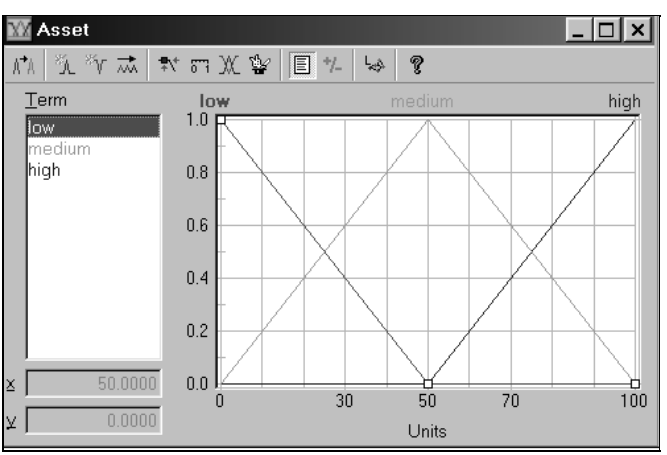
б

Рис. 20.18. Графики функций принадлежности для термов лингвистической переменной "*Workmanship*" (*б*)

В качестве терм-множества пятой входной лингвистической переменной "*Interest*" используется множество T_5={"*low*", "*medium*", "*high*"} с функциями принадлежности термов, изображенными на рис. 20.20, *а*.

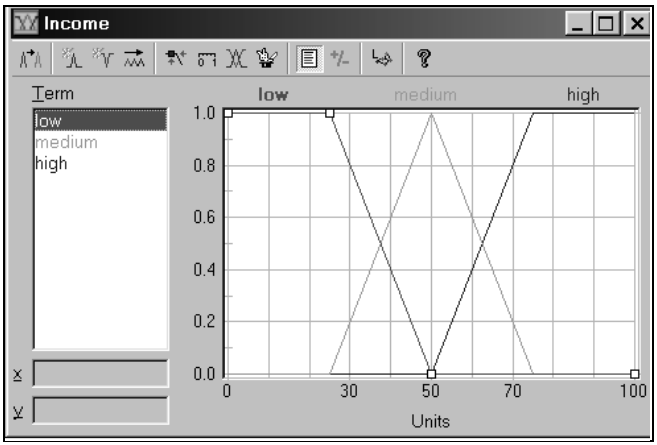
В качестве терм-множества выходной лингвистической переменной "*Credit*" используется множество T_6={"*very_low*", "*low*", "*medium*", "*high*", "*very_high*"} с функциями принадлежности термов, изображенными на рис. 20.20, *б*.

Рассматриваемая нечеткая модель содержит 4 блока правил, 2 из которых имеют промежуточные лингвистические переменные с именами соответствующих блоков правил.



а

Рис. 20.19. Графики функций принадлежности для термов лингвистической переменной "*Asset*" (*а*)

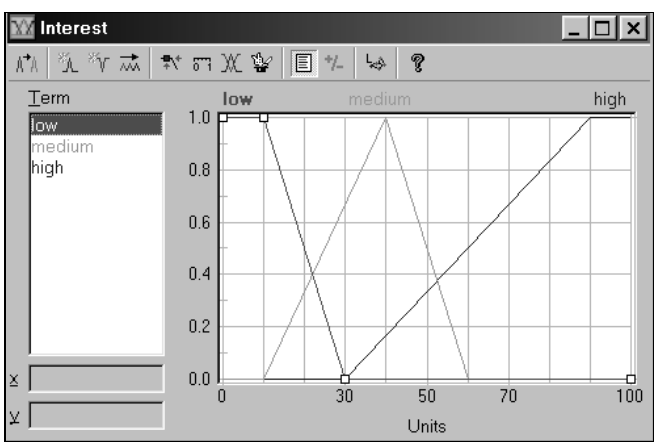


б

Рис. 20.19. Графики функций принадлежности для термов лингвистической переменной "*Income*" (*б*)

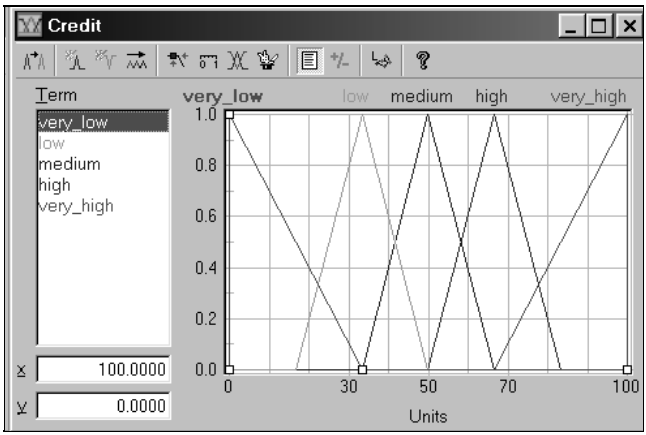
Первый блок правил с именем "*Building*" используется для промежуточной оценки общего качества местоположения и отделки и для рассматриваемой системы нечеткого вывода содержит 7 правил нечетких продукций. Входными лингвистическими переменными этого блока правил являются первые две входные лингвистические переменные проекта, а выходной лингвистической переменной этого блока правил является промежуточная переменная данного проекта с именем "*Building*".

Вид графического окна табличного редактора для этого блока правил изображен на рис. 20.21.



а

Рис. 20.20. Графики функций принадлежности для термов лингвистической переменной "*Interest*" (*а*)



б

Рис. 20.20. Графики функций принадлежности для термов лингвистической переменной "*Credit*" (*б*)

Spreadsheet Rule Editor - Building

#	IF		THEN	
	Location	Workmanship	DoS	Building
1		lousy	1.00	low
2	bad	fair	1.00	medium
3	bad	pretty_good	1.00	medium
4	fair	fair	1.00	medium
5	fair	pretty_good	1.00	high
6	best	fair	1.00	medium
7	best	pretty_good	1.00	high
8				

Рис. 20.21. Графическое окно табличного редактора первого блока правил с именем "*Building*"

Второй блок правил с именем "*Applicant's Assessment*" используется для промежуточной оценки общего количества активов и доходов и для рассматриваемой системы нечеткого вывода содержит 9 правил нечетких продукций. Входными лингвистическими переменными этого блока правил являются третья и четвертая входные лингвистические переменные проекта, а выходной лингвистической переменной этого блока правил является промежуточная переменная данного проекта с именем "*Applicant*".

Вид графического окна табличного редактора для этого блока правил изображен на рис. 20.22.

Третий блок правил с именем "*Evaluation of Credit Worthiness*" используется для частичной итоговой оценки кредитоспособности потенциальных клиентов и для

рассматриваемой системы нечеткого вывода содержит 6 правил нечетких продукций. Входными лингвистическими переменными этого блока правил являются промежуточные лингвистические переменные проекта, а выходной переменной этого блока правил — выходная лингвистическая переменная проекта с именем "*Credit*".

Spreadsheet Rule Editor - Applicant

#	IF		THEN	
	Asset	Income	DoS	Applicant
1	low	low	1.00	low
2	low	medium	1.00	low
3	low	high	1.00	medium
4	medium	low	1.00	low
5	medium	medium	1.00	medium
6	medium	high	1.00	high
7	high	low	1.00	medium
8	high	medium	1.00	medium
9	high	high	1.00	high
10				

Рис. 20.22. Графическое окно табличного редактора второго блока правил с именем "*Applicant*"

Вид графического окна табличного редактора для этого блока правил изображен на рис. 20.23.

Spreadsheet Rule Editor - Creditworthines

#	IF		THEN	
	Applicant	Building	DoS	Credit
1	low		1.00	very_low
2	medium	low	1.00	very_low
3	medium	medium	1.00	medium
4		high	1.00	high
5	high	high	1.00	very_high
6	high	medium	1.00	high
7				

Рис. 20.23. Графическое окно табличного редактора третьего блока правил с именем "*Creditworthiness*"

Наконец, четвертый блок правил с именем "*Killer Criteria*" используется для отрицательной оценки кредитоспособности потенциальных клиентов, когда выдача кредита клиенту практически исключается. Для рассматриваемой системы

нечеткого вывода этот блок правил содержит 3 правила нечетких продукций. Входными лингвистическими переменными этого блока правил являются промежуточные лингвистические переменные проекта, а выходной переменной этого блока правил — выходная лингвистическая переменная проекта с именем "*Credit*".

Вид графического окна табличного редактора для этого блока правил изображен на рис. 20.24.

Spreadsheet Rule Editor - KillerCriteria

#	IF		THEN	
	Income	Interest	DoS	Credit
1	low	medium	1.00	very_low
2	low	high	1.00	very_low
3	medium	high	1.00	very_low
4				

Рис. 20.24. Графическое окно табличного редактора четвертого блока правил с именем "*KillerCriteria*"

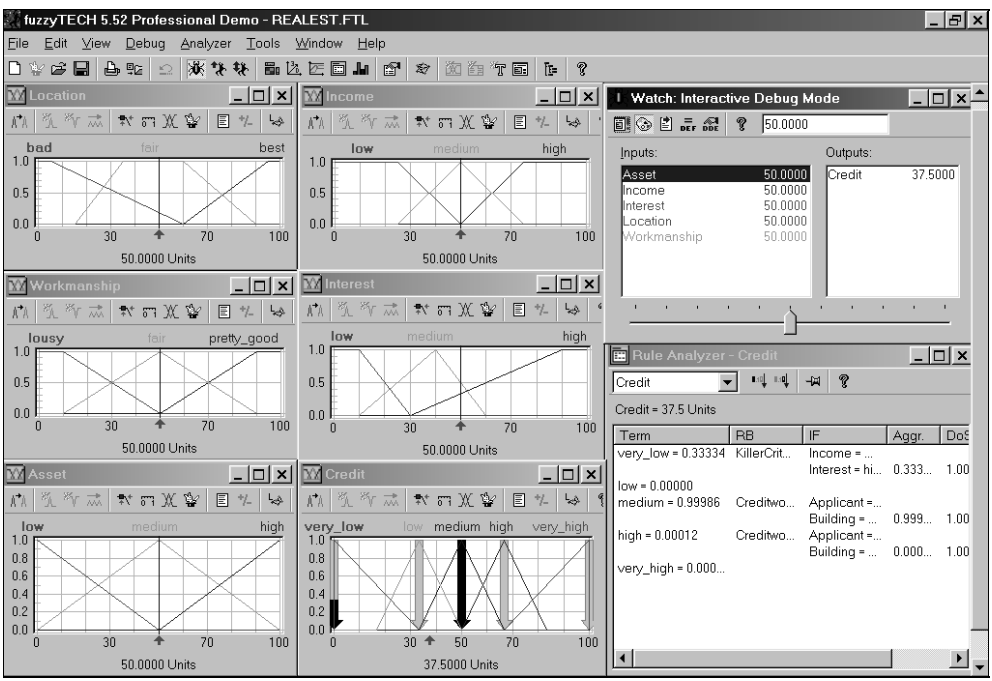


Рис. 20.25. Вид рабочего интерфейса программы fuzzyTECH в режиме отладки проекта "*Оценка кредитоспособности клиентов*"

Для выполнения анализа данной системы нечеткого вывода перейдем в интерактивный режим отладки и дополнительно откроем окно анализатора правил. Внешний вид рабочего интерфейса программы fuzzyTECH в режиме отладки "*Оценка кредитоспособности клиентов*" изображен на рис. 20.25.

Для общего анализа рассматриваемой нечеткой модели можно воспользоваться графическими окнами просмотра поверхности нечеткого вывода на плоскости (рис. 20.26, *а* и *20.27*, *а*) и графическими окнами просмотра трехмерной поверхности нечеткого вывода (рис. 20.26, *б* и *20.27*, *б*).

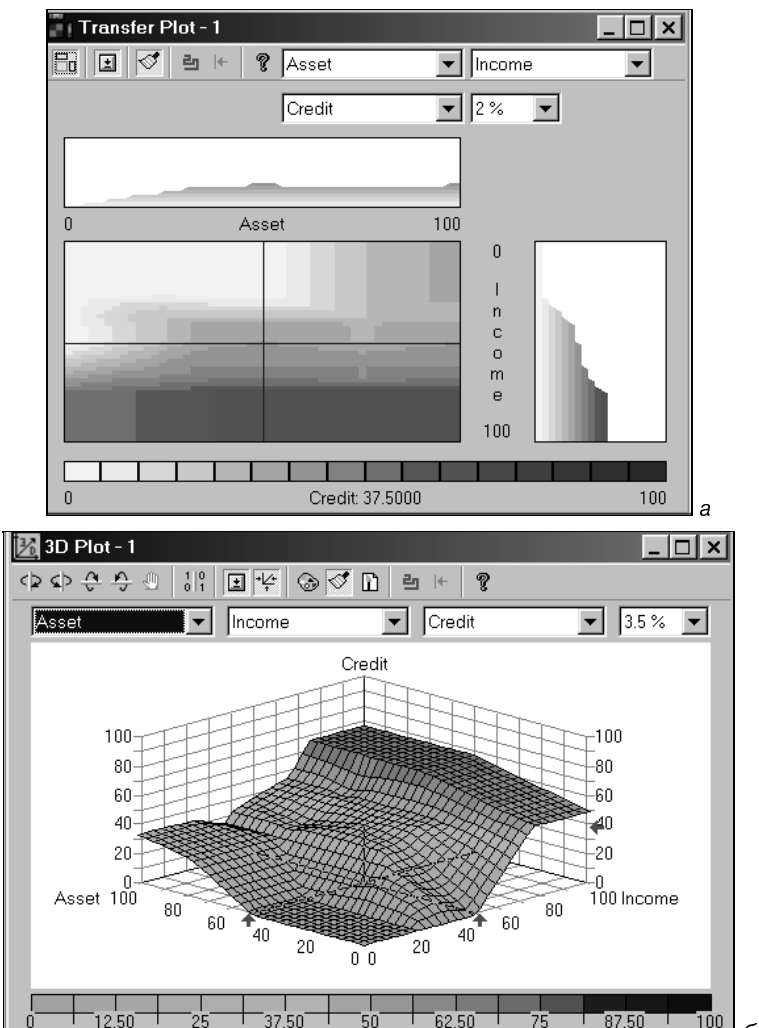
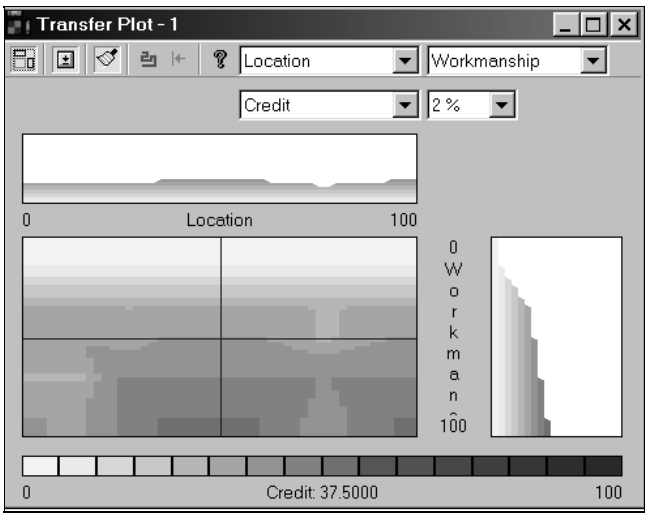
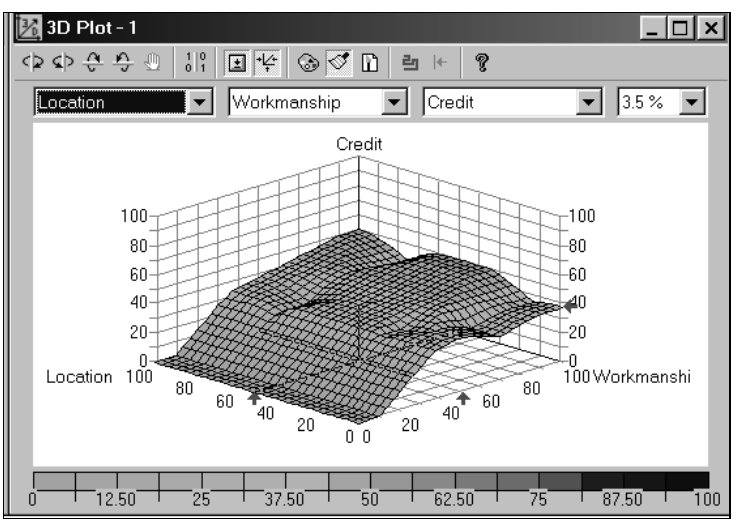


Рис. 20.26. Вид поверхности нечеткого вывода на плоскости (*а*) трехмерной поверхности (*б*) в интерактивном режиме отладки для входных лингвистических переменных первого блока правил

Данные поверхности нечеткого вывода позволяют установить зависимость значений выходной переменной от значений отдельных входных переменных нечеткой модели. Анализ этих зависимостей может служить основанием для изменения функций принадлежности входных переменных или нечетких правил с целью повышения адекватности системы нечеткого вывода для конкретных стратегий банков.



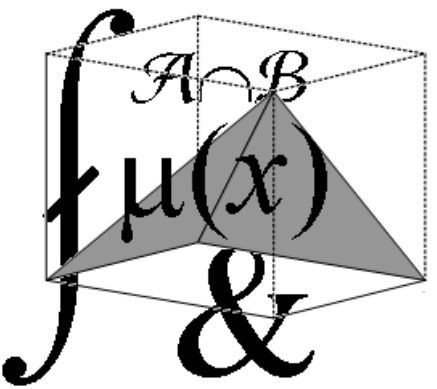
а



б

Рис. 20.27. Вид поверхности нечеткого вывода на плоскости (*а*) трехмерной поверхности (*б*) в интерактивном режиме отладки для входных лингвистических переменных второго блока правил

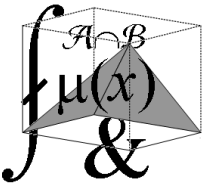
Сравнительный анализ нечетких моделей оценки кредитоспособности клиентов, построенных в средах MATLAB *(см. главу 17)* и fuzzyTECH, показывает, что обе эти модели обладают одинаковой степенью адекватности, поскольку графики поверхностей нечеткого вывода соответствующих лингвистических переменных хорошо согласуются между собой. Отсутствие разрывов на поверхности нечеткого вывода может свидетельствовать об устойчивом характере данных нечетких моделей, что является одним из веских аргументов в пользу применения этих моделей в практике принятия решений при оценке кредитоспособности клиентов.



ЧАСТЬ IV

Приложения

Приложение 1



Основы классической теории множеств и отношений

Основы классической теории множеств как специального раздела математики были разработаны немецким математиком Георгом Кантором (1845—1918) в серии его работ в период 1871—1883 гг. Поскольку теория нечетких множеств является обобщением и дальнейшим развитием обычных множеств и отношений, в настоящем приложении рассматриваются основные теоретико-множественные понятия и обозначения, необходимые для понимания соответствующих конструкций теории нечетких множеств. При этом следует отметить, что сама теоретико-множественная терминология имеет непосредственное отношение не только к современным концепциям естествознания и методологии математических исследований, но и к адекватному описанию базовых концепций системного моделирования.

Множество и способы его задания

Исходным понятием теории множеств является само понятие *множество*, под которым принято понимать некоторую совокупность объектов, хорошо различимых нашей мыслью или интуицией. При этом не делается никаких априорных предположений ни о природе этих объектов, ни о способе их включения в данную совокупность. Отдельные объекты, составляющие то или иное множество, называют *элементами* данного множества. В классической теории множеств природа элементов, из которых состоят множества, не имеет принципиального значения. Предметом данной теории является изучение таких свойств множеств, которые не зависят от природы составляющих их элементов.

Вопрос "*Почему мы рассматриваем ту или иную совокупность элементов как множество*?" в классической теории множеств не требует ответа, поскольку в общее определение множества не входит никаких дополнительных условий на включение отдельных элементов в множество. Если нам хочется, например, рассмотреть множество, состоящее из трех элементов: "*солнце*, *море*, *апельсин*", то никто и ничто не сможет запретить это сделать. В то же время неявно предполагается, что границы множества должны быть *четко* определены, т. е. относительно любого элемента можно с полной определенностью сказать: принадлежит

ли он рассматриваемому множеству или нет. Именно этот аспект в определении множества послужил исходной предпосылкой для введения в рассмотрение нечетких множеств.

Можно привести много примеров конкретных множеств. Это множество квартир жилого дома, множество натуральных чисел, множество планет Солнечной системы, множество сотрудников коммерческой фирмы, множество субъектов Российской Федерации. Совокупность компьютеров в офисе тоже представляет собой множество, хотя, возможно, они и не соединены между собою в сеть. Множество живущих на планете людей, так же как и множество звезд на небосводе, тоже может служить примером множеств. В дальнейшем перечень примеров множеств будет продолжен по мере рассмотрения их отдельных свойств.

Примечание

Создается впечатление, что ситуация с заданием множеств более или менее очевидна. Но это впечатление обманчиво. Даже не говоря об известных парадоксах теории множеств, как быть с "множеством" мыслей отдельного человека? Или множеством всех красок, которые встречаются в природе? Именно такой нетрадиционный взгляд на неопределенность самого *способа* определения состава того или иного множества будет интересовать нас на всем протяжении книги. Вернее, наше внимание будет сосредоточено на таких ситуациях, когда идентификация состава отдельных множеств превращается в самостоятельную проблему. Как было отмечено выше, процесс нечеткого моделирования сопряжен с преодолением именно подобного рода трудностей.

Для записи множеств, их свойств и операций используются специальные обозначения. В общем случае сами множества принято обозначать прописными буквами латинского алфавита. При этом отдельные элементы множества, которые могут иметь различную природу, обозначаются строчными буквами, иногда с индексами, которые хотя и вносят некоторую упорядоченность в последовательность рассмотрения этих элементов, но не являются необходимым атрибутом их задания. Важно понимать, что какой бы то ни было порядок, вообще говоря, не входит в исходное определение множества, а может быть установлен на основе аксиоматизации свойств отношений.

П р и м е р П1.1. Рассмотрим в качестве простого примера множество квартир некоторого 100-квартирного жилого дома. В нашем контексте совершенно безразличны архитектурные особенности этого дома, район его месторасположения и планировка отдельных квартир. Именно в этом и проявляется абстрагирование теоретико-множественного подхода при построении концептуальных моделей той или иной предметной области. Это множество с использованием специальных обозначений можно записать следующим образом: $A=\{a_1, a_2, a_3,\ldots, a_{100}\}$. Здесь фигурные скобки служат обозначением совокупности элементов, каждый из которых имеет свой уникальный числовой индекс. Важно понимать, что для данного конкретного множества элемент a_{10} обозначает отдельную квартиру в рассматриваемом жилом доме. При этом вовсе не обязательно, чтобы номер этой квартиры был равен 10, хотя с точки зрения удобства — это было бы желательно.

Принято называть элементы отдельного множества *принадлежащими* данному множеству. Данный факт записывается с использованием специального символа "$\in$", который так и называется — *символом принадлежности.* Например, запись $a_{10} \in A$ означает тот простой факт, что отдельная квартира (возможно с номером 10) принадлежит рассматриваемому множеству квартир некоторого жилого дома. Если через a_{110} обозначить квартиру с номером 110, то применительно к данному примеру она не входит в множество A, или, говоря более строго, *не* принадлежит этому множеству (рис. П1.1). Формально это записывается как $a_{110} \notin A$, где символ "$\notin$" означает отрицание принадлежности элемента множеству. По определению кроме указанных квартир в множество A не входят никакие другие объекты. Например, такие элементы, как *дача* и *автомобиль*, по определению не принадлежат множеству A.

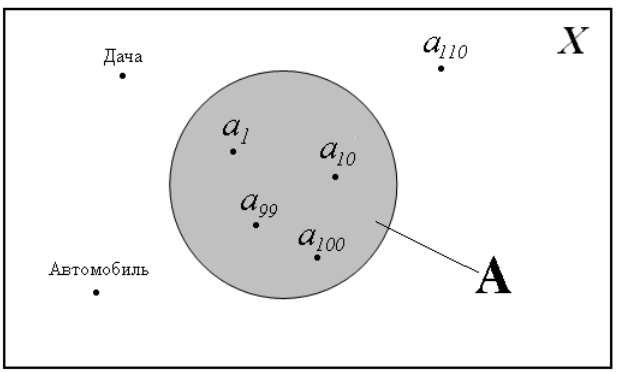


Рис. П1.1. Условное изображение границы и элементов множества квартир *A* рассматриваемого примера

Как следствие вышесказанного, с любым множеством можно связать или ассоциировать некоторую функцию, которая принимает значение 1 для каждого из элементов данного множества, и значение 0 — для всех остальных элементов, не входящих в рассматриваемое множество. Такая функция получила название *характеристической* функции множества. С использованием специальных обозначений характеристическая функция множества A, обозначаемая через χ_A и формально заданная на некотором универсуме $\boldsymbol{X}$, определяется следующим образом:

$$\chi_A(x) = \begin{cases} 1, & \text{если} \quad x \in A \\ 0, & \text{если} \quad x \notin A \end{cases} \quad (\text{для любого } x \in \boldsymbol{X}). \qquad (\text{П1.1})$$

Можно сказать, что задание любого конкретного множества *неявно* определяет соответствующую характеристическую функцию и наоборот. Обычно в классической теории множеств характеристическая функция χ редко используется и играет второстепенную роль. Однако в теории нечетких множеств обобщение этой функции имеет фундаментальное значение, поскольку именно с этой конструкцией связано базовое определение нечеткого множества.

Применительно к рассматриваемому примеру характеристическую функцию множества χ_A можно изобразить графически в форме графика. Для этого на горизонтальной оси X отметим точками обозначения квартир — a_1, a_2, a_3,..., a_{100}, a_{101}, a_{102} и, возможно, других интересующих нас объектов. На вертикальной оси Y рассмотрим только два числовых значения: 0 и 1. Тогда график соответствующей характеристической функции будет состоять из отдельных изолированных точек, что предопределено конечным числом элементов множества A, причем только для 100 элементов значение χ_A будет равно 1, а для всех остальных — 0 (рис. П1.2).

Рис. П1.2. График характеристической функции χ_A рассматриваемого множества квартир A

Между множествами могут иметь место различные отношения, простейшим из которых является *равенство* двух множеств. Два множества считаются равными, если они состоят из одних и тех же элементов. При этом порядок следования элементов в этих множествах не имеет значения. Формально равенство двух множеств можно записать с помощью характеристических функций этих множеств. А именно, множество A равно множеству B (записывается как $A=B$) тогда и только тогда, когда их характеристические функции равны, т. е. выполняется следующее условие:

$$\chi_A(x)=\chi_B(x) \qquad \text{(для любого } x\in \boldsymbol{X}\text{)}, \qquad \text{(П1.2)}$$

где $\chi_A(x)$, $\chi_B(x)$ — характеристические функции множеств A и B соответственно.

Для иллюстрации различных теоретико-множественных операций традиционно используются не характеристические функции, а специальные графические конструкции — *диаграммы Венна*. Последние получили свое название в честь английского логика Джона Венна (1834—1923), который предложил эти обозначения для наглядной интерпретации множеств. Тот факт, что некоторая совокупность элементов образует множество, на диаграмме Венна обозначается графически в форме круга. В этом случае окружность приобретает содержательный смысл или, выражаясь более точным языком, семантику *границы* данного

множества. В этом случае равенство двух множеств можно изобразить графически следующим образом (рис. П1.3).

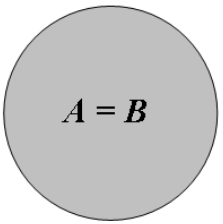


Рис. П1.3. Диаграмма Венна
для равенства двух множеств *A* и *B*

Следующим важным понятием, которое служит прототипом для многих других терминов системного и нечеткого моделирования, является понятие *подмножества*. Интуитивно ситуация достаточно прозрачна. Если есть некоторая совокупность, рассматриваемая как множество, то любая ее часть и будет являться подмножеством этого множества. Так, например, совокупность квартир на первом этаже жилого дома есть ничто иное, как подмножество множества квартир рассматриваемого нами примера. Для обозначения подмножества используется специальный символ "$\subset$". Если утверждается, что множество A является подмножеством множества B, то это записывается как: $A \subset B$.

Примечание

Строго говоря, следует различать два различных варианта подмножества. Рассмотренное выше определение характерно для так называемого *собственного* подмножества, когда исключается случай возможного равенства двух множеств $A=B$. Если же при определении подмножества допускается равенство двух множеств, то оно называется *несобственным* подмножеством и имеет обозначение $A \subseteq B$.

Формально это отношение можно записать и с помощью характеристических функций этих множеств. А именно, множество A является подмножеством (собственным подмножеством) множества B тогда и только тогда, когда справедливо следующее неравенство:

$$\chi_A(x) \le \chi_B(x) \qquad (\text{соответственно, } \chi_A(x) < \chi_B(x)) \qquad (\text{П1.3})$$

$$(\text{для любого } x \in \boldsymbol{X}).$$

Здесь $\chi_A(x)$, $\chi_B(x)$ — характеристические функции множеств A и B соответственно.

Подмножество или факт включения элементов одного множества в другое множество можно изобразить графически следующим образом (рис. П1.4). На этом рисунке бóльшему множеству B соответствует внешний круг, а меньшему множеству (подмножеству) A — внутренний круг.

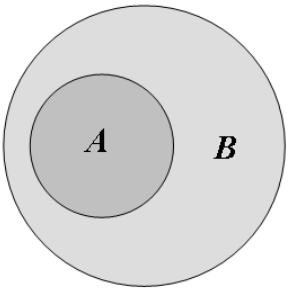


Рис. П1.4. Диаграмма Венна
для строгого включения двух множеств $A \subset B$

В теории множеств некоторые специальные множества играют особую роль. Одним из таких множеств является так называемое *пустое* множество или множество, которое не содержит ни одного элемента. Это множество имеет специальное обозначение: $\varnothing$. Из этого определения следует, что пустое множество является собственным подмножеством любого множества, не являющегося в свою очередь пустым. То есть для любого множества A всегда справедливо утверждение: $\varnothing \subset A$. Очевидно, что характеристическая функция пустого множества тождественно равна нулю для каких бы то ни было элементов: $\chi_{\varnothing} = 0$.

Другим специальным множеством является так называемый *универсум* или множество, содержащее все возможные элементы. Это множество также имеет специальное обозначение: $\boldsymbol{X}$. Из определения универсума следует, что любое множество, не являющееся в свою очередь универсумом, является собственным подмножеством универсума. То есть для любого множества A всегда справедливо утверждение: $A \subset \boldsymbol{X}$. Очевидно, что характеристическая функция универсума тождественно равна единице для каких бы то ни было элементов: $\chi x = 1$.

В зависимости от количества элементов множества бывают конечные и бесконечные. Множество называется *конечным*, если оно содержит конечное число элементов. Про такое множество еще говорят, что оно имеет конечную *мощность*, которая численно равна количеству элементов этого множества. Для обозначения мощности произвольного множества A используется специальный символ $\mathrm{card}(A)$.

Примечание

В литературе для обозначения мощности множества A также используются и другие символы, как, например: $|A|$ или $\overline{\overline{A}}$. Последний из символов может ввести в заблуждение, поскольку, как будет видно из последующего изложения, совпадает по внешнему виду с символом двойного дополнения. По этой причине в дальнейшем мы не будем его использовать.

Возвращаясь к примеру с множеством квартир жилого дома, можно сказать, что его мощность равна 100. Множество всех возможных комбинаций из 8 символов, которые могут служить для ввода некоторого пароля с клавиатуры компьютера,

конечное, хотя и достаточно большое. Или, говоря строгим языком, это множество имеет конечную мощность.

Ситуация усложняется, когда рассматриваются *бесконечные* множества, т. е. множества, не являющиеся конечными. Эта сложность связана с тем, что бесконечные множества могут быть счетными и несчетными. *Счетным* множеством принято называть множество, содержащее бесконечное число элементов, которые, однако, можно перенумеровать натуральными числами 1, 2, 3 и т. д. При этом важно иметь в виду, что достичь последнего элемента при такой нумерации принципиально невозможно, иначе множество окажется конечным. Про такие множества говорят, что они имеют *счетную мощность*, которая обозначается $\aleph_0$ (читается — "*алеф нуль*"). Есть все основания считать множество всех звезд бесконечным, хотя многие из звезд имеют свое уникальное название. Именно по этой причине данное множество может быть отнесено к категории счетных.

Несчетным множеством принято называть множество, содержащее бесконечное число элементов, которые принципиально нельзя перенумеровать натуральными числами. Г. Кантор доказал, что множество всех действительных чисел несчетно. В этой связи иногда говорят, что данное множество имеет несчетную мощность или мощность *континуума* (обозначается символами c или $\aleph$).

Примечание

Проблема бесконечного могла бы показаться отвлеченной и имеющей некоторый философский оттенок, если бы не ее связь с системным моделированием. Так, при рассмотрении некоторой предметной области с целью построения ее концептуальной модели приходится выделять конечное число сущностей или объектов, образующих определенный "скелет" будущей модели. Это при том, что реальность окружающих нас предметов допускает бесконечное рассмотрение их свойств, атрибутов и взаимосвязей.

Множества могут быть заданы двумя основными способами:

- Явным *перечислением* или указанием всех элементов, входящих в рассматриваемое множество. Очевидно, этот способ подходит только для задания конечных множеств с небольшим числом элементов. Например, множество из пяти первых натуральных чисел или множество планет Солнечной системы.
- Указанием некоторого *свойства*, которым обладает каждый из элементов рассматриваемого множества и не обладает всякий другой элемент, не входящий в рассматриваемое множество. Этот способ может быть использован для задания как конечных, так, что наиболее важно, и бесконечных множеств. Например, только таким способом можно определить множество всех четных натуральных чисел или множество действительных чисел, строго больших 10.

В первом случае множество символически записывается в виде: $A=\{a_1, a_2, \ldots, a_n\}$, где n — общее число элементов множества A. Другими словами, множество A имеет конечную мощность $n=\mathrm{card}(A)$.

Во втором случае вводят в рассмотрение некоторое характеристическое свойство, которое может быть записано в виде одноместного *предиката* $P(x)$. Для фор-

мальной строгости предикат $P(x)$ определяется на универсуме $\boldsymbol{X}$ элементов, из которых формируется множество A. Предполагается, что в качестве универсума $\boldsymbol{X}$ в каждом конкретном случае используется такая совокупность объектов, для которых рассматриваемое свойство имеет содержательный смысл или, другими словами, означивание соответствующего предиката семантически корректно. При этом предикат может принимать одно из двух значений истинности: "*истина*" или "*ложь*". Если элемент $x \in \boldsymbol{X}$ обладает рассматриваемым свойством, то соответствующий предикат $P(x)$ принимает значение "*истина*". Если же элемент $x \in \boldsymbol{X}$ не обладает рассматриваемым свойством, то соответствующий предикат $P(x)$ принимает значение "*ложь*". Тогда в общем случае задание множества A с использованием специального свойства, выраженного в форме одноместного предиката $P(x)$, может быть записано в виде: $A=\{a \mid P(a),\ a \in \boldsymbol{X}\}$.

Примечание

Вообще говоря, понятие предиката и значения истинности относятся к математической логике, основы которой изложены в *приложении 2*. Однако для полноты представления основ теории множеств необходимо уже сейчас использовать указанные конструкции.

Например, множество неотрицательных действительных чисел, которое имеет специальное обозначение $\mathbb{R}_+$ или $[0, +\infty)$, может быть задано в виде: $\mathbb{R}_+=\{x \mid x \in \mathbb{R},\ x \geq 0\}$. В этом случае предикат $P(x)$ определяется в форме некоторого составного высказывания "*x является действительным числом и, одновременно, x больше или равно* 0", которое допускает подстановку вместо x любого числа (т. е. $\boldsymbol{X}=\mathbb{R}$, где $\mathbb{R}$ — множество всех действительных чисел). Другим интересным для нас множеством, которое также может быть задано вторым способом, является *замкнутый интервал* действительных чисел, заключенных между 0 и 1, включая сами эти числа. Этот интервал, который традиционно обозначается как $I_{\mathbb{R}}$ или $[0, 1]$, определяется следующим образом: $I_{\mathbb{R}}=\{x \mid x \in \mathbb{R},\ 0 \leq x \leq 1\}$. Эти множества также могут быть определены на основе задания соответствующих характеристических функций $\chi_{[0,+\infty)}$ и $\chi_{[0,\ 1]}$, графики которых представлены на рис. П1.5.

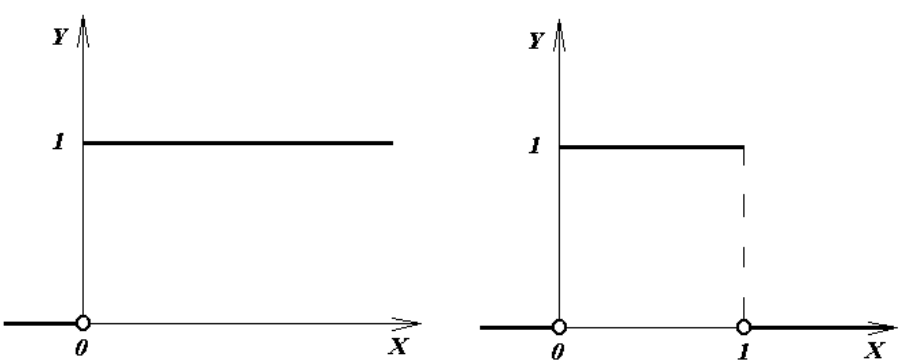


Рис. П1.5. Графики характеристических функций $\chi_{[0,+\infty)}$ (*а*) и $\chi_{[0,\ 1]}$ (*б*)

Примечание

В связи с рассмотрением последнего примера следует обратить внимание на различие множеств {0, 1} и [0, 1]. Первое из них является конечным множеством, которое состоит только из двух элементов: чисел 0 и 1. Второе является бесконечным множеством и, более того, имеет мощность континуума. Тем не менее, справедливо включение: $\{0, 1\} \subset [0, 1]$.

Как нетрудно заметить, в общем случае для произвольного множества A существует формальная взаимосвязь между характеристической функцией множества $\chi_A(x)$ и некоторым предикатом $P_A(x)$, который, возможно неявно, выражает некоторое характеристическое свойство этого множества. А именно, если элемент $a \in \boldsymbol{X}$ обладает свойством $P_A(x)$, т. е. $P_A(a)$ — "*истинно*", то $\chi_A(a)=1$. Если же элемент $a \in \boldsymbol{X}$ не обладает свойством $P_A(x)$, т. е. $P_A(a)$ — "*ложно*", то $\chi_A(a)=0$. Верно и обратное заключение. Эта связь характеристической функции множества и соответствующего ей предиката играет важную роль при установлении взаимосвязей между теорией нечетких множеств и нечеткой логикой.

Основные теоретико-множественные операции

Пусть A и B — произвольные (конечные или бесконечные) множества. *Пересечением* двух множеств A и B называется некоторое третье множество C, которое состоит из тех и только тех элементов двух исходных множеств, которые *одновременно* принадлежат и множеству A, и множеству B. Для этой операции имеется специальное обозначение: "$\cap$". Тогда результат операции пересечения двух множеств можно записать в виде: $C=A \cap B$, где $C=\{x | x \in A$ *и одновременно* $x \in B\}$.

Примечание

При этом важно обратить внимание на тот факт, что характеристическая функция χ_C множества $C=A \cap B$ зависит некоторым образом от характеристических функций исходных множеств χ_A и χ_B. А именно, значение функции $\chi_C(x)$ для любого $x \in \boldsymbol{X}$ может быть получено как минимальное из двух значений: $\chi_A(x)$ и $\chi_B(x)$. Это свойство может быть положено в основу определения операции пересечения множеств. Формально же оно записывается в следующем виде:

$$\chi_C(x)=\min\{\chi_A(x), \chi_B(x)\} \quad \text{(для любого } x \in \boldsymbol{X}\text{)}. \qquad \text{(П1.4)}$$

Например, если в качестве множества A рассмотреть множество сотрудников некоторой фирмы, а в качестве множества B — множество всех мужчин, то нетрудно догадаться, что множество C как результат операции пересечения $A \cap B$ будет состоять из элементов — всех сотрудников мужского пола данной фирмы. Операция пересечения множеств может быть проиллюстрирована с использованием диаграмм Венна (рис. П1.6). На этом рисунке условно изображены два множества A и B, затемненной области как раз и соответствует множество C, являющееся пересечением множеств A и B.

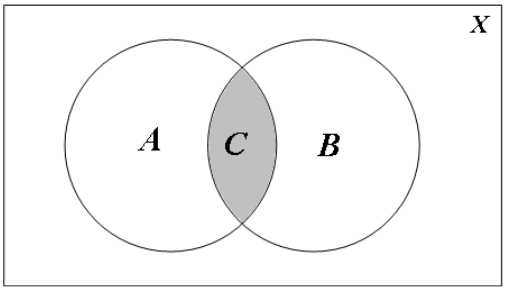


Рис. П1.6. Диаграмма Венна для пересечения двух множеств *A* и *B*

Под *объединением* двух множеств A и B понимается некоторое третье множество, которое обозначим как D, состоящее из тех и только тех элементов, которые принадлежат или A, или B, или им обоим одновременно. Для этой операции также существует специальное обозначение: $D=A\cup B$, где $D=\{x|x\in A$ или $x\in B\}$.

Примечание

При этом важно заметить, что характеристическая функция χ_D множества $D=A\cup B$ также зависит некоторым образом от характеристических функций исходных множеств χ_A и χ_B. А именно, значение функции $\chi_D(x)$ для любого $x\in\boldsymbol{X}$ может быть получено как максимальное из двух значений: $\chi_A(x)$ и $\chi_B(x)$. Это свойство может быть положено в основу определения операции объединения множеств. Формально же оно записывается в следующем виде:

$$\chi_D(x)=\max\{\chi_A(x),\ \chi_B(x)\} \qquad \text{для любого} \quad x\in\boldsymbol{X}. \qquad (П1.5)$$

Так, например, если в качестве множества A рассмотреть множество, состоящее из клавиатуры и мыши, а в качестве множества B — множество, состоящее из системного блока и монитора, то нетрудно догадаться, что их объединение, т. е. множество $D=A\cup B$ будет сдержать все основные компоненты персонального компьютера. Операция объединения множеств также может быть проиллюстрирована с использованием диаграмм Венна (рис. П1.7). На этом рисунке объединению двух исходных множеств также соответствует затемненная область, только размеры и форма ее отличаются от случая пересечения двух множеств.

Разностью двух множеств A и B называется некоторое третье множество E (обозначается $E=A\backslash B$, которое состоит из тех и только тех элементов, которые принадлежат множеству A и не принадлежат множеству B, т. е. $E=\{x|x\in A$ *и одновременно* $x\notin B\}$. Если множества A и B не пересекаются (т. е. $A\cap B=\varnothing$), то $A\backslash B=A$. Из определения этой операции также следует, что: $A\backslash A=\varnothing$ и $A\backslash\varnothing=A$. Формально справедливо следующее утверждение:

$$\chi_E(x)=\max\{\chi_A(x)-\chi_B(x),\ 0\} \qquad \text{для всех} \quad x\in\boldsymbol{X}, \qquad (П1.6)$$

где под знаком максимума применяется обычная операция арифметической разности двух чисел. Это свойство в свою очередь может быть использовано для исходного определения операции разности множеств.

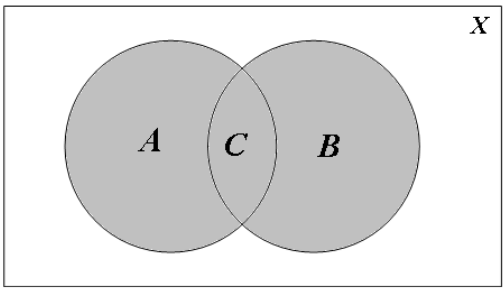


Рис. П1.7. Диаграмма Венна для объединения двух множеств *A* и *B*

Так, например, если в качестве множества A рассмотреть множество, состоящее из студентов некоторой группы, а в качестве множества B — множество студентов, получивших неудовлетворительные оценки на сессии или не аттестованных вовсе, то их теоретико-множественная разность $E=A\setminus B$ — множество студентов группы, успешно сдавших сессию. Операция разности множеств также может быть проиллюстрирована с использованием диаграмм Венна (рис. П1.8). На этом рисунке разности двух исходных множеств также соответствует затемненная область.

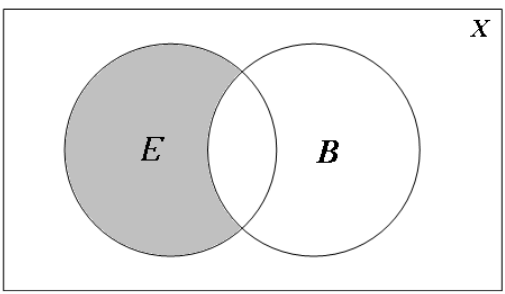


Рис. П1.8. Диаграмма Венна для разности двух множеств *A* и *B*

Следует заметить, что операция разности множеств в отличие от операций объединения и пересечения не является симметричной, т. е. в общем случае $A\setminus B \neq B\setminus A$. Поэтому иногда удобно рассматривать так называемую *симметрическую разность* двух множеств A и B (обозначается через $A\Delta B$). По определению $A\Delta B=(A\setminus B)\cup(B\setminus A)$, т. е. симметрическая разность двух множеств представляет собой объединение двух разностей множеств A и B. При этом оказывается справедливым следующее утверждение:

$$\chi_{A\Delta B}(x)=max\{|\chi_A(x)-\chi_B(x)|, 0\} \qquad \text{для всех} \quad x\in \boldsymbol{X}, \tag{П1.7}$$

где под знаком максимума применяется операция модуля (или вычисления абсолютного значения) числа. Это свойство также может быть использовано для исходного определения операции симметрической разности множеств.

Операция симметрической разности множеств иллюстрируется с помощью диаграмм Венна следующим образом (рис. П1.9). На этом рисунке симметрической разности двух исходных множеств соответствует затемненная область. Можно показать, что для операции симметрической разности имеет место следующее свойство: $A\Delta B=(A\cup B)\backslash(A\cap B)$.

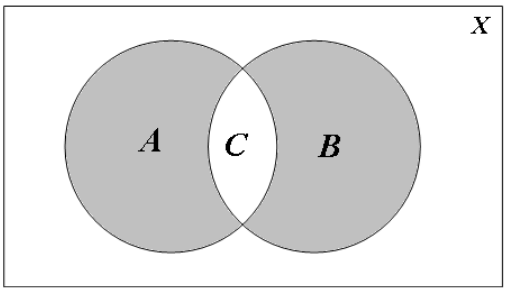


Рис. П1.9. Диаграмма Венна для симметрической разности двух множеств A и B

В целом ряде случаев оказывается полезной унарная операция *дополнения* множества. Дополнение множества A обозначается через $\overline{A}$ и определяется следующим образом: $\overline{A}=\{x|\, x\in \boldsymbol{X}$ *и одновременно* $x\notin A\}$ или $\overline{A}=\boldsymbol{X}\backslash A$, где множество $\boldsymbol{X}$ — универсум. Легко проверяется следующее свойство характеристической функции для дополнения множества:

$$\chi_{\overline{A}}=1-\chi_{\overline{A}} \text{ для всех } x\in \boldsymbol{X}. \qquad (П1.8)$$

Операция дополнения множества иллюстрируется с помощью диаграмм Венна следующим образом (рис. П1.10). На этом рисунке дополнению множества A соответствует затемненная область.

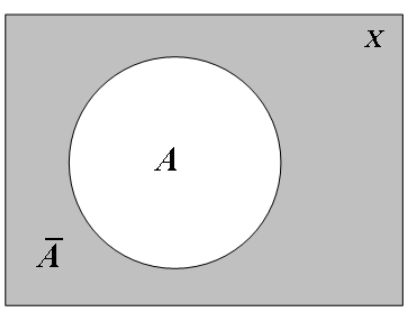


Рис. П1.10. Диаграмма Венна для дополнения множества A

Пример П1.2. Рассмотрим два числовых множества: $A=\{1, 3, 5, 7, 8, 9\}$ и $B=\{2, 4, 6, 7, 8, 10\}$. Для этих множеств результаты выполнения рассмотренных

выше теоретико-множественных операций будут следующими: $A \cap B = \{7, 8\}$, $A \cup B = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, $A \setminus B = \{1, 3, 5, 9\}$, $A \Delta B = \{1, 2, 3, 4, 5, 6, 9, 10\}$. Если в качестве универсума $\boldsymbol{X}$ взять множество натуральных чисел $\mathbb{N}$, то $\overline{A} = \{2, 4, 6, 10, 11, 12, 13, ...\}$ и $\overline{B} = \{1, 3, 5, 9, 11, 12, 13, \ldots\}$.

Для теоретико-множественных операций имеют место следующие свойства, часть из которых непосредственно следует из их определения, а доказательство других может служить в качестве упражнения:

- *Коммутативность* операций объединения и пересечения:

$$A \cup B = B \cup A; \qquad A \cap B = B \cap A. \tag{П1.9}$$

- *Ассоциативность* операций объединения и пересечения:

$$A \cup (B \cup C) = (A \cup B) \cup C; \qquad A \cap (B \cap C) = (A \cap B) \cap C. \tag{П1.10}$$

- *Дистрибутивность* операций объединения и пересечения относительно друг друга:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C); \qquad A \cup (B \cap C) = (A \cup B) \cap (A \cup C). \tag{П1.11}$$

- *Идемпотентность* операций объединения и пересечения:

$$A \cup A = A; \qquad A \cap A = A. \tag{П1.12}$$

- *Поглощение* пустого множества и универсума при операциях объединения и пересечения соответственно:

$$A \cup \varnothing = A; \qquad A \cap \boldsymbol{X} = A. \tag{П1.13}$$

- Универсальные границы (верхняя и нижняя):

$$A \cup \boldsymbol{X} = \boldsymbol{X}; \qquad A \cap \varnothing = \varnothing. \tag{П1.14}$$

- *Дополняемость* (или дополнительность):

$$A \cap \overline{A} = \varnothing; \qquad A \cup \overline{A} = \boldsymbol{X}. \tag{П1.15}$$

- *Инволюция* или двойное дополнение:

$$\overline{\overline{A}} = A. \tag{П1.16}$$

- Законы де Моргана:

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}; \qquad \overline{(A \cap B)} = \overline{A} \cup \overline{B}. \tag{П1.17}$$

Перечисленные свойства получили название аксиом *булевой алгебры* в честь английского логика Дж. Буля (1815—1864), отмечая тем самым его вклад в разработку основ математической логики.

Формула (П1.9) означает, что теоретико-множественные операции объединения и пересечения конечного числа множеств не зависят от порядка следования отдельных множеств при выполнении этих операций. Формула (П1.10) означает, что эти операции не зависят от того, выполняем ли мы их в некоторой очередности или разбиваем на отдельные группы с последующим выполнением соответствующих операций над этими группами. Первый из законов дистрибутивности (П1.11) полностью аналогичен закону дистрибутивности операции умножения относительно сложения в арифметике. Как и в арифметике, из этого закона сле-

дует, что для того, чтобы найти пересечение двух групп множеств, каждая из которых представляет собой объединение конечного числа множеств, следует найти пересечение каждого из множеств первой группы с каждым из множеств второй группы, а к полученным результатам применить операцию объединения. Например, $(A \cup B \cup C \cup D) \cap (E \cup F \cup G) = (A \cap E) \cup (A \cap F) \cup (A \cap G) \cup (B \cap E) \cup \ldots \cup (D \cap E) \cup (D \cap F) \cup (D \cap G)$. Идемпотентность операций объединения и пересечения не имеет аналогии в арифметике, поскольку в результате их применения к одному и тому же множеству всегда получается исходное множество. Свойства поглощения и универсальных границ аналогичны наличию единицы для операции умножения и нуля для операции сложения в арифметике (кроме первой формулы (П1.14), которая не имеет аналогии).

Первая формула дополняемости (П1.15) имеет аналогию в формальной логике в виде так называемого закона *противоречия*. Этот закон утверждает, что в процессе рассуждения о каком-либо определенном предмете или явлении нельзя одновременно утверждать и отрицать что-либо в одном и том же отношении, в противном случае оба суждения не могут быть одновременно истинными. Действительно, если мы утверждаем, что некоторый элемент a_i принадлежит множеству A, то тем самым мы исключаем противоположное утверждение, что этот же элемент a_i *не* принадлежит множеству A. И наоборот, истинность утверждения "элемент a_j не принадлежит множеству A" влечет ложность противоположного утверждения "элемент a_j принадлежит множеству A". Например, утверждая, что квартира с номером 100 одновременно принадлежит и не принадлежит множеству A рассмотренного выше примера П1.1, мы неизменно приходим к противоречию, поскольку в данном примере по определению возможен только один случай, а именно: $a_{100} \in A$.

Вторая формула дополняемости (П1.15) также имеет аналогию в формальной логике в виде так называемого закона *исключенного третьего*. Этот закон утверждает, что в процессе рассуждения о каком-либо определенном предмете или явлении необходимо доводить дело до некоторого законченного утверждения или его отрицания. При этом, если одно из таких утверждений истинно, то другое должно быть обязательно ложно, и наоборот. Третьего в этом отношении быть не может: либо истина, либо ложь. Действительно, можно сформулировать только *два* утверждения относительно принадлежности того или иного элемента множеству: "*элемент a_i принадлежит множеству A*" и "*элемент a_i не принадлежит множеству A*". Третьего утверждения в классической теории множеств быть не может. Например, относительно квартиры с номером 100 можно утверждать только следующее: "*квартира a_{100} принадлежит множеству A*" или "*квартира a_{100} не принадлежит множеству A*". В этом контексте важно понимать, что нельзя сформулировать какое бы то ни было третье утверждение относительно принадлежности данной квартиры рассматриваемому множеству A.

Что касается инволюции или двойного дополнения, то это свойство легко доказывается, если обозначить $\overline{A} = B$. Тогда, если некоторый элемент $a_i \in A$, то тем самым $a_i \notin \overline{A}$ и, следовательно, $a_i \notin B$ или $a_i \in B$. Таким образом, $A = \overline{B}$ или, вводя операцию двойного дополнения, $\overline{\overline{A}} = A$. В обратную сторону доказательство фор-

мулы (П1.16) аналогично. Предположим, что $a_i \in B$. Это означает, что $a_i \notin B$ или, с учетом введенных обозначений, $a_i \notin \overline{A}$. Последнее утверждение в свою очередь влечет $a_i \in A$. Поскольку $\overline{B}$ по определению равно $\overline{\overline{A}}$, то тем самым $\overline{\overline{A}} = A$, что и завершает доказательство данного свойства.

В заключение докажем справедливость первого из законов, получивших свое название в честь шотландского математика и логика Огастеса де Моргана (De Morgan Augustus, 1806—1871): $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$. Для этого обозначим $A \cup B$ через C, т. е. $C = A \cup B$. Предположим, что некоторый элемент $a_i \in \overline{C}$ или $a_i \notin C$. Тем самым справедливо $a_i \notin A$, поскольку $A \subset C$, и одновременно $a_i \notin B$, поскольку также и $B \subset C$. С другой стороны, это означает, что $a_i \in \overline{A}$ и одновременно $a_i \in \overline{B}$. Последнее означает ни что иное как: $a_i \in \overline{A} \cap \overline{B}$. Это доказывает первую часть закона. Теперь докажем рассматриваемое равенство в обратную сторону. Если предположить, что некоторый элемент $a_i \notin A$ и одновременно $a_i \notin B$, то это будет означать, что $a_i \notin A \cup B$. Действительно, если существует такой элемент a_j, для которого справедливо условие: $a_j \in A \cup B$, то тем самым исключается одновременное выполнение условий: $a_j \notin A$ и в то же время $a_j \notin B$. Помня, что $C = A \cup B$, условие $a_i \notin A \cup B$ будет означать $a_i \notin C$, а значит $a_i \in \overline{C}$. Поскольку $a_i \notin A$ означает $a_i \in \overline{A}$, а $a_i \notin B$ означает $a_i \in \overline{B}$, одновременное выполнение этих условий означает ни что иное, как: $a_i \in \overline{A} \cap \overline{B}$. Поскольку мы начали рассуждения с последнего утверждения, а пришли к утверждению $a_i \in \overline{C} = \overline{A \cup B}$, то тем самым получили доказательство первого закона де Моргана в обратную сторону. Второй закон доказывается аналогично и предоставляется выполнить читателю в качестве самостоятельного упражнения.

Булеан или множество всех подмножеств

Как можно заметить, в определении множества присутствует некоторая двойственность, которая касается входящих в множество элементов. С одной стороны, между элементом и множеством имеет место отношение принадлежности $a_i \in A$, которое устанавливает исходный состав множества $A = \{a_1, a_2, \ldots, a_i, \ldots\}$. С другой стороны, каждый из элементов множества A можно рассматривать как одноэлементное подмножество $\{a_i\}$, и тогда имеет место отношение включения между каждым из этих одноэлементных подмножеств и исходным множеством: $\{a_i\} \subset A$. При этом оказывается справедливым следующее утверждение: $\bigcup_i \{a_i\} = A$, где операция объединения выполняется по всем одноэлементным подмножествам, соответствующим элементам множества A.

Именно последний аспект рассмотрения множеств послужил исходной идеей для введения специальной конструкции, которая играет особую роль при определении целого ряда других понятий. Речь идет о множестве всех подмножеств (булеане или степенном множестве), элементами которого по определению являют-

ся все без исключения подмножества некоторого фиксированного множества A. А именно, формально *булеан* определяется как $\mathbb{B}(A)=\{A_i \mid A_i \subseteq A\}$. По этому определению пустое множество $\varnothing$ и само множество A входят в булеан или являются его элементами. Если исходное множество A конечное, т. е. имеет конечную мощность n=card(A), то булеан также будет являться конечным множеством, а его мощность будет равна card($\mathbb{B}(A)$)=2^n (доказывается методом индукции по количеству элементов множества A).

Таким образом, имеет место следующее утверждение: card($\mathbb{B}(A)$)=$2^{\text{card}(A)}$. По аналогии с конечными множествами это свойство распространяют и на бесконечные множества, допуская при этом некоторую вольность, поскольку показатель степени в этом случае не является числом.

Мультимножество или комплект

Одним из обобщений понятия множества является так называемое мультимножество, в качестве синонима которого часто применяется термин *комплект*. Это понятие иногда используется для формального определения математических конструкций, в которых допускается повторение одинаковых элементов или экземпляров некоторого класса.

М у л ь т и м н о ж е с т в о. Содержательно *мультимножество* можно определить как такое множество, в котором те или иные элементы могут повторяться или встречаться несколько раз. Формально мультимножество определяется как множество кортежей: $A^*=\{<a_1, \alpha_1>, <a_2, \alpha_2>, \ldots, <a_n, \alpha_n>\}$, где a_i — элемент мультимножества, α_i — количество повторений элемента a_n в этом мультимножестве A^*. Как видно из этого определения, мультимножества будем обозначать так же, как и обычные множества, но с дополнительным знаком "*". Очевидно, порядок следования кортежей в записи мультимножества не имеет никакого значения.

П р и м е р П1.3. Для иллюстрации данного понятия рассмотрим два абстрактных мультимножества: $A^*=\{<a_1, 2>, <a_2, 1>, <a_3, 2>, <a_4, 1>, <a_5, 3>\}$ и $B^*=\{<a_1, 1>, <a_2, 2>, <a_3, 3>, <a_4, 1>, <a_5, 2>\}$ и два числовых мультимножества: $C^*=\{<1, 1>, <2, 2>, <3, 1>, <4, 2>, <5, 3>\}$ и $D^*=\{<1, 1>, <2, 2>, <3, 3>, <4, 4>, <5, 5>\}$. Они могут быть также записаны в альтернативной форме как: $A^*=\{a_1, a_1, a_2, a_3, a_3, a_4, a_5, a_5, a_5\}$ и $B^*=\{a_1, a_2, a_2, a_3, a_3, a_3, a_4, a_5, a_5\}$, а также: $C^*=\{1, 2, 2, 3, 4, 4, 5, 5, 5\}$ и $D^*=\{1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5\}$.

Примечание

Следует заметить, что в альтернативной форме записи порядок следования элементов в записи мультимножества не имеет значения. Другими словами, записи {1, 2, 5, 2, 4, 5, 3, 5, 4} и {2, 1, 2, 5, 3, 4, 5, 4, 5} соответствуют одному и тому же мультимножеству, а именно, C^*. Очевидно, что для мультимножеств с достаточно большим числом элементов эта форма записи не совсем удобна, поскольку затрудняет визуальное установление подобного факта.

Более строгое определение мультимножества может быть дано на основе предварительного рассмотрения некоторого универсума $\boldsymbol{X}$, как правило, имеющего конечную мощность, и некоторого отображения α: $\boldsymbol{X}\to\mathbb{N}_0$, ставящего в соответствие каждому элементу универсума неотрицательное целое число, равное количеству его включений в мультимножество. При этом элементы универсума, для которых значение этого отображения равно нулю, не записываются в состав данного мультимножества. Здесь и далее через $\mathbb{N}_0$ будет обозначаться множество натуральных чисел и ноль, т. е. $\mathbb{N}_0=\{0, 1, 2, 3,\ldots\}$.

Как и в случае обычных множеств, с любым мультимножеством A^* можно связать некоторую *обобщенную характеристическую функцию* $\chi_{A^*}(x)$, которая принимает значение $\alpha(x_i)$ для каждого из элементов x_i данного мультимножества. С использованием этой обобщенной характеристической функции для мультимножеств могут быть определены отношения равенства и включения, аналоги теоретико-множественных операций и другие понятия, соответствующие случаю обычных множеств.

В общем случае два мультимножества A^* и B^* считаются *равными* по определению, если равны множества соответствующих им кортежей. Другими словами, равные мультимножества не только состоят из одинаковых элементов универсума $\boldsymbol{X}$, но и количество повторений каждого из элементов в этих мультимножествах также равно. Это эквивалентно выполнению условия (П1.2) для соответствующих обобщенных характеристических функций этих мультимножеств. В противном случае мультимножества считаются неравными. Равные мультимножества записываются в виде: $A^*=B^*$.

Таким же образом можно ввести определения отношения включения мультимножеств на основе выполнения условия (П1.3), операции пересечения мультимножеств на основе выполнения условия (П1.4), операции объединения мультимножеств на основе выполнения условия (П1.5), операции разности мультимножеств на основе выполнения условия (П1.6) и операции симметрической разности мультимножеств на основе выполнения условия (П1.7) для соответствующих обобщенных характеристических функций этих мультимножеств.

Так, например, для мультимножеств из примера П1.3 справедливы записи $A^*\neq B^*$, $A^*\cap B^*=\{<a_1, 1>, <a_2, 1>, <a_3, 2>, <a_4, 1>, <a_5, 2>\}$, $A^*\cup B^*=\{<a_1, 2>, <a_2, 2>, <a_3, 2>, <a_4, 1>, <a_5, 3>\}$, $A^*\setminus B^*=\{<a_1, 1>, <a_5, 1>\}$, $A^*\Delta B^*=\{<a_1, 1>, <a_2, 1>, <a_3, 1>, <a_5, 1>\}$.

Примечание

На основе обобщенных характеристических функций для мультимножеств могут быть определены некоторые дополнительные операции, аналогичные арифметическим операциям.

Отношения и способы их задания

Понятие отношения наряду с понятием самого множества является фундаментальным не только в теории множеств, но и в других теоретических и приклад-

ных дисциплинах. Следует помнить, что это понятие часто заменяется терминами связь, ассоциация, взаимосвязь или соотношение. В общем случае отношение определяется как любое подмножество упорядоченных кортежей, построенных из элементов некоторых исходных множеств. При этом под *кортежем* понимается просто набор или список упорядоченных элементов.

Хотя и существует некоторая неоднозначность в принятых обозначениях, кортеж из двух элементов удобно обозначать как $<a_1, a_2>$, из трех элементов — $<a_1, a_2, a_3>$ и т. д. Кортеж из k элементов будем обозначать через $<a_1, a_2,\ldots, a_k>$ и говорить, что он имеет длину k, где k — некоторое натуральное число. При этом отдельные элементы кортежа могут принадлежать как одному множеству, так и различным множествам. Важно иметь в виду, что порядок выбора элементов для построения кортежей строго *фиксирован* для каждого конкретного случая. Речь идет о том, что первый элемент всегда выбирается из первого множества, второй — из второго и т. д.

Совокупность всех кортежей длины 2, образованных из элементов двух произвольных (конечных или бесконечных) множеств A и B, образует множество, которое получило специальное название — *декартово произведение* двух множеств. Данное произведение названо в честь известного французского философа и математика Рене Декарта (1596—1650). Формально декартово произведение определяется как: $A\times B=\{<a_i, b_j>| a_i\in A, b_j\in B\}$. Поскольку декартово произведение является множеством в обычном смысле, то для него справедливы все теоретико-множественные свойства и операции. В частности, мощность декартова произведения двух конечных множеств $A=\{a_1, a_2,\ldots, a_n\}$ и $B=\{b_1, b_2,\ldots, b_m\}$ равна произведению (арифметическому) мощностей этих множеств: $\text{card}(A\times B)=n\cdot m$. Этот факт непосредственно следует из простого подсчета всех кортежей, составленных из элементов этих множеств.

Декартово произведение может быть расширено на случай произвольного конечного числа множеств: $A_1\times A_2\times\ldots\times A_k$ на основе рассмотрения всех кортежей длины k, составленных из элементов этих множеств. С другой стороны, в качестве множеств декартова произведения может выступать и одно множество A. Соответствующее декартово произведение примет вид: $A\times A$ или $A\times A\times\ldots\times A$ (k раз). В последнем случае иногда используется запись: $\prod_{i=1}^{k} A$.

В общем случае *отношением*, заданным на множествах A_1, A_2,..., A_k (иногда говорят — над множествами), называется некоторое подмножество декартова произведения этих множеств. Другими словами, если обозначить произвольное отношение через Q, то по определению $Q\subseteq A_1\times A_2\times\ldots\times A_k$. При этом не исключается случай равенства отношения самому декартову произведению множеств.

Чтобы характеризовать количество множеств, на основе которых строится то или иное отношение, принято называть отношение между двумя множествами — *бинарным*, между тремя множествами — *тернарным*, а в общем случае — *k-арным* отношением. В каждом конкретном случае рассматриваемое отношение неявно предопределяет некоторый способ или семантику выбора отдельных элементов

из одного или нескольких множеств для образования соответствующего множества кортежей.

Примечание

Поскольку отношение по своей сути является множеством, то каждому отношению будет соответствовать некоторая характеристическая функция этого отношения χ_Q, заданная на декартовом произведении исходных множеств данного отношения. При этом, если кортеж $<a_1, a_2,\ldots, a_k>$ входит или принадлежит отношению Q, то значение характеристической функции этого отношения χ_Q для данного кортежа равно 1, и 0 — в противном случае.

Существуют различные способы, которыми могут быть заданы те или иные конкретные отношения. Наибольшее распространение из них получили следующие.

1. Непосредственное *перечисление кортежей* отношения. Как и для случая одного множества, этот способ может быть использован только для задания конечных отношений произвольной арности с небольшим количеством кортежей.
2. *Матричный* способ. Основан на представлении отношения в форме матрицы, строки которой соответствуют одному из множеств отношения, а столбцы — другому множеству. При этом элементы матрицы принимают одно из двух значений: 1 или 0, в зависимости от того, принадлежит или не принадлежит соответствующий кортеж данному отношению. Как не трудно представить, данный способ может быть использован только для задания конечных бинарных отношений. Иногда определенную таким образом матрицу называют *матрицей бинарного отношения*, которую будем обозначать через $\mathbf{M_Q}$.
3. *Табличный* способ, с одной стороны, может рассматриваться как разновидность матричного, поскольку конечная матрица всегда может быть представлена в форме таблицы. С другой стороны, при представлении отношений (не обязательно бинарных) в форме таблиц удобно указывать только те кортежи, которые принадлежат рассматриваемому отношению. Вообще говоря, таблицы отношений являются центральным объектом теории и практики *реляционных баз данных*. Данная тематика, в рамках которой разработаны специальные средства манипулирования данными (реляционная алгебра, правила нормализации, язык SQL, коммерческие базы данных и пр.) выходит за рамки настоящей книги.
4. *Графический* способ. В общем случае основан на представлении отношения с использованием некоторой графической нотации или системы специальных обозначений. Наиболее распространены две разновидности этого способа, которые подходят для задания бинарных отношений.
   - Представление отношения в виде некоторой геометрической фигуры (кривой или совокупности отдельных точек) на плоскости или в пространстве. При этом могут использоваться различные системы координат. График математической функции, такой как, например, парабола $y=x^2$, является примером этого способа задания отношения. Здесь функция рассматривается как частный случай отношения, о чем будет сказано ниже. Этот способ зачастую является единственно возможным для визуализации бесконечных отношений.

- Представление отношения в виде ориентированного (направленного) графа. Хотя основные понятия теории графов получили свое развитие задолго до появления теории множеств как самостоятельной научной дисциплины, формальное определение графа удобно представить в теоретико-множественных терминах. Рассмотрим этот способ более подробно.

  Ориентированным графом называется совокупность двух множеств: множества точек или *вершин* и множества соединяющих их линий или дуг. Формально граф задается в виде двух конечных множеств: $G=(V, E)$, где $V=\{v_1, v_2,\ldots, v_n\}$ — множество вершин графа, $E=\{e_1, e_2,\ldots, e_m\}$ — множество дуг графа. Вершины графа изображаются точками, а дуги — отрезками прямых линий со стрелкой на одном из концов. Рядом с вершинами и дугами записываются условные обозначения соответствующих вершин и дуг, что позволяет идентифицировать их однозначным образом. Натуральное число n определяет общее количество вершин конкретного графа, а натуральное число m — общее количество дуг графа.

  Следует заметить, что в общем случае не все вершины в графе могут соединяться между собой. Именно этот факт ставит в соответствие каждому ориентированному графу некоторое бинарное отношение Q_G, состоящее из всех пар вида $<v_i, v_j>$, где $v_i, v_j \in V$. При этом пара $<v_i, v_j>$ принадлежит соответствующему отношению Q_G в том и только в том случае, если вершины v_i и v_j соединяются в графе G некоторой дугой $e_k \in E$, направленной из вершины v_i в вершину v_j. При графическом задании отношения $Q \subseteq A \times A$ каждому элементу множества $a_i \in A$ будет соответствовать отдельная вершина $v_i \in G$ графа этого отношения, а каждому кортежу отношения $<a_i, a_j> \in Q$ будет соответствовать дуга графа $<v_i, v_j> \in E$ с началом в вершине v_i и концом в вершине v_j.

Примечание

Вообще говоря, графы бывают различных типов. Кроме рассмотренных выше ориентированных графов, в теории графов рассматриваются *неориентированные* графы, т. е. такие графы, у которых соединяющие вершины ребра не имеют направления или ориентации. Неориентированные графы удобно считать частным случаем ориентированных, у которых каждая дуга имеет дугу противоположной ориентации. В современной теории графов рассматриваются также и *бесконечные* графы, т. е. графы с бесконечным числом вершин. Бесконечные графы выходят за рамки тематики настоящей книги. Именно поэтому задание отношения с помощью графа используется только для конечных отношений. В общем случае графы широко используются для представления информации о структуре систем и процессов. Примерами подобных графических моделей могут служить: схемы автомобильных дорог региона, которые соединяют отдельные населенные пункты; схемы телекоммуникаций, используемых для передачи информации между отдельными узлами системы; схемы программ, на которых указываются варианты ветвления вычислительного процесса. Общей для всех подобных моделей является возможность представления информации в графическом виде. Достоинством данного способа является наглядность представления отношений и возможность их визуализации. При этом отдельные модели могут обладать дополнительной семантикой и специальными обозначениями, характерными для той или иной предметной области.

5. Задание отношения на основе явного указания некоторого *свойства*, которым должны обладать все элементы рассматриваемого отношения. Этот способ аналогичен описанному ранее способу для множеств и подходит для задания как конечных, так бесконечных отношений. В этом случае вводится в рассмотрение некоторое характеристическое свойство, которое может быть записано в виде *многоместного предиката* $P(<x_1, x_2,..., x_k>)$. Данный предикат $P(<x_1, x_2,..., x_k>)$ определяется на декартовом произведении $A_1 \times A_2 \times ... \times A_k$. При этом предикат может принимать одно из двух значений истинности: "*истина*" или "*ложь*". Если кортеж $<a_1, a_2,..., a_k>$ принадлежит рассматриваемому отношению Q, то соответствующий предикат $P_Q(<a_1, a_2,..., a_k>)$ принимает значение "*истина*". Если же кортеж $<a_1, a_2,..., a_k>$ не принадлежит отношению Q, то соответствующий предикат $P_Q(<a_1, a_2,..., a_k>)$ принимает значение "*ложь*". Тогда в общем случае отношение Q может быть записано в виде: $Q=\{<a_1, a_2,..., a_k> \mid P_Q(<a_1, a_2,..., a_k>), <a_1, a_2,..., a_k> \in A_1 \times A_2 \times ... \times A_k\}$.

П р и м е р П1.4. Для иллюстрации различных способов задания отношений рассмотрим в качестве примера фрагмент расписания международных авиарейсов аэропорта Пулково-2 (Санкт-Петербург, 2001 г.). Для простоты ограничим наше внимание только номерами авиарейсов, аэропортами назначения и типами самолетов. С этой целью рассмотрим следующие базисные множества: множество номеров авиарейсов $A=\{$LH6370, MA232, Z8221, Z8229, Z8257, Z8277, Z8285$\}$, множество аэропортов назначения $B=\{$*Амстердам*, *Афины*, *Барселона*, *Каир*, *Нью-Йорк*, *Париж*, *Хельсинки*$\}$, множество типов самолетов $C=\{$Боинг-737, Боинг-747, Ил-86, Ту-134, Ту-154$\}$. Одним из бинарных отношений $Q_1 \subseteq A \times B$ является отношение, устанавливающее соответствие между каждым номером авиарейса и аэропортом назначения. Другим отношением $Q_2 \subseteq A \times C$ является отношение между номерами авиарейсов и типом самолета, осуществляющим перелет по соответствующему маршруту.

Традиционно подобная информация записывается в форме таблиц, которые являются исходными объектами представления информации в реляционных базах данных. В нашем простом случае эти таблицы имеют следующий вид (табл. П1.1 и П1.2), что соответствует табличному способу задания этих отношений.

Таблица П1.1. *Авиарейсы и аэропорты назначения*

Номер авиарейса	Аэропорт назначения
Z8277	Амстердам
MA232	Афины
Z8221	Барселона
Z8285	Каир
LH6370	Нью-Йорк
Z8257	Париж
Z8229	Хельсинки

Таблица П1.2. *Авиарейсы и типы самолетов*

Номер авиарейса	Тип самолета
Z8277	Ту-154
MA232	Боинг-737
Z8221	Ту-134
Z8285	Ту-134
LH6370	Боинг-747
Z8257	Ил-86
Z8229	Ту-154

Первый из указанных выше способов задания отношений — непосредственное перечисление кортежей. В этом случае рассматриваемые отношения могут быть записаны в следующем виде: Q_1={<Z8277, *Амстердам*>, <MA232, *Афины*>, <Z8221, *Барселона*>, <Z8285, *Каир*>, <LH6370, *Нью-Йорк*>, <Z8257, *Париж*>, <Z8229, *Хельсинки*>}; Q_2={<Z8277, Ту-154>, <MA232, Боинг-737>, <Z8221, Ту-134>, <Z8285, Ту-134>, <LH6370, Боинг-747>, <Z8257, Ил-86>, <Z8229, Ту-154>}.

Второй из способов — матричный, с использованием которого эти отношения могут быть заданы в виде двух матриц $\mathbf{M}_1$ и $\mathbf{M}_2$, первая из которых соответствует отношению Q_1, а вторая — отношению Q_2. При этом неявно предполагается, что на исходных базисных множествах введено естественное отношение порядка, которое соответствует перечислению элементов в множествах A, B, C.

$$\mathbf{M}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{M}_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Примечание

Напомним, что при задании отношений матричным способом строки матрицы соответствуют элементам первого множества, а столбцы матрицы — элементам второго множества. В этом случае каждый из элементов матрицы может принимать одно из двух возможных значений: 0 или 1 в зависимости от того, принадлежит или нет соответствующий кортеж исходному отношению. Как нетрудно заметить, количество единиц в матрице отношения в точности равно количеству различных кортежей, из которых состоит то или иное отношение.

Третий из способов задания отношений (табличный) был использован для исходной записи рассматриваемых отношений.

Рассмотрим четвертый способ задания отношений — графический. В этом случае отношения Q_1 и Q_2 могут быть заданы в виде двух ориентированных графов: $G_1=(V_1, E_1)$, $G_2=(V_2, E_2)$. При этом множество вершин V_1 графа G_1 должно соответствовать объединению множеств $A \cup B$, а множество вершин V_2 графа G_2 — объединению множеств $A \cup C$. Соответствующие графы отношений представлены на рис. П1.11.

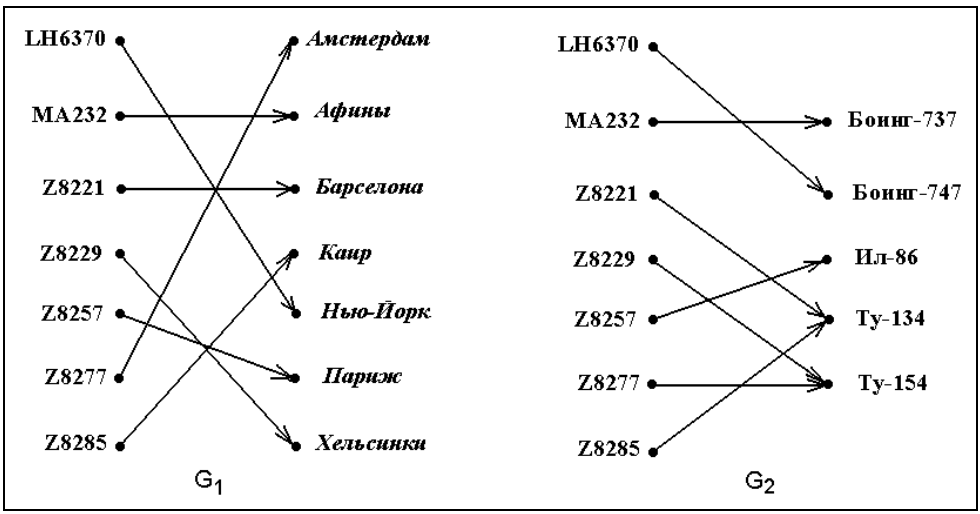


Рис. П1.11. Графы отношений Q_1 и Q_2

Что касается пятого способа задания отношений, то применительно к рассматриваемому примеру для кортежей отношений Q_1 и Q_2 нельзя выявить общие свойства или характеристические признаки. Поскольку в этом случае для задания самих предикатов $P_1(<x_1, x_2>)$ и $P_2(<x_1, x_2>)$ отношений Q_1 и Q_2 необходимо использовать один из способов задания соответствующих отношений, этот способ из соображений конструктивности здесь не приводится.

Операции над бинарными отношениями

Поскольку отношения являются некоторыми специальным образом определенными множествами, то для них в полной мере применимы все теоретико-множественные операции: пересечение, объединение, разность, симметрическая разность и дополнение. В то же время для отношений могут быть определены дополнительные операции, не имеющие аналогов среди множеств в общем случае. Ограничив наше внимание бинарными отношениями, рассмотрим унарную операцию "*обратное отношение*" и бинарную операцию *композиции отношений*.

Обратное отношение. Пусть задано некоторое бинарное отношения $Q \subseteq A \times B$. *Обратным отношением* Q^{-1} к отношению Q называется такое бинарное отношение, заданное на декартовом произведении $B \times A$, которое содержит кортеж элементов $<b_j, a_i>$ ($b_j \in B$, $a_i \in A$) тогда и только тогда, когда $<a_i, b_j> \in Q$. Другими словами: $Q^{-1} = \{<b_j, a_i> | <a_i, b_j> \in Q, a_i \in A, b_j \in B\}$.

В качестве примера рассмотрим обратное отношение Q_2^{-1} к отношению $Q_2 \subseteq A \times C$ из примера П1.4. В этом случае $Q_2^{-1} \subseteq C \times A$ и может быть задано матричным способом в виде:

$$M_{Q_2^{-1}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Примечание

Как нетрудно заметить, матрица обратного отношения может быть получена транспонированием матрицы исходного бинарного отношения, а граф обратного отношения может быть получен обращением стрелок (изменением их ориентации на противоположную) графа исходного отношения и возможно перемещением некоторых его вершин для большей наглядности.

Композиция отношений. Пусть задано два бинарных отношения $Q_1 \subseteq A \times B$ и $Q_2 \subseteq B \times C$ (правое базисное множество отношения Q_1 должно быть обязательно равно левому базисному множеству отношения Q_2). *Композицией бинарных отношений* Q_1 и Q_2 (обозначается $Q_1 \bullet Q_2$) называется такое бинарное отношение Q, которое заданно на декартовом произведении $A \times C$ и содержит кортеж элементов $<a_i, c_k>$ ($a_i \in A$, $c_k \in C$) тогда и только тогда, когда существует элемент $b_j \in B$, такой что одновременно $<a_i, b_j> \in Q_1$ и $<b_j, c_k> \in Q_2$. Операцию композиции бинарных отношений иногда называют *произведением* отношений.

В качестве примера рассмотрим композицию конечных отношений Q_2^{-1} и Q_1, где Q_2^{-1} — обратное к отношению $Q_2 \subseteq A \times C$ из примера П1.3. В этом случае $Q_2^{-1} \subseteq C \times A$, $Q_1 \subseteq A \times B$, что обеспечивает выполнение формального условия, необходимого для этой операции. Результат композиции этих отношений, который содержательно устанавливает соответствие между типами самолетов и аэропортами назначения, может быть представлен в матричном виде следующим образом:

$$M_{Q_2^{-1} \bullet Q_1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Для наглядности представим этот же результат в форме таблицы (табл. П1.3).

Таблица П1.3. *Типы самолетов и аэропорты назначения*

Тип самолета	Аэропорт назначения
Боинг-737	Афины
Боинг-747	Нью-Йорк
Ил-86	Париж
Ту-134	Барселона, Каир
Ту-154	Амстердам, Хельсинки

Примечание

Как нетрудно заметить, матрица композиции бинарных отношений может быть получена произведением матрицы первого бинарного отношения на матрицу второго бинарного отношения. Равенство правого базисного множества первого отношения и левого базисного множества второго отношения гарантирует согласование размерности соответствующих матриц. Однако при выполнении операции сложения над элементами матриц следует придерживаться следующих правил: 0+0=0, 0+1=1+0=1, 1+1=1. Операция умножения выполняется обычным образом: $0 \cdot 0=0$, $0 \cdot 1=1 \cdot 0=0$, $1 \cdot 1=1$. Очевидно, эти бинарные операции эквивалентны операциям max и min, что устанавливает концептуальную связь с операцией композиции нечетких отношений, рассматриваемой в *главе 4*.

Отображение

Бинарное отношение F, заданное на декартовом произведении $A \times B$, называется *отображением*, если из одновременного выполнения условий $<a_i, a_j> \in F$ и $<a_i, a_s> \in F$ всегда следует равенство вторых элементов этих кортежей, а именно, $a_j=a_s$. Другими словами, каждому из элементов множества A отображение F ставит в соответствие не более одного элемента множества B. В этом случае говорят, что отображение F действует *из* множества A в множество B. Для формальной записи отображения используется следующее обозначение $F: A \to B$, при этом не исключается случай, когда $A=B$.

С точки зрения теории множеств известное в математике понятие *функции* является частным случаем отображения, когда множества A и B являются числовыми. Общепринятый способ обозначения функциональной зависимости малыми латинскими буквами часто используется и в теории множеств. В этом случае отображение может быть записано в форме $f: A \to B$ или более привычной $y=f(x)$, что, если не указано дополнительно, не имеет принципиального значения. Обе эти формы записи отображений (функций) будут использоваться далее как эквивалентные.

Понятие отображения допускает обобщение на декартово произведение произвольного конечного числа множеств слева от стрелки. Поэтому в общем случае отображение записывается в виде $f: A_1 \times A_2 \times \ldots \times A_k \to B$ и ставит в соответствие каждому кортежу $<a_1, a_2, \ldots, a_k> \in A_1 \times A_2 \times \ldots \times A_k$ некоторый единственный элемент b из множества B. Известное в алгебре понятие *операция* является частным случаем такого отображения, когда все множества $A_1, A_2, \ldots, A_k$ и B равны. В этом случае операция, точнее, *k-местная операция*, может быть записана в форме $f: A \times A \times \ldots \times A \to A$ или более привычной $y = x_1 x_2 \ldots x_k$. Так, например, арифметическая операция сложения есть не что иное, как бинарная операция $+: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, где через $\mathbb{R}$ обозначено множество всех действительных чисел.

При анализе отображений часто вводят в рассмотрение два отдельных множества, которые характеризуют особенности структурного строения отображений. Во-первых, отображение f может быть задано не на всем множестве A, а только на некотором его подмножестве $D_f \subseteq A$. В этом случае множество D_f называется *областью (множеством) определения* отображения f. Если же $D_f = A$, то отображение f (соответственно, функцию) называют *всюду определенным*. Во-вторых, подмножество $Im_f \subseteq B$, определяемое как $Im_f = \{f(x) | \ x \in D_f\}$, получило название *области (множества) значений* отображения f. Здесь можно отметить тот случай, когда множество обозначено двумя буквами (сокращение от англ. *image*).

Имеются некоторые дополнительные понятия, связанные с произвольным отображением $f: A \to B$. Зафиксируем произвольный элемент $a \in D_f$. По определению этому элементу a отображение f ставит в соответствие некоторый единственный элемент $b \in Im_f$. При этом элемент b называется *образом* элемента a при отображении f, а сам элемент a — *прообразом* элемента b при отображении f.

Примечание

Как нетрудно заметить, характеристическая функция множества по определению является отображением $\chi_A : X \to \{0, 1\}$, которое задается выражением (П1.1). Здесь в качестве области определения отображения используется некоторый универсум X, а в качестве области значений — двухэлементное множество {0, 1}. Важно понимать, что задание в качестве области значений множества {0, 1} гарантирует, что никаких других значений, кроме 0 и 1, данная функция принимать не может.

Свойства бинарных отношений, заданных на одном базисном множестве

Как уже было упомянуто выше, в теории множеств физическая природа элементов, из которых могут быть образованы множества и отношения, не имеет принципиального значения. Предметом исследования являются такие особенности множеств и отношений, которые характеризуют отдельные абстрактные свойства их структуры. В контексте нечеткого моделирования наибольший интерес представляют такие общие свойства бинарных отношений, заданных на одном

множестве, как рефлексивность, симметричность и транзитивность. Эти свойства играют важную роль при определении некоторых специальных отношений. Наш интерес к этим отношениям обусловлен тем обстоятельством, что эти свойства допускают естественное нечеткое обобщение, что широко используется в нечетком моделировании.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *рефлексивным*, если каждый из кортежей $<a_i, a_i>$ принадлежит отношению Q. Формально это можно записать в следующем виде: Q — рефлексивно тогда и только тогда, когда $<a_i, a_i> \in Q$, $\forall a_i \in A$. Следует заметить, что главная диагональ матрицы рефлексивного отношения состоит из одних 1.

Примечание

В определении рефлексивного отношения использован специальный символ "$\forall$", который называется *квантором общности* и читается "*для любого, для каждого, для всех*" (этот символ ассоциируется с перевернутой первой буквой английского *All*). Квантор общности широко применяется в различных логических исчислениях, включая нечеткую логику. Здесь его употребление обусловливается соображениями удобства и сокращением формальных записей.

Кроме квантора общности $\forall$, имеется двойственный ему *квантор существования*, который обозначается символом "$\exists$" (этот символ ассоциирован с первой буквой англ. *Exist*). В математических и логических выражениях он читается как "*существует, найдется хотя бы один*". При этом не исключается случай, когда искомых элементов может оказаться несколько, а в предельном случае — даже все. Важно понимать, что применение квантора существования гарантирует наличие *хотя бы одного* элемента, удовлетворяющего заданному логическому условию. Более подробно семантика использования этих кванторов рассмотрена в *приложении 2*.

Примечание

При использовании квантора существования ничего не говорится о способе определения указанного элемента (или элементов, которых, вообще говоря, может быть несколько). Данное обстоятельство послужило источником критики классической теории множеств и целых разделов математики, особенно теорем существования, со стороны так называемого *конструктивного* направления. Однако эти аспекты теории множеств лежат за пределами тематики настоящей книги.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *антирефлексивным*, если ни один из кортежей $<a_i, a_i>$ не принадлежит отношению Q. Формально это можно записать в следующем виде: Q — антирефлексивно тогда и только тогда, когда $<a_i, a_i> \notin Q$, $\forall a_i \in A$. Следует заметить, что главная диагональ матрицы антирефлексивного отношения состоит из одних 0.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *симметричным*, если наряду с каждым кортежем $<a_i, a_j>$, принадлежащим отно-

шению Q, этому отношению принадлежит и кортеж $<a_j, a_i>$, в котором элементы $a_i, a_j \in A$ взяты в обратном порядке. Формально это можно записать в следующем виде: Q — симметрично тогда и только тогда, когда $<a_i, a_j> \in Q$ и $<a_j, a_i> \in Q$ одновременно. Следует заметить, что матрица симметричного отношения симметрична относительно главной диагонали.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *асимметричным*, если из двух кортежей $<a_i, a_j>$ и $<a_j, a_i>$ по меньшей мере один не принадлежит отношению Q (а может быть и оба). Формально это можно записать в следующем виде: Q — асимметрично тогда и только тогда, когда условия $<a_i, a_j> \in Q$ и $<a_j, a_i> \in Q$ не могут выполняться одновременно.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *антисимметричным*, если оба кортежа $<a_i, a_j>$ и $<a_j, a_i>$ могут принадлежать отношению Q только в одном случае, а именно, когда $a_i = a_j$. Формально это можно записать в следующем виде: Q — антисимметрично тогда и только тогда, когда условия $<a_i, a_j> \in Q$ и $<a_j, a_i> \in Q$ могут выполняться одновременно только при $a_i = a_j$.

Бинарное отношение Q, заданное на декартовом произведении $A \times A$, называется *транзитивным*, если из условия принадлежности двух кортежей $<a_i, a_j>$ и $<a_j, a_s>$ всегда следует, что данному отношению принадлежит и кортеж $<a_i, a_s>$. Формально это можно записать в следующем виде: Q — транзитивно тогда и только тогда, когда из условий $<a_i, a_j> \in Q$ и $<a_j, a_s> \in Q$ всегда следует выполнение условия: $<a_i, a_s> \in Q$.

Проиллюстрируем свойства отношений на следующем примере.

П р и м е р П1.5. Каждая бизнес-система имеет некоторую структуру подчинения своих сотрудников согласно занимаемым ими должностям. Рассмотрим подобное отношение "*субординации*" на множестве сотрудников в общем случае. Данное отношение антирефлексивно (сотрудник не может быть подчинен самому себе); асимметрично (если сотрудник a_i подчинен сотруднику a_j, то сотрудник a_j не может быть подчинен сотруднику a_i) и транзитивным (если сотрудник a_i подчинен сотруднику a_j, а сотрудник a_j подчинен сотруднику a_s, то сотрудник a_i подчинен сотруднику a_s).

Рассмотренный пример иллюстрирует тот важный факт, что те или иные отношения, рассматриваемые в самом общем контексте, могут одновременно обладать несколькими из введенных в рассмотрение свойств.

Некоторые специальные виды бинарных отношений, заданных на одном базисном множестве

Совместное наличие нескольких свойств у тех или иных бинарных отношений позволяет выполнить их дальнейшую специализацию с целью получения более конструктивных результатов относительно наличия целого ряда дополнительных свойств и уточнения характера взаимосвязей элементов этих бинарных отношений.

Отношение строгого частичного порядка

Бинарное отношение Q, заданное на одном базисном множестве $Q \subset A \times A$, называется отношением *строгого частичного порядка*, если оно одновременно является антирефлексивным, асимметричным и транзитивным. Если дополнительно отношение удовлетворяет условию: для любой пары элементов $<a_i, a_j>$, где $a_i, a_j \in A$, справедливо либо $<a_i, a_j> \in Q$, либо $<a_j, a_i> \in Q$, то такое отношение называется отношением *строгого линейного порядка*.

Примерами отношений строгого частичного порядка могут служить отношение субординации на множестве сотрудников некоторой бизнес-системы (пример П1.4), а также отношение включения на множестве всех подмножеств (булеане) некоторого базисного множества. Примерами отношений строгого линейного порядка могут служить отношение "*строго больше*" на множестве действительных чисел, а также отношения типа "*быть выше ростом*", "*иметь более высокую скорость движения*" на соответствующих базисных множествах.

Отношение толерантности

Бинарное отношение Q, заданное на одном базисном множестве $Q \subset A \times A$, называется отношением *толерантности*, если оно является рефлексивным и симметричным.

Отношение толерантности иногда называют отношением покрытия или сходства, поскольку оно связано с определением соответствующей системы множеств. А именно, система подмножеств $\Im(A) = \{A_k \mid A_k \subseteq A\}$ множества A называется *покрытием*, если выполняется следующее условие:

$$\bigcup_k A_k = A, \qquad (A_k \in \Im), \qquad (\text{П1.18})$$

т. е. объединение всех или части подмножеств из $\Im(A)$ совпадает с исходным множеством A (или "*покрывает*" исходное множество A).

Примером отношения толерантности может служить факт знакомства среди некоторого множества субъектов. Для конкретности рассмотрим в качестве исходного множества $A=\{a_1, a_2, a_3,..., a_n\}$ совокупность сотрудников некоторой достаточно крупной бизнес-системы. Сформируем бинарное отношение Q на этом множестве, включив в него такие пары элементов $<a_i, a_j>$, где $a_i, a_j \in A$, для которых выполняется условие: "*сотрудник a_i знаком с сотрудником a_j*". Легко проверить, что это отношение является симметричным, поскольку если "*сотрудник a_i знаком с сотрудником a_j*", то определенно и "*сотрудник a_j знаком с сотрудником a_i*". Свойство рефлексивности также выполняется, поскольку по определению будем считать, что каждый из сотрудников знаком с самим собой.

С другой стороны, это же отношение порождает некоторое покрытие $\Im(A)=\{A_k \mid A_k \subseteq A\}$ множества A, если в качестве подмножеств $A_k \subseteq A$ взять указанные выше пары элементов $A_k=\{a_i, a_j\}$ — знакомых между собой сотрудников.

Отношение эквивалентности

Бинарное отношение Q, заданное на одном базисном множестве $Q \subset A \times A$, называется отношением *эквивалентности*, если оно одновременно является рефлексивным, симметричным и транзитивным.

Отношение эквивалентности также называют отношением разбиения, поскольку оно связано с определением соответствующей системы множеств. А именно, система подмножеств $\Re(A) = \{A_k \mid A_k \subseteq A\}$ множества A называется *разбиением*, если выполняются следующие условия:

$$\bigcup_k A_k = A, \qquad (A_k \in \Re), \qquad (\text{П1.19})$$

$$A_l \bigcap A_m = \varnothing, \qquad (\forall A_l, A_m \in \Re), \qquad (\text{П1.20})$$

т. е. объединение всех или части подмножеств из $\Re(A)$ совпадает с исходным множеством A (или "*покрывает*" исходное множество A), при этом подмножества разбиения попарно не пересекаются между собой. В этом случае отдельные множества $A_k \in \Re$ разбиения получили специальное название — *классы* разбиения.

Нетрудно заметить, что каждое разбиение является в то же время и покрытием, которое дополнительно удовлетворяет свойству (П1.20). Вообще говоря, это свойство существенно обогащает структуру соответствующих отношений эквивалентности, позволяя утверждать, что каждый из элементов исходного множества A может принадлежать не более одному из классов разбиения. Свойство (П1.19) гарантирует наличие такого класса A_k для каждого элемента $a_i \in A$.

Примерами отношений эквивалентности могут служить отношение "*быть родственником*", "*иметь одинаковый рост*", "*иметь одинаковую цену*", "*обладать равным доходом*" и многие другие. В общем случае с отношением эквивалентности связаны логические правила "хорошей" классификации, которые используются в системном моделировании для концептуальной структуризации предметной области при построении моделей сложных систем.

Мультиотношение

В общем случае *мультиотношением,* заданным на множествах A_1, A_2,..., A_k (иногда говорят — над множествами), называется некоторое мультимножество кортежей элементов этих множеств.

Другими словами, если обозначить произвольное мультиотношение через Q^*, то по определению: $Q^* = \{<\boldsymbol{r}_1, \alpha_1>, <\boldsymbol{r}_2, \alpha_2>, \ldots, <\boldsymbol{r}_n, \alpha_n>, \ldots\}$, где $\boldsymbol{r}_i$ кортеж длины k из элементов $<a_1^i, a_2^i, \ldots, a_k^i>$, при этом: $a_1^i \in A_1$, $a_2^i \in A_2, \ldots$, $a_k^i \in A_k$, α_i — количество повторений кортежа $\boldsymbol{r}_i$ в этом мультиотношении Q^*.

Поскольку мультиотношение является, с одной стороны, мультимножеством, а с другой стороны, обобщением обычного отношения, для него оказываются справедливыми операции, определенные выше для мультимножеств. В общем случае

мультиотношения могут быть заданы перечислением кортежей мультимножества, матрицей мультиотношения, а также с помощью так называемого мультиграфа.

Ориентированным мультиграфом называется совокупность двух множеств: множества точек или *вершин* и мультимножества соединяющих их линий или дуг.

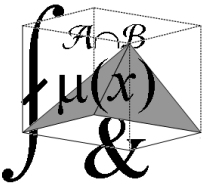
Формально мультиграф задается в виде двух конечных множеств: $G'=(V, E^*)$, где $V=\{v_1, v_2,\ldots, v_n\}$ — множество вершин графа, $E^*=\{<e_1, \alpha_1>, <e_2, \alpha_2>,\ldots, <e_m, \alpha_m>\}$ — мультимножество дуг мультиграфа.

Вершины мультиграфа изображаются точками, аналогично обычным графам, а дуги — отрезками прямых линий со стрелкой на одном из концов. Рядом с вершинами и дугами записываются условные обозначения соответствующих вершин и дуг. Натуральное число n определяет общее количество вершин конкретного мультиграфа, а натуральное число m — общее количество дуг мультиграфа.

Следует заметить, что в общем случае при графическом задании бинарного мультиотношения Q^* каждому элементу множества $a_i \in A$ будет соответствовать отдельная вершина $v_i \in G$ мультиграфа этого отношения, а каждому кортежу мультиотношения $\boldsymbol{r_k} \in Q^*$ будет соответствовать α_i количество дуг мультиграфа с началом в вершине v_i и концом в вершине v_j.

Понятие мультиграфа используется при графическом представлении структуры сетей Петри и их обобщений в *главе 10*.

Приложение 2



Основы математической логики

Л о г и к а как наука об особенностях и формах мышления имеет весьма древнюю историю, истоки которой берут начало с творчества таких античных философов, как Аристотель (384—322 до н. э.), Зенон (336—264 до н. э.), Демокрит (460—370 до н. э.). Изучая структуру и формы человеческой мысли, логика отвлекается от конкретного содержания, которое могут заключать в себе те или иные формы мысли. Тем самым логика абстрагируется от реальных объектов, предметов и явлений, которые являются элементами мыслительного процесса.

Начиная с античных философов, предметом логики принято считать изучение структуры и взаимосвязей форм, в рамках которых реализуется процесс человеческого мышления. Именно в этом аспекте логику называют также формальной логикой.

Разработка формально-логических средств для анализа логических основ математики привела к появлению математической логики. Идея построения универсального символического языка для математики и формализации на его основе процесса математического доказательства теорем принадлежит немецкому математику и философу Г. Лейбницу (1646—1716). Важный вклад в начальный этап разработки аппарата математической логики внесли Дж. Буль (1815—1864), Г. Фреге (1948—1925), Дж. Пеано (1858—1932), Б. Рассел (1872—1070), А. Уайтхед (1061—1947), Н. И. Лобачевский (1792—1856). Именно в этот период были сформулированы принципы аксиоматического метода в математике, а также разработаны основы логики высказываний и логики предикатов.

Важнейшим событием не только в истории логики, но и в науке в целом, стала программа обоснования математики на базе математической логики, с которой выступил на II Математическом Конгрессе математиков в 1900 г. немецкий математик Д. Гильберт (1862—1943). Сформулированные им проблемы математики на несколько десятилетий определили развитие некоторых направлений фундаментальной математики и математической логики. Именно этот период принято считать началом современного этапа развития математической логики, характерной особенностью которого является применение строгих математических средств для изучении формальных аксиоматических теорий. Наибольший вклад в разработку математической логики на этом этапе внесли К. Гёдель (1906—1978), А. Чёрч (1903—1995), А. М. Тьюринг (1912—1954), Г. Генцен (1909—1945), П. С. Новиков (1901—1975), А. И. Мальцев (1909—1967), А. А. Марков (1903—1979).

Отличие современной формальной логики от классической заключается не в предмете исследования, а в применяемых методах, которые основаны на математических средствах аксиоматизации и формализации точных рассуждений. Именно эти методы способствовали интенсивному развитию математической логики, в рамках которой были решены многие проблемы, такие как установление или доказательство полноты, непротиворечивости и разрешимости различных формально-логических теорий.

На развитие математической логики в последние десятилетия оказали влияние две основные тенденции.

С одной стороны, инженерная разработка вычислительных устройств потребовала развития математических средств моделирования и проектирования соответствующих технических устройств, что послужило стимулом к исследованию различных логико-алгебраических систем.

С другой стороны, интенсивное развитие проблематики искусственного интеллекта привело не только к разработке логико-лингвистических моделей представления знаний и появлению основанных на правилах экспертных систем (продукционных экспертных систем), но и созданию неклассических логик, которые специально предназначены для моделирования тех или иных аспектов человеческого мышления. В этом контексте нечеткая логика представляет собой вариант неклассической логики.

В контексте нечеткого моделирования наибольший интерес представляют три раздела математической логики, а именно: логика высказываний, логика предикатов (логика предикатов первого порядка) и продукционные системы (системы, основанные на правилах). Выбор этих тем определяется исключительно возможностью использования соответствующих базовых логических конструкций для обобщения в рамках нечеткой логики и нечеткого логического вывода.

Классическая логика высказываний

Хотя исторически тремя базовыми конструкциями формальной логики являются понятие, суждение и умозаключение, в математической логике главным образом исследуются формы взаимосвязей между логическими высказываниями или предложениями. Важное значение при этом имеют используемые для этих целей символические обозначения или синтаксис формально-логических теорий, а также логические формулы, которые могут быть построены на основе высказываний, предикатных и функциональных символов, логических связок и кванторов.

Основные понятия логики высказываний

В ы с к а з ы в а н и е. *Высказыванием (элементарным высказыванием)* в математической логике называется повествовательное предложение, выражающее законченную мысль, относительно которой мы можем судить истинна она или ложна, но не то и другое одновременно.

Важной особенностью логики высказываний является то, что само элементарное высказывание не подлежит какому-либо синтаксическому или лингвистическому анализу. Другими словами, ответ на вопрос: "Почему то или иное элементарное высказывание является истинным или ложным?" лежит за пределами логики и может служить предметом специальных исследований.

П р и м е р П2.1. Ниже приводятся примеры элементарных высказываний:

1. *Москва — столица Российской Федерации.*
2. *Сегодня пасмурная погода.*
3. 4 *является корнем уравнения*: y^2=16.
4. 4>10.
5. 10 *делится на* 5 *без остатка.*

Элементарные высказывания будем обозначать заглавными буквами: A, B, C, D (возможно, с индексами). По определению множество значений истинности высказываний состоит из двух элементов: {"*истина*", "*ложь*"}. Традиционно эти значения истинности обозначают буквами {**И**, **Л**}, при этом значению "*истина*" соответствует буква **И**, а значению "*ложь*" — буква **Л**.

Примечание

Иногда значения истинности обозначают цифрами {0, 1}, при этом значению "*истина*" соответствует цифра 1, а значению "*ложь*" — цифра 0. В общем случае то или иное обозначение значений истинности в математической логике не имеет принципиального значения, поскольку отражает лишь синтаксический аспект логических исчислений. Однако в контексте нечеткого моделирования для нас будет удобным использовать значения {0, 1}, которые в нечеткой логике естественным образом обобщаются до интервала [0, 1].

Применительно к рассмотренным выше примерам высказываний истинными являются высказывания: 1, 3, 5; а высказывание 4 является ложным. Относительно высказывания 2 ситуация представляется не столь тривиальной, поскольку его истинность зависит от того, какая в действительности погода на текущий день. Специфический характер подобного рода высказываний послужил основой для рассмотрения так называемых *переменных высказываний*, т. е. элементарных высказываний, которые в том или ином контексте могут принимать одно из двух значений истинности.

Примечание

Для оценки истинности произвольного элементарного высказывания можно ввести в рассмотрение специальное отображение T, которое действует из множества рассматриваемых элементарных высказываний U в двухэлементное множество: {**И**, **Л**} или {0, 1}, т. е. $T : U \rightarrow$ {**И**, **Л**} или $T : U \rightarrow \{0, 1\}$. В этом случае значение истинности некоторого элементарного высказывания $A \in U$ будем обозначать через T(A). Так, если обозначить элементарное высказывание (1) из примера П2.1 через A_1, то его истинность формально может быть записана как $T(A_1)$, а количественно равна 1, т. е. $T(A_1) = 1$.

В логике высказываний представляют интерес более сложные высказывания, которые состоят из элементарных, соединенных логическими связками или операциями.

Основные логические операции над высказываниями

Л о г и ч е с к о е о т р и ц а н и е. Это унарная логическая операция над элементарным высказыванием, результат которой является также высказыванием и принимает значение "*истина*", если исходное высказывание ложно и значение "*ложь*", если исходное высказывание истинно.

Используя символические обозначения и обозначив произвольное элементарное высказывание, например, через A, его отрицание записывается как: ¬A (читается как "*не A*", "*неверно, что A*").

Примечание

Кроме указанного обозначения в литературе можно встретить и другие обозначения отрицания, например: —, **~**, **not**, **не**.

В математической логике принято определять логические операции с помощью так называемых *таблиц истинности*, которые определяют истинность результата рассматриваемой операции для каждого из значений истинности исходного или исходных высказываний. Так операция логического отрицания полностью определяется своей таблицей истинности (табл. П2.1).

Таблица П2.1. *Таблица истинности логического отрицания*

A	¬A	A	¬A
И	Л	Л	И

Примером применения операции логического отрицания к высказыванию "*Москва — столица Российской Федерации*" будет высказывание "*Неверно, что Москва — столица Российской Федерации*", которое принимает значение "*ложь*". Отрицанием высказывания "4>10" будет высказывание "4≤10", которое принимает значение "*истина*".

Л о г и ч е с к а я к о н ъ ю н к ц и я (обозначается через "A∧B"). Это бинарная операция над двумя высказываниями, результат которой равен значению "*истина*", если истинны оба высказывания одновременно и значение "*ложь*" во всех остальных случаях.

Высказывание: A∧B читается — "*A и B*". Логическую конъюнкцию также называют логическим "И". Эта операция может быть определена соответствующей таблицей истинности (3-й столбец в табл. П2.2).

Примечание

Кроме указанного обозначения в литературе можно встретить и другие обозначения для конъюнкции, например: &, •, **and**, **и**.

Примером логической конъюнкции может служить составное высказывание: "*Москва — столица Российской Федерации и* 4 *является корнем уравнения*: y^2=16", которое принимает значение "*истина*". Результат логической конъюнкции высказываний "4>10 *и* 10 *делится на* 5 *без остатка*" принимает значение "*ложь*".

Логическая дизъюнкция (обозначается через "A∨B"). Это бинарная операция над двумя высказываниями, результат которой равен значению "*ложь*", если ложны оба высказывания одновременно и значение "*истина*" во всех остальных случаях.

Высказывание: A∨B читается — "*A или B*". Логическую дизъюнкцию также называют логическим *неисключающим* "ИЛИ". Эта операция также может быть определена соответствующей таблицей истинности (4-й столбец в табл. П2.2).

Примечание

Кроме указанного обозначения в литературе можно встретить и другие обозначения для дизъюнкции, например: |, **or**, **или**.

Примером логической дизъюнкции может служить составное высказывание: "*Москва — столица Российской Федерации или* 4 *является корнем уравнения*: y^2=16", которое принимает значение "*истина*". Результат логической дизъюнкции высказываний "4>10 *или* 10 *делится на* 5 *без остатка*" также принимает значение "*истина*".

Таблица П2.2. *Таблица истинности основных бинарных логических операций*

A	B	A∧B	A∨B	A⊃B	A≡B
И	И	И	И	И	И
И	Л	Л	И	Л	Л
Л	И	Л	И	И	Л
Л	Л	Л	Л	И	И

Логическая импликация (обозначается через "A⊃B"). Бинарная операция над двумя высказываниями, результат которой принимает значение "*ложь*" в случае истинности высказывания A и ложности высказывания B и значение "*истина*" во всех остальных случаях.

Высказывание: A⊃B читается — "*A влечет B*", "*из A следует B*", "*если A, то B*". Последнее прочтение для импликации использовать не целесообразно, чтобы отличать ее от правила продукции. Логическая импликация может быть определена соответствующей таблицей истинности (5-й столбец в табл. П2.2).

Примечание

Кроме указанного обозначения в литературе можно встретить и другие обозначения для импликации, например: ⇒, →. Что касается условных конструкций типа `if...then` (`если...то`), которые используются в различных языках программирования, то, как будет видно из последующего изложения, они применяются для обозначения правил продукций. Как будет показано далее, операцию импликации высказываний в общем случае следует отличать от правила продукции.

Примером логической импликации может служить составное высказывание: "*Если Москва — столица Российской Федерации, то* 4 *является корнем уравнения*: y^2=16", которое принимает значение "*истина*". Результат логической импликации высказываний "*Если* 4>10, *то* 10 *делится на* 5 *без остатка*" также принимает значение "*истина*". С другой стороны, составное высказывание: "*Если Москва — столица Российской Федерации, то* 4>10" принимает значение "*ложь*".

Импликация играет особую роль в процессе логических рассуждений. Первый ее операнд (высказывание) называется *посылкой* (или *антецедентом*), а второй — *заключением* (или *консеквентом*). Из таблицы истинности импликации можно заметить, что если посылка истинна, то импликация принимает значение истинности заключения. У тех, кто впервые сталкивается с этой логической связкой, может вызвать недоумение тот факт, что импликация может быть истинной даже тогда, когда ее посылка ложна (3-я и 4-я строки в 5-м столбце табл. П2.2). Тем не менее, в логике принято считать, что при импликации из лжи может следовать все, что угодно, а из истины — только истина. Следует также отметить, что в отличие от дизъюнкции и конъюнкции, импликация является некоммутативной логической связкой.

Примечание

Говоря о логической импликации, следует отметить, что она не заключает в себе никакой причины, временной или иной взаимосвязи между посылкой и заключением. Чтобы явно отметить этот факт, определенную подобным образом импликацию еще называют *материальной импликацией*. Эта связка является основной в математических рассуждениях, поскольку соединяет условие P и утверждение Q в теоремах. Однако в общем случае запись P⊃Q может оказаться некорректной, поскольку в качестве посылки и заключения могут использоваться произвольные высказывания, в том числе и не принадлежащие тому или иному логическому формализму. Именно по этой причине для обозначения логического следования в математических теоремах используется другое обозначение: P⇒Q.

Логическая эквивалентность (обозначается через "≡"). Это бинарная операция над двумя высказываниями, которая принимает значение "*истина*" в случае истинности или ложности обоих высказываний одновременно и значение "*ложь*" во всех остальных случаях.

Высказывание: A≡B читается — "*A эквивалентно B*", которая может быть определена соответствующей таблицей истинности (6-й столбец в табл. П2.2).

Примечание

Кроме указанного обозначения в литературе можно встретить и другие обозначения для эквивалентности, например: $\Leftrightarrow$, $\leftrightarrow$, $\approx$.

Примером логической эквивалентности может служить составное высказывание: "*Москва — столица Российской Федерации эквивалентно тому, что* 4 *является корнем уравнения*: y^2=16", которое принимает значение "*истина*". С другой стороны, результат логической эквивалентности высказываний "4>10 *эквивалентно тому, что* 10 *делится на* 5 *без остатка*" принимает значение "*ложь*".

Поскольку с помощью рассмотренных логических связок могут быть образованы достаточно сложные логические высказывания, для явного указания порядка их применения по аналогии с арифметическими операциями используются круглые скобки. Так, например, составное высказывание: $(P \supset Q) \supset (\neg Q \supset \neg P)$ остается истинным при любых конкретных высказываниях, которые можно подставить вместо букв P и Q. В этом случае обязательным условием корректности логических рассуждений является требование одновременной подстановки вместо одинаковых букв одних и тех же высказываний.

Примечание

Кроме рассмотренных логических связок существуют и другие бинарные логические операции. Поскольку каждая бинарная связка может быть задана своей таблицей истинности, которая имеет 4 строки, а каждая строка может содержать одно из двух значений истинности {**И**, **Л**}, то общее количество различных бинарных связок равно: 2^4=16. С другими логическими связками можно познакомиться в дополнительной литературе по математической логике.

Истинность произвольного высказывания может быть установлена с помощью построения так называемой таблицы истинности. *Таблицей истинности высказывания* называется специальным образом построенная таблица, содержащая значения истинности всех элементарных высказываний и результатов выполнения логических операций, входящих в состав данной составного высказывания.

Правило (алгоритм) построения таблицы истинности для произвольного высказывания следующее:

1. Определить все элементарные высказывания, входящие в состав рассматриваемого составного высказывания. Пусть число различных элементарных высказываний в составном высказывании равно k.
2. Определить последовательность выполнения логических операций в данном составном высказывании, при необходимости расставив для наглядности скобки. Пусть общее количество логических операций, которые используются с различными операндами, равно m.
3. Изобразить таблицу, содержащую 2^k +1 строк (первая из которых используется в качестве заголовка) и $k+m$ столбцов.

4. Первые k столбцов таблицы должны содержать все комбинации значений истинности элементарных высказываний, входящих в состав составного высказывания.
5. Оставшиеся столбцы заполняются слева направо в последовательности выполнения логических операций в составном высказывании, при этом каждой операции соответствует свой столбец таблицы. Тогда последний правый столбец построенной таким образом таблицы содержит искомые значения истинности рассматриваемого составного высказывания.

П р и м е р П2.2. В качестве примера рассмотрим процесс построения таблицы истинности для следующего составного высказывания: $((P \supset Q) \wedge (Q \supset R)) \supset (P \supset R)$.

Для начала определим общее число различных элементарных высказываний: $k=3$ и общее число логических операций, которые используются с различными операндами: $m=5$. Далее изобразим таблицу, содержащую $2^3+1=9$ строк и $3+5=8$ столбцов. В данной таблице (табл. П2.3) первые три столбца содержат все комбинации значений истинности элементарных высказываний данной формулы. Следующие четыре столбца таблицы содержат значения истинности отдельных частных высказываний рассматриваемого составного высказывания. Наконец последний столбец таблицы содержит значения истинности всего составного высказывания в целом.

Как нетрудно заметить, рассмотренное в примере П2.2 составное высказывание принимает значение "*истина*" для всех возможных значений истинности элементарных высказываний, входящих в состав этого составного высказывания. Последнее получило даже специальное название — *закон силлогизма*.

Таблица П2.3. *Таблица истинности формулы исчисления высказываний из примера П2.2*

P	Q	R	P⊃Q	Q⊃R	(P⊃Q)∧(Q⊃R)	P⊃R	((P⊃Q)∧(Q⊃R))⊃(P⊃Q)
И	И	И	И	И	И	И	И
И	И	Л	И	Л	Л	Л	И
И	Л	И	Л	И	Л	И	И
И	Л	Л	Л	И	Л	Л	И
Л	И	И	И	И	И	И	И
Л	И	Л	И	Л	Л	И	И
Л	Л	И	И	И	И	И	И
Л	Л	Л	И	И	И	И	И

Формальные теории

Более общий и строгий подход к установлению истинности произвольного высказывания основывается на рассмотрении формальных теорий, которые позволяют изучать высказывания с помощью так называемого аксиоматического метода.

В общем случае формальная или аксиоматическая теория $\mathcal{T}$ считается определенной, если выполнены следующие условия:

1. Задан некоторый алфавит A в виде не более чем счетного множества символов теории $\mathcal{T}$. При этом произвольная конечная последовательность символов теории называется *выражением* или *словом* теории $\mathcal{T}$. Множество всех выражений или слов теории $\mathcal{T}$ обозначим через $A(\mathcal{T})$.
2. Определено подмножество выражений теории $\mathfrak{F} \subseteq A(\mathcal{T})$, называемых *формулами* или правильно построенными выражениями теории $\mathcal{T}$.
3. Среди формул множества $\mathfrak{F}$ выделено некоторое подмножество формул $\mathfrak{A} \subseteq \mathfrak{F}$, которые называются аксиомами теории $\mathcal{T}$.
4. Задано некоторое конечное число отношений $R_1, R_2, \ldots, R_n$ между формулами, которые называются *правилами вывода*. Совокупность этих отношений обозначим через $\mathfrak{R}$.

Формальную теорию будем обозначать принятым в математике способом для обозначения математических объектов, а именно — в форме кортежа составляющих ее конструкций: $\mathcal{T}=<A, \mathfrak{F}, \mathfrak{A}, \mathfrak{R}>$. При этом слово "*формальная*" для краткости опускают.

В общем случае формула $G \in \mathfrak{F}$ формальной теории $\mathcal{T}$ называется *непосредственным следствием* формул $F_1, F_2, \ldots, F_k \in \mathfrak{F}$ по правилу R_i этой же формальной теории, если формулы $F_1, F_2, \ldots, F_k, G$ удовлетворяют отношению R_i. *Выводом* в формальной теории $\mathcal{T}$ называется всякая упорядоченная последовательность формул или кортеж произвольной длины $< F_1, F_2, \ldots, F_k >$, такая, что любая из формул последовательности является или аксиомой теории, или непосредственным следствием каких-либо предыдущих формул этой последовательности.

Формула $G \in \mathfrak{F}$ формальной теории $\mathcal{T}$ называется *теоремой* этой теории, если в $\mathcal{T}$ существует такой вывод, в котором последней формулой является G. Такой вывод называется *выводом теоремы* (формулы) G.

Одной из центральных проблем изучения формальных теорий является решение вопроса о том, является ли некоторая формула теории теоремой или нет. Наиболее конструктивным ответом на поставленный вопрос служит нахождение такой процедуры или алгоритма, с помощью которого для любой произвольной формулы теории можно установить, существует ли ее вывод в этой теории или нет.

Формула, для которой такая процедура существует, называется *разрешимой* в этой теории, в противном случае — *неразрешимой*. Другими словами, для неразрешимых формул отсутствует алгоритм выяснения свойства формулы быть теоремой в данной теории.

Исчисление высказываний как формальная теория

Как было отмечено выше, математическая логика изучает структуру логических рассуждений с формальной точки зрения, абстрагируясь от содержания фраз естественного языка, которыми могут быть выражены те или иные конкретные высказывания. Формальная теория, в основе которой лежит логика высказываний, получила название исчисления высказываний.

Исчисление высказываний представляет собой формальную теорию, в рамках которой составные высказывания являются формулами исчисления высказываний, а правила образования таких высказываний строго определяются некоторым рекурсивным образом.

Таким образом, *алфавит* исчисления высказываний содержит три группы символов:

1. A, B, C, P, Q, … — элементарные высказывания или атомы (буквы).
2. $\neg$, $\vee$, $\wedge$, $\supset$, $\equiv$ — символы логических операций или связок.
3. (,) — вспомогательные символы (скобки).

Формулой исчисления высказываний называется выражение, составленное из символов алфавита и удовлетворяющее следующим *правилам построения*:

1. Всякое элементарное высказывание является формулой исчисления высказываний *(базис)*.
2. Если P, Q — формулы исчисления высказываний, то выражения: $\neg$P, $\neg$Q, (P$\vee$Q), (P$\wedge$Q), (P$\supset$Q), (P$\equiv$Q) являются формулами исчисления высказываний *(индукционный шаг)*.
3. Формулой исчисления высказываний является любое выражение, которое может быть получено с помощью правил (1) и (2) *(ограничение)*.

Если при построении некоторой формулы другая формула используется как элемент в индуктивном шаге (2), то последняя называется *подформулой* исходной формулы.

Правила построения формул исчисления высказываний (1)—(3) позволяют строить формулы какой угодно сложности. Например, выражение: ((¬P∨Q) ⊃ ⊃ ((P∧Q) ≡ Q)) является формулой исчисления высказываний. Напротив, выражение: ((¬P∨Q) ⊃ ((∧Q) ¬P)) не является формулой исчисления высказываний, поскольку подвыражения (∧Q) и (∧Q) ¬P), входящие в его состав, не удовлетворяют второму из правил построения формул.

Множество формул исчисления высказываний является подмножеством множества всех выражений, составленных из алфавита исчисления высказываний. Поскольку последнее множество является *счетным* (можно установить взаимно однозначное соответствие между этим множеством и множеством натуральных чисел), следовательно, множество формул исчисления высказываний, являясь бесконечным, также является счетным.

Порядок применения логических операций в формулах исчисления высказываний устанавливается аналогично принятому в алгебре — слева направо. Иногда в дополнение к этому задается приоритет логических операций. Для исключения неоднозначности или изменения порядка выполнения операций используются вспомогательные символы — круглые скобки.

Примечание

Выше было показано, что в логике высказываний существует общая процедура, которая позволяет установить истинность произвольной формулы на основе построения ее таблицы истинности. Поскольку число всех возможных интерпретаций той или иной формулы является конечным, то методом полного перебора мы всегда можем установить выполнимость (общезначимость, противоречивость) любой формулы. Говоря более строго, проблема выявления общезначимости или выполнимости формулы исчисления высказываний является NP-полной. Это означает, что не существует (возможно, пока) эффективного алгоритма с полиномиальной оценкой трудоемкости, который бы позволил конструктивно решить указанную проблему. Тем не менее наличие рассмотренной выше процедуры построения таблиц истинности для формул позволяет говорить об *алгоритмической разрешимости* исчисления высказываний, что имеет важное значение в математической логике.

Приписывание конкретных значений истинности для входящих в некоторую формулу элементарных высказываний называется *интерпретацией* данной формулы. Другими словами, если задана некоторая формула исчисления высказываний G, в состав которой входят элементарные высказывания: A_1, A_2, A_3,..., A_n, то интерпретацией этой формулы является приписывание некоторой фиксированной комбинации значений истинности для этих элементарных высказываний, при которой каждому из A_i поставлено в соответствие либо значение "**И**", либо значение "**Л**", но не оба значения одновременно. Очевидно, подобная формула будет иметь 2^n возможных интерпретаций. Нетрудно заметить, что каждая строка таблицы истинности некоторой формулы представляет собой некоторую ее интерпретацию.

Если произвольная формула G исчисления высказываний истинна при некоторой интерпретации, то говорят, что эта формула *выполнима в этой интерпретации* или данная интерпретация *удовлетворяет* рассматриваемой формуле G. *Моделью* формулы G называется произвольная ее интерпретация, которая удовлетворяет этой формуле. Таким образом, семантическая выполнимость некоторой формулы означает существование для нее хотя бы одной модели.

Формула исчисления высказываний, которая принимает значение "*истина*" для всех возможных значений истинности входящих в состав этой формулы элементарных высказываний называется *тождественно истинной*.

Формула исчисления высказываний, которая принимает значение "*ложь*" для всех возможных значений истинности входящих в состав этой формулы элементарных высказываний называется *тождественно ложной* или *невыполнимой формулой*. Наконец, формула исчисления высказываний, которая принимает значение "*истина*" хотя бы для одной комбинации значений истинности входя-

щих в состав этой формулы элементарных высказываний, называется *семантически выполнимой формулой* или просто *выполнимой*.

П р и м е р П2.3. Примерами тождественно истинных формул, которые играют особую роль не только в исчислении высказываний, но и в математической логике в целом, являются следующие формулы:

1. $(P \supset Q) \supset (\neg Q \supset \neg P)$ — *закон контрапозиции*.
2. $(\neg P \supset Q) \wedge (\neg P \supset \neg Q) \supset P$ — *закон доказательства от противного*.
3. $\neg(\neg P) \equiv P$ — *закон двойного отрицания*.
4. $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$ — *закон отрицания дизъюнкции*.
5. $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$ — *закон отрицания конъюнкции*.

Примерами тождественно ложных формул являются следующие формулы:

1. $P \wedge \neg P$ — закон противоречия.
2. $((P \supset Q) \wedge (P \supset \neg Q)) \equiv P$.

Примерами выполнимых формул являются следующие формулы:

1. $(P \supset Q) \wedge (P \supset \neg Q)$.
2. $(P \wedge Q) \vee R$.

Примечание

Доказательство этих утверждений с помощью построения таблиц истинности для соответствующих формул предлагается выполнить читателям самостоятельно в качестве упражнения.

Вернемся к формальному определению исчисления высказываний как формальной логической системы, для чего необходимо определить систему или множество аксиом и правила вывода.

В качестве системы *аксиом* исчисления высказываний могут быть использованы следующие формулы:

1. $A \supset (B \supset A)$.
2. $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$.
3. $A \wedge B \supset A$.
4. $A \wedge B \supset B$.
5. $(A \supset B) \supset ((A \supset C) \supset (A \supset (B \wedge C)))$.
6. $A \supset A \vee B$.
7. $B \supset A \vee B$.
8. $(A \supset C) \supset ((B \supset C) \supset ((A \vee B) \supset C)))$.
9. $(A \supset B) \supset (\neg B \supset \neg A)$.
10. $A \supset (\neg\neg A)$.
11. $(\neg\neg A) \supset A$.

Примечание

Поскольку вместо формул A, B, C могут быть подставлены произвольные формулы исчисления высказываний, аксиомы (1—11), строго говоря, представляют собой схемы аксиом. В этом случае множество аксиом исчисления высказываний является бесконечным. Эта система аксиом предложена П. С. Новиковым.

Правилами вывода исчисления высказываний являются следующие правила:

- *Правило заключения* или так называемое правило MP (*modus ponens* — читается, как "*модус поненс*"). Если A и A⊃B произвольные выводимые формулы исчисления высказываний, то B — также выводимая формула. Символически это записывается в виде: $\frac{A,\ A \supset B}{B}$.
- *Правило подстановки.* Пусть F — выводимая в исчислении высказываний формула, которая содержит символ некоторого элементарного высказывания A, а G — произвольная формула исчисления высказываний. Тогда формула B, которая получается из F заменой всех вхождений элементарного высказывания A на формулу G, также является выводимой в исчислении высказываний.

Пример П2.4. В качестве примера вывода в исчислении высказываний рассмотрим вывод простой формулы: A⊃A, т. е. докажем, что эта формула является теоремой исчисления высказываний. С этой целью рассмотрим следующую последовательность формул:

1. A⊃(B⊃A) — аксиома (1).
2. A⊃((A⊃A)⊃A) — правило подстановки.
3. (A⊃(B⊃C))⊃((A⊃B)⊃(A⊃C)) — аксиома (2).
4. (A⊃((A⊃A)⊃A))⊃((A⊃(A⊃A))⊃(A⊃A)) — правило подстановки.
5. (A⊃(A⊃A))⊃(A⊃A) — правило вывода MP (2, 4).
6. A⊃(A⊃A) — правило подстановки.
7. A⊃A — правило вывода MP (6, 5).

Тем самым доказано исходное утверждение о том, что формула A⊃A является теоремой исчисления высказываний.

Логика предикатов

Рассматриваемые в исчислении высказываний элементарные высказывания и формулы позволяют формализовать лишь незначительную часть логических рассуждений. Причина этого заключается в том, что высказывания представляют собой неделимый объект, не имеющий своей внутренней структуры. В то же время очевидно, что такие высказывания, как: "*Все автомобили имеют белый цвет*" и "*Некоторые автомобили имеют белый цвет*" имеют различные значения истинности в силу различия их внутренней структуры. Более сложная ситуация

связана с невозможностью в рамках исчисления высказываний формализовать логические рассуждения, представленные ниже.

- Все государства имеют свою столицу. Российская Федерация — государство. Следовательно, Российская Федерация имеет свою столицу.
- Российская Федерация имеет морские границы. Российская Федерация — государство. Следовательно, некоторые государства имеют морские границы.
- Все натуральные числа, которые делятся на 4 без остатка, являются четными. Следовательно, некоторые из четных чисел делятся на 4 без остатка.

Формализация подобных семантических аспектов логических рассуждений оказалась возможной на основе введения в рассмотрение специальных объектов, которые получили название предикатов.

Основные понятия логики предикатов первого порядка

П р е д и к а т. *Предикатом* называется некоторое высказывание относительно некоторых внутренних переменных и принимающее значение "*истина*" или "*ложь*" только при подстановке вместо этих переменных тех или иных конкретных объектов.

Более строгое определение основано на введении в рассмотрение некоторого множества $A=\{a_1, a_2,\ldots, a_i,\ldots\}$ и переменных $x, y, z,\ldots$, которые могут принимать значения из этого множества, т. е. $x, y, z \in A$. Каждая из таких переменных получила название *предметной переменной*, а само множество A допустимых значений предметных переменных называется *предметной областью*. То или иное высказывание об элементах множества A будем обозначать через $P(a_1)$, $Q(a_1, a_2)$, $R(a_1, a_2, a_3)$ и т. д. При этом каждое такое высказывание может принимать одно из множества значений истинности {**И**, **Л**}, но не оба одновременно.

Если вместо элементов a_1, a_2, a_3 в высказывания $P(a_1)$, $Q(a_1, a_2)$ и $R(a_1, a_2, a_3)$ подставить предметные переменные: $x, y, z \in A$, то получим предикаты: $P(x)$, $Q(x, y)$ и $R(x, y, z)$. При этом $P(x)$ называется *одноместным* предикатом, $Q(x, y)$ — двуместным предикатом, $R(x, y, z)$ — трехместным предикатом и т. д.

П р и м е р П2.5. Примерами предикатов могут служить следующие переменные высказывания:

- Рассмотрим в качестве предметной области множество натуральных чисел, а в качестве предиката $P(x)$ — высказывание: "x — *простое число*". Тогда для значения $x = 7$ предикат $P(7)$ принимает значение "*истина*", а для значения $x = 6$ предикат $P(6)$ принимает значение "*ложь*".
- Для той же предметной области рассмотрим предикат $Q(x, y)$, семантика которого выражается переменным высказыванием: "x *делится на* y *без остатка*". Тогда высказывания: $Q(4, 2)$ и $Q(10, 5)$ принимают значение "*истина*", а высказывания: $Q(9, 2)$ и $Q(10, 3)$ — значения "*ложь*".

❐ Рассмотрим в качестве предметной области декартово произведение множества авиарейсов: A={LH6370, MA232, Z8221, Z8229, Z8257, Z8277, Z8285} и множества городов из примера П1.3: B={*Амстердам*, *Афины*, *Барселона*, *Каир*, *Нью-Йорк*, *Париж*, *Хельсинки*}. Тогда двухместный предикат $P(x, y)$ будет описывать бинарное отношение $Q_1 \subseteq A \times B$, устанавливающее соответствие между каждым номером авиарейса и аэропортом назначения. В этом случае высказывание P(Z8277, *Амстердам*) принимает значение "*истина*", а высказывание P(MA232, *Хельсинки*) — значение "*ложь*".

Примечание

Как было отмечено в *приложении 1*, в общем случае для произвольного множества A с элементами из универсума $\boldsymbol{X}$ существует формальная взаимосвязь между характеристической функцией этого множества $\chi_A(x)$ и некоторым одноместным предикатом $P_A(x)$, который, возможно неявно, выражает какое-нибудь характеристическое свойство этого множества. А именно, если элемент универсума $a \in \boldsymbol{X}$ обладает рассматриваемым характеристическим свойством, т. е. значение предиката $P_A(a)$ — "*истина*", то $\chi_A(a)$=1 и $a \in A$. Если же элемент $a \in \boldsymbol{X}$ не обладает этим свойством, т. е. значение предиката $P_A(a)$ — "*ложь*", то $\chi_A(a)$=0 и $a \notin A$. Верно и обратное заключение. Эта связь характеристической функции множества и соответствующего ей предиката играет важную роль при установлении взаимосвязей между теорией нечетких множеств и нечеткой логикой.

Логические операции над предикатами

Поскольку предикаты являются переменными высказываниями, то применительно к ним могут быть обобщены все рассмотренные выше логические операции: $\{\neg, \wedge, \vee, \supset, \equiv\}$. При этом результатом выполнения соответствующих операций будет то значение истинности, которое получается при одновременной подстановке вместо предметных переменных конкретных значений элементов из предметной области.

П р и м е р П2.6. Будем использовать в качестве предметной области множество натуральных чисел. Рассмотрим два предиката: $P(x)$ — "*x — четное число*" и $Q(x, y)$ — "*x строго больше y*". Тогда переменное высказывание: "*x — четное число и x строго больше y*" формально может быть записано с помощью логической конъюнкции исходных предикатов: $P(x) \wedge Q(x, y)$. По определению конъюнкции составное высказывание $P(4) \wedge Q(4, 3)$ принимает значение "*истина*", а высказывание $P(5) \wedge Q(7, 4)$ — значение "*ложь*". Аналогичным образом может быть установлена истинность высказываний $P(4) \vee Q(3, 4)$, $P(3) \supset Q(8, 3)$, $P(5) \equiv Q(5, 7)$ и ложность высказываний $P(3) \vee Q(3, 6)$, $P(4) \supset Q(3, 6)$, $P(5) \equiv Q(7, 5)$.

Кванторы логики предикатов

В логике предикатов нашли применение так называемые *кванторы*, которые также рассматриваются в *приложении 1*. Применительно к предикатам используются эти же кванторы и их обозначения: знак "$\forall$" для квантора общности и знак "$\exists$" для квантора существования. Каждый из кванторов может быть применен только к отдельной предметной переменной, входящей в состав того или иного предиката.

Семантика этих предикатов следующая. Пусть A — предметная область, на которой определен некоторый предикат $P(x)$. Тогда выражение $\forall x$, $P(x)$ означает: "*Для всякого (любого, каждого) x из предметной области A имеет место (справедливо, истинно) высказывание $P(x)$*". Выражение $\exists x$, $P(x)$ в свою очередь означает: "*Существует (найдется хотя бы один) элемент x из предметной области A, для которого имеет место (справедливо, истинно) высказывание $P(x)$*".

Применение этих кванторов естественным образом расширяется на многоместные предикаты. При этом вхождение предметной переменной в предикат называется *связанным*, если эта переменная используется с одним из кванторов или находится в области действия применяемого к этому предикату квантора. В противном случае вхождение предметной переменной в рассматриваемый предикат называется *свободным*. Если все вхождения предметных переменных в некоторый предикат связанные, то такой предикат становится обычным высказыванием.

П р и м е р П2.7. Для простоты будем по-прежнему использовать в качестве предметной области множество натуральных чисел. Рассмотрим два предиката: $P(x)$ — "*x — четное число*" и $Q(x, y)$ — "*x строго больше y*". Тогда высказывание: "*Любое x — четное число*" формально может быть записано с помощью квантора общности следующим образом: $\forall x$, $P(x)$. Очевидно, это высказывание принимает значение "*ложь*". Напротив, высказывание: "*Существует натуральное x, которое является четным числом*" формально может быть записано с помощью квантора существования следующим образом: $\exists x$, $P(x)$. Очевидно так же, что это высказывание принимает значение "*истина*". Более сложный случай применения кванторов — выражение: $\exists x \forall y$, $(P(x) \wedge Q(x, y))$, которое является высказыванием со значением истинности "*ложь*".

В общем случае предикат $P(x_1, x_2,\ldots, x_m)$ от предметных переменных $x_1, x_2,\ldots, x_m$ называется *тождественно истинным*, если он принимает значение "*истина*" при любой подстановке элементов из предметной области вместо предметных переменных, входящих в состав этого предиката. Предикат $P(x_1, x_2,\ldots, x_m)$ от предметных переменных $x_1, x_2,\ldots, x_m$ называется *тождественно ложным*, если он принимает значение "*ложь*" при любой подстановке элементов из предметной области вместо предметных переменных, входящих в состав этого предиката. Наконец, предикат $P(x_1, x_2,\ldots, x_m)$ называется *выполнимым*, если он принимает значение "*истина*" хотя бы для одной из подстановок элементов из предметной области вместо предметных переменных, входящих в состав этого предиката.

Исходя из определения кванторов, предикат $P(x_1, x_2,\ldots, x_m)$ — тождественно истинный, если высказывание $\forall x_1 \forall x_2 \ldots \forall x_m$, $P(x_1, x_2,\ldots, x_m)$ принимает значение

"*истина*". С другой стороны, предикат $P(x_1, x_2,\ldots, x_m)$ — тождественно ложный, если высказывание $\exists x_1 \exists x_2 \ldots \exists x_m$, $P(x_1, x_2,\ldots, x_m)$ принимает значение "*ложь*".

Исчисление предикатов первого порядка как формальная теория

Формально-логический анализ рассуждений в логике предикатов безотносительно к содержанию отдельных предикатов приводит к определению соответствующей формальной системы — *исчислению предикатов первого порядка*. По аналогии с исчислением высказываний в этом случае также можно определить строгий логический формализм, который позволяет рассматривать формулы, построенные с помощью введенных выше понятий и обозначений.

Алфавит исчисления предикатов первого порядка (далее — просто исчисления предикатов) содержит следующие группы символов:

1. Не более чем счетное множество предметных переменных: $\{x_1, x_2,\ldots, x_i,\ldots\}$.
2. Не более чем счетное (возможно, пустое) множество предметных констант или элементов предметной области: $A=\{a_1, a_2,\ldots, a_j,\ldots\}$.
3. Не более чем счетное (возможно, пустое) множество функциональных символов (букв): $\{f_1, f_2,\ldots, f_k,\ldots\}$.
4. Не более чем счетное множество предикатных символов (букв): $\{P_1, P_2,\ldots, P_m,\ldots\}$.
5. Символы логических операций (связок): $\{\neg, \vee, \wedge, \supset, \equiv\}$.
6. Символы логических кванторов: $\{\exists, \forall\}$.
7. Вспомогательные символы (скобки): (,) и запятая.

Из этих групп символов пояснения требуют лишь множество функциональных символов (букв): $\{f_1, f_2,\ldots, f_k,\ldots\}$, каждый из которых представляет собой некоторое отображение из предметной области A (в случае одноместного функционального символа) или декартова произведения $A\times A\times\ldots\times A$ (в случае многоместного функционального символа) в эту же предметную область A. Другими словами, $f_k: A\to A$ или $f_k: A\times A\times\ldots\times A\to A$. При этом их часто записывают в виде: $f_k(x)$ или $f_k(x_1, x_2,\ldots, x_n)$, где натуральное n обозначает количество аргументов функционального символа или его местность. Примерами двухместных функциональных символов могут служить обычные арифметические операции, а также бинарные операции максимума и минимума на множестве всех действительных чисел.

Прежде чем определить собственно формулы исчисления предикатов, необходимо предварительно определить некоторые дополнительные понятия — термы и атомы.

Термы в исчислении предикатов определяются рекурсивно следующим образом:

1. Всякая предметная переменная или предметная константа является термом.
2. Если f_k — некоторый n-местный функциональный символ, а $t_1, t_2,\ldots, t_k$ — термы, то $f_k(t_1, t_2,\ldots, t_n)$ является термом.
3. Других правил образования термов нет.

Например, если x, y — термы, а max и min — двухместные функциональные символы, то $\max(x, y)$ и $\min(x, y)$ — также термы.

Атомы в исчислении предикатов определяются рекурсивно следующим образом:

1. Если P_m — некоторый n-местный предикатный символ, а t_1, t_2,..., t_n — термы, то $P_m(t_1, t_2, \ldots, t_n)$ является атомом.
2. Если t_1, t_2 — некоторые термы, то выражение $t_1 = t_2$, где знак "=" — равенство, является атомом.
3. Других правил образования атомов нет.

Примерами атомов могут служить выражения: $P(x+y)$, $Q(max(x, y), z)$, $R(x, f(x, y))$.

Формулой исчисления предикатов первого порядка называется выражение, составленное из символов алфавита и удовлетворяющее следующим *правилам построения*:

1. Всякий атом является формулой исчисления предикатов *(базис)*.
2. Если P, Q — формулы исчисления предикатов, а x — предметная переменная, то выражения: $\neg P$; $\neg Q$; $(P \vee Q)$; $(P \wedge Q)$; $(P \supset Q)$; $(P \equiv Q)$; $\forall x, P$; $\forall x, Q$; $\exists x, P$; $\exists x, Q$; являются формулами исчисления предикатов *(индукционный шаг)*.
3. Формулой исчисления предикатов является любое выражение, которое может быть получено с помощью правил (1) и (2) *(ограничение)*.

Если при построении некоторой формулы другая формула используется как элемент в индуктивном шаге (2), то последняя называется *подформулой* исходной формулы.

Порядок применения логических операций и кванторов в формулах исчисления предикатов также устанавливается слева направо. Для исключения неоднозначности или изменения этого порядка могут быть использованы скобки.

В качестве системы *аксиом* исчисления предикатов могут быть использованы следующие формулы:

1. $P \supset (Q \supset P)$.
2. $(P \supset (Q \supset C)) \supset ((P \supset Q) \supset (P \supset C))$.
3. $P \wedge Q \supset P$.
4. $P \wedge Q \supset Q$.
5. $(P \supset Q) \supset ((P \supset C) \supset (P \supset (Q \wedge C)))$.
6. $P \supset P \vee Q$.
7. $Q \supset P \vee Q$.
8. $(P \supset C) \supset ((Q \supset C) \supset ((P \vee Q) \supset C)))$.
9. $(P \supset Q) \supset (\neg Q \supset \neg P)$.
10. $P \supset (\neg\neg P)$.

11. $(\neg\neg P)\supset P$.

12. $(\forall x, \mathrm{P}(x))\supset P(y)$.

13. $P(y)\supset(\exists x, P(x))$.

В аксиомах (12) и (13) предполагается, что терм y свободен по переменной x в формуле P.

Примечание

Поскольку вместо формул P, Q могут быть подставлены произвольные формулы исчисления предикатов, аксиомы (1—13) также представляют собой схемы аксиом. В этом случае множество аксиом исчисления предикатов является бесконечным.

Правилами вывода исчисления высказываний являются следующие правила:

- *Правило заключения* или так называемое правило MP (*modus ponens* — читается, как "*модус поненс*"). Если P и $P\supset Q$ произвольные выводимые формулы исчисления высказываний, то Q — также выводимая формула. Символически это записывается в виде: $\dfrac{P,\ P\supset Q}{Q}$.

- *Правило обобщения*: $\dfrac{P}{\forall x, P(x)}$, где P — произвольная формула, в которой некоторый терм заменяется предикатной переменной x.

- *Первое правило Бернайса*: $\dfrac{P\supset Q(x)}{P\supset(\forall x, Q(x))}$, где формула $Q(x)$ содержит свободное вхождение переменной x, а формула P — не содержит.

- *Второе правило Бернайса*: $\dfrac{P(x)\supset Q}{(\exists x, P(x))\supset Q}$, где формула $P(x)$ содержит свободное вхождение переменной x, а формула Q — не содержит.

Установление истинности произвольной формулы исчисления предикатов может быть связано с серьезными трудностями, имеющими принципиальный характер. Действительно, в общем случае отсутствует конструктивный алгоритм, который бы за конечное, хотя и возможно большое число шагов, позволил бы установить тождественную истинность или ложность произвольной формулы. Хотя, казалось бы, используемый для аналогичной цели в исчислении высказываний алгоритм построения таблицы истинности может быть обобщен для формулы исчисления предикатов, тем не менее, он совершенно не годится в случае предметной области с более чем счетной мощностью. Действительно, в этом случае потребовалось бы построить таблицу с бесконечным числом строк, что уже в принципе не позволит установить истинность произвольной формулы для подобной предметной области.

Поиск выхода из создавшегося положения послужил движущей силой развития математической логики в XX столетии. Были предложены различные формаль-

ные системы, в рамках которых удалось разработать алгоритмы (например, метод резолюций или обратный метод С. Ю. Маслова), предназначенные для поиска доказательств общезначимости формул исчисления предикатов первого порядка. Однако данная проблематика выходит за рамки настоящей книги.

Продукционные системы

Продукционные системы были разработаны в рамках исследований по методам искусственного интеллекта и нашли широкое применение для представления знаний и вывода заключений в так называемых *экспертных системах, основанных на правилах* (rule-based expert systems, продукционные ЭС). Поскольку нечеткий вывод реализован в форме нечетких правил продукций, рассмотрение базового формализма продукционных моделей приобретает самостоятельное значение. При этом продукционные системы не только во многом близки к логическим моделям, но и, что наиболее важно, позволяют наглядно представить практические знания экспертов в той или иной проблемной области.

Продукционная система или система правил продукций представляет собой некоторое согласованное множество отдельных *продукций* (или правил продукций в форме "ЕСЛИ A, ТО B").

В общем виде под *продукцией* понимается выражение следующего вида:

$$(i) : Q; P; A \Rightarrow B; N,$$

где (i) — имя продукции; Q — сфера применения продукции; P — условие применимости ядра продукции; $A \Rightarrow B$ — ядро продукции, в котором A — условие ядра (или антецедент); B — заключение ядра (или консеквент); "$\Rightarrow$" — знак логической секвенции (или следования); N — постусловия продукции.

Дадим краткую характеристику компонентам продукции. В качестве имени (i) продукции может выступать та или иная совокупность букв или символов, позволяющая однозначным образом идентифицировать продукцию в системе. Наиболее часто продукции хранятся в системе со своим уникальным номером.

Сфера применения продукции Q описывает явно или неявно предметную область знания, которую представляет отдельная продукция. При этом декомпозиция предметной области на отдельные независимые области может существенно повысить эффективность рассуждений в продукционной системе. Именно этот аспект при необходимости может описывать данный элемент продукции.

Условие применимости ядра продукции P представляет собой логическое выражение (как правило, предикат). Если оно присутствует в продукции, то активизация ядра продукции становится возможной только в случае истинности этого условия. Во многих случаях этот элемент продукции может быть опущен или введен в ядро продукции.

Ядро продукции $A \Rightarrow B$, как следует уже из его названия, является центральным компонентом продукции. Наиболее часто ядро продукции читается как "ЕСЛИ A, ТО B", где A и B — некоторые логические выражения, которые в том или

ином контексте могут принимать значения "*истина*" или "*ложь*". При этом секвенция интерпретируется в обычном логическом смысле как знак логического следования *B* из истинного *A*. Другими словами, если истинно выражение *A*, то истинным следует считать и выражение *B*. В остальных случаях, когда *A* не является истинным, об истинности *B* ничего сказать нельзя. Во многих случаях в качестве выражения *A* могут использоваться сложные логические выражения, т. е. выражения, содержащие основные логические связки, такие как отрицание, конъюнкцию или дизъюнкцию.

Наконец, постусловие продукции *N* описывает действия и процедуры, которые необходимо выполнить в случае реализации ядра продукции, т. е. получения информации об истинности *B*. Характер этих действий может быть самым различным и отражать вычислительный или иной аспект продукционной системы.

Примечание

Рассмотренная выше общая форма продукции характерна для так называемых *детерминированных* ядер продукции. В недетерминированных продукционных системах каждая из продукций может дополнительно содержать количественную оценку степени истинности правила.

Системы продукций нашли широкое практическое применение при построении экспертных систем, которые формализуют опыт экспертов в некоторой проблемной области. Как оказалось, принимая решение, многие специалисты оперируют знаниями именно в форме правил вида "ЕСЛИ "*некоторое условие*" ТО "*заключение*". При этом в качестве условий выступают известные факты, отражающие конкретно сложившуюся ситуацию в проблемной области на текущий момент времени. Что касается заключения, то его вывод или подтверждение оказывает непосредственное влияние на процедуру принятия решения по изменению сложившейся ситуации в необходимом направлении.

Таким образом, система правил продукций представляет собой некоторое согласованное множество отдельных правил продукций, которое представляет знания экспертов в той или иной проблемной области. Такая система служит составной частью систем управления в случае их реализации в форме продукционных экспертных систем. В этом случае построение системы правил продукций или базы правил является важным этапом разработки соответствующих систем, который должен быть выполнен до начала их применения по целевому назначению.

П р и м е р П2.7. В качестве системы продукций рассмотрим следующее множество правил продукций (базу правил), которое упрощенно представляет ситуацию таможенного досмотра граждан при пересечении границы:

- ПРАВИЛО_1: ЕСЛИ "Гражданин не является высокопоставленным чиновником", ТО "он подвергается таможенному досмотру"
- ПРАВИЛО_2: ЕСЛИ "Гражданин является высокопоставленным чиновником", ТО "он не подвергается таможенному досмотру"
- ПРАВИЛО_3: ЕСЛИ "Гражданин не подвергается таможенному досмотру", ТО "не исключается возможность провоза наркотиков"

- ПРАВИЛО_4: ЕСЛИ "Количество граждан, проходящих таможенный досмотр, велико", ТО "контролер испытывает чувство усталости"
- ПРАВИЛО_5: ЕСЛИ "Контролер испытывает чувство усталости", ТО "не исключается возможность провоза наркотиков"
- ПРАВИЛО_6: ЕСЛИ "Гражданин подвергается таможенному досмотру" И "в отношении этого гражданина имеется агентурная информация", ТО "исключается возможность провоза наркотиков"
- ПРАВИЛО_7: ЕСЛИ "Гражданин подвергается таможенному досмотру" И "контролер использует новейшие технические средства", ТО "исключается возможность провоза наркотиков"

Примечание

Конечно, рассмотренная база правил не претендует на полноту и построена на основе лишь здравого смысла. Тем не менее, даже этот простой пример характеризует нетривиальный характер возможности реального использования подобной базы правил, которое должно быть дополнено механизмами исключения возможных конфликтов и противоречий, о которых будет сказано ниже. Более адекватный пример системы нечетких продукций был рассмотрен в *главе 6*.

Прямой и обратный методы вывода заключений в продукционных системах

Следующим важным компонентом продукционных систем является так называемый метод (или схема) вывода заключений на основе известных значений условий в базе правил продукций. Наиболее известными являются два таких метода вывода заключений: прямой и обратный.

- *Прямой метод вывода* заключений в продукционных системах, называемый также методом восходящего вывода или методом прямой цепочки рассуждений (classic forward-chaining reasoning), основан на использовании правила вывода MP. А именно, если это правило позволяет определить истинность заключения правила продукции при известной истинности его условия. Действительно, дедукция по правилу MP позволяет заключить следующее: если утверждается, что обе формулы P и $P \supset Q$ принимают значение "*истина*", то значение "*истина*" принимает и формула Q.

  Применительно к системам правил продукций прямой метод вывода реализуется посредством преобразования отдельных фактов проблемной области в конкретные значения истинности условий правил продукций. После этого преобразования те из правил продукций, для которых становятся истинными соответствующие условия, генерируют заключения своих правых частей. Эти заключения принимаются как истинные и становятся новыми фактами, которые могут быть использованы в качестве условий в рассматриваемой базе правил. При этом правила, для которых имеются истинные условия, часто называют *активными*.

Процесс вывода прямым методом имеет рекурсивный характер и может быть остановлен либо в случае отсутствия новых активных правил, либо в случае получения заключения, которое является целевым в контексте решения исходной проблемы. Подобное подтверждение целевого заключения характеризует успех процесса вывода, поскольку только в этом случае использование системы продукций характеризует решение поставленной проблемы.

- *Обратный метод вывода* в продукционных системах, называемый также методом нисходящего вывода или методом обратной цепочки рассуждений (classic backward-chaining reasoning), основан использовании правила вывода, известного как правило вывода MT (modus tollens, читается как "*модус толленс*"). Это правило также является *правилом заключения*, которое символически записывается в виде: $\frac{\overline{Q}, P \supset Q}{\overline{P}}$, где $\overline{Q}$ и $P \supset Q$ — произвольные выводимые формулы исчисления высказываний. В случае применения этого правила становится справедливым заключение о том, что $\overline{P}$ — также выводимая формула. Тем самым это правило позволяет определить ложность условия правила продукции при известной ложности его заключения.

  Применительно к системам правил продукций обратный метод вывода реализуется в несколько модифицированном виде посредством исследования возможности применения правил для подтверждения некоторых заранее заданных заключений. А именно, форма импликации $P \supset Q$ остается без изменения, а отрицание заключения $\overline{Q}$ заменяется вопросом о его истинности. В этом случае ставится вопрос об истинности условия $\overline{P}$. Символически это записывается в виде: $\frac{Q?, P \supset Q}{P?}$. Содержательно это означает, что в случае истинности импликации "P *влечет* Q" достаточным условием истинности формулы Q является истинность формулы P. Таким образом, если целью вывода является доказательство истинности заключения Q, то для этого достаточно доказать истинность условия P, рассматриваемого как подцель. Поэтому обратный метод служит обоснованием достаточных условий для истинности заключений правил продукций. Именно в этом проявляется различие между классическим методом заключения MT и обратным методом вывода в продукционных ЭС.

  Применительно к системам правил продукций процесс обратного вывода начинается с подстановки отдельных интересующих нас заключений в правые части соответствующих правил, которые в этом случае становятся *активными*. После анализа каждого из активных правил фиксируются условия, которые подтверждают эти правила. Эти условия принимаются как истинные и становятся новыми фактами, которые могут быть использованы в качестве новых целевых заключений в рассматриваемой базе правил.

  Процесс вывода обратным методом также имеет рекурсивный характер и может быть остановлен либо в случае отсутствия новых активных правил, либо в случае получения подтверждения условий, которые являются истинными или известными фактами проблемной области. Подобное подтвержде-

ние условий характеризует успех процесса вывода и подтверждение исходных заключений.

Примечание

Рассмотренные методы вывода лишь упрощенно представляют основные идеи, лежащие в их основе. Конкретные реализации механизмов вывода в системах правил продукций должны обладать целым рядом дополнительных особенностей, позволяющих эффективно управлять процессом вывода на каждой из рекурсивных итераций. Обсуждение этих механизмов управления выводом выходит за рамки книги.

П р и м е р П2.8. Рассмотрим использование базы правил, построенной в предыдущем примере П2.7, для получения выводов в двух конкретных ситуациях.

Предположим, что текущая ситуация на таможенном пункте контроля сложилась таким образом, что среди граждан, въезжающих в страну, находятся высокопоставленные чиновники. Количество граждан, проходящих таможенный досмотр, невелико, а таможенный пункт контроля не оснащен новейшими техническими средствами. Какая-либо предварительная информация о наличии наркотиков у отдельных граждан отсутствует.

Проблема заключается в оценке истинности заключения об исключении возможности провоза наркотиков через данный пункт контроля. Рассмотрим возможный способ решения данной проблемы с использованием прямого метода вывода.

При прямом методе вывода на первом этапе активными является правило 2, которое позволяет получить заключение о том, что "некоторые из граждан не будут подвергнуты таможенному досмотру". В свою очередь это условие активизирует правило 3, которое позволяет получить заключение о том, что в сложившейся ситуации "не исключается возможность провоза наркотиков". Поскольку другие активные правила в системе отсутствуют, то данное заключение принимается в качестве результата вывода в рассматриваемой базе правил для заданной исходной ситуации. Тем самым исходное целевое заключение об исключении возможности провоза наркотиков через данный пункт контроля не подтверждается.

Примечание

Обычные продукционные системы оставляют открытым вопрос о том, как учитывать тот факт, что подтверждение некоторого заключения может следовать на основе сразу нескольких правил продукций. Одним из достоинств систем нечеткого вывода является то, что они позволяют аккумулировать факты, свидетельствующие в пользу тех или иных заключений *(см. главу 7)*.

П р и м е р П2.9. Предположим, другая ситуация на таможенном пункте контроля характеризуется тем, что среди граждан, въезжающих в страну, отсутствуют высокопоставленные чиновники. Количество граждан, проходящих таможенный досмотр, велико, но таможенный пункт контроля оснащен новейшими техниче-

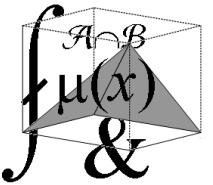
скими средствами и имеется предварительная информация о наличии наркотиков у одного из граждан.

Проблема заключается в оценке истинности заключения об исключении возможности провоза наркотиков через данный пункт контроля. Рассмотрим возможный способ решения данной проблемы с использованием обратного метода вывода.

Очевидно, в этом случае правило 6 сразу дает ответ на поставленный вопрос, поскольку его условием является известный факт текущей ситуации проблемной области. Продолжая процесс обратного вывода, можно активизировать также правило 7, после чего — правило 1, которые в совокупности также свидетельствуют в пользу заключения об исключении возможности провоза наркотиков через данный пункт контроля.

Попытка найти решение для последней ситуации методом прямого вывода сталкивается с противоречием, суть которого в том, что активизация нескольких правил приводит к взаимно исключающим заключениям. Еще одним достоинством систем нечеткого вывода является то, что они позволяют без внесения дополнительных механизмов исключать противоречивость на основе оперирования степенью истинности заключений, отличной от бинарных значений "*истина*" и "*ложь*" *(см. главу 7)*. Хотя некоторые из читателей могут поставить под сомнение целесообразность практического применения противоречивых баз правил, подобных рассмотренной выше, тем не менее, практически интересным проблемным областям присущи свойства противоречивости и неполноты. Именно по этой причине системы нечеткого вывода оказываются наиболее адекватным средством формализации экспертных знаний в таких проблемных областях и решения задач вывода в нечетких продукционных системах.

Приложение 3



Справочник функций пакета Fuzzy Logic Toolbox системы MATLAB

Настоящее приложение содержит описание всех функций Fuzzy Logic Toolbox MATLAB, предназначенных для работы с нечеткими множествами, создания и редактирования систем нечеткого вывода, а также выполнения процедур нечеткого вывода. Функции перечислены в алфавитном порядке, некоторые из них снабжены рисунками и примерами. По каждой из функций указывается ее назначение, приводится описание общего синтаксиса и всех формальных параметров, используемых в качестве аргументов.

addmf

Назначение. Добавляет новую функцию принадлежности к системе нечеткого вывода FIS.

Синтаксис. `a = addmf(a, 'varType', varIndex, 'mfName', 'mfType', mfParams)`

Описание. Функция принадлежности может быть добавлена только к существующей переменной FIS для текущей рабочей области MATLAB. При этом номера или индексы функциям принадлежности назначаются в том порядке, в котором они добавляются. Таким образом, первая добавленная к переменной функция принадлежности будет всегда являться для этой переменной функцией принадлежности с номером 1. Если в FIS определена только одна входная переменная, то добавить функцию принадлежности для любой другой входной переменной с номером 2 нельзя.

Функция использует 6 аргументов (входных параметров) в следующем порядке:

1. `a` — имя переменной структуры FIS в рабочей области MATLAB.
2. `'varType'` — строка, представляющая *тип* переменной, к которой добавляется функция принадлежности. Может принимать одно из двух значений: `'input'` или `'output'`.
3. `varIndex` — индекс переменной, к которой добавляется функция принадлежности.

4. 'mfName' — строка, представляющая *имя* новой функции принадлежности.
5. 'mfType' — строка, представляющая *тип* новой функции принадлежности.
6. mfParams — вектор параметров, которые специфицируют добавляемую функцию принадлежности.

Пример добавления трех функций принадлежности для одной входной переменной

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
plotmf(a,'input',1)
```

Результат этого примера изображен на рис. П3.1.

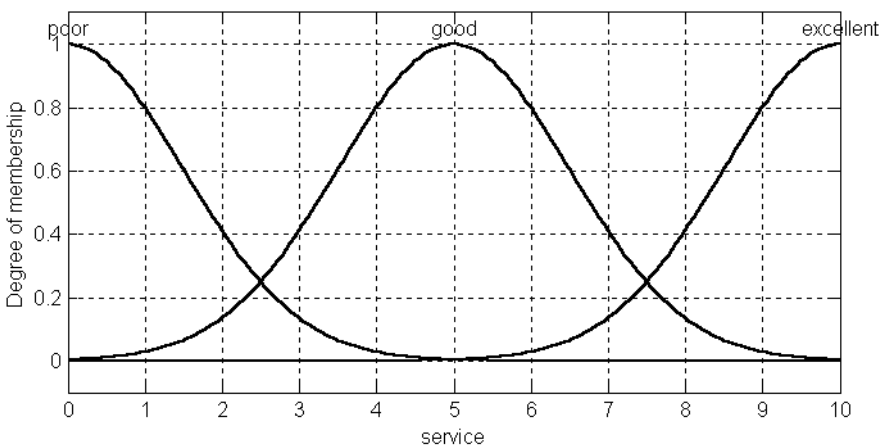


Рис. П3.1. Результат добавления трех функций принадлежности для одной входной переменной системы нечеткого вывода

См. также: addrule, addvar, plotmf, rmmf, rmvar.

addrule

Назначение. Добавляет новое правило к системе нечеткого вывода FIS.

Синтаксис. a = addrule(a, ruleList)

Описание. Функция `addrule` имеет два аргумента. Первый аргумент — имя переменной FIS в рабочей области MATLAB. Второй аргумент `ruleList` — матрица, содержащая одну или несколько строк, каждая из которых представляет данное правило. Следует помнить, что для представления этой матрицы используется специальный формат. Если система вывода имеет m входных переменных и n выходных переменных, то соответствующая матрица `ruleList` должна иметь в точности $m + n + 2$ столбцов.

Первые m столбцов относятся к входным переменным системы. При этом номер столбца должен соответствовать индексу функции принадлежности для конкретной входной переменной. Следующие n столбцов относятся к выходным переменным системы вывода. При этом каждый столбец также имеет номер, который должен соответствовать индексу функции принадлежности для выходной переменной.

Столбец с номером $m + n + 1$ содержит вес, с которым применяется данное правило. Вес может принимать любое значение между нулем и единицей. В общем случае целесообразно задавать вес равным 1.

Столбец с номером $m + n + 2$ содержит число 1, если для подзаключений данного правила используется нечеткий оператор `AND` (нечеткое И). Этот столбец содержит число 2, если для подусловий данного правила используется нечеткий оператор `OR` (нечеткое ИЛИ).

Пример для системы вывода с двумя входными переменными и одной выходной переменной

```
ruleList=[ 1 1 1 1 1
           1 2 2 1 1 ];
a = addrule(a,ruleList);
```

Для данного примера добавляемое правило может быть интерпретировано следующим образом: "Если входная переменная 1 имеет функцию принадлежности 1 и входная переменная 2 имеет функцию принадлежности 1, то выходная переменная 1 имеет функцию принадлежности 1".

См. также: `addmf`, `addvar`, `rmmf`, `rmvar`, `parsrule`, `showrule`.

addvar

Назначение. Добавляет новую переменную к системе нечеткого вывода FIS.

Синтаксис. `a = addvar(a, 'varType', 'varName', varBounds)`

Описание. Функция использует 4 аргумента в следующем порядке:

1. `a` — имя структуры FIS в рабочей области MATLAB.
2. `'varType'` — строка, представляющая *тип* добавляемой переменной. Может принимать одно из двух значений: `'input'` или `'output'`.

3. `'varName'` — строка, представляющая *имя* добавляемой переменной.
4. `varBounds` — вектор, задающий границы области определения добавляемой переменной.

Номера или индексы переменным назначаются в том порядке, в котором они добавляются. Таким образом, первая добавленная переменная будет всегда являться переменной с номером 1. Нумерация входных и выходных переменных производится независимо друг от друга.

Пример добавления одной входной переменной с именем `'service'`

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
getfis(a,'input',1)
```

В результате выполнения этой последовательности функций в окне команд появится следующий результат:

```
Name =          service
NumMFs =        0
MFLabels =
Range =         [0 10]
```

См. также: `addmf`, `addrule`, `rmmf`, `rmvar`.

anfis

Назначение. Программа обучения для системы нечеткого вывода FIS типа Сугено (только MEX).

Синтаксис.

```
[fismat, error1, stepsize] = anfis(trnData)
[fismat, error1, stepsize] = anfis(trnData, fismat)
[fismat1, error1, stepsize] = ...
      anfis(trnData, fismat, trnOpt, dispOpt)
[fismat1, error1, stepsize, fismat2, error2] = ...
      anfis(trnData, trnOpt, dispOpt, chkData)
[fismat1, error1, stepsize, fismat2, error2] = ...
      anfis(trnData,   trnOpt,   dispOpt,   chkData,
optMethod)
```

Описание. Это основная программа обучения для адаптивных систем нейро-нечеткого вывода ANFIS. Функция `anfis` использует алгоритм гибридного обучения для определения параметров систем нечеткого вывода типа Сугено. Этот алгоритм представляет собой комбинацию метода наименьших квадратов и метода убывания обратного градиента для обучения параметров функции принадлежности FIS с целью воспроизведения заданного множества обучающей выборки.

Функция `anfis` может также вызываться с необязательным аргументом с целью проверки правильности нечеткой модели. Типом проверки модели, который применяется в этом случае, является проверка совпадения данных модели, а аргумент представляет собой некоторое множество данных, которое называется множеством проверочных данных.

Функция `anfis` может быть использована со следующими аргументами:

- `trnData` — имя обучающей выборки. Представляет собой матрицу, все столбцы которой, кроме последнего, содержат входные данные. Последний столбец матрицы представляет собой обычный вектор выходных данных;
- `fismat` — имя системы нечеткого вывода FIS, используемой для того, чтобы предоставить функции `anfis` некоторое исходное множество функций принадлежности для обучения. Без этого аргумента функция `anfis` будет использовать по умолчанию функцию `genfis1` для задания некоторой системы нечеткого вывода FIS для выполнения последующего обучения. Эта система нечеткого вывода FIS, задаваемая по умолчанию, будет иметь 2 функции принадлежности типа функции Гаусса, в то время как функция `anfis` вызывается только с одним аргументом. Если `fismat` представляет собой некоторое число (или вектор), то этот аргумент принимается равным номеру функции принадлежности (или вектору, компоненты которого представляют номера функций принадлежности соответствующих входных переменных). В этом случае оба эти аргумента функции `anfis` используются функцией `genfis1` для задания некоторой "правильной" структуры FIS до начала процесса обучения;
- `trnOpt` — вектор параметров обучения. Если любой из параметров обучения вводится как NaN, то будут использованы соответствующие параметры по умолчанию. Эти параметры обучения следующие:
  - `trnOpt(1)` — количество циклов (эпох) обучения (по умолчанию: 10);
  - `trnOpt(2)` — целевой уровень ошибки обучения (по умолчанию: 0);
  - `trnOpt(3)` — величина исходного шага алгоритма обучения (по умолчанию: 0.01);
  - `trnOpt(4)` — коэффициент уменьшения величины шага (по умолчанию: 0.9);
  - `trnOpt(5)` — коэффициент увеличения величины шага (по умолчанию: 1.1);
- `dispOpt` — вектор параметров отображения информации, которые определяют характер сообщений, выводимых в окно команд системы MATLAB в ходе процесса обучения. По умолчанию значение любого из этих параметров равно 1, что означает отображение соответствующей информации. В случае, если значение некоторого параметра равно 0, то соответствующая информации не отображается. Когда какой-либо из параметров отображения принимает значение NaN, то для этого параметра используется значение по умолчанию. Эти параметры отображения следующие:
  - `dispOpt(1)` — ANFIS-информация, такая как число входных и выходных функций принадлежности и т. д. (по умолчанию: 1);

- `dispOpt(2)` — ошибка обучения (по умолчанию: 1);
- `dispOpt(3)` — величина шага обновления по каждому из параметров (по умолчанию: 1);
- `dispOpt(4)` — конечные результаты (по умолчанию: 1);

- `chkData` — имя необязательного множества проверочных данных для проверки правильности модели. Представляет собой матрицу, формат которой совпадает с форматом матрицы обучающей выборки;
- `optMethod` — применение необязательного метода оптимизации для обучения параметров функций принадлежности. Этот параметр может принимать одно из следующих значений: 1 — гибридный алгоритм, 0 — алгоритм обратного распространения ошибки. Гибридный алгоритм представляет собой комбинацию метода наименьших квадратов и метода обратного распространения ошибки. Гибридный алгоритм используется по умолчанию и вызывается всякий раз, когда соответствующий параметр принимает любое отличное от 0 значение.

Процесс обучения сети заканчивается, когда выполнено заданное число циклов обучения или достигнут целевой уровень ошибки обучения.

В случае если функция `anfis` вызывается с двумя или большим числом аргументов, а любой из необязательных аргументов задан в форме NaN или равен пустой матрице, то этот параметр принимает значение по умолчанию. Значения параметров по умолчанию могут быть изменены непосредственной модификацией файла nfis.m. Если пользователь не желает специфицировать некоторые из параметров, но желает специфицировать следующие за ними параметры, то на месте первых из них необходимо указать значение NaN или пустую матрицу.

Функция `anfis` имеет следующие выходные параметры:

- `fismat1` — имя структуры FIS, параметры которой представляют собой множество, полученное согласно критерию минимизации ошибки обучения;
- `error1` — массив (вектор) значений ошибок для обучающей выборки;
- `error2` — массив (вектор) значений ошибок для проверочной выборки;
- `stepsize` — массив значений величины шага в алгоритме обучения. Эти значения следуют в убывающем порядке, если мера ошибки подвергается двум чередующимся комбинациям увеличения и уменьшения. Величины шага следуют в возрастающем порядке, если мера ошибки подвергается четырем последовательным уменьшениям;
- `fismat2` — имя структуры FIS, параметры которой представляют собой множество, полученное согласно критерию минимизации ошибки для проверочной выборки.

Пример использования функции `anfis` с тремя аргументами

```
x = (0:0.1:10)';
y = sin(2*x)./exp(x/5);
```

```
trnData = [x y];
numMFs = 5;
mfType = 'gbellmf';
epoch_n = 20;
in_fismat = genfis1(trnData, numMFs, mfType);
out_fismat = anfis(trnData, in_fismat, 20);
plot(x, y, x, evalfis(x, out_fismat));
legend('Training Data','ANFIS Output');
```

Результат выполнения этой последовательности команд приведет к созданию и обучению гибридной сети ANFIS, графики обучающей выборки и выхода которой изображены на рис. П3.2.

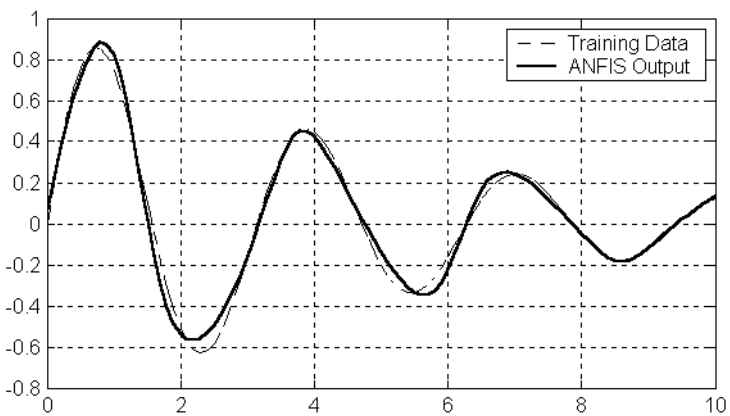


Рис. П3.2. Графики обучающей выборки и выхода гибридной сети с архитектурой ANFIS

См. также: `genfis1`, `anfisedit`.

anfisedit

Назначение. Открывает редактор гибридных сетей ANFIS.

Синтаксис.

```
anfisedit('a')
anfisedit(a)
anfisedit
```

Описание. Функция `anfisedit` вызывает графический интерфейс редактора ANFIS, который позволяет пользователю загрузить некоторое множество дан-

ных и обучить гибридную сеть ANFIS. Эта функция в формате `anfisedit('a')` вызывает редактор с загруженной структурой FIS, которая хранится во внешнем файле с именем a.fis. Если функция используется в формате `anfisedit(a)`, то она вызывает редактор с загруженной структурой FIS, которая представляет собой переменную типа структуры с именем `a` в рабочей области MATLAB. Функция в формате `anfisedit` вызывает редактор без загруженной структуры FIS. Более подробно особенности работы с редактором гибридных сетей ANFIS изложены в *главе 15*.

Редактор ANFIS имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора правил содержит такие же операции, что и соответствующий пункт меню редактора FIS.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **FIS Properties...** — вызывает графический интерфейс редактора FIS;
  - **Membership Functions...** — вызывает графический интерфейс редактора функций принадлежности;
  - **Rules** — вызывает графический интерфейс редактора правил системы нечеткого вывода;
  - **Anfis...** — вызывает графический интерфейс редактора ANFIS.
- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил системы нечеткого вывода;
  - **Surface** — вызывает программу просмотра поверхности системы нечеткого вывода.

Пример вызова графического интерфейса редактора ANFIS с исходными данными из рабочей области

```
anfisedit(out_fismat)
```

Результат вызова графического интерфейса редактора ANFIS с исходными данными из рабочей области изображен на рис. П3.3.

См. также: `fuzzy`, `mfedit`, `ruleedit`, `ruleview`, `surfview`.

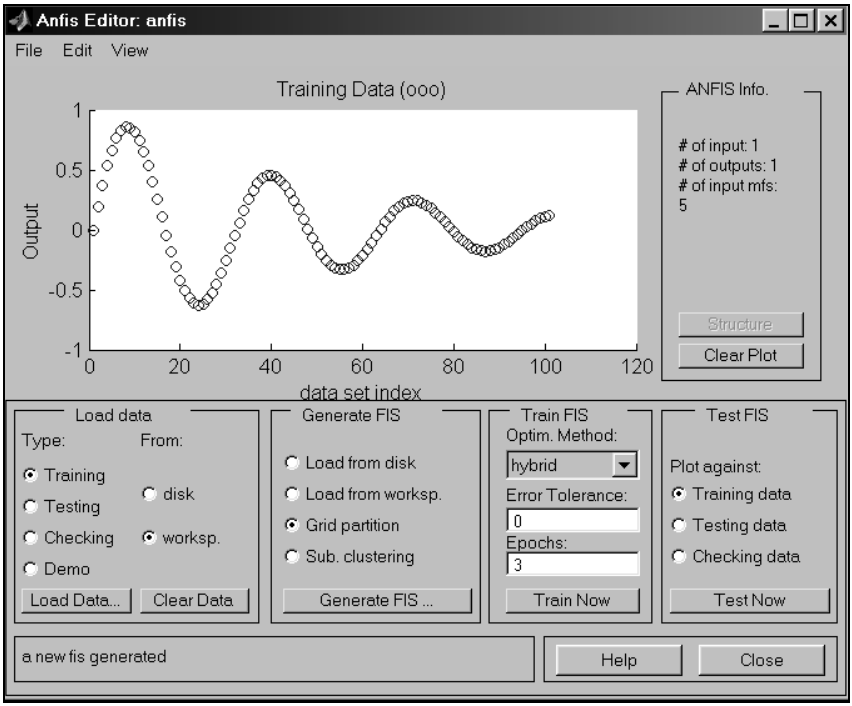


Рис. П3.3. Результат вызова графического интерфейса редактора ANFIS с загруженными исходными данными

convertfis

Назначение. Преобразует матрицу FIS Fuzzy Logic Toolbox версии 1.0 в структуру FIS версии 2.0.

Синтаксис. `fis_new=convertfis(fis_old)`

Описание. Функция `convertfis` использует в качестве аргумента матрицу FIS версии 1.0 и преобразует ее в структуру FIS версии 2.0.

defuzz

Назначение. Выполняет дефаззификацию функции принадлежности.

Синтаксис. `out = defuzz(x, mf, type)`

Описание. Функция `defuzz(x, mf, type)` позволяет получить число, которое является результатом дефаззификации функции принадлежности `mf` для соответствующей переменной `x`. При этом может быть использован один из методов

дефаззификации, который определяется аргументом `type`. Этот аргумент может принимать одно из следующих значений:

- `'centroid'` — метод центра тяжести для дискретного множества значений функции принадлежности (7.10);
- `'bisector'` — метод центра площади (модификация (7.11);
- `'som'` — метод левого модального значения (7.12);
- `'lom'` — метод правого модального значения (7.13);
- `'mom'` — метод среднего максимума, определяемый как среднее арифметическое левого и правого модальных значений.

Примечание

Метод среднего максимума выдает корректный результат только в случае унимодального нечеткого множества. В противном случае результат может оказаться даже не модальным значением соответствующего нечеткого множества.

Если для аргумента `type` не указано ни одно из перечисленных выше значений, то предполагается использование метода дефаззификации, который определен самим пользователем. Это можно сделать посредством добавления в m-файл этой функции defuzz.m соответствующих операторов, реализующих дополнительный метод дефаззификации.

Пример использования метода центра тяжести для дефаззификации функции принадлежности

```
x = -10:0.1:10;
mf = trapmf(x,[-10 -8 -4 7]);
xx = defuzz(x,mf,'centroid');
plot(x,mf)
line(xx, 0, 'linestyle','none','marker','*','color','r');
```

В результате выполнения данной последовательности команд будет получено следующее изображение (рис. П3.4).

dsigmf

Назначение. Встроенная *П*-образная функция принадлежности, которая является разностью двух сигмоидальных функций.

Синтаксис. `y = dsigmf(x,[a1 c1 a2 c2])`

Описание. Используемая в MATLAB сигмоидальная функция принадлежности зависит от двух параметров a и b и аналитически задается следующим выраже-

нием (2.14). Функция принадлежности `dsigmf` зависит от 4 параметров: `a1`, `c1`, `a2`, `c2` и представляет собой разность двух сигмоидальных функций:

$$f_1(x;a_1,c_1)-f_2(x;a_2,c_2).$$

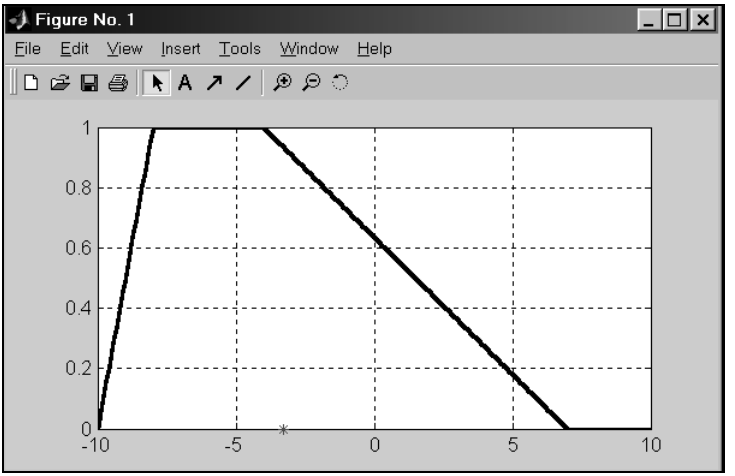


Рис. П3.4. График трапециевидной функции принадлежности и результат ее дефаззификации методом центроида ("*")

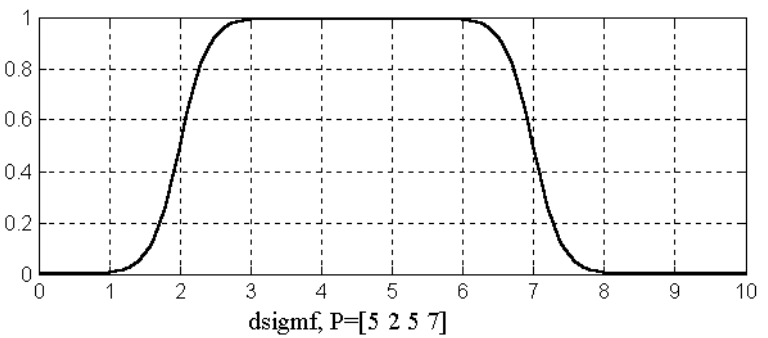


Рис. П3.5. График разности двух сигмоидальных функций

Следует обратить внимание на правильную запись параметров, которые должны следовать вначале для первой функции, а потом для второй: `[a1 c1 a2 c2 ]`.

Пример задания функции принадлежности `dsigmf`

```
x=0:0.1:10;
y=dsigmf(x,[5 2 5 7]);
```

```
plot(x,y)
xlabel('dsigmf, P=[5 2 5 7]')
```

График соответствующей функции принадлежности изображен на рис. П3.5.

См. также: `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

evalfis

Назначение. Выполняет нечеткий вывод в системе FIS.

Синтаксис.

```
output= evalfis(input, fismat)
output= evalfis(input, fismat, numPts)
[output, IRR, ORR, ARR]= evalfis(input, fismat)
[output, IRR, ORR, ARR]= evalfis(input, fismat, numPts)
```

Описание. Функция `evalfis` имеет следующие аргументы:

- `input` — число или матрица, которая определяет входные переменные. Если `input` является матрицей порядка ($m \times n$), где n — число входных переменных, то функция `evalfis` воспринимает каждую строку `input` как вектор, соответствующий отдельному набору значений входных переменных. Функция возвращает матрицу порядка ($m \times l$), в которой каждая строка соответствует вектору значений выходных переменных, где l — число всех выходных переменных системы;
- `fismat` — имя структуры FIS, для которой выполняется нечеткий вывод;
- `numPts` — необязательный аргумент, задающий общее число выбранных точек, в которых оцениваются функции принадлежности в областях определения входных и выходных переменных. Если этот аргумент не указан, то по умолчанию используется значение 101.

Функция `evalfis` возвращает значения, которые имеют следующий смысл:

- `output` — матрица выходов размерности ($m \times l$), где m равно числу определенных ранее значений входных переменных, а l — числу выходных переменных системы нечеткого вывода FIS.

Следующие выходные значения функции являются необязательными и возвращаются функцией `evalfis` только в том случае, когда указан один набор значений входов (m=1):

- `IRR` — результат вычисления значений функций принадлежности для соответствующих входных переменных. Представляет собой матрицу размерности ($numRules \times n$), где $numRules$ — общее количество правил в системе FIS, а n — число входных переменных;
- `ORR` — результат вычисления значений функций принадлежности для соответствующих выходных переменных. Представляет собой матрицу размерно-

сти (*numPts* × *numRules***l*), где *numRules* — общее количество правил в системе FIS, а *l* — число выходных переменных. Первые *numRules* столбцов этой матрицы соответствуют первой выходной переменной, следующие *numRules* столбцов этой матрицы соответствуют второй выходной переменной и т. д.;

- `ARR` — матрица размерности (*numPts*×*l*) агрегированных значений в области задания каждой выходной переменной.

Если функция вызывается для одного набора значений входных переменных, то в качестве результата возвращается вектор значений выходных переменных. Если в структуре FIS определена одна выходная переменная, то возвращается скалярное значение этой выходной переменной после ее дефаззификации.

Пример использования функции `evalfis` для двух наборов значений входных переменных

```
fismat = readfis('tipper');
out = evalfis([2 1; 4 9],fismat)
```

Выполнение функции `evalfis` в этом случае приведет к получению следующего результата в окне команд:

```
out =
       7.0169
       19.6810
```

См. также: `ruleview`, `gensurf`.

evalmf

Назначение. Задает одну из встроенных функций принадлежности.

Синтаксис. `y = evalmf(x, mfParams, mfType)`

Описание. Функция `evalmf` позволяет задать любую из имеющихся в системе MATLAB функций принадлежности, тип которой устанавливается строкой `mfType`. При этом значение аргумента `x` устанавливает область определения задаваемой функции принадлежности (универсум) и должна быть предварительно определена. Строка `mfType` должна соответствовать имени m-файла, в котором определена задаваемая функция принадлежности. Второй аргумент `mfParams` представляет собой вектор параметров, определяющих задаваемую функцию принадлежности.

При необходимости пользователь может задать собственную функцию принадлежности. Для этого ее следует определить в отдельном m-файле как функцию и использовать имя этого файла в качестве третьего аргумента функции `evalmf`. При этом второй аргумент `mfParams` должен соответствовать параметрам этой функции принадлежности.

Пример использования функции `evalmf`

```
x=0:0.1:10;
y=evalmf(x,[4.5 1.5],'tri2mf');
plot(x,y)
```

Результат выполнения этой последовательности команд приведет к заданию треугольной функции принадлежности, график которой изображен на рис. П3.6.

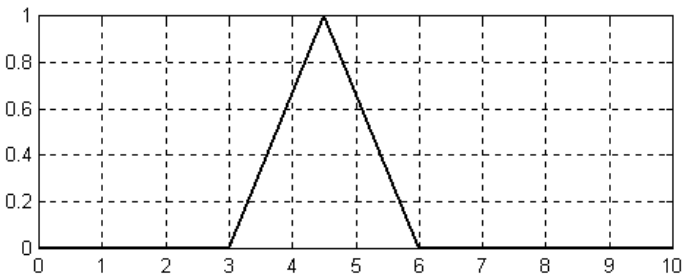


Рис. П3.6. График треугольной функции принадлежности, заданной с использованием функции `evalmf`

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

fcm

Назначение. Кластеризация методом нечетких *c*-средних (алгоритм FCM).

Синтаксис.

```
[center, U, obj_fcn] = fcm(data, cluster_n)
[center, U, obj_fcn] = fcm(data, cluster_n, options)
```

Описание. Функция `fcm(data, cluster_n)` реализует нахождение нечеткой кластеризации заданного множества объектов алгоритмом FCM, который, рассмотрен в *главе 13*.

Входными аргументами этой функции являются:

- `data` — матрица исходных данных кластеризации, каждая строка которой представляет информацию об одном объекте нечеткой кластеризации;
- `cluster_n` — число искомых нечетких кластеров (должно быть больше единицы).

Выходными аргументами этой функции являются:

- `center` — матрица центров искомых нечетких кластеров, каждая строка которой представляет координаты центра одного из нечетких кластеров;

- `U` — матрица функций принадлежности искомого нечеткого разбиения;
- `obj_fcn` — значения целевой функции на каждой из итераций работы алгоритма.

Функция `fcm(data, cluster_n, options)` использует переменную дополнительных аргументов `options` для управления процессом нечеткой кластеризации, для изменения критерия остановки работы алгоритма и/или отображения информации на экране монитора.

Эти дополнительные аргументы имеют следующее значение:

- `options(1)` — экспоненциальный вес для расчета матрицы нечеткого разбиения `U` (по умолчанию это значение равно 2);
- `options(2)` — максимальное число итераций (по умолчанию это значение равно 100);
- `options(3)` — параметр сходимости алгоритма (по умолчанию это значение равно 0.00001);
- `options(4)`: информация о текущей итерации, отображаемая на экране монитора (по умолчанию это значение равно 1).

Если любое из значений дополнительных аргументов равно NaN, то для этого аргумента используется значение по умолчанию. Функция `fcm` заканчивает свою работу, когда алгоритм FCM выполнит максимальное количество итераций, или когда разность между значениями целевых функций на двух последовательных итерациях будет меньше заданного априори значения параметра сходимости алгоритма.

Пример решения задачи нечеткой кластеризации с помощью функции `fcm`

```
data = rand(100, 2);
[center,U,obj_fcn] = fcm(data, 2);
plot(data(:,1), data(:,2),'o');
maxU = max(U);
index1 = find(U(1,:)== maxU);
index2 = find(U(2, :)== maxU);
line(data(index1,1), data(index1, 2), 'linestyle', 'none',...
'marker', '*','color', 'g');
line(data(index2,1), data(index2, 2), 'linestyle', 'none',...
'marker', '*','color', 'r');
```

Выполнение этой последовательности команд приведет к графическому представлению результата нечеткой кластеризации, график которой изображен на рис. П3.7.

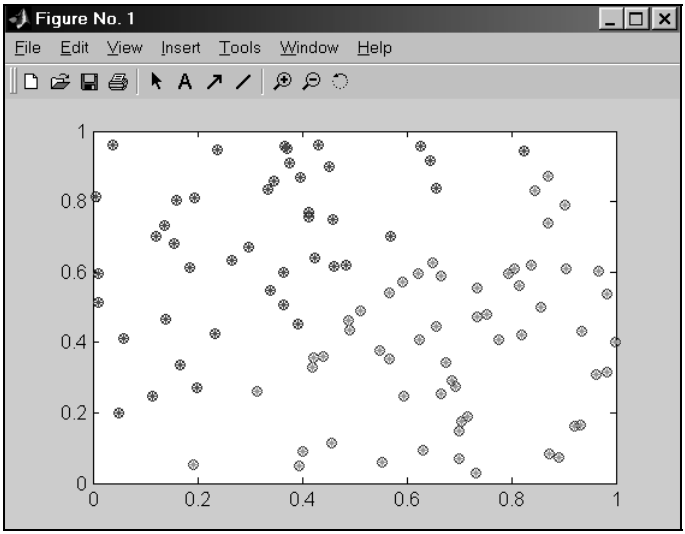


Рис. П3.7. Результат нечеткой кластеризации множества с использованием функции `fcm`

findcluster

Назначение. Вызывает графический интерфейс для интерактивной кластеризации методами нечетких *c*-средних и субтрактивной кластеризации.

Синтаксис.
```
findcluster
findcluster('file.dat')
```

Описание. Функция `findcluster` вызывает графический интерфейс GUI для выполнения нечеткой кластеризации алгоритмом FCM и/или субтрактивной (subtractive) кластеризации. Выбор метода нечеткой кластеризации осуществляется с помощью раскрывающегося списка **Methods**. Исходные данные загружаются в рабочую область из внешнего файла с помощью кнопки **Load Data**.

Для каждого из методов нечеткой кластеризации в соответствующих строках ввода установлены значения параметров алгоритмов по умолчанию. Эти значения могут быть изменены пользователем. Для этого необходимо установить курсор ввода в соответствующее поле и набрать нужные цифры с учетом допустимых значений параметров. Назначение этих параметров для алгоритма FCM описано при рассмотрении функции `fcm`. Назначение этих параметров для алгоритма субтрактивной кластеризации описано при рассмотрении функции `subclust`.

Хотя графический интерфейс нечеткой кластеризации позволяет работать с многомерными исходными данными, он визуализирует данные измерений только по 2-м признакам. Выбор признаков осуществляется с помощью раскрывающихся

списков **X-axis** и **Y-axis** в нижней части интерфейса. Например, для исходных данных из внешнего файла clusterdemo.dat можно изобразить 2 из 3-х признаков, выбрав в полях меню `data_1`, `data_2` или `data_3`. Нажатие на кнопку **Start** приводит к началу работы соответствующего алгоритма, после окончания его работы результаты отображаются на графике. Найденные центры кластеров можно сохранить во внешнем файле с целью их последующего анализа.

Если функция используется в виде `findcluster ('file.dat')`, то при вызове графического интерфейса в рабочую область автоматически загружаются данные кластеризации из внешнего файла file.dat, а на графике отображаются значения матрицы данных для первых двух признаков.

Пример вызова графического интерфейса нечеткой кластеризации с исходными данными из внешнего файла clusterdemo.dat

```
findcluster('clusterdemo.dat')
```

Результат вызова графического интерфейса нечеткой кластеризации для этого случая изображен на рис. П3.8.

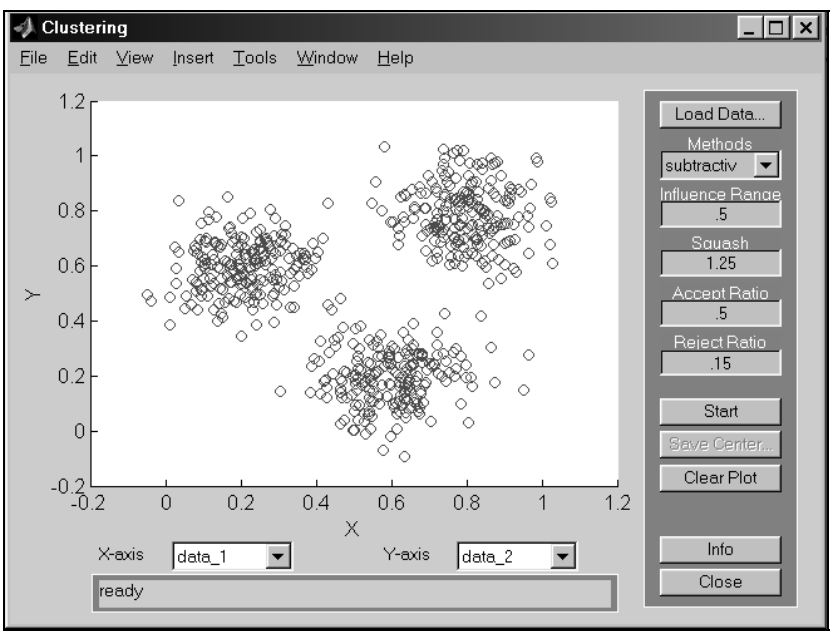


Рис. П3.8. Результат вызова графического интерфейса нечеткой кластеризации с использованием функции `findcluster`

См. также: `fcm`, `subclust`.

fuzarith

Назначение. Выполнение операций нечеткой арифметики.

Синтаксис. `C = fuzarith(X, A, B, operator)`

Описание. Используя операции с нечеткими интервалами *(см. главу 5)*, функция `C = fuzarith(X, A, B, operator)` возвращает нечеткое множество `C` в качестве результата выполнения бинарной операции, определяемой строкой `operator`, над выпуклыми нечеткими множествами `A` и `B`. При этом нечеткие множества `A` и `B` предварительно должны быть заданы выпуклыми функциями принадлежности на универсуме `X`.

Аргументами этой функции являются:

- `A`, `B`, `X` — векторы одинаковой размерности;
- `operator` — строка, принимающая одно из следующих значений: `'sum'`, `'sub'`, `'prod'`, `'div'`.

В качестве результата функция возвращает `C` — вектор-столбец той же размерности, что и вектор универсума `X`.

Примечание

Нечеткое сложение может генерировать сообщение "*деление на нуль*" ('divide by zero'), но это сообщение не влияет на правильность выполнения этой операции.

Пример использования функции `fuzarith`

```
point_n = 1001; % определяет количество точек функции принадлежности
min_x = -20; max_x = 20;% универсум нечеткого множества [min_x, max_x]
x = linspace(min_x, max_x, point_n)';
A = trapmf(x, [-2 -1 1 3]); % A — трапециевидный нечеткий интервал
B = trimf(x, [2 3 5]); % B — треугольное нечеткое число
C1 = fuzarith(x, A, B, 'sum');
plot(x, A, x, B, x, C1);
title('нечеткое сложение A+B');
C2 = fuzarith(x, A, B, 'sub');
plot(x, A, x, B, x, C2);
title('нечеткое вычитание A-B');
C3 = fuzarith(x, A, B, 'prod');
plot(x, A, x, B, x, C3);
title('нечеткое умножение A·B');
C4 = fuzarith(x, A, B, 'div');
plot(x, A, x, B, x, C4);
title('нечеткое деление A/B');
```

Результат выполнения этой последовательности команд изображен на рис. П3.9.

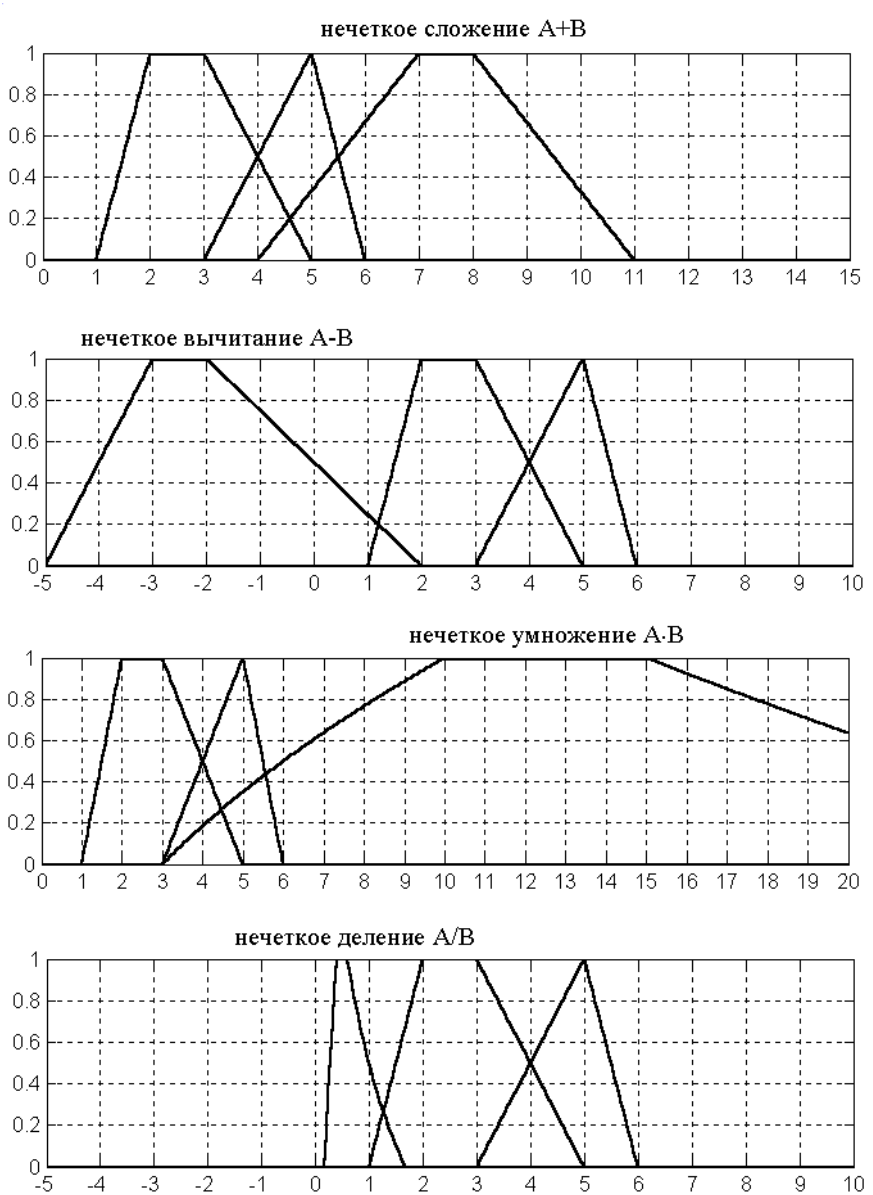


Рис. П3.9. Результаты выполнения операций с нечетким интервалом `A` и нечетким числом `B`

fuzblock

Назначение. Блоки нечеткой логики из пакета Simulink.

Синтаксис. `fuzblock`

Описание. Эта функция вызывает систему Simulink, в которой кроме демонстрационных примеров, содержатся три специальных блока.

- Контроллер нечеткой логики (Fuzzy Logic Controller).
- Контроллер нечеткой логики с окном просмотра правил (Fuzzy Logic Controller with Rule Viewer).
- Блок функций принадлежности MF (рис. П3.10).

Последний из блоков позволяет просматривать правила в ходе моделирования, выполняемого в системе Simulink. После двойного щелчка левой кнопкой мыши на блоке контроллера нечеткой логики открывается диалоговое окно свойств этого блока. Это окно содержит имя структуры FIS, которая находится в рабочей области системы MATLAB и которую предполагается моделировать в системе Simulink.

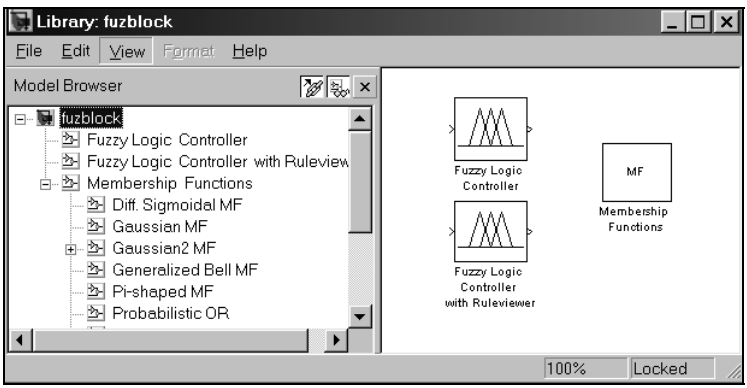


Рис. П3.10. Графический интерфейс блоков нечеткой логики системы Simulink

Чтобы открыть диалоговое окно свойств контроллера нечеткой логики с окном просмотра правил, необходимо:

1. Выполнить двойной щелчок левой кнопкой мыши на этом блоке, в результате чего откроется диалоговое окно свойств контроллера нечеткой логики.
2. Выполнить двойной щелчок левой кнопкой мыши на втором блоке контроллера нечеткой логики.

Если система нечеткого вывода имеет несколько входов, эти входы должны быть соединены вместе до размещения их в блоке контроллера нечеткой логики или в блоке контроллера нечеткой логики с окном просмотра правил. Аналогично,

если система нечеткого вывода имеет несколько выходов, эти выходы должны быть соединены вместе до размещения их в блоке контроллера нечеткой логики или в блоке контроллера нечеткой логики с окном просмотра правил.

См. также: `sffis`, `ruleview`.

fuzdemos

Назначение. Список всех демонстрационных примеров в пакете Fuzzy Logic Toolbox.

Синтаксис. `fuzdemos`

Описание. Эта функция вызывает графический интерфейс, который позволяет выбрать для анализа любой из демонстрационных примеров, поставляемых с пакетом Fuzzy Logic Toolbox. С демонстрационными примерами можно также познакомиться с использованием следующих функций командной строки:

- `defuzzdm` — пример методов дефаззификации;
- `fcmdemo` — пример кластеризации точек на плоскости методом нечетких c-средних FCM;
- `gasdemo` — пример применения системы для эффективного управления топливом с использованием метода субтрактивной кластеризации;
- `juggler` — пример просмотра правил для задачи жонглирования мяча;
- `invkine` — пример инверсной кинематики захвата механической руки робота;
- `irisfcm` — пример кластеризации точек в четырехмерном пространстве методом нечетких c-средних FCM;
- `noisedm` — пример адаптивного устранения шума;
- `slbb` — пример управления движением мяча на качающейся балке с помощью средств пакета Simulink;
- `slcp` — пример управления инвертированным маятником с помощью средств пакета Simulink;
- `sltank` — пример управления уровнем воды с помощью средств пакета Simulink;
- `sltankrule` — пример управления уровнем воды с использованием программы просмотра правил с помощью средств пакета Simulink;
- `sltbu` — пример нечеткого управления движением грузового автомобиля (только Simulink).

fuzzy

Назначение. Вызывает редактор систем нечеткого вывода FIS.

Синтаксис.

```
fuzzy
fuzzy(fismat)
```

Описание. Эта функция предоставляет пользователю возможность редактировать на высоком уровне свойства системы нечеткого вывода, такие как число входных и выходных переменных, используемый метод дефаззификации и т. д.

Редактор систем нечеткого вывода FIS (или просто редактор FIS) обладает графическим интерфейсом и позволяет вызывать все другие редакторы и программы просмотра систем нечеткого вывода. Графический интерфейс этого редактора обладает максимальным удобством и гибкостью, необходимой для интерактивной работы с отдельными компонентами системы нечеткого вывода.

В верхней части рабочего интерфейса редактора FIS изображается диаграмма, представляющая в визуальной форме входы и выходы системы нечеткого вывода, в центре которых находится процессор нечетких правил. Щелчок на прямоугольнике с изображением входа или выхода выделяет соответствующую переменную и делает ее текущей. Прямоугольник текущей переменной при этом выделяется красным цветом.

Двойной щелчок на прямоугольнике с изображением входной или выходной переменной вызывает редактор функций принадлежности с загруженной в него соответствующей переменной. Двойной щелчок на изображении процессора нечетких правил вызывает редактор правил для соответствующей системы нечеткого вывода. Если некоторая переменная существует в системе нечеткого вывода, но не используется в правилах вывода, то связь ее с процессором нечетких правил изображается не сплошной, а пунктирной линией.

Редактор FIS имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

❒ Пункт меню **File** (Файл) редактора FIS содержит следующие операции:

- **New FIS...** — позволяет выбрать тип задаваемой новой системы нечеткого вывода: **Mamdani** — типа Мамдани или **Sugeno** — типа Сугено. При этом задаваемая система нечеткого вывода не имеет ни входных, ни выходных переменных, а ее имя задается по умолчанию как Untitled;
- **Import** — позволяет загрузить в редактор FIS существующую систему нечеткого вывода одним из следующих способов: **From Workspace...** — из рабочего пространства программы MATLAB или **From Disk...** — из внешнего файла. В последнем случае вызывается стандартное диалоговое окно открытия файла;
- **Export** — позволяет сохранить редактируемую систему нечеткого вывода одним из следующих способов: **To Workspace...** — в рабочем пространстве программы MATLAB или **To Disk...** — во внешнем файле. В последнем случае вызывается стандартное диалоговое окно сохранения файла;
- **Print** — позволяет распечатать на принтере редактируемую систему нечеткого вывода. В этом случае вызывается стандартное диалоговое окно настройки свойств печати на подключенном к данному компьютеру принтере;
- **Close** — закрывает редактор FIS, при этом вызывается диалоговое окно с предложениями сохранить или отказаться от сохранения редактируемой системы нечеткого вывода.

- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **Add Variable...** — позволяет добавить в редактируемую систему нечеткого вывода переменную одного из следующих типов: **Input** — входную переменную или **Output** — выходную переменную;
  - **Remove Selected Variable** — удаляет выбранную переменную из редактируемой системы нечеткого вывода;
  - **Membership Functions...** — вызывает редактор функций принадлежности;
  - **Rules** — вызывает программу редактирования правил.
- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил;
  - **Surface** — вызывает программу просмотра поверхности вывода.

В левой нижней части рабочего интерфейса редактора FIS имеется 5 раскрывающихся меню:

- **And method** (Метод логической конъюнкции) — позволяет задать один из следующих методов для выполнения логической конъюнкции в условиях нечетких правил:
  - **min** — метод минимального значения (6.2);
  - **prod** — метод алгебраического произведения (6.3);
  - **Custom** — метод, определенный самим пользователем.
- **Or method** (Метод логической дизъюнкции) — позволяет задать один из следующих методов для выполнения логической дизъюнкции в условиях нечетких правил:
  - **max** — метод максимального значения (6.6);
  - **probor** — метод алгебраической суммы (6.7);
  - **Custom** — метод, определенный самим пользователем.
- **Implication method** (Метод вывода заключения) — позволяет задать один из следующих методов для выполнения (активизации) логического заключения в каждом из нечетких правил:
  - **min** — метод минимального значения (7.6);
  - **prod** — метод алгебраического произведения (7.7);
  - **Custom** — метод, определенный пользователем. Это меню не используется для систем нечеткого вывода типа Сугено.
- **Aggregation method** (Метод агрегирования) — позволяет задать один из следующих методов для агрегирования значений функции принадлежности каждой из выходных переменных в заключениях нечетких правил:
  - **max** — метод максимального значения (6.6);
  - **sum** — метод граничной суммы (6.8);

- **probor** — метод алгебраической суммы (6.7);
- **Custom** — метод, определенный пользователем.

Это меню не используется для систем нечеткого вывода типа Сугено.

❒ **Defuzzification method** (Метод дефаззификации) — позволяет задать один из следующих методов для выполнения дефаззификации выходных переменных в системе нечеткого вывода типа Мамдани:

- **centroid** — метод центра тяжести для дискретного множества значений функции принадлежности (7.10);
- **bisector** — метод центра площади (модификация (7.11));
- **mom** (middle of maximum) — метод среднего максимума, определяемый как среднее арифметическое левого и правого модальных значений;
- **som** (smallest of maximum) — метод левого модального значения (7.12);
- **lom** (largest of maximum) — метод правого модального значения (7.13);
- **Custom** — метод, определенный самим пользователем.

Для систем нечеткого вывода типа Сугено можно выбрать один из следующих методов дефаззификации:

- **wtaver** (weighted average) — метод взвешенного среднего (7.15);
- **wtsum** (weighted sum) — метод взвешенной суммы (7.16).

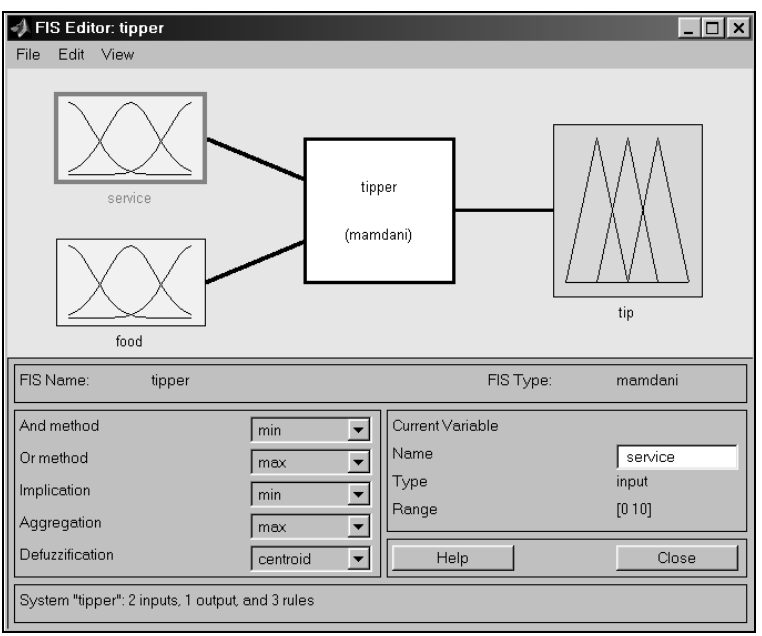


Рис. П3.11. Графический интерфейс редактора FIS, вызываемый функцией `fuzzy('tipper')`

Более подробно особенности работы с редактором FIS были изложены в *главе 12*.

Пример использования функции `fuzzy`

```
fuzzy('tipper')
```

Результат выполнения функции `fuzzy` изображен на рис. П3.11.

См. также: `mfedit`, `ruleedit`, `ruleview`, `surfview`, `anfisedit`.

gauss2mf

Назначение. Встроенная *П*-образная функция принадлежности, которая является комбинацией двух функций Гаусса.

Синтаксис. `y = gauss2mf(x, [sig1 c1 sig2 c2])`

Описание. Функция Гаусса описывает плотность нормального распределения, определяется выражением (2.22) и зависит от двух параметров `sig` (или σ) и `c`. Функция `gauss2mf` является комбинацией двух таких функций. Первая функция Гаусса задается параметрами `sig1` и `c1` и определяет форму кривой функции принадлежности слева от модального значения. Вторая функция Гаусса задается параметрами `sig2` и `c2` и определяет форму кривой функции принадлежности справа от модального значения.

Если выполняется неравенство `c1<c2`, то функция `gauss2mf` достигает своего максимального значения, равного 1. В противном случае ее максимальное значение меньше 1. Параметры должны быть указаны в следующем порядке: `[sig1 c1 sig2 c2]`.

Пример задания функции принадлежности `gauss2mf`

```
x = 0:0.1:10;
y1 = gauss2mf(x, [2 4 1 8]);
y2 = gauss2mf(x, [2 5 1 7]);
y3 = gauss2mf(x, [2 6 1 6]);
y4 = gauss2mf(x, [2 7 1 5]);
y5 = gauss2mf(x, [2 8 1 4]);
plot(x,y1,x,y2,x,y3,x,y4,x,y5)
```

Графики соответствующих функций принадлежности, полученные в результате выполнения указанной последовательности команд, изображены на рис. П3.12.

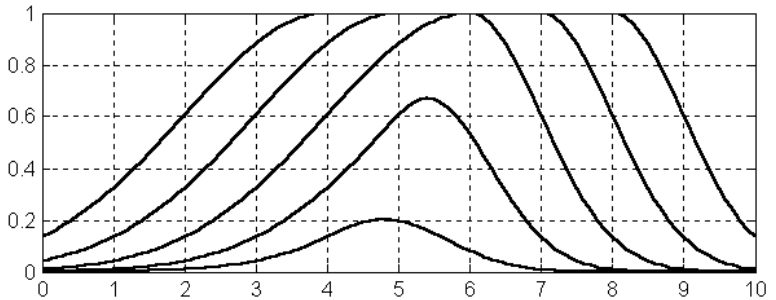


Рис. П3.12. Графики нескольких функций принадлежности, являющихся комбинацией двух функций Гаусса

См. также: `dsigmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

gaussmf

Назначение. Встроенная *П*-образная функция принадлежности типа функции Гаусса.

Синтаксис. `y = gaussmf(x, [sig c])`

Описание. Симметричная функция Гаусса описывает плотность нормального распределения и определяется выражением (2.22). Эта функция имеет два параметра: `sig` (или σ) и `c`, которые задаются в форме вектора `[sig c]`.

Пример задания функции принадлежности `gaussmf`

```
x=0:0.1:10;
y=gaussmf(x,[2 4]);
plot(x,y)
```

График этой функции принадлежности изображен на рис. 2.18, *б*.

См. также: `dsigmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

gbellmf

Назначение. Встроенная *П*-образная функция принадлежности типа колоколообразной кривой.

Синтаксис. `y = gbellmf(x, params)`

Описание. Данная обобщенная функция принадлежности типа колоколообразной кривой (*П*-образная функция принадлежности) определяется выражением (2.21) и зависит от трех параметров: `a`, `b`, `c`, при этом параметр `b` должен быть обязательно положительным. Эти параметры являются вторым аргументом функции и задаются в форме вектора `params`, компоненты которого записываются в следующем порядке: `[a b c]`.

Пример задания функции принадлежности `gbellmf`

```
x=0:0.1:10;
y=gbellmf(x,[2 3 6]);
plot(x,y)
```

График этой функции принадлежности изображен на рис. 2.18, *а*.

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

genfis1

Назначение. Генерирует структуру системы нечеткого вывода FIS на основе экспериментальных данных без кластеризации этих данных.

Синтаксис.

```
fismat = genfis1(data)
fismat = genfis1(data,numMFs,inmftype, outmftype)
```

Описание. Функция `genfis1` генерирует систему нечеткого вывода FIS типа Сугено, которая может быть использована в качестве исходной структуры для инициализации параметров функций принадлежности в процессе обучения гибридной сети anfis. Эта функция в формате `genfis1(data, numMFs, inmftype, outmftype)` генерирует структуру FIS на основе некоторой обучающей выборки с именем `data` с использованием некоторого сеточного разбиения этих данных без их кластеризации.

Аргументами этой функции являются:

- `data` — матрица данных (обучающая выборка), последний столбец которой соответствует единственной выходной переменной, а остальные столбцы соответствуют входным переменным;
- `numMFs` — вектор, компоненты которого определяют количество функций принадлежности у каждой из входных переменных. Если пользователю необходимо задать одинаковое количество функций принадлежности для всех входных переменных, то для этого достаточно задать аргумент `numMFs` как обычное число (скаляр);
- `inmftype` — массив строковых значений, каждое из которых специфицирует тип функции принадлежности для соответствующей входной переменной. Аналогично, если пользователю необходимо определить один и тот же тип

функции принадлежности для всех входных переменных, то для этого достаточно задать этот аргумент как простую строку;

- `outmftype` — строка, которая специфицирует тип функции принадлежности выходной переменной. Поскольку используется система нечеткого вывода типа Сугено, то существует только одна выходная переменная. Тип функции принадлежности выходной переменной может быть или линейным, или константой.

Количество функций принадлежности для выходной переменной должно быть равно количеству правил, генерируемых функцией `genfis1`. По умолчанию значение аргумента `numMFs` равно 2, а тип функций принадлежности входных и выходной переменных равен `'gbellmf'`. Эти значения используются всякий раз, когда функция `genfis1` вызывается без последних трех аргументов.

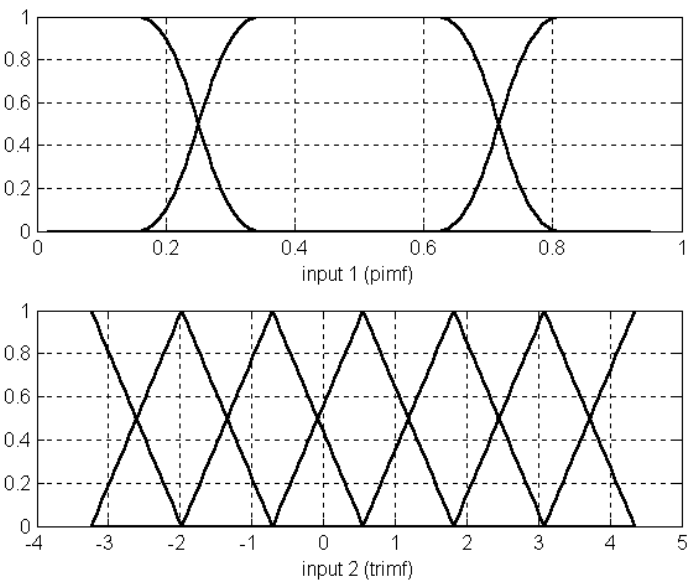


Рис. П3.13. Результат выполнения функции `genfis1`

Пример использования функции `genfis1`

```
data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];
numMFs = [3 7];
mfType = str2mat('pimf','trimf');
fismat = genfis1(data,numMFs,mfType);
[x,mf] = plotmf(fismat,'input',1);
```

```
subplot(2,1,1), plot(x,mf);
xlabel('input 1 (pimf)');
[x,mf] = plotmf(fismat,'input',2);
subplot(2,1,2), plot(x,mf);
xlabel('input 2 (trimf)');
```

Результат выполнения функции `genfis1` изображен на рис. П3.13.

См. также: `anfis`.

genfis2

Назначение. Генерирует структуру системы нечеткого вывода FIS с использованием метода субтрактивной кластеризации.

Синтаксис.

```
fismat = genfis2(Xin,Xout,radii)
fismat = genfis2(Xin,Xout,radii,xBounds)
fismat = genfis2(Xin,Xout,radii,xBounds,options)
```

Описание. Функция `genfis2` генерирует структуру системы нечеткого вывода FIS методом субтрактивной кластеризации на основе экспериментальных данных, заданных в виде двух отдельных множеств входных и выходных данных. Если задана только одна выходная переменная, функция `genfis1` может быть использована для генерации структуры адаптивной системы нейро-нечеткого вывода (FIS типа Сугено) с целью ее последующего обучения на основе предварительной субтрактивной кластеризации экспериментальных данных.

Функция `genfis2` выполняет генерацию структуры посредством спецификации множества правил, которые соответствуют исходному множеству данных. При формирования правил нечетких продукций для определения количества правил и функций принадлежности их условий на первом этапе используется функция `subclust`. На втором этапе методом наименьших квадратов определяются заключения правил нечетких продукций. В итоге функция `genfis2` возвращает структуру системы нечеткого вывода FIS, база правил которой охватывает все пространство экспериментальных данных.

Аргументами этой функции являются:

- `Xin` — матрица, каждая строка которой представляет исходные значения входных переменных;
- `Xout` — матрица, каждая строка которой представляет исходные значения выходных переменных;
- `radii` — вектор, который задает интервал (радиус) влияния центров кластеров по каждой координате векторов исходных данных в предположении, что все данные находятся внутри некоторого единичного гиперкуба;

  Например, если используются 3-мерные исходные данные, т. е. матрица `Xin` имеет 2 столбца, а матрица `Xout` — 1 столбец, то значения `radii = [0.5 0.4 0.3]`

задают относительные интервалы влияния в пределах рассматриваемого гиперкуба по каждой из трех переменных: по первой входной переменной 0.5, по второй входной переменной — 0.4, и по выходной переменной — 0.3. Если параметр `radii` является скаляром, то соответствующее значение относится к каждой из переменных нечеткой модели, т. е. центр каждого кластера будет иметь сферическую окрестность влияния заданного радиуса;

- `xBounds` — необязательная матрица размерности ($2 \times n$), специфицирующая диапазоны значений по каждой из входных и выходных переменных нечеткой модели (здесь n — общее количество входных и выходных переменных). Эта спецификация необходима для преобразования исходных значений в значения единичного гиперкуба. Первая строка матрицы `xBounds` задает минимальные значения диапазона значений по каждой из осей многомерных данных, а вторая строка — дает максимальные значения диапазона значений по каждой их осей многомерных данных.

  Например, значения матрицы `xBounds = [-10 0 -1; 10 50 1]` определяют, что исходные данные по первой оси нормализуются в интервале [–10, 10], исходные данные по второй оси — в интервале [0, 1] и исходные данные по третьей оси нормализуются в интервале [–1, 1]. Если матрица `xBounds` не задана, то диапазоны нормализации значений определяются по умолчанию — как минимальные и максимальные значения каждой их переменных нечеткой модели, присутствующие в исходных данных;

- `options` — необязательный вектор, который специфицирует параметры алгоритма генерации структуры FIS аналогично параметрам функции `subclust`. Более подробно эти функции рассматриваются в *главе 13*.

Пример использования функции `genfis2` с минимальным числом аргументов

```
fismat = genfis2(Xin,Xout,0.5);
```

Пример использования функции `genfis2` с нормализацией исходных данных

```
fismat = genfis2(Xin,Xout,0.5,[-10 -5 0; 10 5 20]);
```

В данном случае задается параметр нормализации для преобразования значений матриц `Xin` и `Xout` в значения единичного гиперкуба, т. е. в интервал [0, 1] по каждой из переменных. Поскольку для рассматриваемого примера матрица `Xin` имеет 2 столбца, а матрица `Xout` — 1 столбец, то выполняется шкалирование по каждой их переменных. При этом значения первой входной переменной нормализуются в интервале [–10, 10], второй входной переменной — в интервале [–5, 5], а выходной переменной — в интервале [0, 20].

См. также: `subclust`.

gensurf

Назначение. Генерирует поверхность нечеткого вывода FIS.

Синтаксис.
```
gensurf(fis)
gensurf(fis, inputs, output)
gensurf(fis, inputs, output, grids)
gensurf(fis, inputs, output, grids, refinput)
```

Описание. Функция `gensurf(fis)` позволяет получить изображение поверхности нечеткого вывода для одной из выходных переменных структуры системы нечеткого вывода FIS с именем `fis`. В этом формате для построения поверхности вывода используются первые две входные переменные и первая выходная переменная структуры `fis`.

Функция `gensurf(fis, inputs, output)` во втором формате позволяет получить изображение поверхности нечеткого вывода для одной или двух входных переменных, номера которых задаются вектором `inputs`, и одной выходной переменной с номером `output` для структуры системы нечеткого вывода FIS с именем `fis`.

Функция `gensurf(fis, inputs, output, grids)` в третьем формате позволяет получить изображение сетки на рисунке поверхности нечеткого вывода для первой горизонтальной переменной и второй вертикальной входной переменной. При этом номера переменных задаются вектором `grids`.

Функция `gensurf(fis, inputs, output, grids, refinput)` в четвертом формате позволяет получить изображение поверхности нечеткого вывода для более чем двух выходных переменных. В этом случае вектор `refinput` задает номера входных переменных, которые рассматриваются как неизменяющиеся.

Функция может быть также вызвана в формате `[x, y, z]=gensurf(...)`. В этом случае она возвращает значения переменных, которые определяют поверхность вывода и могут быть использованы для изображения этой поверхности средствами графики MATLAB.

Пример использования функции `gensurf`

```
a = readfis('tipper');
gensurf(a)
```

График полученной поверхности нечеткого вывода изображен на рис. П3.14.

См. также: `evalfis`, `surfview`.

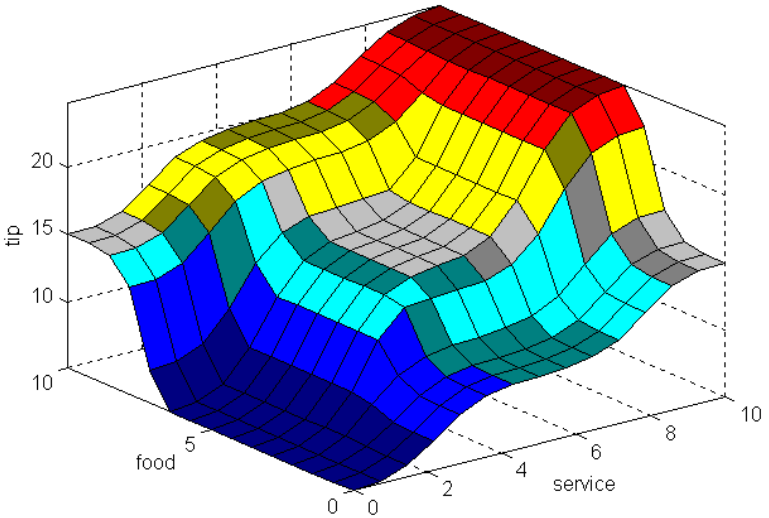


Рис. П3.14. График поверхности нечеткого вывода, полученный с помощью функции `gensurf`

getfis

Назначение. Позволяет отобразить различные свойства системы нечеткого вывода FIS.

Синтаксис.

```
getfis(a)
getfis(a, 'fisprop')
getfis(a, 'vartype', varindex, 'varprop')
getfis(a, 'vartype', varindex, 'mf', mfindex)
getfis(a, 'vartype', varindex, 'mf',mfindex, 'mfprop')
```

Описание. Это основная функция, предназначенная для отображения в окне команд отдельных свойств структуры системы нечеткого вывода FIS. Функция `getfis` может использоваться в одном из 5 указанных форматов со следующими аргументами:

- `a` — имя структуры FIS в рабочей области MATLAB;
- `'vartype'` — строка с указанием типа отображаемой переменной (`input` или `output`);
- `varindex` — целое число, соответствующее номеру отображаемой переменной (например, 1 для первой из входных переменных);
- `'mf'` — строка, указывающая на отображение информации о функции принадлежности;
- `mfindex` — целое число, соответствующее номеру отображаемой функции принадлежности (например, 1 для первой из функций принадлежности).

Пример использования функции `getfis` с одним аргументом

```
a = readfis('tipper');
getfis(a)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name =         tipper
Type =         mamdani
NumInputs =    2
InLabels =
       service
       food
NumOutputs =   1
OutLabels =
       tip
NumRules =     3
AndMethod =    min
OrMethod =     max
ImpMethod =    min
AggMethod =    max
DefuzzMethod = centroid
ans =
       tipper
```

Пример использования функции `getfis` с двумя аргументами

```
getfis(a,'type')
```

В этом случае в окно команд будет выведена следующая информация:

```
ans =
       mamdani
```

Пример использования функции `getfis` с тремя аргументами

```
getfis(a,'input',1)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name =         service
NumMFs =       3
MFLabels =
```

```
       poor
       good
       excellent
Range =        [0 10]
ans =
       [ ]
```

Пример использования функции `getfis` с четырьмя аргументами

```
getfis(a,'input',1,'name')
```

В этом случае в окно команд будет выведена следующая информация:

```
ans =
       service
```

Пример использования функции `getfis` с пятью аргументами

```
getfis(a,'input',1,'mf',2)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name = good
Type = gaussmf
Params =[1.5  5]
ans =
       [ ]
```

Пример использования функции `getfis` с шестью аргументами

```
getfis(a,'input',1,'mf',2,'name')
```

В этом случае в окно команд будет выведена следующая информация:

```
ans =
       good
```

См. также: `setfis`, `showfis`.

mam2sug

Назначение. Преобразует систему нечеткого вывода FIS типа Мамдани в систему нечеткого вывода FIS типа Сугено.

Синтаксис. `sug_fis=mam2sug(mam_fis)`

Описание. Функция `mam2sug` преобразует структуру FIS типа Мамдани с именем `mam_fis` (не обязательно с единственным выходом) в структуру FIS типа Сугено с именем `sug_fis`. Получаемая в результате система нечеткого вывода типа Сугено имеет функции принадлежности, каждая из которых имеет некоторое постоянное значение. Эти константы определяются методом центроида для соответствующих функций исходной системы нечеткого вывода типа Мамдани. При этом условия правил нечеткого вывода остаются без изменений.

Пример использования функции `mam2sug` для преобразования структуры FIS типа Мамдани из внешнего файла tipper.fis в структуру FIS типа Сугено

```
mam_fismat = readfis('tipper.fis');
sug_fismat = mam2sug(mam_fismat);
subplot(2,2,1);
gensurf(mam_fismat, [1 2], 1);
title('Система нечеткого вывода типа Мамдани');
subplot(2,2,2);
gensurf(sug_fismat, [1 2], 1);
title('Система нечеткого вывода типа Сугено');
```

Результат выполнения этой последовательности команд изображен на рис. П3.15.

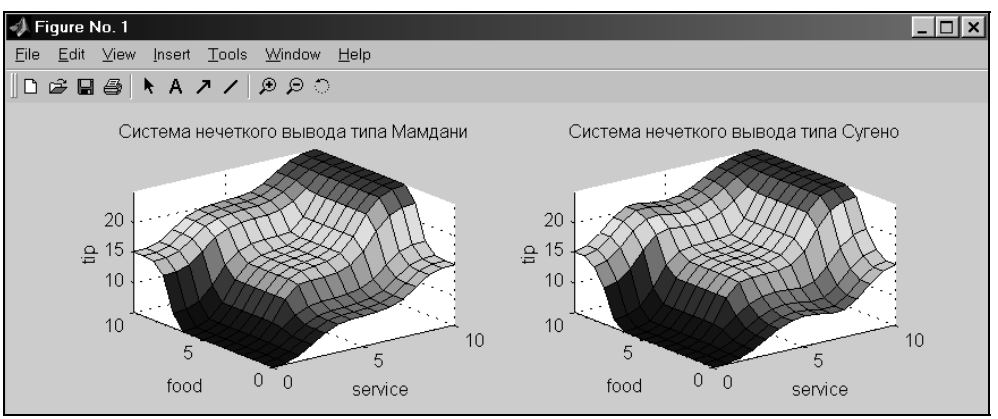


Рис. П3.15. Преобразование системы нечеткого вывода типа Мамдани в систему нечеткого вывода типа Сугено

mf2mf

Назначение. Преобразует параметры двух функций принадлежности.

Синтаксис. `outParams = mf2mf(inParams, inType, outType)`

Описание. Эта функция преобразует одну из встроенных функций принадлежности в любую другую встроенную функцию принадлежности. При этом подобное преобразование выполняется на уровне параметров соответствующих функций принадлежности. В принципе функция `mf2mf` сохраняет точки симметрии старой и новой функций принадлежности. Однако результат выполнения преобразования может привести к потере информации при выполнении обратного преобразования новой функции принадлежности в функцию принадлежности исходного типа. В этом случае преобразованная функция принадлежности может по своему виду отличаться от первоначальной функции принадлежности.

Функция `mf2mf` имеет следующие входные аргументы:

- `inParams` — параметры исходной функции принадлежности, которая подлежит преобразованию;
- `inType` — строка с именем типа исходной функции принадлежности, которая подлежит преобразованию;
- `outType` — строка с именем типа новой функции принадлежности, в который преобразуется исходная функция принадлежности.

Пример использования функции `mf2mf` для преобразования структуры FIS типа Мамдани из внешнего файла tipper.fis в структуру FIS типа Сугено

```
x=0:0.1:5;
mfp1 = [1 2 3];
mfp2 = mf2mf(mfp1,'gbellmf','trimf');
plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
```

Результат выполнения этой последовательности команд изображен на рис. П3.16.

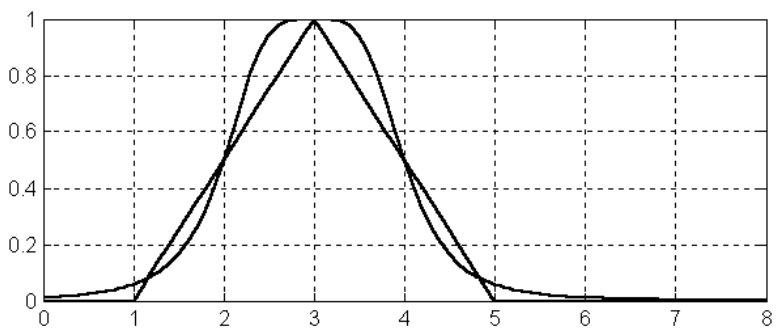


Рис. П3.16. Графики *П*-образной исходной функции принадлежности и треугольной функции принадлежности, полученной в результате преобразования с помощью функции `mf2mf`

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

mfedit

Назначение. Вызывает графический интерфейс редактора функций принадлежности системы нечеткого вывода FIS.

Синтаксис.
```
mfedit('a')
mfedit(a)
mfedit
```

Описание. Эта функция, записанная в формате `mfedit('a')`, вызывает редактор функций принадлежности, который позволяет пользователю в графическом режиме анализировать и модифицировать все функции принадлежности некоторой структуры FIS, сохраненной во внешнем файле с именем a.fis. Функция в формате `mfedit(a)` работает с переменной рабочего пространства MATLAB, соответствующей структуре FIS с именем `a`.

Функция в формате `mfedit` просто вызывает редактор функций принадлежности без загрузки какой бы то ни было структуры FIS. Для каждой функции принадлежности можно изменить ее имя, тип и параметры. Редактор предоставляет пользователю не только возможность выбрать любую из 11 встроенных функций принадлежности, но и задать собственную функцию принадлежности.

Для отображения графиков функций принадлежности следует выбрать необходимую переменную в левой части графического интерфейса редактора (под заголовком **FIS Variables** (Переменные FIS). Чтобы выбрать нужную функцию принадлежности, следует щелкнуть на ней или ее метке.

Редактор функций принадлежности имеет главное меню программы, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора функций принадлежности содержит такие же операции, что и соответствующий пункт меню редактора FIS.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **Add MF...** — позволяет добавить встроенную функции принадлежности термов для выбранной переменной;
  - **Add Custom MF...** — позволяет добавить пользовательскую функцию принадлежности для отдельной переменной;
  - **Remove Current MF** — позволяет удалить отдельную функцию принадлежности;
  - **Remove All MFs** — позволяет удалить все функции принадлежности для отдельной переменной;

- **FIS Properties...** — вызывает графический интерфейс редактора FIS;
- **Rules...** — вызывает графический интерфейс редактора правил FIS.

❒ Пункт меню **View** (Вид) содержит следующие операции:

- **Rules** — вызывает программу просмотра правил FIS;
- **Surface** — вызывает программу просмотра поверхности вывода FIS.

Раскрывающееся меню типов функций принадлежности позволяет выбрать одну из 11 встроенных функций принадлежности. При этом можно также модифицировать существующие функции принадлежности или определить пользовательские функции принадлежности.

Пример использования функции `mfedit`

```
mfedit('tipper')
```

Результат выполнения функции `mfedit('tipper')` изображен на рис. П3.17.

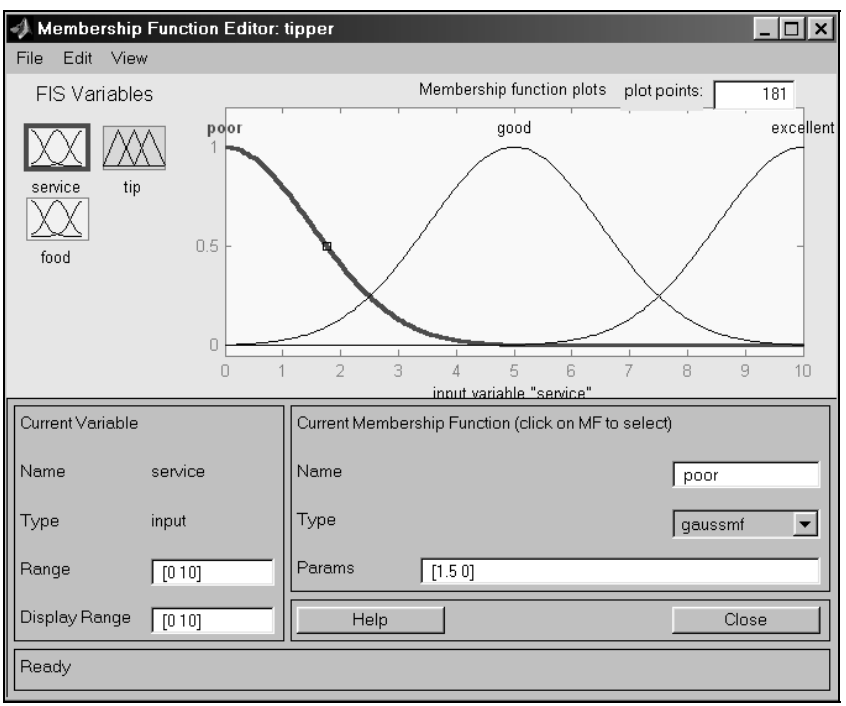


Рис. П3.17. Редактор функций принадлежности, вызванный функцией `mfedit('tipper')`

Более подробно особенности работы с редактором функций принадлежности изложены в *главе 12*.

См. также: `fuzzy`, `ruleedit`, `ruleview`, `surfview`.

newfis

Назначение. Создает новую систему нечеткого вывода FIS.

Синтаксис.
```
a=newfis(fisName, fisType, andMethod, orMethod,
impMethod, ... aggMethod, defuzzMethod)
```

Описание. Эта функция предназначена для создания новых структур систем нечеткого вывода FIS. Функция `newfis` имеет 7 аргументов и возвращает единственное значение — структуру FIS с именем `a`. Аргументами функции являются:

- `fisName` — строка с именем создаваемой структуры FIS, при этом также создается файл с именем fisName.fis;
- `fisType` — тип системы нечеткого вывода FIS;
- `andMethod`, `orMethod`, `impMethod`, `aggMethod`, `defuzzMethod` — соответственно задают методы для выполнения нечетких логических операций `AND`, `OR`, импликации, агрегирования и дефаззификации. Если эти аргументы не указаны, то задаются соответствующие им значения по умолчанию.

Пример использования функции `newfis` с аргументами, принятыми по умолчанию

```
a=newfis('newsys');
getfis(a)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name =         newsys
Type =         mamdani
NumInputs =    0
InLabels =
NumOutputs =   0
OutLabels =
NumRules =     0
AndMethod =    min
OrMethod =     max
ImpMethod =    min
AggMethod =    max
DefuzzMethod =centroid
```

```
ans =
       newsys
```

См. также: `readfis`, `writefis`.

parsrule

Назначение. Осуществляет грамматический анализ нечетких правил.

Синтаксис.
```
fis2 = parsrule(fis,txtRuleList)
fis2 = parsrule(fis,txtRuleList,ruleFormat)
fis2 = parsrule(fis,txtRuleList,ruleFormat,lang)
```

Описание. Эта функция выполняет грамматическую проверку текста в правилах нечетких продукций для структуры нечеткого вывода в рабочей области системы MATLAB. Функция может иметь следующие аргументы:

- `fis` — имя структуры системы нечеткого вывода в рабочей области;
- `txtRuleList` — строка текста, которая определяет некоторое правило нечеткой продукции;
- `ruleFormat` — строка, определяющая формат представления правил нечетких продукций, которая может принимать одно из следующих значений:
  - `'verbose'` — текстовый формат (по умолчанию);
  - `'symbolic'` — символический формат;
  - `'indexed'` — цифровой формат;
- `lang` — строка, определяющая язык записи правил нечетких продукций, которая может принимать одно из следующих значений:
  - `'english'` — английский язык;
  - `'francais'` — французский язык;
  - `'deutsch'` — немецкий язык.

Пример использования функции `parsrule` с тремя аргументами

```
a = readfis('tipper');
ruleTxt = 'if service is poor then tip is generous';
a2 = parsrule(a,ruleTxt,'verbose');
showrule(a2)
```

В этом случае в окно команд будет выведена следующая информация:

```
ans =
       1. If (service is poor) then (tip is generous) (1)
```

См. также: `addrule`, `ruleedit`, `showrule`.

pimf

Назначение. Встроенная *П*-образная функция принадлежности.

Синтаксис. `y = pimf(x, [a b c d])`

Описание. Эта функция принадлежности определяется как алгебраическое произведение двух сплайн-функций (2.19): *Z*-образной функции вида (2.11) и *S*-образной функции вида (2.13). Вторым аргументом этой функции является вектор параметров `[a b c d]`. При этом параметры `a` и `b` характеризуют *Z*-образную кривую, а параметры `c` и `d` характеризуют *S*-образную кривую.

Пример задания функции принадлежности `pimf`

```
x=0:0.1:10;
y=pimf(x,[1 4 5 9]);
plot(x,y)
```

График этой функции принадлежности изображен на рис. 2.16, *б*.

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `psigmf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

plotfis

Назначение. Изображает диаграмму системы нечеткого вывода FIS.

Синтаксис. `plotfis(fismat)`

Описание. Эта функция изображает диаграмму верхнего уровня системы нечеткого вывода FIS с именем `fismat`. Входные переменные и их функции принадлежности располагаются слева от структурных характеристик FIS, в то время как выходные переменные и их функции принадлежности располагаются справа.

Пример использования функции `plotfis`

```
a = readfis('tipper')
plotfis(a)
```

Результат этого вызова функции `plotfis` изображен на рис. П3.18.

См. также: `plotfis`, `evalmf`.

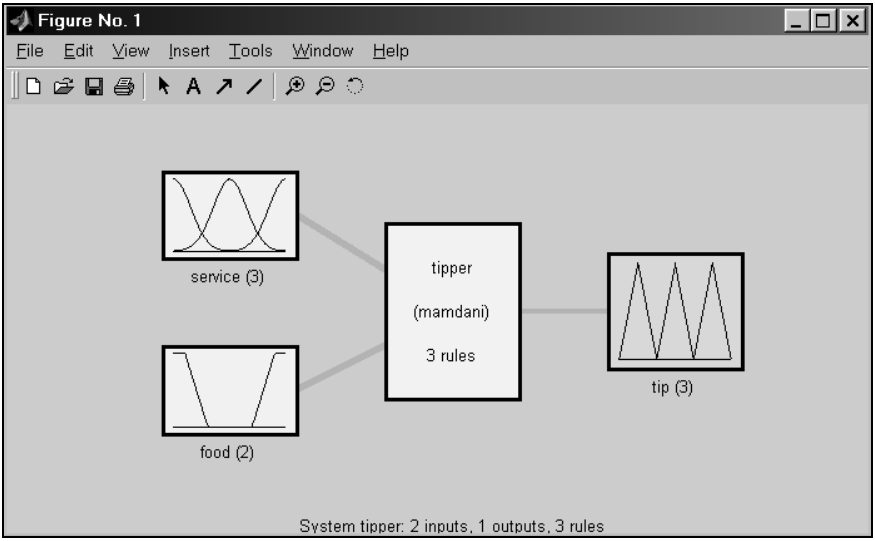


Рис. П3.18. Результат использования функции `plotfis` для изображения диаграммы системы нечеткого вывода FIS

plotmf

Назначение. Изображает графики всех функций принадлежности для заданной переменной.

Синтаксис. `plotmf(fismat, varType, varIndex)`

Описание. Эта функция изображает графики всех функций принадлежности для некоторой переменной из системы нечеткого вывода FIS с именем `fismat`, которое является первым аргументом этой функции. Вторым и третьим аргументами являются тип и номер соответствующей переменной: `varType` (`'input'` или `'output'`) и `varIndex`. Эта функция также может быть использована с функцией `subplot`.

Пример использования функции `plotmf`

```
a = readfis('tipper');
plotmf(a,'input',1)
```

Результат этого вызова функции `plotmf` изображен на рис. П3.19.

См. также: `evalmf`, `plotfis`.

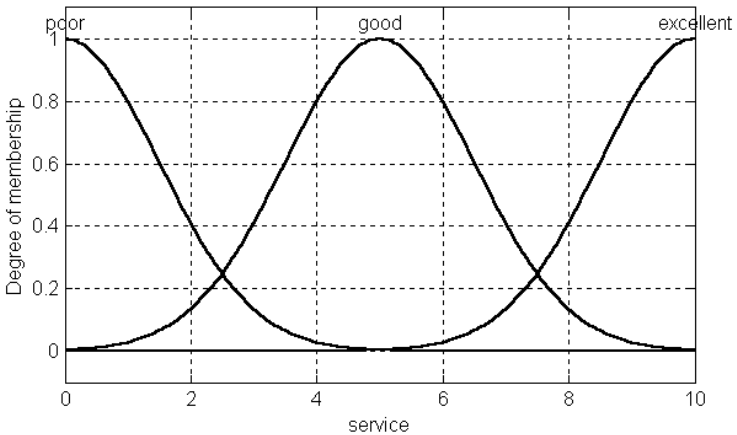


Рис. П3.19. Результат использования функции `plotmf` для изображения функций принадлежности первой из входных переменных системы нечеткого вывода FIS

psigmf

Назначение. Встроенная *П*-образная функция принадлежности, являющаяся произведением двух сигмоидальных функций.

Синтаксис. `y = psigmf(x, [a b c d])`

Описание. Эта функция принадлежности определяется как алгебраическое произведение двух сигмоидальных функций (2.20), каждая из которых задается как функция вида (2.14). Вторым аргументом этой функции является вектор параметров `[a b c d]`. При этом параметры `a` и `b` характеризуют первую сигмоидальную функцию, а параметры `c` и `d` — вторую сигмоидальную функцию.

Пример задания функции принадлежности `psigmf`

```
x=0:0.1:10;
y=psigmf(x,[2 4 -5 9]);
plot(x,y)
```

График этой функции принадлежности изображен на рис. 2.17, *б*.

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `sigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

readfis

Назначение. Загружает систему нечеткого вывода FIS из внешнего файла.

Синтаксис. `fismat = readfis('filename')`

Описание. Эта функция считывает систему нечеткого вывода из внешнего файла с именем `filename`, имеющего расширение fis, и загружает его данные в рабочую область. Если данная функция вызывается без аргумента, то открывается стандартное диалоговое окно открытия файла.

Пример использования функции `readfis`

```
fismat = readfis('tipper');
getfis(fismat)
```

В этом случае в окне команд будет выведена следующая информация:

```
Name =          tipper
Type =          mamdani
NumInputs =     2
InLabels =
        service
        food
NumOutputs =    1
OutLabels =
                tip
NumRules =      3
AndMethod =     min
OrMethod =      max
ImpMethod =     min
AggMethod =     max
DefuzzMethod = centroid
ans =
                tipper
```

См. также: `writefis`.

rmmf

Назначение. Удаляет функцию принадлежности из системы нечеткого вывода FIS.

Синтаксис. `fis = rmmf(fis, 'varType', varIndex, 'mf', mfIndex)`

Описание. Функция командной строки `fis = rmmf(fis, varType, varIndex, 'mf', mfIndex)` удаляет функцию принадлежности с номером `mfIndex` для типа переменной `varType` (`'input'` или `'output'`) и номером переменной `varIndex` из системы нечеткого вывода FIS, ассоциированной со структурой с именем `fis`. Структура `fis` должна находиться в рабочей области системы MATLAB. При этом требуется еще один аргумент `'mf'` — строка, указывающая тип удаляемой функции принадлежности.

Пример использования функции `rmmf`

```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
a = addmf(a,'input',1,'cold','trimf',[0 30 60]);
getfis(a,'input',1)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name =         temperature
NumMFs =       1
MFLabels =
       cold
Range =        [0 100]
```

Пример использования функции `rmmf` (продолжение)

```
b = rmmf(a,'input',1,'mf',1);
getfis(b,'input',1)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name =         temperature
NumMFs =       0
MFLabels =
Range =        [0 100]
```

См. также: `addmf`, `addrule`, `addvar`, `plotmf`, `rmvar`.

rmvar

Назначение. Удаляет переменные из системы нечеткого вывода FIS.

Синтаксис.

```
[fis2, errorStr] = rmvar(fis, 'varType', varIndex)
fis2 = rmvar(fis, 'varType', varIndex)
```

Описание. Функция командной строки `fis2 = rmvar(fis, 'varType', varIndex)` удаляет переменную типа `'varType'` (`'input'` или `'output'`) с номером `varIndex` из системы нечеткого вывода FIS, ассоциированной со структурой `fis`. Эта структура должна находиться в рабочей области системы MATLAB. При этом происходит автоматическое изменение списка правил с целью согласования с новым числом переменных.

До удаления переменной из системы нечеткого вывода необходимо удалить все правила, в которых используется удаляемая переменная. Нельзя удалить переменную, которая присутствует в списке правил. При наличии ошибки функция возвращает сообщение об ошибке в строке с именем `errorStr`.

Пример использования функции `rmvar`

```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
getfis(a)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name =          mysys
Type =          mamdani
NumInputs =     1
InLabels =
        temperature
NumOutputs =    0
OutLabels =
NumRules =      0
AndMethod =     min
OrMethod =      max
ImpMethod =             min
AggMethod =             max
DefuzzMethod =          centroid
ans =
        mysys
```

Пример использования функции `rmvar` (продолжение)

```
b = rmvar(a,'input',1);
getfis(b)
```

В этом случае в окно команд будет выведена следующая информация:

```
Name =         mysys
Type =         mamdani
NumInputs =    0
InLabels =
NumOutputs =   0
OutLabels =
NumRules =     0
AndMethod =    min
OrMethod =     max
ImpMethod =    min
AggMethod =    max
DefuzzMethod = centroid
ans =
       mysys
```

См. также: `addmf`, `addrule`, `addvar`, `rmmf`.

ruleedit

Назначение. Вызывает графический интерфейс редактора правил системы нечеткого вывода.

Синтаксис.
```
ruleedit('a')
ruleedit(a)
```

Описание. Эта функция, записанная в формате `ruleedit('a')`, вызывает редактор правил, который позволяет пользователю в графическом режиме анализировать и модифицировать правила продукций системы нечеткого вывода FIS, сохраненной во внешнем файле с именем a.fis. Эта функция позволяет также выполнять грамматический анализ правил, которые используются в некоторой системе нечеткого вывода FIS.

Чтобы использовать данный редактор для создания правил, необходимо предварительно определить все входные и выходные переменные, для чего можно воспользоваться редактором системы нечеткого вывода FIS. При этом задать правила можно с помощью выбора соответствующих значений термов входных и выходных переменных.

Функция в формате `ruleedit(a)` вызывает редактор правил для переменной рабочего пространства MATLAB, которая соответствует структуре FIS с именем `a`.

Редактор правил имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора правил содержит такие же операции, что и соответствующий пункт меню редактора FIS.

- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **FIS Properties...** — вызывает редактор FIS;
  - **Membership Functions...** — вызывает редактор функций принадлежности.
- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил;
  - **Surface** — вызывает программу просмотра поверхности вывода.
- Пункт меню **Options** (Сервис) содержит следующие операции:
  - **Language** — позволяет выбрать язык для записи правил в форме текста: **English** (английский), **Deutsch** (немецкий) или **Francais** (французский);
  - **Format** — позволяет выбрать формат записи правил системы нечеткого вывода: **Verbose** (в форме текста), **Symbolic** (в символической форме) или **Indexed** (в цифровой форме).

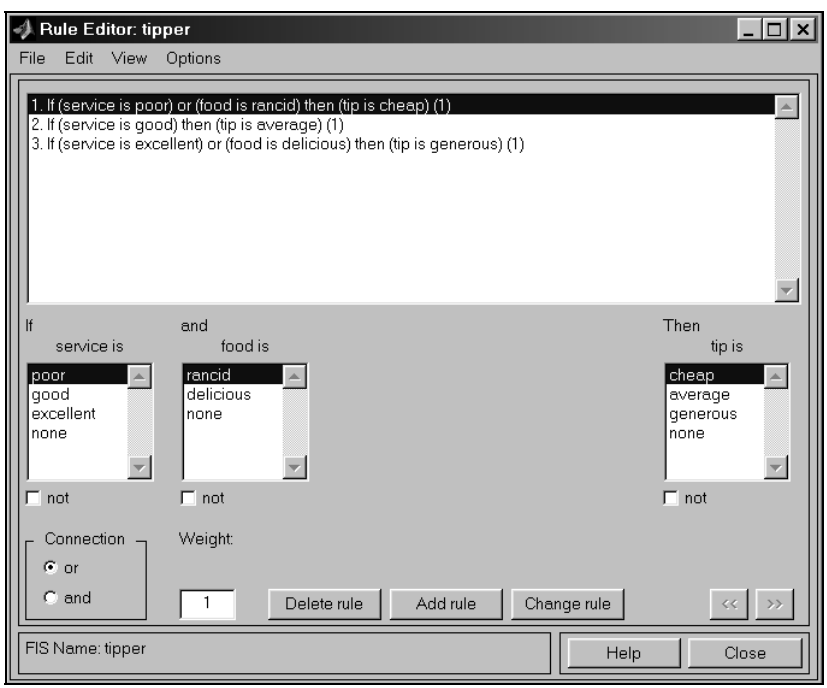


Рис. П3.20. Редактор правил нечетких продукций, вызываемый функцией `ruleedit('tipper')`

При записи правил в форме текста для создания законченных предложений используются служебные слова "if", "then", "AND", "OR" и т. д. При записи правил

в символической форме эти служебные слова заменяются символами соответствующих операций. Например, правило "if A AND B then C" преобразуется к виду: "A & B => C". Правила, записанные в цифровой форме, соответствуют формату их представления в структуре FIS. Более подробно особенности работы с редактором правил изложены в *главе 12*.

Пример использования функции `ruleedit`

```
ruleedit('tipper')
```

Результат выполнения функции `ruleedit('tipper')` изображен на рис. П3.20.

См. также: `addrule`, `fuzzy`, `mfedit`, `parsrule`, `ruleview`, `showrule`, `surfview`.

ruleview

Назначение. Вызывает графический интерфейс программы просмотра правил системы нечеткого вывода FIS.

Синтаксис.
```
ruleview('a')
ruleview(a)
```

Описание. Эта функция, записанная в формате `ruleview('a')`, вызывает программу просмотра правил, которая изображает диаграмму нечеткого вывода для структуры FIS, сохраненной во внешнем файле с именем a.fis. Функцию целесообразно использовать в том случае, когда необходимо визуально представить весь процесс нечеткого вывода от начала до конца. При этом пользователь имеет возможность оценить влияние каждого из правил на результат нечеткого вывода посредством изменения значений входных переменных.

Функция в формате `ruleview(a)` вызывает программу просмотра правил для переменной рабочего пространства MATLAB, соответствующей структуре FIS с именем `a`.

Программа просмотра правил имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора правил содержит такие же операции, что и соответствующий пункт меню редактора FIS.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **FIS Properties...** — вызывает редактор FIS;
  - **Membership Functions...** — вызывает редактор функций принадлежности;
  - **Rules** — вызывает программу редактирования правил.

- Пункт меню **View** (Вид) содержит следующие операции:
  - **Surface** — вызывает программу просмотра поверхности вывода.
- Пункт меню **Options** (Сервис) содержит следующие операции:
  - **Format** — позволяет выбрать формат записи правил системы нечеткого вывода: **Verbose** (в форме текста), **Symbolic** (в символической форме) или **Indexed** (в цифровой форме).

Если щелкнуть на номере правила в левой части диаграммы нечеткого вывода, то соответствующее правило появится в строке состояния в нижней части графического интерфейса программы просмотра правил в том формате, который выбран операцией **Options>Format**. Более подробно особенности работы с программой просмотра правил изложены в *главе 12*.

Пример использования функции `ruleview`

```
ruleview('tipper')
```

Результат выполнения функции `ruleview('tipper')` изображен на рис. П3.21.

См. также: `fuzzy`, `mfedit`, `ruleedit`, `surfview`.

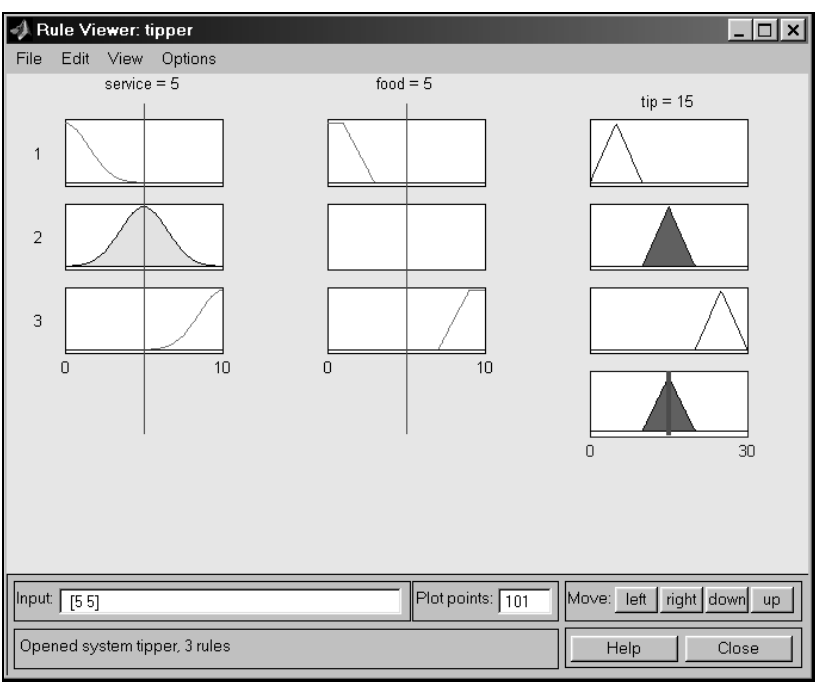


Рис. П3.21. Программа просмотра правил нечетких продукций, вызываемая функцией `ruleview('tipper')`

setfis

Назначение. Задает свойства системы нечеткого вывода FIS.

Синтаксис.

```
a = setfis(a, 'fispropname', 'newfisprop')
   a = setfis(a, 'vartype', varindex, 'varpropname',
                'newvarprop')
a = setfis(a, 'vartype', varindex, 'mf', mfindex, ...
'mfpropname', 'newmfprop')
```

Описание. Функция `setfis` может быть вызвана с тремя, пятью или семью аргументами в зависимости от того, необходимо ли задать структуру FIS в целом, только переменную структуры FIS или отдельную функцию принадлежности для одной из переменных структуры FIS. Аргументы функции имеют следующие значения:

- `a` — имя структуры FIS в рабочей области MATLAB;
- `'vartype'` — строка с указанием типа переменной (`input` или `output`);
- `varindex` — номер входной или выходной переменной;
- `'mf'` — строка, необходимая для вызова функции с пятью или семью аргументами, с указанием на то, что функция `setfis` задает функцию принадлежности для некоторой переменнаой;
- `mfindex` — номер функции принадлежности для выбранной переменной;
- `'fispropname'` — строка с указанием свойства поля структуры FIS, которая может принимать следующие значения: `name`, `type`, `andmethod`, `ormethod`, `impmethod`, `aggmethod`, `defuzzmethod`;
- `'newfisprop'` — строка с указанием имени свойства FIS или метода, который задается данной функцией;
- `'varpropname'` — строка с указанием имени поля структуры FIS, которая задается данной функцией (`name` или `range`);
- `'newvarprop'` — строка с указанием имени задаваемой переменной (для `name`) или диапазона изменения ее значений (для `range`);
- `'mfpropname'` — строка с указанием имени поля задаваемой функции принадлежности (`name`, `type` или `params`);
- `'newmfprop'` — строка с указанием имени или типа поля задаваемой функции принадлежности (для `name` или `type`) или вектор с указанием параметров задаваемой функции принадлежности (`params`).

Пример использования функции `setfis` с тремя аргументами

```
a = readfis('tipper');
a2 = setfis(a, 'name', 'eating');
getfis(a2, 'name')
```

В этом случае в окне команд будет выведена следующая информация:

```
out =
       eating
ans =
       eating
```

Пример использования функции `setfis` с пятью аргументами

```
a2 = setfis(a,'input',1,'name','help');
getfis(a2,'input',1,'name')
```

В этом случае в окне команд будет выведена следующая информация:

```
ans =
       help
```

Пример использования функции `setfis` с семью аргументами

```
a2 = setfis(a,'input',1,'mf',2,'name','wretched');
getfis(a2,'input',1,'mf',2,'name')
```

В этом случае в окне команд будет выведена следующая информация:

```
ans =
       wretched
```

См. также: `getfis`

sffis

Назначение. *S*-функция нечеткого вывода из пакета Simulink.

Синтаксис. `output = sffis(t, x, u, flag, fismat)`

Описание. Для выполнения расчетов, выполняемых обычно функцией `evalfis`, в системе Simulink используется собственный MEX-файл. Этот файл специально оптимизирован для работы в среде Simulink. Это означает, кроме всего прочего, что функция `sffis` создает некоторую структуру данных в рабочей области в ходе выполнения фазы инициализации моделирования в системе Simulink и продолжает ее использовать до завершения процесса моделирования.

Аргументы `t`, `x` и `flag` являются стандартными параметрами *S*-функции пакета Simulink. Аргумент `u` является входом в структуру системы нечеткого вывода FIS с именем `fismat` в рабочей области MATLAB. Например, если структура `fismat` имеет две входных переменных, то `u` представляет собой вектор с двумя компонентами.

См. также: `evalfis`, `fuzblock`.

showfis

Назначение. Отображает структуру указываемой системы нечеткого вывода FIS.

Синтаксис. `showfis(fismat)`

Описание. Функция `showfis(fismat)` выводит в окне команд системы MATLAB значения всех полей структуры `fismat`, которая определяет систему нечеткого вывода FIS и должна находиться в рабочей области.

Пример вызова функции `showfis`

```
a = readfis('tipper');
showfis(a)
```

В этом случае в окне команд будет выведена следующая информация:

```
1.  Name                tipper
2.  Type                mamdani
3.  Inputs/Outputs              [2 1]
4.  NumInputMFs         [3 2]
5.  NumOutputMFs        3
6.  NumRules            3
7.  AndMethod           min
8.  OrMethod            max
9.  ImpMethod           min
10. AggMethod           max
11. DefuzzMethod        centroid
12. InLabels            service
13.                                         food
14. OutLabels           tip
15. InRange                     [0 10]
16.                                          [0 10]
17. OutRange            [0 30]
18. InMFLabels          poor
19.                                         good
20.                                         excellent
21.                                         rancid
22.                                         delicious
23. OutMFLabels         cheap
24.                                         average
25.                                         generous
```

```
26. InMFTypes           gaussmf
27.                                         gaussmf
28.                                         gaussmf
29.                                         trapmf
30.                                         trapmf
31. OutMFTypes          trimf
32.                                          trimf
33.                                          trimf
34. InMFParams          [1.5 0 0 0]
35.                                         [1.5 5 0 0]
36.                                         [1.5 10 0 0]
37.                                         [0 0 1 3]
38.                                         [7 9 10 10]
39. OutMFParams         [0 5 10 0]
40.                                         [10 15 20 0]
41.                                         [20 25 30 0]
42. Rule Antecedent           [1 1]
43.                                          [2 0]
44.                                          [3 2]
42. Rule Consequent     1
43.                                        2
44.                                        3
42. Rule Weigth         1
43.                                        1
44.                                        1
42. Rule Connection           2
43.                                        1
44.                                        2
```

См. также: `getfis`.

showrule

Назначение. Отображает правила системы нечеткого вывода FIS.

Синтаксис.

```
showrule(fis)
showrule(fis, indexList)
showrule(fis, indexList, format)
showrule(fis, indexList, format, Lang)
```

Описание. Эта функция используется для визуализации правил системы нечеткого вывода FIS. Она может быть вызвана с различным числом аргументов: от 1 до 4.

- `fis` — обязательный аргумент, который должен соответствовать имени структуры системы нечеткого вывода FIS в рабочей области MATLAB;
- `indexList` — необязательный аргумент, который представляет собой вектор номеров правил, которые требуется отобразить в окне команд;
- `format` — необязательный аргумент, который представляет собой строку формата, в котором требуется отобразить правила. Эта строка может принимать одно из следующих значений: `'verbose'` (значение по умолчанию, которое соответствует отображению правил в форме текста на английском языке), `'symbolic'` (соответствует отображению правил в символической форме), `'indexed'` (соответствует отображению правил в цифровой форме, при этом указываются только номера правил);
- `Lang` — необязательный аргумент, который представляет собой строку с указанием языка отображения правил в окне команд в случае использования формата `verbose`. Эта строка может принимать одно из следующих значений: `'english'`, `'francais'`, или `'deutsch'`.

Пример использования функции `showrule` с двумя аргументами

```
a = readfis('tipper');
showrule(a,1)
```

В результате выполнения этой последовательности функций в окне команд появится следующий результат:

```
ans =
 1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
```

Пример использования функции `showrule` с тремя аргументами

```
showrule(a,[3 1],'symbolic')
```

В результате выполнения этой функции в окне команд появится следующий результат:

```
ans =
3. (service= =excellent) | (food= =delicious) => (tip=generous) (1)
1. (service= =poor) | (food= =rancid) => (tip=cheap) (1)
```

Пример использования функции `showrule` с тремя аргументами (продолжение)

```
showrule(a,1:3,'indexed')
```

В результате выполнения этой функции в окне команд появится следующий результат:

```
ans =
       1 1, 1 (1) : 2
       2 0, 2 (1) : 1
       3 2, 3 (1) : 2
```

См. также: `parsrule`, `ruleedit`, `addrule`.

sigmf

Назначение. Встроенная сигмоидальная функция принадлежности.

Синтаксис. `y = sigmf(x, [a b])`

Описание. Сигмоидальная функция принадлежности определяется выражением (2.14). Вторым аргументом этой функции является вектор параметров `[a b]`, которые принимают произвольные действительные значения и упорядочены отношением: $a<b$. При этом в случае $a>0$ может быть получена S-образная функция принадлежности, а в случае $a<0$ — Z-образная функция принадлежности.

Пример задания функции принадлежности `sigmf`

```
x=0:0.1:10;
y=sigmf(x,[3 6]);
plot(x,y)
```

График этой функции принадлежности изображен на рис. 2.14, *а*.

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `smf`, `trapmf`, `trimf`, `zmf`.

smf

Назначение. Встроенная S-образная функция принадлежности.

Синтаксис. `y = smf(x, [a b])`

Описание. Эта функция принадлежности является сплайн-функцией и определяется выражением (2.13). Вторым аргументом этой функции является вектор параметров `[a b]`, которые принимают произвольные действительные значения и упорядочены отношением: $a<b$. Эти параметры характеризуют наклонную часть соответствующей кривой.

Пример задания функции принадлежности `smf`

```
x=0:0.1:10;
y=smf(x,[3 6]);
plot(x,y)
```

График этой функции принадлежности изображен на рис. 2.13, *б*.

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `trapmf`, `trimf`, `zmf`.

subclust

Назначение. Находит центры кластеров методом субтрактивной кластеризации.

Синтаксис. `[C,S] = subclust(X, radii, xBounds, options)`

Описание. Эта функция предназначена для нахождения центров нечетких кластеров в множестве данных методом субтрактивной кластеризации. Метод субтрактивной кластеризации предполагает, что каждая точка данных является центром потенциального кластера и рассчитывает меру подобия каждой точки данных представлять центр кластера. При этом мера подобия основана на оценке плотности точек данных вокруг соответствующего центра кластера.

Данный алгоритм, который является обобщением метода кластеризации Р. Ягера (R. Yager), позволяет:

- выбрать точку данных с максимальным потенциалом для представления центра первого кластера;
- удалить все точки данных в окрестности центра первого кластера, величина которой задается параметром `radii`, чтобы определить следующий нечеткий кластер и координаты его центра.

Эти две процедуры повторяются до тех пор, пока все точки данных не окажутся внутри окрестностей радиуса `radii` некоторого центра кластера.

Матрица `X` содержит данные кластеризации, каждая строка которой соответствует координатам отдельной точки данных. Параметр `radii` представляет собой вектор, компоненты которого принимают значения из интервала [0, 1] и задают диапазон расчета центров кластеров по каждому из признаков измерений. При этом делается предположение, что все данные содержатся в некотором единичном гиперкубе. В общем случае небольшие значения параметра `radii` приводят к нахождению малого числа точек кластеров. Наилучшие результаты получаются при значениях `radii` между 0.2 и 0.5.

Например, для двух признаков измерения (матрица `X` содержит 2 столбца) значения параметра `radii = [0.5 0.25]` задает диапазон расчета по первому признаку равным 0.5 ширины пространства данных, а по второму признаку

равным 0.25 ширины пространства данных. Если параметр `radii` задан в форме скаляра, то это значение относится к каждому признаку, т. е. центр каждого из кластеров имеет сферическую окрестность указанного радиуса для расчета значений своих координат.

Аргумент `xBounds` представляет собой матрицу размерности $(2\times q)$, которая определяет способ отображения матрицы данных `X` в некотором единичном гиперкубе. Здесь q — количество рассматриваемых признаков в множестве данных. Этот аргумент является необязательным, если матрица `X` уже нормализована. Первая строка этой матрицы содержит минимальные значения интервала измерения каждого из признаков, а вторая строка — максимальные значения измерения каждого из признаков. Например, `xBounds = [-10 -5; 10 5]` определяет диапазон измерения первого признака в интервале [–10 +10], а диапазон измерения второго признака в интервале [–5 +5]. Если значения этой матрицы не указаны, то по умолчанию диапазон измерения признаков полагается равным интервалу от наименьшего значения до наибольшего значения по каждому из признаков в матрице данных.

Для изменения заданных по умолчанию параметров алгоритма кластеризации может быть использован дополнительный вектор `options`. Компоненты этого вектора могут принимать следующие значения:

- `options(1) = quashFactor` — параметр, используемый в качестве коэффициента для умножения значений `radii`, которые определяют окрестность центра кластера. Это осуществляется с целью уменьшения влияния потенциала граничных точек, рассматриваемых как часть нечеткого кластера (по умолчанию это значение равно 1.25);
- `options(2) = acceptRatio` — параметр, устанавливающий потенциал как часть потенциала центра первого кластера, выше которого другая точка данных может рассматриваться в качестве центра другого кластера (по умолчанию это значение равно 0.5);
- `options(3) = rejectRatio` — параметр, устанавливающий потенциал как часть потенциала центра первого кластера, ниже которого другая точка данных не может рассматриваться в качестве центра другого кластера (по умолчанию это значение равно 0.15);
- `options(4) = verbose` — если значение этого параметра не равно нулю, то на экран монитора выводится информация о выполнении процесса кластеризации (по умолчанию это значение равно 0).

Функция `subclust` возвращает матрицу `C` значений координат центров нечетких кластеров. При этом каждая строка этой матрицы содержит координаты одного центра кластера. Эта функция также возвращает вектор `S`, компоненты которого представляют σ-значения, которые определяют диапазон влияния центра кластера по каждому из рассматриваемых признаков. При этом все центры кластеров обладают одинаковым множеством σ-значений.

Пример вызова функции `subclust` с минимальным числом аргументов

```
[C,S] = subclust(X,0.5)
```

В этом случае для всех признаков множества данных задается один диапазон влияния 0.5.

Пример вызова функции `subclust` с дополнительными значениями аргументов

```
[C,S] = subclust(X,[0.5 0.25 0.3],[],[2.0 0.8 0.7])
```

В этом случае для множества данных используется 3 признака (матрица `X` имеет 3 столбца) с радиусами окрестностей 0.5, 0.25, 0.3 по первому, второму и третьему признакам соответственно. Коэффициенты шкалирования для отображения множества данных в единичный гиперкуб получаются на основе минимального и максимального значений данных. Аргумент `squashFactor`, равный 2.0, указывает на то, что необходимо определить кластеры, далеко расположенные друг от друга. Аргумент `acceptRatio` равный 0.8, указывает на то, что для нахождения центров кластеров будет нужен очень высокий потенциал. Наконец, аргумент `rejectRatio` равен 0.7, что указывает на необходимость исключения из рассмотрения всех точек данных, не обладающих высоким потенциалом.

См. также: `genfis2`.

surfview

Назначение. Вызывает графический интерфейс программы просмотра поверхности системы нечеткого вывода.

Синтаксис.

```
surfview('a')
surfview(a)
```

Описание. Эта функция, записанная в формате `surfview('a')`, вызывает программу просмотра поверхности, которая изображает поверхность нечеткого вывода для структуры FIS, сохраненной во внешнем файле с именем a.fis, для любой одной или двух из ее входных переменных. Эта программа не позволяет вносить изменения в систему нечеткого вывода и соответствующую ей структуру FIS. Используя главное меню программы, пользователь может выбрать входные переменные и соответствующие им горизонтальные оси системы координат (`X` и `Y`), а также выходную переменную, которой соответствует вертикальная ось системы координат (`Z`). Функция в формате `surfview(a)` вызывает программу просмотра поверхности вывода для переменной рабочего пространства MATLAB, соответствующей структуре FIS с именем `a`.

Щелкнув и задержав левую кнопку мыши, посредством последующего перемещения курсора мыши в том или ином направлении можно изменить угол просмотра поверхности вывода. Если рассматривается система нечеткого вывода с более чем двумя входными переменными, то для невизуализируемых входных переменных следует задать некоторые постоянные значения (константы).

Программа просмотра поверхности вывода имеет главное меню, которое позволяет пользователю вызывать другие графические средства работы с системой нечеткого вывода FIS, загружать и сохранять структуру FIS во внешних файлах и т. д.

- Пункт меню **File** (Файл) редактора правил содержит такие же операции, что и соответствующий пункт меню редактора FIS.
- Пункт меню **Edit** (Редактирование) содержит следующие операции:
  - **Undo** — отменяет выполнение последнего действия;
  - **FIS Properties...** — вызывает редактор FIS;
  - **Membership Functions...** — вызывает редактор функций принадлежности;
  - **Rules** — вызывает программу редактирования правил.
- Пункт меню **View** (Вид) содержит следующие операции:
  - **Rules** — вызывает программу просмотра правил.
- Пункт меню **Options** (Сервис) содержит следующие операции:
  - **Plot** — позволяет выбрать один из 8 стилей изображения графика поверхности вывода;
  - **Color Map** — позволяет выбрать одну из 4 цветовых схем изображения графика поверхности вывода;
  - **Always evaluate** — пометка галочкой этого пункта вложенного меню приводит к автоматическому формированию новой поверхности вывода всякий раз, когда вносятся изменения в систему нечеткого вывода, влияющие на форму графика поверхности вывода (такие, как изменение количества точек сетки графика). Это значение принято по умолчанию. Чтобы его отменить, необходимо снять галочку у этого пункта вложенного меню, щелкнув на этой позиции меню.

Более подробно особенности работы с программой просмотра поверхности вывода изложены в *главе 12*.

Пример использования функции `surfview`

```
surfview('tipper')
```

Результат выполнения функции `surfview('tipper')` изображен на рис. П3.22.

См. также: `anfisedit`, `fuzzy`, `gensurf`, `mfedit`, `ruleedit`, `ruleview`.

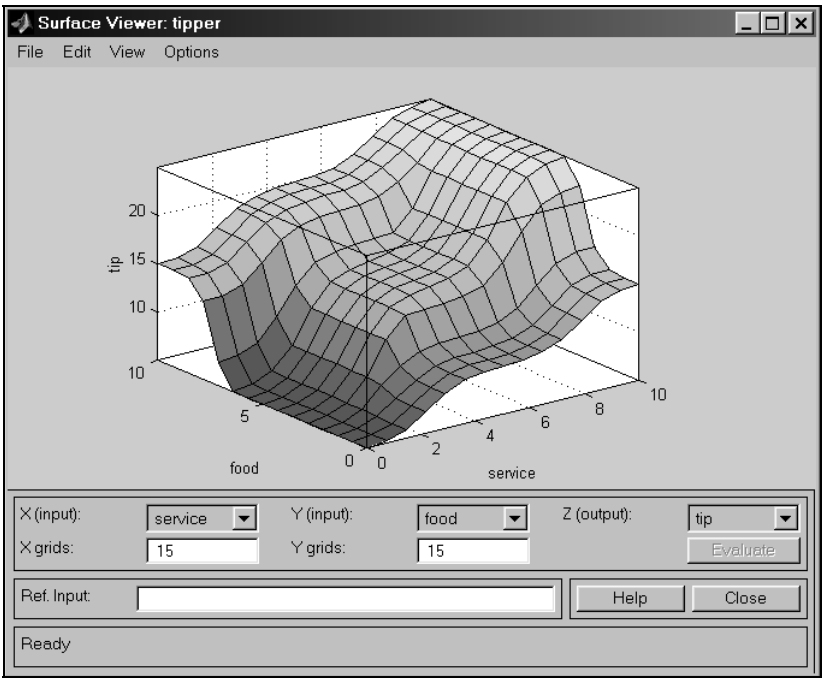


Рис. П3.22. Программа просмотра поверхности вывода, вызываемая функцией `surfview('tipper')`

trapmf

Назначение. Встроенная трапециевидная функция принадлежности.

Синтаксис. `y = trapmf(x, [a b c d])`

Описание. Эта функция используется для задания трапециевидных функций принадлежности вида (2.9). Функция `trapmf` использует четыре аргумента: a, b, c и d, в качестве некоторых числовых параметров, принимающих произвольные действительные значения и упорядоченные отношением: $a \le b \le c \le d$. При этом значения a и d задают носитель соответствующего нечеткого множества, а значения b и c — ядро этого нечеткого множества.

Пример задания трапециевидной функции принадлежности `trapmf`

```
x=0:0.1:10;
y=trapmf(x,[1 3 5 8]);
plot(x,y)
```

График полученной в этом случае функции принадлежности изображен на рис. 2.11, *б*.

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trimf`, `zmf`.

trimf

Назначение. Встроенная треугольная функция принадлежности.

Синтаксис.
```
y = trimf(x, [a b c])
```

Описание. Эта функция используется для задания треугольных функций принадлежности вида (2.8). Функция `trimf` использует три аргумента: a, b и c — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a \le b \le c$. При этом значения a и c определяют носитель соответствующего нечеткого множества, а значение b — модальное значение этого нечеткого множества.

Пример задания треугольной функции принадлежности `trimf`

```
x=0:0.1:10;
y=trimf(x,[3 4 7]);
plot(x,y)
```

График полученной в этом случае функции принадлежности изображен на рис. 2.11, *а*.

См. также: `dsigmf`, `gaussmf`, `gauss2mf`, `gbellmf`, `evalmf`, `mf2mf`, `pimf`, `psigmf`, `sigmf`, `smf`, `trapmf`.

writefis

Назначение. Сохраняет систему нечеткого вывода FIS во внешнем файле.

Синтаксис.
```
writefis(fismat)
writefis(fismat, 'filename')
writefis(fismat, 'filename', 'dialog')
```

Описание. Функция `writefis` сохраняет структуру системы нечеткого вывода FIS с именем `fismat`, находящуюся в рабочей области системы MATLAB, во внешнем файле с расширением fis. Вызов этой функции в формате `writefis(fismat)` открывает стандартное диалоговое окно сохранения файла. Использование функции в формате `writefis(fismat, 'filename')` записывает структуру системы нечеткого вывода FIS с именем `fismat`, находящуюся в рабочей области системы MATLAB, во внешнем файле с расширением fis и именем

filename. При этом диалоговое окно сохранения файла не открывается, а файл сохраняется в текущем каталоге на диске.

Вызов функции в формате writefis(fismat, 'filename', 'dialog') открывает диалоговое окно сохранения файла, при этом запись будет производиться в файл с именем filename.fis. В том случае, если расширение fis не содержится в имени файла, оно будет добавлено автоматически.

Пример использования функции writefis

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
```

См. также: readfis.

zmf

Назначение. *Z*-образная функция принадлежности.

Синтаксис. y = zmf(x, [a b])

Описание. Эта функция используется для задания *Z*-образных функций принадлежности вида сплайн-функции (2.11). Функция zmf использует два аргумента: *a* и *b* — некоторые числовые параметры, принимающие произвольные действительные значения и упорядоченные отношением: $a<b$.

Пример задания Z-образной функции принадлежности zmf

```
x=0:0.1:10;
y=zmf(x,[3 6]);
plot(x,y)
```

График полученной в этом случае функции принадлежности изображен на рис. 2.12, *а*.

См. также: dsigmf, gaussmf, gauss2mf, gbellmf, evalmf, mf2mf, pimf, psigmf, sigmf, smf, trapmf, trimf.

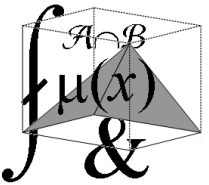
Пакет Fuzzy Logic Toolbox также содержит ряд демонстрационных примеров, которые могут быть использованы для самостоятельного изучения соответствующих нечетких моделей, их модификации и исследования. Эти демонстрационные примеры можно открыть как функцией demo, которая позволяет познако-

миться со всеми примерами, доступными в системе MATLAB, так и с помощью соответствующих функций в командном режиме. Поскольку некоторые из этих функций (например, `fcmdemo`) рассматриваются в тексте книги, а другие тесно интегрированы с пакетом расширения Simulink, здесь лишь указывается их назначение (табл. П3.1).

Таблица П3.1. *Функции вызова демонстрационных примеров*

Функция	Назначение
`defuzzdm`	Примеры методов дефаззификации
`fcmdemo`	Пример кластеризации точек на плоскости методом нечетких *c*-средних FCM
`fuzdemos`	Примеры графического интерфейса пользователя в пакете Fuzzy Logic Toolbox
`gasdemo`	Пример применения системы для эффективного управления топливом на основе метода субтрактивной кластеризации
`juggler`	Пример просмотра правил для задачи жонглирования мяча
`invkine`	Пример инверсной кинематики захвата механической руки робота
`irisfcm`	Пример кластеризации точек в четырехмерном пространстве методом нечетких *c*-средних FCM
`noisedm`	Пример адаптивного устранения шума
`slbb`	Пример управления движением мяча на качающейся балке с использованием средств пакета Simulink
`slcp`	Пример управления инвертированным маятником с использованием средств пакета Simulink
`sltank`	Пример управления уровнем воды с использованием средств пакета Simulink
`sltankrule`	Пример управления уровнем воды с использованием программы просмотра правил средствами пакета Simulink
`sltbu`	Пример нечеткого управления движением грузового автомобиля с использованием средств пакета Simulink

Приложение 4



Пример файла проекта для программы fuzzyTECH

В настоящем приложении приводится текст файла проекта системы нечеткого вывода в формате FTL среды fuzzyTECH для примера "*Чаевые в ресторане*". Как нетрудно заметить, запись системы нечеткого вывода в формате FTL, используемом для спецификации всех параметров нечеткого проекта в среде fuzzyTECH, очень похожа на спецификацию системы нечеткого вывода на языке нечеткого управления FCL *(см. главу 8)*.

Текст файла TIPPER.FTL для нечеткой модели "Чаевые в ресторане"

```
PROJECT {
  NAME        = TIPPER.FTL;
  TITLE       = TIPPER;
  AUTHOR      = A. Leonenkoff;
  DATEFORMAT  = YYYY.DD.MM;
  LASTCHANGE  = 2003.19.01;
  CREATED     = 2003.06.01;
  SHELLOPTIONS {
    ONLINE_REFRESHTIME   = 55;
    ONLINE_TIMEOUTCOUNT = 1100;
    ONLINE_CODE          = ON;
    ONLINE_TRACE_BUFFER = (ON, PAR(1024));
    COMMENTS             = ON;
    FTL_BUFFER           = (OFF, PAR(1));
    PASSWORD             = OFF;
    PUBLIC_IO            = ON;
    FAST_CMBF            = OFF;
    FAST_COA             = ON;
```

```
  BTYPE               = DOUBLE;
  C_TYPE              = ANSI;
} /* SHELLOPTIONS */
MODEL {
  VARIABLE_SECTION {
    LVAR {
      NAME       = Service;
      BASEVAR    = degree;
      LVRANGE    = MIN(0.000000), MAX(10.000000),
                   MINDEF(0), MAXDEF(65535),
                   DEFAULT_OUTPUT(0.0);
      RESOLUTION = XGRID(1.0), YGRID(1.0),
                   SHOWGRID (ON), SNAPTOGRID(ON);
      COLOR      = RED (255), GREEN (0), BLUE (0);
      INPUT      = CMBF;
      POS        = -225, 30;
      TERM {
        TERMNAME = poor;
        POINTS   = (0.0, 1.0),
                   (5.0, 0.0),
                   (10.0, 0.0);
        SHAPE = (S_SHAPE, PAR (0.800000));
        COLOR    = RED (255), GREEN (0), BLUE (0);
      }
      TERM {
        TERMNAME = good;
        POINTS   = (0.0, 0.0),
                   (5.0, 1.0),
                   (10.0, 0.0);
       SHAPE = (S_SHAPE, PAR (0.800000));
        COLOR    = RED (0), GREEN (0), BLUE (255);
      }
      TERM {
        TERMNAME = excellent;
        POINTS   = (0.0, 0.0),
                   (5.0, 0.0),
                   (10.0, 1.0);
        SHAPE = (S_SHAPE, PAR (0.800000));
```

```
    COLOR    = RED (0), GREEN (255), BLUE (0);
  }
 }  /* LVAR */
LVAR {
  NAME       = Food;
  BASEVAR    = degree;
  LVRANGE    = MIN(0.00), MAX(10.00),
               MINDEF(0), MAXDEF(65535),
               DEFAULT_OUTPUT(0.0);
  RESOLUTION = XGRID(0.2), YGRID(1.0),
               SHOWGRID (ON), SNAPTOGRID(ON);
  COLOR      = RED (0), GREEN (255), BLUE (0);
  INPUT      = CMBF;
  POS        = -225, -35;
  TERM {
    TERMNAME = rancid;
    POINTS   = (0.0, 1.0),
               (1.0, 1.0),
               (3.0, 0.0),
               (30.0, 0.0);
    SHAPE    = LINEAR;
    COLOR    = RED (255), GREEN (0), BLUE (0);
  }
  TERM {
    TERMNAME = delicious;
    POINTS   = (0.0, 0.0),
               (7.0, 0.0),
               (9.0, 1.0),
               (10.0, 1.0);
    SHAPE    = LINEAR;
    COLOR    = RED (0), GREEN (0), BLUE (255);
  }
}  /* LVAR */
LVAR {
  NAME       = Tip;
  BASEVAR    = percentage;
  LVRANGE    = MIN(0.000000), MAX(30.000000),
               MINDEF(0), MAXDEF(65535),
               DEFAULT_OUTPUT(0.0);
```

```
    RESOLUTION = XGRID(0.25), YGRID(1.0),
                 SHOWGRID (ON), SNAPTOGRID(ON);
    COLOR      = RED (0), GREEN (0), BLUE (255);
    OUTPUT     = COM;
    POS        = 75, 0;
    TERM {
      TERMNAME = cheap;
      POINTS   = (0.0, 0.0),
                 (5.0, 1.0),
                 (10.0, 0.0),
                 (30.0, 0.0);
      SHAPE    = LINEAR;
      COLOR    = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = average;
      POINTS   = (0.0, 0.0),
                 (10.0, 0.0),
                 (15.0, 1.0),
                 (20.0, 0.0),
                 (30.0, 0.0);
      SHAPE    = LINEAR;
      COLOR    = RED (0), GREEN (0), BLUE (255);
    }
    TERM {
      TERMNAME = generous;
      POINTS   = (0.0, 0.0),
                 (20.0, 0.0),
                 (25.0, 1.0),
                 (30.0, 0.0);
      SHAPE    = LINEAR;
      COLOR    = RED (0), GREEN (255), BLUE (0);
    }
  }  /* LVAR */
}  /* VARIABLE_SECTION */

OBJECT_SECTION {
  REMARK {
    TEXT = Чаевые в ресторане;
```

```
  POS = -175, -135;
  FONTSPEC = 36, 0, 0, 0, 0, 16, 0;
  FONTNAME =Times New Roman Cyr;
  COLOR = RED (161), GREEN (0), BLUE (160);
}
RULEBLOCK {
  NAME        = RB1;
  INPUT       = Service, Food;
  OUTPUT      = Tip;
  AGGREGATION = (MIN_MAX, PAR (0.0));
  RESULT_AGGR = MAX;
  POS         = -80, -35;
  RULES {
    IF    Service = poor
      AND Food = rancid
    THEN  Tip = cheap   WITH 1.000;
    IF    Service = good
    THEN  Tip = average   WITH 1.000;
    IF    Service = excellent
      AND Food = delicious
    THEN  Tip = generous  WITH 1.000;
  }  /* RULES */
}  /* RULEBLOCK */
REMARK {
  TEXT = Блок Правил;
  POS = -40, -70;
  FONTSPEC = 18, 0, 0, 0, 0, 16, 0;
  FONTNAME =Times New Roman Cyr;
  COLOR = RED (0), GREEN (112), BLUE (80);
}
REMARK {
  TEXT = Входные Переменные;
  POS = -205, -70;
  FONTSPEC = 18, 0, 0, 0, 0, 16, 0;
  FONTNAME =Times New Roman Cyr;
  COLOR = RED (0), GREEN (80), BLUE (80);
}
REMARK {
  TEXT = Выходные Переменные;
```

```
        POS = 90, -70;
        FONTSPEC = 18, 0, 0, 0, 0, 16, 0;
        FONTNAME =Times New Roman Cyr;
        COLOR = RED (0), GREEN (80), BLUE (80);
      }
    }  /* OBJECT_SECTION */
  }  /* MODEL */
}  /* PROJECT */
ONLINE {
  TIMESTAMP = 20010119143151UT;
}  /* ONLINE */

NEUROFUZZY {
  LEARNRULE     =RandomMethod;
  STEPWIDTHDOS  = 0.100000;
  STEPWIDTHTERM = 1.000000;
  MAXDEVIATION  = (50.000000, 1.000000, 0.750000);
  AVGDEVIATION  = 0.100000;
  MAXSTEPS      = 100;
  NEURONS       = 1;
  DATASEQUENCE  = RANDOM;
  UPDATEDBGWIN  = OFF;
}  /* NEUROFUZZY */
```

Прокомментируем отдельные фрагменты текста приведенного файла нечеткого проекта. Описание проекта начинается после ключевого слова `PROJECT` и заключается внутри соответствующих фигурных скобок. Система нечеткого вывода описывается после ключевого слова `MODEL` и содержит секцию лингвистических переменных `VARIABLE_SECTION` и секцию объектов `OBJECT_SECTION`.

Описание каждой лингвистической переменной заключается в блок `LVAR`, в котором задается имя переменной после ключевого слова `NAME`, единицы измерения ее значений после слова `BASEVAR` и диапазон возможных значений после слова `LVRANGE`. Для входных лингвистических переменных после ключевого слова `INPUT` указывается метод фаззификации значений этой переменной, а для выходных лингвистических переменных после ключевого слова `OUTPUT` указывается метод дефаззификации значений этой переменной.

Описание каждого терма лингвистических переменных заключается в блок `TERM`, в котором задается имя терма после ключевого слова `TERMNAME`, перечисляются пары значений терма и его функции принадлежности в некоторых характеристических точках, а также указывается форма ее функции принадлежности после ключевого слова `SHAPE`.

Секция объектов `OBJECT_SECTION` содержит описания блоков текста `REMARK` и описание блока правил `RULEBLOCK`. Заметим, что в блоках текста можно использовать символы кириллицы при установке имени шрифта: `FONTNAME =Times New Roman (Cyr)`. Ключевое слово `POS` используется для задания координат отображения блока текста или других объектов, а после ключевого слова `COLOR` можно определить цвет в трехцветной модели RGB (красный/зеленый/синий) изображения текста в окне редактора проекта.

В секции блока правил `RULEBLOCK` после ключевого слова `RULES` специфицируются все правила соответствующего блока в виде структурируемого текста: "IF <*Имя терма* = *Значение терма*> AND <*Имя терма* = *Значение терма*> THEN <*Имя терма* = *Значение терма*> WITH <*Значение DoS*>". В блоке правил должны быть определены все необходимые методы для агрегирования и композиции результатов нечетких выводов, аналогично рассмотренным в языке FCL.

Блоки `ONLINE` и `NEUROFUZZY` предназначены для использования дополнительных модулей программы fuzzyTECH, и их смысл в данной модели не имеет принципиального значения. После загрузки данного файла в редакторе проекта будет изображена структура соответствующей системы нечеткого вывода, которая обладает полной функциональностью в контексте ее дальнейшего редактирования и отладки.

Глоссарий

Настоящий глоссарий поясняет содержательный смысл основных понятий теории нечетких множеств, нечеткой логики и нечеткого управления, которые используются для построения нечетких моделей систем. Некоторые из терминов имеют статус рекомендованного стандарта IEC 1131-7, другие традиционно применяются в нечеткой математике. Все термины расположены в алфавитном порядке, в скобках приводится англоязычный эквивалент каждого из терминов.

Агрегирование (aggregation) — этап нечеткого вывода, представляющий собой процедуру определения степени истинности условия каждого правила системы нечеткого вывода.

Аккумуляция (accumulation) — этап нечеткого вывода, представляющий собой процедуру или процесс нахождения функции принадлежности каждой из выходных лингвистических переменных множества правил нечетких продукций. Синоним — композиция.

Активизация (activation) — этап нечеткого вывода, представляющий собой процедуру или процесс нахождения степени истинности каждого из подзаключений правил нечетких продукций.

Антецедент — *см.* Условие.

База правил (rule base) — совокупность или множество правил нечетких продукций, согласованных относительно использования входных и выходных лингвистических переменных.

Весовой коэффициент (weighting factor, DoS) — количественное значение из интервала [0, 1], которое определяет степень влияния условия правила нечеткой продукции на степень истинности заключения этого правила или степень уверенности в достоверности получаемого результата.

Входная переменная (input variable) — лингвистическая переменная, которая используется в формировании условий правил нечетких продукций.

Вывод, нечеткий вывод (inference) — применение лингвистических правил к значениям входных переменных с целью получения значений выходных переменных.

Выходная переменная (output variable) — лингвистическая переменная, которая используется в формировании заключений правил нечетких продукций.

Дефаззификация (defuzzification) — в общем случае преобразование некоторого нечеткого множества в четкое множество. В контексте нечеткого управления имеет более узкий смысл, означающий получение для заданной функции принадлежности выходной лингвистической переменной единственного количественного значения. Синоним — приведение к четкости.

Заключение (conclusion) — в общем случае правая часть правила продукции. В контексте правил нечеткой продукции означает нечеткое лингвистическое высказывание, следующее после связки THEN (ТО). Синонимы — консеквент, следствие.

Композиция — *см.* Аккумуляция.

Консеквент — *см.* Заключение.

Лингвистическая переменная (linguistic variable) — переменная, которая принимает свои значения из множества лингвистических термов.

Лингвистический терм (linguistic term) — некоторое фиксированное значение, которое может принимать лингвистическая переменная. Как правило, является нечетким множеством с соответствующей функцией принадлежности.

Лингвистическое правило (linguistic rule) — правило продукции вида ЕСЛИ-ТО (IF-THEN) с условием и заключением, которые являются нечеткими лингвистическими высказываниями в форме лингвистических переменных. Синоним — правило нечеткой продукции.

Множество обычное, не нечеткое множество (crisp set) — специальный случай нечеткого множества, функция принадлежности которого для каждого из элементов принимает значение 0 или 1. Это понятие соответствует классическому определению множества в смысле Г. Кантора.

Нечеткая логика (fuzzy logic) — одна из разновидностей неклассических логик, в которой допускается непрерывное множество значений истинности высказываний и применяются специальные логические операции или связки.

Нечеткая операция (fuzzy operator) — операция, используемая в теории нечетких множеств или в нечеткой логике. Синоним — нечеткий оператор.

Нечеткая сеть Петри (fuzzy Petri net) — разновидность сетей Петри, в которых правила функционирования определяются на основе концепции теории нечетких множеств.

Нечеткий вывод (fuzzy inference) — процедура установления истинности заключения на основании известных значений истинности условий.

Нечеткое множество (fuzzy set) — совокупность элементов произвольной природы, относительно которых нельзя с полной определенностью утверждать — принадлежит ли тот или иной элемент данной совокупности или нет. Математически нечеткое множество определяется как множество упорядоченных пар вида: $<x, \mu(x)>$, где x является элементом универсума $\boldsymbol{X}$, а $\mu(x)$ — функция принадлежности, которая ставит в соответствие каждому из элементов $x \in \boldsymbol{X}$ некоторое действительное число из интервала [0,1].

Нечеткое управление (fuzzy control) — специальный раздел теории управления, при котором алгоритмы управления основываются на концепции теории нечетких множеств и нечеткой логики.

Обратная цепочка рассуждений (fuzzy backward-chaining reasoning) — метод установления истинности заключений правил, который основан на последовательной активизации правил нечетких продукций в порядке проверки предварительно заданного заключения.

Одноэлементное множество (singleton) — специальный случай нечеткого множества, состоящего только из одного элемента, значение функции принадлежности которого равно 1. Для всех остальных элементов универсума функция принадлежности этого множества равна 0.

Программируемый логический контроллер (programmable logic controller) — электронное устройство, содержащее в своем составе программно-аппаратные компоненты, такие как один или несколько микропроцессоров, модули памяти, порты ввода/вывода, и предназначенное для сбора данных о состоянии технологического процесса, а также для автоматического управления им.

Прямая цепочка рассуждений (fuzzy forward-chaining reasoning) — метод установления истинности заключений правил, который основан на последовательной активизации правил нечетких продукций с целью проверки всех условий правил с наибольшей степенью истинности.

Степень принадлежности (degree of membership) — отдельное значение функции принадлежности для фиксированного элемента нечеткого множества.

Условие (condition) — в общем случае левая часть правила продукции. В системах нечеткого вывода — совокупность лингвистических термов, записанных между ключевыми словами ЕСЛИ и ТО (IF-THEN) нечеткого правила продукции. Синоним — антецедент.

Фаззификация (fuzzification) — преобразование некоторого четкого множества в нечеткое множество. Применительно к нечеткому управлению имеет более узкий смысл, означающий получение для конкретного значения входной переменной значений функций принадлежности, определенных для лингвистических термов данной входной переменной. Синоним — приведение к нечеткости.

Функция принадлежности (membership function) — математическая функция, определяющая степень или уверенность, с которой элементы некоторого множества принадлежат заданному нечеткому множеству. Данная функция ставит в соответствие каждому элементу нечеткого множества действительное число из интервала [0, 1]. Задать конкретное нечеткое множество означает определить соответствующую ему функцию принадлежности.

Ниже приводится список используемых в книге сокращений с кратким пояснением их смысла.

ANFIS (Adaptive Neuro-Fuzzy Inference System) — адаптивная система нейро-нечеткого вывода.

CF (certainty factor) — фактор уверенности или весовой коэффициент правила нечетких продукций.

CoA, COA (Centre of Area) — метод дефаззификации по формуле центра площади.

CoG, COG (Centre of Gravity) — метод дефаззификации по формуле центра тяжести.

CoGS, COGS (Centre of Gravity for Singletons) — метод дефаззификации по формуле центра тяжести для одноточечных множеств.

DoS (Degree of Support) — весовой коэффициент правила нечеткой продукции в среде fuzzyTECH.

DDE (Dynamic Data Exchange) — технология, которая позволяет нескольким программам обмениваться данными в среде MS Windows.

FB (Function Block) — функциональный блок в языке FCL.

FBD (Function Block Diagram) — диаграмма функционального блока на графическом языке управления.

FC (Fuzzy Control) — нечеткое управление.

FCL (Fuzzy Control Language) — язык нечеткого управления.

FDCL (Fuzzy Dependency and Command Language) — язык нечетких зависимостей и команд.

FIS (Fuzzy Inference System) — система нечеткого вывода в среде MATLAB.

FL (Fuzzy Logic) — нечеткая логика.

FMP (fuzzy modus ponens) — нечеткий модус поненс.

FMT (fuzzy modus tollens) — нечеткий модус толленс.

FTL (Fuzzy Technology Language) — специальный текстовый формат для сохранения файла нечеткого проекта в среде fuzzyTECH.

FTR (fuzzyTECH Runtime) — специальный бинарный формат для сохранения файла нечеткого проекта в среде fuzzyTECH.

GAMMA — коэффициент компенсации для агрегирования в среде fuzzyTECH.

IEC (International Electrotechnical Commission) — Международная электротехническая комиссия, комиссия по электротехнике.

IL (Instruction List) — список инструкций.

ISO (International Standardization Organisation) — Международная организация по стандартизации.

LM (Left Most Maximum) — метод дефаззификации по формуле левого модального значения.

LV (Linguistic Variable) — лингвистическая переменная.

MAX (Maximum operator) — операция максимизации произвольного количества аргументов.

MBF/MF (Membership Function) — функции принадлежности.

MFLIPS (Million Fuzzy Logic Inferences per Second) — миллион нечетких логических выводов в секунду.

MIN (Minimum operator) — операция минимизации произвольного количества аргументов.

PC/PLC (Programmable/Logic Controller) — программируемый/логический контроллер.

PROD (Product operator) — операция произведения произвольного количества аргументов.

RB (Rule Block) — блок правил нечетких продукций.

RM (Right Most Maximum) — метод дефаззификации по формуле правого модального значения.

SOFCL (Self-Organising Fuzzy Logic Controllers) — самоорганизующиеся контроллеры с нечеткой логикой.

ST (Structured Text) — структурируемый текст.

Литература

1. Ансофф И. Новая корпоративная стратегия. — СПб: Питер, 1999. — 416 с.
2. Дьяконов В. MATLAB: учебный курс. — СПб: Питер, 2001. — 560 с.
3. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB. Специальный справочник. — СПб: Питер, 2001. — 480 с.
4. Дюбуа Д., Прад А. Теория возможностей. Приложение к представлению знаний в информатике. — М.: Радио и связь, 1990. — 288 с.
5. Городецкий В. И. Прикладная алгебра и дискретная математика. Часть 3. Формальные системы логического типа. — МО СССР, 1987. — 177 с.
6. Гультяев А. Визуальное моделирование в среде MATHLAB: учебный курс. — СПб: Питер, 2000. — 432 с.
7. Искусственный интеллект. — Кн. 2. Модели и методы: Справочник / Под ред. Д. А.Поспелова. — М.: Радио и связь, 1990. — 304 с.
8. Карлоф Б. Деловая стратегия. — М.: Экономика, 1991. — 239 с.
9. Классификация и кластер /Под. ред. Дж. Вэн Райзина. — М.: Мир, 1980. — 392 с.
10. Клини С. Математическая логика. — М.: Мир, 1973. — 480 с.
11. Ковальски Р. Логика в решении проблем. — М.: Наука, 1990. — 280 с.
12. Котов В. Е. Сети Петри. — М.: Наука, 1984. — 432 с.
13. Кофман А. Введение в теорию нечетких множеств. — М.: Радио и связь, 1982. — 432 с.
14. Круглов В. В., Борисов Н. Н. Искусственные нейронные сети. Теория и практика. — М.: Горячая линия — Телеком, 2001. — 382 с.
15. Кузьмин В. Б. Построение групповых решений в пространствах четких и нечетких бинарных отношений. — М.: Наука, 1982. — 168 с.
16. Куратовский К., Мостовский А. Теория множеств. — М.: Мир, 1970. — 416 с.
17. Леоненков А. В. Самоучитель UML. — БХВ-Петербург, 2001. — 304 с.
18. Леоненков А. В. Алгоритм нечеткого кластерного анализа в задачах структуризации сложных систем. — Сборник алгоритмов и программ типовых задач. Вып. 11. — МО РФ, 1993, с. 74—79.
19. Лескин А. А., Мальцев П. А., Спиридонов А. М. Сети Петри в моделировании и управлении. — Л.: Наука, 1989. — 136 с.
20. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. — М.: Мир, 1990. — 432 с.
21. Мандель Н. Д. Кластерный анализ. — М.: Финансы и статистика, 1988. — 176 с.

22. Мелихов А. Н., Бернштейн Л. С., Коровин С. Я. Ситуационные советующие системы с нечеткой логикой. — М.: Наука, 1990. — 272 с.

23. Методы анализа данных / Под. ред. Э. Дидэ и др. — М.: Финансы и статистика, 1985. — 360 с.

24. Нечеткие множества в моделях управления и искусственного интеллекта/ Под. ред. Д. А. Поспелова. — М.: Наука, 1986. — 312 с.

25. Нечеткие множества и теория возможностей / Под. ред. Р. Р. Ягера. — М.: Радио и связь, 1986. — 408 с.

26. Новиков П. С. Элементы математической логики. — М.: Наука,1973. — 400 с.

27. Борисов А. Н. и др. Обработка нечеткой информации в системах принятия решений. — М.: Радио и связь, 1989. — 304 с.

28. Орловский С. А. Проблемы принятия решений при нечеткой исходной информации. — М.: Наука, 1981. — 208 с.

29. Питерсон Дж. Теория сетей Петри и моделирование систем. — М.: Мир, 1984. — 230 с.

30. Прикладные нечеткие системы / Под ред. Т. Тэрано, К. Асаи, М. Сугено. — М.: Мир, 1993. — 368 с.

31. Резников Б. А. Системный анализ и методы системотехники (Методология системных исследований). Часть 1. — МО СССР, 1990. — 522 с.

32. Сети Петри и их реализация на ЭВМ. Вып. 1 / Леоненков А. В. — МО СССР, 1990. — 61 с.

33. Сети Петри и их реализация на ЭВМ. Вып. 2. Временные сети Петри / Леоненков А. В. — СПб., ВИКА, 1996. — 87 с.

34. Сети Петри и их реализация на ЭВМ. Вып. 3. Стохастические сети Петри/ Леоненков А. В. — МО РФ, 1995. — 63 с.

35. Arnould T., Tano S. Interval-valued fuzzy backward reasoning. — IEEE Transactions on Fuzzy Systems, vol. 3, no. 4, 1995, pp. 425—437.

36. Bezdek J. C. A convergence theorem for the fuzzy ISODATA clustering algorithm. — IEEE Transactions on Pattern Analysis and Mashine Intelligence, vol. 2, no. 1, 1980, pp. 1—8.

37. Bezdek J.C. Some recent applications of fuzzy c-means in pattern recognition and image processing. — IEEE Workshop Lang. Autom, 1983, pp. 247—252.

38. Chen S. M., Ke J. S., Chang J. F. Knowledge representation using fuzzy Petri nets. — IEEE Transactions on Knowledge and Data Engineering, vol. 2, no. 3, 1990, pp. 311—319.

39. Dunn J. C. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. — Journal on Cybernetics, vol. 3, no. 3, 1974, pp. 32—37

40. Fukami S., Mizumoto M., Tanaka, K. Some considerations of fuzzy conditional inference. — Fuzzy Sets and Systems, vol. 4, 1980, pp. 243—273.

41. Kiszka J. B., Kochanska M. E., Sliwinska D. S. The influence of some fuzzy implication operators on the accuracy of a fuzzy model. — Fuzzy Sets and Systems, vol. 15, 1985, pp. 111 — 128; 223 — 240.

42. Konar A., Mandal A. K. Uncertainty management in expert systems using fuzzy Petri nets. — IEEE Transactions on Knowledge and Data Engneering, vol. 8, no. 1, 1996, pp. 96—104.

43. Looney C. G. Fuzzy Petri nets for rule-based decision making. — IEEE Transactions on Systems, Man, and Cybernetics, v. 18, no. 1, 1988, pp. 178—183.

44. Mamdani E. H. Application of fuzzy logic to approximate reasoning using linguistic synthesis. — IEEE Transactions on Computers, vol. 26, no. 12, 1977, pp. 1182—1191.

45. Mamdani E. H., Assilian S. An experiment in linguistic yn thesis with a fuzzy logic controller. — International Journal of Man-Machine Studies, vol. 7, no. 1, 1975, pp. 1—13.

46. Mamdani E. H. Advances in the linguistic synthesis of fuzzy □ontrollers. — International Journal of Man-Machine Studies, vol. 8, 1976. pp. 669—678.

47. Mamdani E. H. Applications of fuzzy logic to approximate reasoning using linguistic synthesis. — IEEE Transactions on Computers, vol. 26, no. 12, 1977, pp. 1182—1191.

48. Murata T. Petri nets: properties, analysis and applications. — Proceedings of the IEEE, vol. 77, no. 4, 1989, pp. 541—580.

49. Pedrycz W., Gomide F. A generalized fuzzy Petri net model. — IEEE Transactions on Fuzzy Systems, vol. 2, no. 4, 1994, pp. 295—301.

50. Ross T. J. Fuzzy logic with engineering applications. — McGraw-Hill, 1995. — 600 p.

51. Sugeno M. Fuzzy measures and fuzzy integrals: a survey. (in M. M. Gupta, G. N. Saridis, and B. R. Gaines, editors) Fuzzy Automata and Decision Processes, pp. 89—102, North-Holland, New York, 1977.

52. Takagi T., Sugeno M. Fuzzy identification of systems and its applications to modeling and control. — IEEE Transactions on Systems, Man, and Cybernetics, vol. 15, no. 1, 1985, pp. 116—132.

53. Windham M. P. Cluster validity for the fuzzy c-means clustering algorithm. — IEEE Transactions on Pattern Analysis and Mashine Intelligence, vol. 4, no. 4, 1982, pp. 357—363.

54. Windham M. P. Cluster validity for the fuzzy c-means clustering algorithm. — Fuzzy Sets and Systems, vol. 10, no. 3, 1983, pp. 271—279.

55. Zadeh L. A. Fuzzy sets. — Information and Control, vol. 8, 1965, pp. 338—353.

56. Zadeh L. A. The concept of a linguistic variable and its application to approximate reasoning. — Information Sciences, vol. 8, 1975, pp. 43—80.

57. Zadeh L. A. Fuzzy logic. — IEEE Transactions on Computers, vol. 21, no. 4, 1988, pp. 83—93.