

А.В.Андрейчиков, О.Н.Андрейчикова

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Рекомендовано
Министерством образования Российской Федерации
в качестве учебника для студентов
высших учебных заведений,
обучающихся по специальности
“Прикладная информатика в экономике”



МОСКВА
“ФИНАНСЫ И СТАТИСТИКА”
2004

РЕЦЕНЗЕНТЫ:

Кафедра информационных систем в экономике
Тверского государственного технического университета;
А. В. Смирнов,
доктор технических наук, профессор
заместитель директора по научной работе
Санкт-Петербургского института информатики и автоматизации РАН

Андрейчиков А. В., Андрейчикова О. Н.
А65 **Интеллектуальные информационные системы: Учебник. —**
М.: Финансы и статистика, 2004. — 424 с.: ил.
ISBN 5-279-02568-2

Рассматриваются методы искусственного интеллекта и их применение для решения задач из различных проблемных областей. Описаны методы приобретения, представления и обработки знаний в интеллектуальных системах, а также технологии проектирования и реализации интеллектуальных систем. Особое внимание уделено вопросам применения интеллектуальных систем для выбора коллективных решений, проектирования сложных систем (объектов), реинжиниринга бизнес-процессов, анализа и прогнозирования деятельности предприятия.

Для студентов старших курсов, обучающихся по специальностям «Прикладная информатика (по областям)», «Системы автоматизированного проектирования», «Вычислительные машины, комплексы, системы и сети».

А $\frac{1402070000 - 038}{010(01) - 2004}$ 353 — 2004

УДК 004.89(075.8)
ББК 32.813я73

ISBN 5-279-02568-2

© Андрейчиков А. В., Андрейчикова О. Н., 2004

Предисловие	11
Глава 1. Искусственный интеллект — основа новых информационных технологий	17
1.1. Основные направления исследований в области искусственного интеллекта	18
1.2. Классификация интеллектуальных информационных систем	20
1.2.1. Системы с интеллектуальным интерфейсом	22
1.2.2. Экспертные системы	24
1.2.3. Самообучающиеся системы	28
1.2.4. Адаптивные информационные системы	31
1.3. Технологии разработки экспертных систем	32
1.3.1. Классификационные признаки экспертных систем	35
1.3.2. Характеристика инструментальных средств	41
1.3.3. Технология проектирования и разработки экспертных систем	46
<i>Литература</i>	<i>52</i>
<i>Контрольные вопросы и задания</i>	<i>53</i>
Глава 2. Традиционные способы представления и обработки знаний в интеллектуальных системах	55
2.1. Отличия знаний от данных	55
2.2. Типичные модели представления знаний	59
2.2.1. Логическая модель представления знаний	59
2.2.2. Представление знаний правилами продукций ..	66
2.2.3. Объектно-ориентированное представление знаний фреймами	68
2.2.4. Модель семантической сети	74
2.3. Традиционные способы обработки знаний	76
2.3.1. Способы доказательства и вывода в логике	77
2.3.2. Прямой и обратный вывод в экспертных системах продукционного типа	88
2.3.3. Обработка знаний в интеллектуальных системах с фреймовым представлением	98

2.4. Примеры разработки интеллектуальных систем с применением типичных моделей представления знаний	102
2.4.1. Оболочка экспертной системы продукционного типа.....	103
2.4.2. Программирование карточной игры со многими участниками	106
<i>Литература</i>	109
<i>Контрольные вопросы и задания</i>	110
 Глава 3. Нечеткие знания и способы их обработки.....	112
3.1. Виды нечеткости знаний, способы их устранения и/или учета в интеллектуальных системах	112
3.2. Нечеткие множества и нечеткие выводы.....	126
3.3. Пример разработки компьютерной игры позиционного типа	134
3.4. Описание компьютерной системы «Нечеткая логика».....	138
3.5. Пример прогнозирования развития предприятия с применением интеллектуальной системы на базе нечеткой логики.....	144
3.6. Программные средства для работы с нечеткими знаниями	152
<i>Литература</i>	155
<i>Контрольные вопросы и задания</i>	156
 Глава 4. Методы приобретения знаний	158
4.1. Стратегии получения знаний	158
4.1.1. Аспекты извлечения знаний	159
4.1.2. Проблемы структурирования знаний.....	167
4.1.3. Семиотический подход к приобретению знаний	170
4.1.4. Методы извлечения знаний	175
4.1.5. Выявление «скрытых» структур знаний	188
4.2. Построение баз знаний для экспертных систем диагностики.....	197
4.3. Проблемы обучения интеллектуальных систем	202
4.3.1. Индуктивные выводы в логике.....	205
4.3.2. ДСМ-метод	208
4.4. Средства компьютерной поддержки приобретения знаний.....	211
4.5. Методы и средства интеллектуального анализа данных.....	214
<i>Литература</i>	225
<i>Контрольные вопросы и задания</i>	229

Глава 5. Нейронные сети.....	231
5.1. Модель искусственного нейрона	231
5.2. Модели нейронных сетей	235
5.3. Построение нейронной сети	241
5.4. Обучение нейронных сетей	242
5.5. Способы реализации нейронных сетей	246
5.6. Практическое применение нейросетевых технологий.....	251
<i>Литература</i>	<i>257</i>
<i>Контрольные вопросы и задания</i>	<i>258</i>
 Глава 6. Эволюционные аналогии в искусственных интеллектуальных системах.....	 260
6.1. Генетические алгоритмы.....	263
6.1.1. Простой генетический алгоритм	270
6.1.2. Разновидности генетических алгоритмов	276
6.1.3. Примеры практического применения генетических алгоритмов	281
6.1.4. Краткий обзор программных средств.....	287
6.2. Методы эволюционного программирования	292
6.2.1. Генетическое программирование.....	292
6.2.2. Эволюционное программирование	298
6.2.3. Эволюционные стратегии	300
<i>Литература</i>	<i>302</i>
<i>Контрольные вопросы и задания</i>	<i>304</i>
 Глава 7. Интеллектуальные мультиагентные системы	 306
7.1. Основные понятия теории агентов	307
7.1.1. Характеристики интеллектуальных агентов.....	308
7.1.2. Архитектуры мультиагентных систем	310
7.2. Коллективное поведение агентов.....	312
7.2.1. Способы и причины взаимодействия между агентами	313
7.2.2. Моделирование взаимодействия в мультиагентных системах.....	315
7.2.3. Координация поведения агентов в мультиагентной системе	318
7.3. Примеры мультиагентных систем.....	321
7.3.1. Координация поведения на основе модели аукциона	321
7.3.2. Мультиагентная система для поддержки процессов принятия решений на предприятии	325

7.4. Технологии проектирования мультиагентных систем.....	328
7.4.1. Инструментальные средства для построения мультиагентных систем	330
7.4.2. Мультиагентные системы для поиска информации	332
7.5. Перспективы мультиагентных технологий.....	335
<i>Литература</i>	336
<i>Контрольные вопросы и задания</i>	338
 Глава 8. Интеллектуальные методы проектирования сложных систем	340
8.1. Проблемы проектирования и реинжиниринга экономических систем.....	341
8.2. Системный подход к проектированию сложных систем	344
8.3. Программные средства для поддержки процессов реинжиниринга	354
8.4. Подход к коллективному выбору решений при проектировании экономических систем	357
8.5. Разрешение конфликтов при коллективном выборе решений.....	373
8.6. Эволюционный синтез систем и объектов	378
8.7. Логический подход к синтезу сценариев развития сложных систем.....	393
<i>Литература</i>	415
<i>Контрольные вопросы и задания</i>	417
 Предметный указатель	419

Contents

Preface	11
Chapter 1. Artificial Intelligence as the base of new information technologies	17
1.1. Main lines of researches in Artificial Intelligence	18
1.2. The classification of intelligent information systems ...	20
1.2.1. Systems with the intelligent interface	22
1.2.2. Expert systems	24
1.2.3. Self-learning systems	28
1.2.4. Adaptive information systems	31
1.3. Technologies for expert systems' development	32
1.3.1. The classification attributes of the expert systems	35
1.3.2. Overview of software tools	41
1.3.3. Technology for designing and development of expert systems	46
References	52
Questions and tasks	53
 Chapter 2. The traditional modes for handling knowledge in intelligent systems	 55
2.1. The difference between knowledge and data	55
2.2. Typical models for representation of knowledge	59
2.2.1. Logical model	59
2.2.2. Production rules	66
2.2.3. Representation of knowledge by frames	68
2.2.4. Semantic networks	74
2.3. The ways of knowledge processing	76
2.3.1. The methods of the proof and reasoning in classic logic	77
2.3.2. Forward and backward chaining in expert systems based on production rules	88
2.3.3. Knowledge handling in intelligent systems with frame representation	98

2.4. Examples of development of intelligent systems with typical models of knowledge representation	102
2.4.1. The shell of expert system based on production rules	103
2.4.2. A card-play with many participants	106
<i>References</i>	109
<i>Questions and tasks</i>	110
Chapter 3. Uncertain knowledge and methods for its processing	112
3.1. Kinds of knowledge uncertainty and the ways of their elimination and/or consideration in intelligent systems	112
3.2. Fuzzy sets and fuzzy reasoning	126
3.3. An example of the computer game development	134
3.4. Computer system «Fuzzy logic»	138
3.5. An example of forecasting of the enterprise future with use of intelligent system «Fuzzy logic»	144
3.6. Overview of software on fuzzy knowledge processing ..	152
<i>References</i>	155
<i>Questions and tasks</i>	156
Chapter 4. Techniques for knowledge acquisition	158
4.1. Strategy of knowledge acquisition	158
4.1.1. Main aspects of knowledge extraction	159
4.1.2. The problems of knowledge structuring	167
4.1.3. The semiotic approach to knowledge acquisition	170
4.1.4. Methods of knowledge extraction	175
4.1.5. The revealing of a latent knowledge	188
4.2. The formation of knowledge base for diagnostic expert system	197
4.3. The problems of teaching intelligent systems	202
4.3.1. Inductive inference in logic	205
4.3.2. DSM-method	208
4.4. The tools for computer support of knowledge acquisition	211
4.5. Technologies for Data Mining and Knowledge Discovery	214
<i>References</i>	225
<i>Questions and tasks</i>	229

Chapter 5. Neural networks	231
5.1. An artificial neuron model	231
5.2. The neural networks' models	235
5.3. Creation of neural networks	241
5.4. Training of a neural network	242
5.5. Implementation of neural network	246
5.6. Neural networks' applications	251
<i>References</i>	<i>257</i>
<i>Questions and tasks</i>	<i>258</i>
 Chapter 6. Evolutionary analogies in artificial intelligent systems ..	260
6.1. Genetic algorithms	263
6.1.1. Simple genetic algorithm	270
6.1.2. Genetic algorithms' modifications	276
6.1.3. Practical applications of genetic algorithms	281
6.1.4. The brief review of evolutionary software	287
6.2. Methods of evolutionary programming	292
6.2.1. Genetic programming	292
6.2.2. Evolutionary programming	298
6.2.3. Evolutionary strategy	300
<i>References</i>	<i>302</i>
<i>Questions and tasks</i>	<i>304</i>
 Chapter 7. Intelligent multiagent systems	306
7.1. The basic concepts of the agents theory	307
7.1.1. Main features of intelligent agents	308
7.1.2. Multiagent systems' architecture types	310
7.2. Collective behavior of the agents	312
7.2.1. The ways and the reasons of agents communication	313
7.2.2. Modeling of communication in multiagent systems	315
7.2.3. Behavior coordination of the agents in multiagent systems	318
7.3. Examples of multiagent systems	321
7.3.1. Coordination of agents behavior with use of auction model	321
7.3.2. Multiagent system for supporting of decision-making process	325

7.4. Technologies for designing of multiagent systems	328
7.4.1. Software tools for development of multiagent systems	330
7.4.2. Multiagent systems for information search	332
7.5. Multiagent technologies prospects	335
<i>References</i>	336
<i>Questions and tasks</i>	338
Chapter 8. Intelligent methods for designing of the complex systems	340
8.1. The problems of economic systems' designing and reengineering	341
8.2. System approach to the designing of complex systems	344
8.3. Software for support of reengineering	354
8.4. The approach to a collaborative decision-making in the economic systems' designing	357
8.5. The conflicts resolution in a collective decision-making	373
8.6. Evolutionary synthesis of complex systems and objects	378
8.7. Logical approach to the scripts' synthesis for forecasting of future	393
<i>References</i>	415
<i>Questions and tasks</i>	417
Index	419

Предисловие

В основе любой организованной деятельности лежит информация. Основоположник кибернетики Норберт Винер определил это понятие так: «Информация есть информация, не материя и не энергия». Следствием нематериальной природы информации является возможность ее неограниченного тиражирования, которая означает неисчерпаемость информационных ресурсов и свидетельствует об их принципиальном отличии от ресурсов материальных. Отличительное свойство информационных ресурсов — их универсальность, так как они необходимы в любой сфере человеческой деятельности.

Информационные процессы имеют место в живой и неживой природе. Для человеческого общества характерно стремление к познанию и преобразованию окружающего мира путем приобретения информации, ее хранения, передачи, обработки и использования. В течение многих веков люди добывают, обрабатывают и используют материальные и энергетические ресурсы. Они создали для этих целей мощное оборудование и эффективные технологии. То же можно сказать об информации: на протяжении своей истории человечество накапливало знания, полученные из наблюдений и опыта, пыталось сохранить их и передать последующим поколениям, делало верные и неверные выводы из совокупности доступных знаний и было неудержимо в попытках применения полученных знаний на практике. В результате этой деятельности человеку удалось существенно изменить лицо нашей планеты. Остается только удивляться тому, что только в XX веке люди создали эффективный инструмент для обработки информации — компьютер. Однако с момента его появления развитие новых технологий обработки информации было настолько стремительным, как будто эти полвека измерялись в другой шкале времени.

Традиционно считается, что естественные и технические науки ориентированы на созидание благ, в то время как гуманитар-

ные науки связаны с распределением различного рода ресурсов, в том числе затрат и доходов. Задачи созидания чаще всего не могут быть решены без привлечения материальных, энергетических и информационных ресурсов. В гуманитарных исследованиях информация является определяющим, а иногда единственным и достаточным видом требуемых ресурсов, поэтому специалистам в области экономики и управления необходимо иметь глубокие знания о технологиях ее обработки. В естественных и технических науках значительную часть изучаемых закономерностей можно проверить экспериментально и описать на количественном уровне с помощью детерминированных или стохастических моделей. В гуманитарных науках, таких, как экономика, социология, социальная психология, политология и другие, эксперимент, как правило, невозможен, а описание области исследования представлено наборами фактов, гипотез, правил и закономерностей, сформулированных на качественном уровне.

В соответствии с классификацией проблем, предложенной Г. Саймоном, задачи, описанные количественными и качественными признаками с преобладанием последних, относятся к слабо-структурированным проблемам. Современный подход к решению таких проблем базируется на методах искусственного интеллекта.

Настоящая книга посвящена интеллектуальным информационным системам, т. е. вопросам организации, проектирования, разработки и применения систем, предназначенных для обработки информации, базирующихся на применении методов искусственного интеллекта. Поскольку в сферу приложений искусственного интеллекта вошли практически все направления современной информатики, мы не ставили перед собой цели «объять необъятное», а включили в учебник разделы, содержащие описание традиционных моделей и технологий создания интеллектуальных систем, а также новых перспективных подходов к решению проблем, возникающих в области искусственного интеллекта. При написании книги мы видели свою цель в том, чтобы познакомить читателя с принципами создания и функционирования интеллектуальных информационных систем. Практический опыт показывает, что люди, не знакомые с этими принципами, испытывают большие трудности, выступая в роли пользователей интеллектуального программного обеспечения.

В главе I приведены краткий обзор приложений искусственного интеллекта, общая характеристика интеллектуальных ин-

формационных систем, а также основные понятия и определения, которые используются в книге.

Глава 2 посвящена описанию классических моделей представления знаний средствами логики высказываний и логики предикатов, правилами продукций, имеющих вид «ЕСЛИ А, ТО В», иерархиями фреймов и семантическими сетями. Здесь также рассмотрены основы компьютерной обработки знаний, представленных различными средствами, знакомство с которыми необходимо как разработчикам интеллектуальных систем, так и пользователям. Приведены примеры разработки интеллектуальных информационных систем с применением описанных моделей.

В главе 3 описаны виды нечеткости знаний, а также способы их устранения или учета в интеллектуальных информационных системах. Неоднозначность, ненадежность, неточность, неполнота знаний и другие так называемые НЕ-факторы вызвали появление нового направления искусственного интеллекта — *мягких вычислений*, а также послужили отправной точкой для создания различных направлений нетрадиционной логики. Разработанные в этой области средства не только получили широкое распространение в виде программного обеспечения для интеллектуальной обработки информации, но нашли применение в сфере нечеткого управления техническими объектами, в частности автомобилями, бытовой техникой и т. д. В разд. 3.5 рассмотрен пример использования интеллектуальной системы с нечетким выводом для оценки и прогнозирования состояния малого предприятия.

Глава 4 содержит сведения о методах и средствах приобретения знаний, которые закладываются в интеллектуальные информационные системы. Проблемы приобретения знаний имеют место на стадиях проектирования, разработки и эксплуатации программного обеспечения, поэтому для их решения применяются различные подходы. Методы приобретения знаний, используемые на стадии проектирования системы, ориентированы на получение необходимой информации от экспертов, проверку элементов этой информации на непротиворечивость и тесно связаны с проблемами ее представления и обработки. На стадии разработки интеллектуальной системы рассматриваются вопросы добавления новой информации в базу знаний, организации ее взаимодействия с внешними информационными источниками и реализации возможностей самообучения. Эксплуатация интел-

лектуальных систем требует поддержания баз знаний и данных в непротиворечивом и актуальном состоянии. Для решения этой задачи необходимы подготовленные пользователи (инженеры по знаниям), владеющие компьютерными технологиями извлечения, добавления, согласования и интерпретации знаний.

Известно, что только небольшую часть своих знаний человек может точно сформулировать вербальным или формальным способом. Обширная область интуитивных знаний специалистов, которые необходимы для успешной работы интеллектуальных систем, остается недоступной из-за отсутствия средств их извлечения и представления. Неуловимый характер человеческих знаний и их постоянное развитие помешали сторонникам «нисходящего метода» в области искусственного интеллекта удержать в своих руках пальму первенства. «Нисходящий метод» соответствует дедуктивному подходу, в рамках которого на этапе становления искусственного интеллекта разрабатывались программы, способные решать сложные задачи на основе логической обработки содержащихся в них знаний. Примерами таких программ являются знаменитый «Логик-Теоретик» и GPS — универсальный решатель задач. В их разработке участвовали известные ученые А. Ньюэлл, А. Тьюринг, К. Шеннон, Г. Саймон, Дж. Шоу и др.

«Восходящий метод» развивался в работах Дж. Маккалоха, У. Питтса, Ф. Розенблата и др., посвященных созданию самоорганизующихся систем и «самообучающихся машин». Эти ученые опирались на идею Н. Винера об обратной связи, благодаря которой все живое приспосабливается к окружающей среде и добивается своих целей. Так возникло направление, связанное с разработкой нейронных сетей, которое, не успев твердо встать на ноги, было подвергнуто суровой критике оппонентов из противоположного лагеря (М. Минский и С. Пейперт) и какое-то время считалось неперспективным. Однако стремительное развитие аппаратных компьютерных средств и не оправдавшиеся надежды на возможности экспертных систем с дедуктивными выводами стали причиной второго рождения нейросетевых технологий в 1980-х гг. Сегодня модели нейронных сетей активно разрабатываются и применяются для решения задач прогнозирования, распознавания, извлечения знаний из хранилищ данных. Этим вопросам посвящена глава 5.

Глава 6 содержит описание методов обработки информации, основанных на эволюционных аналогиях. Наличие символической

информации и отсутствие детерминированных алгоритмов ее обработки в интеллектуальных системах послужило причиной возникновения задач поиска в пространстве высокой размерности, а также комбинаторных задач, для решения которых успешно применяются генетические алгоритмы, методы эволюционного программирования и эволюционные стратегии. При этом для решения сложных и плохо обусловленных задач все чаще применяются комбинированные подходы, в которых сочетаются методы нечеткого представления знаний, модели нейронных сетей для получения результата при отсутствии четко заданного алгоритма и генетические методы для оптимизации полученных решений.

В главе 7 рассмотрены вопросы проектирования и применения мультиагентных систем, ориентированных на автономное выполнение интеллектуальных задач в распределенных компьютерных средах. Это одно из самых новых направлений в искусственном интеллекте, которое имеет большие перспективы в связи с широким распространением Интернета и представляет особый интерес для специалистов в области экономики и бизнеса, так как является базой для создания виртуальных предприятий.

Глава 8 знакомит читателя с интеллектуальными методами проектирования сложных систем. Вопросы проектирования и реинжиниринга (перепроектирования) экономических и социально-производственных систем особенно актуальны в настоящее время, так как их решение позволяет качественно улучшить показатели функционирования за счет совершенствования системы управления, а не за счет привлечения дополнительных материальных и денежных ресурсов. Методологической базой проектирования сложных систем являются системный анализ и математическое моделирование. В главе рассмотрены подходы, методы и системы, которые изначально были разработаны нами в области проектирования технических систем и впоследствии адаптированы к проблемам экономики и управления, в том числе: подход к решению задач коллективного выбора с взаимными требованиями участников, мультиагентная система для разрешения конфликтов, метод эволюционного синтеза сложных систем и метод генерации сценариев на основе причинно-следственных связей. Эти средства могут использоваться для компьютерной поддержки решения задач прогнозирования, стратегического планирования, генерации и выбора рациональных решений, что проиллюстрировано примерами.

В конце каждой главы учебника приведены список литературы, а также контрольные вопросы и задания.

При написании книги мы не стремились ориентировать читателя на применение конкретных видов интеллектуального программного обеспечения, имеющегося на рынке в настоящее время, и сознательно не приводили подробного описания программных систем и методик решения прикладных задач с использованием тех или иных пакетов. Это объясняется большим разнообразием таких средств и высокой сложностью их освоения. Эффективная работа с инструментальным комплексом G2 или другими интеллектуальными системами требует подробного ознакомления со специальной литературой и документацией. Ограниченный объем книги не позволил нам привести подробного и понятного описания применения упоминаемых программных продуктов. Надеемся, что знания, полученные при изучении этой книги, и приведенные краткие обзоры программных средств позволят читателю совершить самостоятельный выбор необходимого программного обеспечения и помогут в его освоении.

Авторы выражают глубокую благодарность профессорам: А.В. Смирнову, В.М. Курейчику, Б.В. Палюху, Н. А. Семенову и А.А. Емельянову за содержательное обсуждение и рецензирование нашей книги, студентам Волгоградского государственного технического университета, которые слушали наши лекции и разрабатывали программное обеспечение, а также сотрудникам и аспирантам кафедры «Информационные системы в экономике» Волгоградского государственного технического университета, помогавшим нам в работе над этой книгой.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ – ОСНОВА НОВЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Знания – орудие, а не цель.

Л. Н. Толстой

Искусственный интеллект (ИИ) как наука существует около полувека. Первой интеллектуальной системой считается программа «Логик-Теоретик», предназначенная для доказательства теорем и исчисления высказываний. Ее работа впервые была продемонстрирована 9 августа 1956 г. В создании программы участвовали такие известные ученые, как А. Ньюэлл, А. Тьюринг, К. Шеннон, Дж. Лоу, Г. Саймон и др. За прошедшее с тех пор время в области ИИ разработано великое множество компьютерных систем, которые принято называть интеллектуальными. Области их применения охватывают практически все сферы человеческой деятельности, связанные с обработкой информации.

На сегодняшний день не существует единого определения, которое однозначно описывает эту научную область. Среди многих точек зрения на нее доминируют следующие три [7]. Согласно первой исследования в области ИИ относятся к фундаментальным, в процессе которых разрабатываются новые модели и методы решения задач, традиционно считавшихся интеллектуальными и не поддававшихся ранее формализации и автоматизации. Согласно второй точке зрения это направление связано с новыми идеями решения задач на ЭВМ, с разработкой новых технологий программирования и с переходом к компьютерам не фон-неймановской архитектуры. Третья точка зрения, наиболее прагматическая, основана на том, что в результате исследований, проводимых в области ИИ, появляется множество прикладных систем, способных решать задачи, для которых ранее создаваемые системы были непригодны. По последней трактовке ИИ является экспериментальной научной дисциплиной, в которой роль эксперимента заключается в проверке и уточнении интеллектуальных систем, представляющих собой аппаратно-программные информационные комплексы.

ОСНОВНЫЕ НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ В ОБЛАСТИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Интеллектуальные информационные системы проникают во все сферы нашей жизни, поэтому трудно провести строгую классификацию направлений, по которым ведутся активные и многочисленные исследования в области ИИ. Рассмотрим кратко некоторые из них.

Разработка интеллектуальных информационных систем или систем, основанных на знаниях. Это одно из главных направлений ИИ. Основной целью построения таких систем являются выявление, исследование и применение знаний высококвалифицированных экспертов для решения сложных задач, возникающих на практике. При построении систем, основанных на знаниях (СОЗ), используются знания, накопленные экспертами в виде конкретных правил решения тех или иных задач. Это направление преследует цель имитации человеческого искусства анализа неструктурированных и слабоструктурированных проблем [9]. В данной области исследований осуществляется разработка моделей представления, извлечения и структурирования знаний, а также изучаются проблемы создания баз знаний (БЗ), образующих ядро СОЗ. Частным случаем СОЗ являются экспертные системы (ЭС).

Разработка естественно-языковых интерфейсов и машинный перевод. Проблемы компьютерной лингвистики и машинного перевода разрабатываются в ИИ с 1950-х гг. Системы машинного перевода с одного естественного языка на другой обеспечивают быстроту и систематичность доступа к информации, оперативность и единообразие перевода больших потоков, как правило, научно-технических текстов [6]. Системы машинного перевода строятся как интеллектуальные системы, поскольку в их основе лежат БЗ в определенной предметной области и сложные модели, обеспечивающие дополнительную трансляцию «исходный язык оригинала — язык смысла — язык перевода». Они базируются на структурно-логическом подходе, включающем последовательный анализ и синтез естественно-языковых сообщений. Кроме того, в них осуществляется ассоциативный поиск аналогичных фрагментов текста и их переводов в специальных базах данных (БД). Данное направление охватывает также исследования методов и разработку систем, обеспечивающих реализацию процесса

общения человека с компьютером на естественном языке (так называемые системы ЕЯ-общения) [6].

Генерация и распознавание речи. Системы речевого общения создаются в целях повышения скорости ввода информации в ЭВМ, разгрузки зрения и рук, а также для реализации речевого общения на значительном расстоянии. В таких системах под текстом понимают фонемный текст (как слышится).

Обработка визуальной информации. В этом научном направлении решаются задачи обработки, анализа и синтеза изображений [6]. Задача обработки изображений связана с трансформированием графических образов, результатом которого являются новые изображения. В задаче анализа исходные изображения преобразуются в данные другого типа, например в текстовые описания. При синтезе изображений на вход системы поступает алгоритм построения изображения, а выходными данными являются графические объекты (системы машинной графики).

Обучение и самообучение. Эта актуальная область ИИ включает модели, методы и алгоритмы, ориентированные на автоматическое накопление и формирование знаний с использованием процедур анализа и обобщения данных [4, 13]. К данному направлению относятся не так давно появившиеся системы добычи данных (Data-mining) и системы поиска закономерностей в компьютерных базах данных (Knowledge Discovery).

Распознавание образов. Это одно из самых ранних направлений ИИ, в котором распознавание объектов осуществляется на основании применения специального математического аппарата, обеспечивающего отнесение объектов к классам [7], а классы описываются совокупностями определенных значений признаков.

Игры и машинное творчество. Машинное творчество охватывает сочинение компьютерной музыки [5], стихов [6], интеллектуальные системы для изобретения новых объектов [2, 14]. Создание интеллектуальных компьютерных игр является одним из самых развитых коммерческих направлений в сфере разработки программного обеспечения. Кроме того, компьютерные игры предоставляют мощный арсенал разнообразных средств, используемых для обучения.

Программное обеспечение систем ИИ. Инструментальные средства для разработки интеллектуальных систем включают специальные языки программирования, ориентированные на об-

работку символьной информации (LISP, SMALLTALK, РЕФАЛ), языки логического программирования (PROLOG), языки представления знаний (OPS 5, KRL, FRL), интегрированные программные среды, содержащие арсенал инструментальных средств для создания систем ИИ (KE, ARTS, GURU, G2), а также оболочки экспертных систем (BUILD, EMYCIN, EXSYS Professional, ЭКСПЕРТ), которые позволяют создавать прикладные ЭС, не прибегая к программированию [8, 11].

Новые архитектуры компьютеров. Это направление связано с созданием компьютеров не фон-неймановской архитектуры, ориентированных на обработку символьной информации. Известны удачные промышленные решения параллельных и векторных компьютеров [1, 8], однако в настоящее время они имеют весьма высокую стоимость, а также недостаточную совместимость с существующими вычислительными средствами.

Интеллектуальные роботы. Создание интеллектуальных роботов составляет конечную цель робототехники. В настоящее время в основном используются программируемые манипуляторы с жесткой схемой управления, названные роботами первого поколения. Несмотря на очевидные успехи отдельных разработок, эра интеллектуальных автономных роботов пока не наступила. Основными сдерживающими факторами в разработке автономных роботов являются нерешенные проблемы в области интерпретации знаний, машинного зрения, адекватного хранения и обработки трехмерной визуальной информации.

1.2. КЛАССИФИКАЦИЯ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Интеллектуальная информационная система (ИИС) основана на концепции использования базы знаний для генерации алгоритмов решения прикладных задач различных классов в зависимости от конкретных информационных потребностей пользователей.

Для ИИС характерны следующие признаки [12]:

- развитые коммуникативные способности;
- умение решать сложные плохо формализуемые задачи;
- способность к самообучению;
- адаптивность.

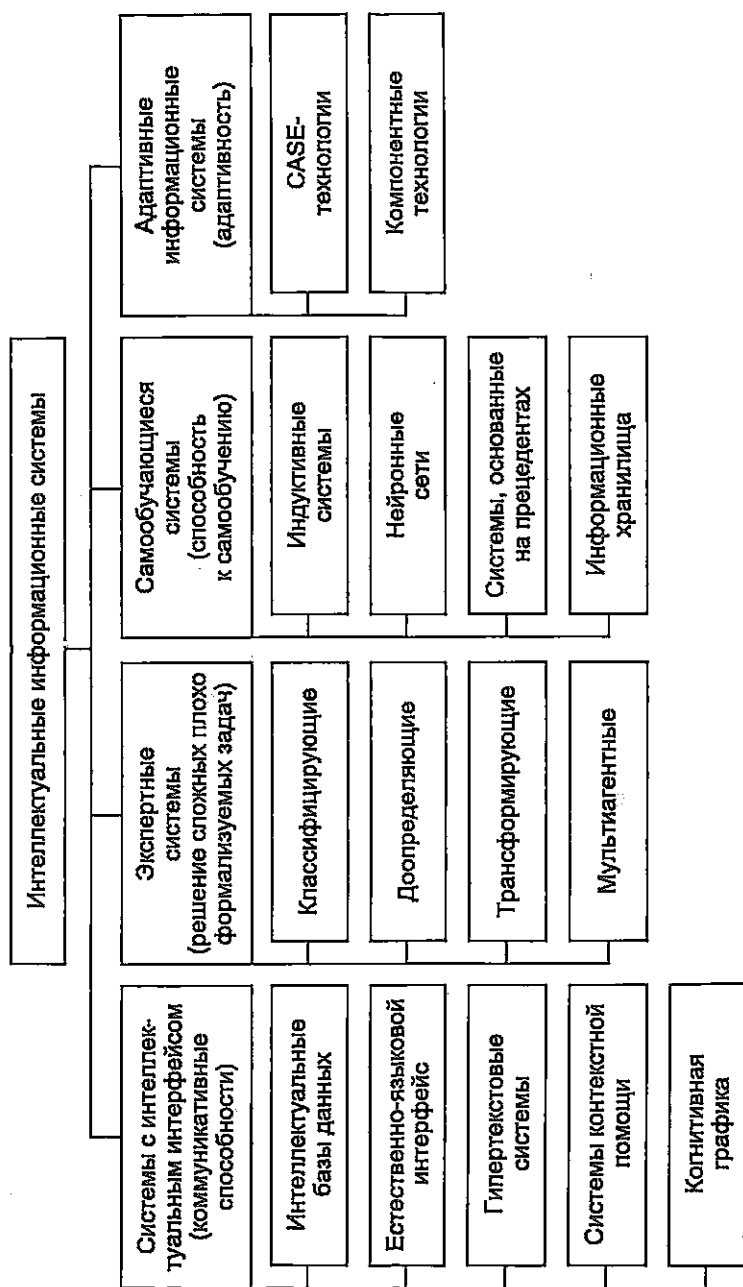


Рис. 1.1. Классификация интеллектуальных информационных систем

Каждому из перечисленных признаков условно соответствует свой класс ИИС. Различные системы могут обладать одним или несколькими признаками интеллектуальности с различной степенью проявления.

Средства ИИ могут использоваться для реализации различных функций, выполняемых ИИС. На рис.1.1 приведена классификация ИИС, признаками которой являются следующие интеллектуальные функции:

- *коммуникативные способности* — способ взаимодействия конечного пользователя с системой;

- *решение сложных плохо формализуемых задач*, которые требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, характеризующейся неопределенностью и динамичностью исходных данных и знаний;

- *способность к самообучению* — умение системы автоматически извлекать знания из накопленного опыта и применять их для решения задач;

- *адаптивность* — способность системы к развитию в соответствии с объективными изменениями области знаний.

1.2.1. СИСТЕМЫ С ИНТЕЛЛЕКТУАЛЬНЫМ ИНТЕРФЕЙСОМ

Применение ИИ для усиления коммуникативных способностей информационных систем привело к появлению систем с интеллектуальным интерфейсом, среди которых можно выделить следующие типы.

1. *Интеллектуальные базы данных*. Позволяют в отличие от традиционных БД обеспечивать выборку необходимой информации, не присутствующей в явном виде, а выводимой из совокупности хранимых данных.

2. *Естественно-языковой интерфейс*. Применяется для доступа к интеллектуальным базам данных, контекстного поиска документальной текстовой информации, голосового ввода команд в системах управления, машинного перевода с иностранных языков. Для реализации ЕЯ-интерфейса необходимо решить проблемы морфологического, синтаксического и семантического анализа, а также задачу синтеза высказываний на естественном языке. При морфологическом анализе осуществляются распознава-

ние и проверка правильности написания слов в словаре. Синтаксический контроль предполагает разложение входных сообщений на отдельные компоненты, проверку соответствия грамматическим правилам внутреннего представления знаний и выявление недостающих частей. Семантический анализ обеспечивает установление смысловой правильности синтаксических конструкций. В отличие от анализа синтез высказываний заключается в преобразовании цифрового представления информации в представление на естественном языке.

3. *Гипертекстовые системы.* Используются для реализации поиска по ключевым словам в базах данных с текстовой информацией. Для более полного отражения различных смысловых отношений терминов требуется сложная семантическая организация ключевых слов. Решение этих задач осуществляется с помощью интеллектуальных гипертекстовых систем, в которых механизм поиска сначала работает с базой знаний ключевых слов, а затем — с самим текстом. Аналогичным образом проводится поиск мультимедийной информации, включающей кроме текста графическую информацию, аудио- и видеообразы.

4. *Системы контекстной помощи.* Относятся к классу систем распространения знаний. Такие системы являются, как правило, приложениями к документации. Системы контекстной помощи — частный случай гипертекстовых и ЕЯ-систем. В них пользователь описывает проблему, а система на основе дополнительного диалога конкретизирует ее и выполняет поиск относящихся к ситуации рекомендаций. В обычных гипертекстовых системах, наоборот, компьютерные приложения навязывают пользователю схему поиска требуемой информации.

5. *Системы когнитивной графики.* Ориентированы на общение с пользователем ИИС посредством графических образов, которые генерируются в соответствии с изменениями параметров моделируемых или наблюдаемых процессов. Когнитивная графика позволяет в наглядном и выразительном виде представить множество параметров, характеризующих изучаемое явление, освобождает пользователя от анализа тривиальных ситуаций, способствует быстрому освоению программных средств и повышению конкурентоспособности разрабатываемых ИИС. Применение когнитивной графики особенно актуально в системах мониторинга и оперативного управления, в обучающих и тренажерных системах, в оперативных системах принятия решений, работающих в режиме реального времени.

Экспертные системы как самостоятельное направление в искусственном интеллекте сформировалось в конце 1970-х гг. История ЭС началась с сообщения японского комитета по разработке ЭВМ пятого поколения, в котором основное внимание уделялось развитию «интеллектуальных способностей» компьютеров с тем, чтобы они могли оперировать не только данными, но и знаниями, как это делают специалисты (эксперты) при выработке умозаключений. Группа по экспертным системам при Комитете British Computer Society определила ЭС как «воплощение в ЭВМ компоненты опыта эксперта, основанной на знаниях, в такой форме, что машина может дать интеллектуальный совет или принять решение относительно обрабатываемой функции». Одним из важных свойств ЭС является способность объяснить ход своих рассуждений понятным для пользователя образом [15].

Область исследования ЭС называют «инженерией знаний». Этот термин был введен Е. Фейгенбаумом и в его трактовке означает «привнесение принципов и инструментария из области искусственного интеллекта в решение трудных прикладных проблем, требующих знаний экспертов». Другими словами, ЭС применяются для решения неформализованных проблем, к которым относят задачи, обладающие одной (или несколькими) из следующих характеристик:

- задачи не могут быть представлены в числовой форме;
- исходные данные и знания о предметной области обладают неоднозначностью, неточностью, противоречивостью;
- цели нельзя выразить с помощью четко определенной целевой функции;
- не существует однозначного алгоритмического решения задачи;
- алгоритмическое решение существует, но его нельзя использовать по причине большой размерности пространства решений и ограничений на ресурсы (времени, памяти).

Главное отличие ЭС и систем искусственного интеллекта от систем обработки данных состоит в том, что в них используется символьный, а не числовой способ представления данных, а в качестве методов обработки информации применяются процедуры логического вывода и эвристического поиска решений.

ЭС охватывают самые разные предметные области (рис.1.2), среди которых лидируют бизнес, производство, медицина, проектирование и системы управления [4, 6, 11, 12, 15, 17].

Во многих случаях ЭС являются инструментом, усиливающим интеллектуальные способности эксперта. Кроме того, ЭС может выступать в роли:

- консультанта для неопытных или непрофессиональных пользователей;
- ассистента эксперта-человека в процессах анализа вариантов решений;

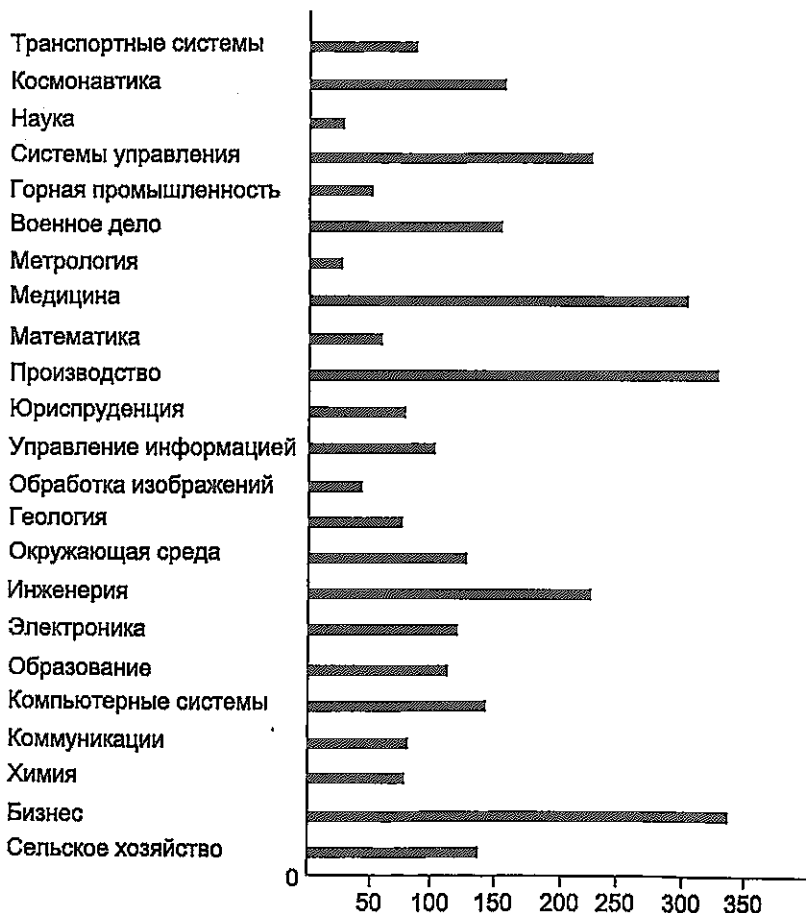


Рис. 1.2. Области применения экспертных систем

• партнера эксперта в процессе решения задач, требующих привлечения знаний из разных предметных областей.

Для классификации ЭС используются следующие признаки:

- способ формирования решения;
- способ учета временного признака;
- вид используемых данных и знаний;
- число используемых источников знаний.

По способу формирования решения ЭС можно разделить на анализирующие и синтезирующие. В системах первого типа осуществляется выбор решения из множества известных решений на основе анализа знаний, в системах второго типа решение синтезируется из отдельных фрагментов знаний.

В зависимости от способа учета временного признака ЭС делят на статические и динамические. Статические ЭС предназначены для решения задач с неизменяемыми в процессе решения данными и знаниями, а динамические ЭС допускают такие изменения.

По видам используемых данных и знаний различают ЭС с детерминированными и неопределенными знаниями. Под неопределенностью знаний и данных понимаются их неполнота, ненадежность, нечеткость.

ЭС могут создаваться с использованием одного или нескольких источников знаний.

В соответствии с перечисленными признаками можно выделить четыре основных класса ЭС (рис. 1.3): классифицирующие, доопределяющие, трансформирующие и мультиагентные [12].

Классифицирующие ЭС решают задачи распознавания ситуаций. Основным методом формирования решений в таких системах является дедуктивный логический вывод.

Доопределяющие ЭС используются для решения задач с не полностью определенными данными и знаниями. В таких ЭС возникают задачи интерпретации нечетких знаний и выбора альтернативных направлений поиска в пространстве возможных решений. В качестве методов обработки неопределенных знаний могут использоваться байесовский вероятностный подход, коэффициенты уверенности, нечеткая логика (см. главу 3).

Трансформирующие ЭС относятся к синтезирующим динамическим экспертным системам, в которых предполагается повторяющееся преобразование знаний в процессе решения задач. В

	Анализ	Синтез	
Детерминированность знаний	Классифицирующие	Трансформирующие	Один источник знаний
Неопределенность знаний	Доопределяющие	Мультиагентные	Несколько источников знаний
	Статика	Динамика	

Рис. 1.3. Основные классы экспертных систем

ЭС данного класса используются различные способы обработки знаний:

- генерация и проверка гипотез;
- логика предположений и умолчаний (когда по неполным данным формируются представления об объектах определенного класса, которые впоследствии адаптируются к конкретным условиям изменяющихся ситуаций);
- использование метазнаний (более общих закономерностей) для устранения неопределенностей в ситуациях.

Мультиагентные системы — это динамические ЭС, основанные на интеграции нескольких разнородных источников знаний. Эти источники обмениваются между собой получаемыми результатами в ходе решения задач. Системы данного класса имеют следующие возможности:

- реализация альтернативных рассуждений на основе использования различных источников знаний и механизма устранения противоречий;
- распределенное решение проблем, декомпозируемых на параллельно решаемые подзадачи с самостоятельными источниками знаний;
- применение различных стратегий вывода заключений в зависимости от типа решаемой проблемы;
- обработка больших массивов информации из баз данных;
- использование математических моделей и внешних процедур для имитации развития ситуаций.

1.2.3. САМООБУЧАЮЩИЕСЯ СИСТЕМЫ

Самообучающиеся интеллектуальные системы основаны на методах автоматической классификации ситуаций из реальной практики, или на методах обучения на примерах. Примеры реальных ситуаций составляют так называемую обучающую выборку, которая формируется в течение определенного исторического периода. Элементы обучающей выборки описываются множеством классификационных признаков.

Стратегия «обучения с учителем» предполагает задание специалистом для каждого примера значений признаков, показывающих его принадлежность к определенному классу ситуаций. При обучении «без учителя» система должна самостоятельно выделять классы ситуаций по степени близости значений классификационных признаков.

В процессе обучения проводится автоматическое построение обобщающих правил или функций, описывающих принадлежность ситуаций к классам, которыми система впоследствии будет пользоваться при интерпретации незнакомых ситуаций. Из обобщающих правил, в свою очередь, автоматически формируется база знаний, которая периодически корректируется по мере накопления информации об анализируемых ситуациях.

Построенные в соответствии с этими принципами самообучающиеся системы имеют следующие недостатки:

- относительно низкую адекватность баз знаний возникающим реальным проблемам из-за неполноты и/или зашумленности обучающей выборки;
- низкую степень объяснимости полученных результатов;
- поверхностное описание проблемной области и узкую направленность применения из-за ограничений в размерности признакового пространства.

Индуктивные системы позволяют обобщать примеры на основе принципа индукции «от частного к общему». Процедура обобщения сводится к классификации примеров по значимым признакам. Алгоритм классификации примеров включает следующие основные шаги.

1. Выбор классификационного признака из множества заданных.
2. Разбиение множества примеров на подмножества по значению выбранного признака.

3. Проверка принадлежности каждого подмножества примеров одному из классов.

4. Проверка окончания процесса классификации. Если какое-то подмножество примеров принадлежит одному подклассу, т.е. у всех примеров этого подмножества совпадает значение классификационного признака, то процесс классификации заканчивается.

5. Для подмножеств примеров с несовпадающими значениями классификационных признаков процесс распознавания продолжается, начиная с первого шага. При этом каждое подмножество примеров становится классифицируемым множеством.

Нейронные сети представляют собой классический пример технологии, основанной на примерах. Нейронные сети – обобщенное название группы математических алгоритмов, обладающих способностью обучаться на примерах, «узнавая» впоследствии черты встреченных образцов и ситуаций. Благодаря этой способности нейронные сети используются при решении задач обработки сигналов и изображений, распознавания образов, а также для прогнозирования [10].

Нейронная сеть – это кибернетическая модель нервной системы, которая представляет собой совокупность большого числа сравнительно простых элементов – нейронов, топология соединения которых зависит от типа сети. Чтобы создать нейронную сеть для решения какой-либо конкретной задачи, следует выбрать способ соединения нейронов друг с другом и подобрать значения параметров межнейронных соединений.

В *системах, основанных на прецедентах*, БЗ содержит описания конкретных ситуаций (прецеденты). Поиск решения осуществляется на основе аналогий и включает следующие этапы:

- получение информации о текущей проблеме;
- сопоставление полученной информации со значениями признаков прецедентов из базы знаний;
- выбор прецедента из базы знаний, наиболее близкого к рассматриваемой проблеме;
- адаптация выбранного прецедента к текущей проблеме;
- проверка корректности каждого полученного решения;
- занесение детальной информации о полученном решении в БЗ.

Прецеденты описываются множеством признаков, по которым строятся индексы быстрого поиска. Однако в системах, ос-

нованных на прецедентах, в отличие от индуктивных систем допускается нечеткий поиск с получением множества допустимых альтернатив, каждая из которых оценивается некоторым коэффициентом уверенности. Наиболее эффективные решения адаптируются к реальным ситуациям с помощью специальных алгоритмов.

Системы, основанные на прецедентах, применяются для распространения знаний и в системах контекстной помощи.

Информационные хранилища отличаются от интеллектуальных баз данных, тем, что представляют собой хранилища значимой информации, регулярно извлекаемой из оперативных баз данных. Хранилище данных — это предметно-ориентированное, интегрированное, привязанное ко времени, неизменяемое собрание данных, применяемых для поддержки процессов принятия управленческих решений [3]. Предметная ориентация означает, что данные объединены в категории и хранятся в соответствии с теми областями, которые они описывают, а не с приложениями, которые их используют. В хранилище данные интегрируются в целях удовлетворения требований предприятия в целом, а не отдельной функции бизнеса. Привязанность данных ко времени выражает их «историчность», т.е. атрибут времени всегда явно присутствует в структурах хранилища данных. Неизменяемость означает, что, попав однажды в хранилище, данные уже не изменяются в отличие от оперативных систем, где данные присутствуют только в последней версии, поэтому постоянно меняются.

Технологии извлечения знаний из хранилищ данных основаны на методах статистического анализа и моделирования, ориентированных на поиск моделей и отношений, скрытых в совокупности данных. Эти модели могут в дальнейшем использоваться для оптимизации деятельности предприятия или фирмы.

Для извлечения значимой информации из хранилищ данных имеются специальные методы (OLAP-анализа, Data Mining или Knowledge Discovery), основанные на применении методов математической статистики, нейронных сетей, индуктивных методов построения деревьев решений и др. (см. главу 4).

Технология OLAP (On-Line Analytical Processing — оперативный анализ данных) предоставляет пользователю средства для формирования и проверки гипотез о свойствах данных или отношениях между ними на основе разнообразных запросов к базе данных. Они применяются на ранних стадиях процесса извлече-

ния знаний, помогая аналитику сфокусировать внимание на важных переменных. Средства Data Mining отличаются от OLAP тем, что кроме проверки предполагаемых зависимостей они способны самостоятельно (без участия пользователя) генерировать гипотезы о закономерностях, существующих в данных, и строить модели, позволяющие количественно оценить степень взаимного влияния исследуемых факторов на основе имеющейся информации.

1.2.4. АДАПТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Потребность в адаптивных информационных системах возникает в тех случаях, когда поддерживаемые ими проблемные области постоянно развиваются. В связи с этим адаптивные системы должны удовлетворять ряду специфических требований, а именно:

- адекватно отражать знания проблемной области в каждый момент времени;
- быть пригодными для легкой и быстрой реконструкции при изменении проблемной среды.

Адаптивные свойства информационных систем обеспечиваются за счет интеллектуализации их архитектуры. Ядром таких систем является постоянно развиваемая модель проблемной области, поддерживаемая в специальной базе знаний — репозитории. Ядро системы управляет процессами генерации или переконfigurирования программного обеспечения.

В процессе разработки адаптивных информационных систем применяется оригинальное или типовое проектирование. Оригинальное проектирование предполагает разработку информационной системы с «чистого листа» на основе сформулированных требований. Реализация этого подхода основана на использовании систем автоматизированного проектирования, или *CASE-технологий* (Designer2000, SilverRun, Natural Light Storm и др.).

При типовом проектировании осуществляется адаптация типовых разработок к особенностям проблемной области. Для реализации этого подхода применяются инструментальные средства *компонентного (сборочного) проектирования* информационных систем (R/3, BAAN IV, Prodis и др.).

Главное отличие подходов состоит в том, что при использовании CASE-технологии на основе репозитория при изменении проблемной области каждый раз выполняется *генерация* программного обеспечения, а при использовании сборочной технологии — *конфигурирование* программ и только в редких случаях — их *переработка*.

1.3. ТЕХНОЛОГИИ РАЗРАБОТКИ ЭКСПЕРТНЫХ СИСТЕМ

Технология создания интеллектуального программного обеспечения существенно отличается от разработки традиционных программ с использованием известных алгоритмических языков (табл. 1.1).

Рассмотрим отработанные на сегодняшний день элементы технологии создания ИИС на примере разработки экспертных систем. Этот выбор обусловлен тем, что ЭС получили весьма широкое распространение во многих сферах человеческой деятельности, а технологии их создания имеют универсальный характер и не требуют аппаратных реализаций.

Экспертными системами называют сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и тиражирующие этот эмпирический

Таблица 1.1

Отличия систем искусственного интеллекта от обычных программных систем

Характеристика	Программирование в системах искусственного интеллекта	Традиционное программирование
Тип обработки Метод Задание шагов решения Искомое решение Управление и данные Знания Модификации	Символьный Эвристический поиск Неявное Удовлетворительное Смешаны Неточные Частые	Числовой Точный алгоритм Явное Оптимальное Разделены Точные Редкие

опыт для консультаций менее квалифицированных пользователей [4].

В самых первых ЭС не учитывалось изменение знаний, используемых в процессе решения конкретной задачи. Их называли *статическими ЭС*. Типичная статическая ЭС содержит следующие основные компоненты:

- базу знаний;
- рабочую память, называемую также базой данных;
- решатель (интерпретатор);
- систему объяснений;
- компоненты приобретения знаний;
- интерфейс с пользователем.

База знаний ЭС предназначена для хранения долгосрочных данных, описывающих рассматриваемую область, и правил, описывающих целесообразные преобразования данных этой области.

База данных (рабочая память) служит для хранения текущих данных решаемой задачи.

Решатель (интерпретатор) формирует последовательность применения правил и осуществляет их обработку, используя данные из рабочей памяти и знания из БЗ.

Система объяснений показывает, каким образом система получила решение задачи и какие знания при этом использовались. Это облегчает тестирование системы и повышает доверие пользователя к полученному результату.

Компоненты приобретения знаний необходимы для заполнения ЭС знаниями в диалоге с пользователем-экспертом, а также для добавления и модификации заложенных в систему знаний.

К разработке ЭС привлекаются специалисты из разных предметных областей, а именно:

- эксперты той проблемной области, к которой относятся задачи, решаемые ЭС;
- инженеры по знаниям, являющиеся специалистами по разработке ИИС;
- программисты, осуществляющие реализацию ЭС.

Эксперты поставляют знания в ЭС и оценивают правильность получаемых результатов.

Инженеры по знаниям помогают экспертам выявить и структурировать знания, необходимые для работы ЭС, выполняют работу по представлению знаний, выбирают методы обработки зна-

ний, проводят выбор инструментальных средств для реализации ЭС, наиболее пригодных для решения поставленных задач.

Программисты разрабатывают программное обеспечение ЭС и осуществляют его сопряжение со средой, в которой оно будет использоваться.

Любая ЭС должна иметь, по крайней мере, два режима работы. В *режиме приобретения знаний* эксперт наполняет систему знаниями, которые впоследствии позволят ЭС самостоятельно (без помощи эксперта) решать определенные задачи из конкретной проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют объекты, их характеристики и значения, существующие в области экспертизы. Правила определяют взаимные связи, существующие между данными, и способы манипулирования данными, характерные для рассматриваемого класса задач.

В *режиме консультации* пользователь ЭС сообщает системе конкретные данные о решаемой задаче и стремится получить с ее помощью результат. Пользователи-неспециалисты обращаются к ЭС за результатом, не умея получить его самостоятельно, пользователи-специалисты используют ЭС для ускорения и облегчения процесса получения результата. Следует подчеркнуть, что термин «пользователь» является многозначным, так как использовать ЭС могут и эксперт, и инженер по знаниям, и программист. Поэтому, когда хотят подчеркнуть, что речь идет о том, для кого делалась ЭС, используют термин «конечный пользователь».

В режиме консультации входные данные о задаче поступают в рабочую память. Решатель на основе входных данных из рабочей памяти и правил из БЗ формирует решение. В отличие от традиционных программ компьютерной обработки данных ЭС при решении задачи не только исполняет предписанную последовательность операций, но и сама формирует ее.

Существует широкий класс приложений, в которых требуется учитывать изменения, происходящие в окружающем мире за время исполнения приложения. Для решения таких задач необходимо применять *динамические ЭС*, которые наряду с компонентами статических систем содержат подсистему моделирования внешнего мира и подсистему связи с внешним окружением. Подсистема моделирования внешнего мира необходима для прогнозирования, анализа и адекватной оценки состояния внешней среды. Изменения окружения решаемой задачи требуют изменения хра-

нимых в ЭС знаний, для того чтобы отразить временную логику происходящих в реальном мире событий. Компонента связи с внешним миром актуальна для автономных интеллектуальных систем (роботов), а также для интеллектуальных систем управления. Связь с внешним миром осуществляется через систему датчиков и контроллеров.

1.3.1. КЛАССИФИКАЦИОННЫЕ ПРИЗНАКИ ЭКСПЕРТНЫХ СИСТЕМ

В основе классификации экспертных систем (будем называть их приложениями) лежат следующие параметры:

- тип приложения;
- стадия существования;
- масштаб;
- тип проблемной среды;
- тип решаемой задачи.

Тип приложения. Характеризуется следующими признаками.

1. Возможность взаимодействия приложения с другими программными средствами:

• *изолированное приложение* — ЭС, не способная взаимодействовать с другими программными системами (например, с БД, электронными таблицами, пакетами прикладных программ, контроллерами, датчиками и т.п.);

• *интегрированное приложение* — ЭС и другие программные системы, с которыми она взаимодействует в ходе работы. Большинство современных ЭС, используемых для решения практически значимых задач, являются интегрированными.

2. Возможность исполнять приложение на разнородной аппаратуре и переносить его на различные платформы:

• *закрытые приложения* — исполняются только в программной среде данной фирмы и могут быть перенесены на другие платформы только путем перепрограммирования приложения;

• *открытые приложения* — ориентированы на исполнение в разнородном программно-аппаратном окружении и могут быть перенесены на другие платформы без перепрограммирования.

3. Архитектура приложения:

• *централизованное приложение* — реализуется на базе центральной ЭВМ, с которой связаны терминалы;

• *распределенное приложение* — обычно используется архитектура клиент-сервер.

Стадия существования. Характеризует степень завершенности разработки ЭС. Принято выделять следующие стадии:

• *исследовательский прототип* — решает представительный класс задач проблемной области, но может быть неустойчив в работе и не полностью проверен. При наличии развитых инструментальных средств для разработки исследовательского прототипа требуется примерно 2 — 4 месяца. База знаний исследовательского прототипа обычно содержит небольшое число исполняемых утверждений;

• *действующий прототип* — надежно решает любые задачи проблемной области, но при решении сложных задач может потребовать чрезмерно много времени и (или) памяти. Доведение системы от начала разработки до стадии действующего прототипа требует примерно 6 — 9 месяцев, при этом количество исполняемых утверждений в БЗ увеличивается по сравнению с исследовательским прототипом;

• *промышленная система* — обеспечивает высокое качество решения всех задач при минимуме времени и памяти. Обычно процесс преобразования действующего прототипа в промышленную систему состоит в расширении базы знаний и ее тщательной отладке. Доведение ЭС от начала разработки до стадии промышленной системы с применением развитых инструментальных средств требует не менее 12 — 18 месяцев;

• *коммерческая система* — пригодна не только для использования разработчиком, но и для продажи различным потребителям. Доведение системы до коммерческой стадии требует примерно 1,5 — 2 года. Приведенные здесь сроки справедливы для ЭС средней сложности.

Масштаб ЭС. Характеризует сложность решаемых задач и связан с типом используемой ЭВМ. По этому признаку различают:

• *малые ЭС* — предназначены для первичного обучения и исследования возможности применения технологии ЭС для рассматриваемого класса задач. Системы такого типа могут быть реализованы на персональных компьютерах;

• *средние ЭС* — охватывают весь спектр необходимых приложений и обычно интегрированы с БД, электронными таблицами и т.д. Системы такого масштаба чаще всего реализуются на рабочих станциях;

• *большие ЭС* — имеют доступ к высокомоощным БД и реализуются на рабочих станциях или на специализированных компьютерах;

• *символьные ЭС* — создаются с исследовательскими целями и реализуются на специализированных компьютерах, ориентированных на обработку символьных данных.

Тип проблемной среды. Это понятие включает описание предметной области (множество сущностей, описывающих множество объектов, их характеристик и отношений между объектами) и решаемых в ней задач. Иначе говоря, проблемная среда включает структуры данных и решаемые с ними задачи, представленные в виде исполняемых утверждений (правил, процедур, формул и т. п.). В связи с этим проблемная среда определяется характеристиками соответствующей предметной области и характеристиками типов решаемых в ней задач.

Характеристики предметной области

1. Тип предметной области:

• *статический* — входные данные не изменяются за время сеанса работы приложения, значения других (не входных) данных изменяются только самой экспертной системой;

• *динамический* — входные данные, поступающие из внешних источников, изменяются во времени, значения других данных изменяются ЭС или подсистемой моделирования внешнего окружения.

2. Способ описания сущностей предметной области:

• совокупность атрибутов и их значений (фиксированный состав сущностей);

• совокупность классов (объектов) и их экземпляров (изменяемый состав сущностей).

3. Способ организации сущностей в БЗ:

• неструктурированная БЗ;

• структурирование сущностей в БЗ по различным иерархиям («частное — общее», «часть — целое», «род — вид»), что обеспечивает наследование свойств сущностей.

Структурирование БЗ способствует:

• ограничению круга сущностей, которые должны рассматриваться механизмом вывода, и сокращению количества перебираемых вариантов в процессе выбора решения;

• обеспечению *наследования свойств сущностей*, т.е. передачи свойств вышерасположенных в иерархии сущностей нижерасположенным, что значительно упрощает процесс приобретения и использования знаний.

Характеристики задач

1. Тип решаемых задач:

• задачи анализа или синтеза. В *задаче анализа* задана модель сущности и требуется определить неизвестные характеристики модели. В *задаче синтеза* задаются условия, которым должны удовлетворять характеристики «неизвестной» модели сущности, и требуется построить модель этой сущности. Решение задачи синтеза обычно включает задачу анализа как составную часть;

• статические или динамические задачи. Если задачи, решаемые ЭС, явно не учитывают фактор времени и/или не изменяют в процессе своего решения знания об окружающем мире, то говорят, что ЭС решает *статические задачи*, в противном случае говорят о решении *динамических задач*. Учитывая значимость времени в динамических проблемных средах, многие специалисты называют их приложениями, работающими в реальном времени. Обычно выделяют следующие системы реального времени: *псевдореального* времени, *«мягкого»* реального времени и *«жесткого»* реального времени. Системы псевдореального времени, как следует из названия, не являются системами реального времени, однако они, в отличие от статических систем, получают и обрабатывают данные, поступающие из внешних источников. Системы псевдореального времени решают задачу быстрее, чем происходят значимые изменения информации об окружающем мире.

2. Общность исполняемых утверждений:

• частные исполняемые утверждения, содержащие ссылки на конкретные сущности (объекты);

• общие исполняемые утверждения, относящиеся к любым сущностям заданного типа (вне зависимости от их числа и имени). Использование общих утверждений позволяет значительно лаконичнее представлять знания. Однако поскольку общие утверждения не содержат явных ссылок на конкретные сущности, для их использования каждый раз требуется определять те сущности, к которым они должны применяться.

Не все сочетания перечисленных выше параметров, характеризующих проблемную среду, встречаются на практике. Наиболее распространены следующие типы проблемных сред:

- **статическая предметная область:**

представление сущностей в виде совокупности атрибутов и их значений, неизменяемый состав сущностей, БЗ не структурирована, решаются статические задачи анализа, используются только частные исполняемые утверждения;

представление сущностей объектами, изменяемый состав сущностей, БЗ структурирована, решаются статические задачи анализа и синтеза, используются общие и частные исполняемые утверждения;

- **динамическая предметная область:**

представление сущностей совокупностью атрибутов и их значений, неизменяемый состав сущностей, БЗ не структурирована, решаются динамические задачи анализа, используются частные исполняемые утверждения;

представление сущностей в виде объектов, изменяемый состав сущностей, БЗ структурирована, решаются динамические задачи анализа и синтеза, используются общие и частные исполняемые утверждения.

Тип решаемой задачи. По этому признаку различают следующие задачи:

- **интерпретация данных** — процесс определения смысла данных, результаты которого должны быть согласованными и корректными. Экспертные системы, как правило, проводят многовариантный анализ данных;

- **диагностика** — процесс соотнесения объекта с некоторым классом объектов и/или обнаружение неисправностей в системе (отклонений параметров системы от нормативных значений);

- **мониторинг** — непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы;

- **проектирование** — создание ранее не существовавшего объекта и подготовка спецификаций на создание объектов с заранее определенными свойствами. Степень новизны может быть разной и определяется видом знаний, заложенных в ЭС, и методами их обработки. Для организации эффективного проектирования и реинжиниринга требуется формировать не только сами проектные решения, но и мотивы их принятия. ЭС, решающие задачи

проектирования, реализуют процедуры вывода решения и объяснения полученных результатов;

- *прогнозирование* — предсказание последствий некоторых событий или явлений на основе анализа имеющихся данных. Прогнозирующие ЭС логически выводят вероятные следствия из заданных ситуаций. В прогнозирующих ЭС в большинстве случаев используются динамические модели, в которых значения параметров «подгоняются» под заданную ситуацию. Выводимые из этих моделей следствия составляют основу для прогнозов с вероятностными оценками;

- *планирование* — построение планов действий объектов, способных выполнять некоторые функции. Работа ЭС по планированию основана на моделях поведения реальных объектов, которые позволяют проводить логический вывод последствий планируемой деятельности;

- *обучение* — использование компьютера для обучения каким-либо дисциплине или предмету. Экспертные системы обучения выполняют такие функции, как диагностика ошибок, подсказывание правильных решений; аккумулирование знаний о гипотетическом «ученике» и его характерных ошибках; диагностирование слабости в познаниях обучаемых и нахождение соответствующих средств для их ликвидации. Системы обучения способны планировать акт общения с учеником в зависимости от успехов ученика для передачи необходимых знаний;

- *управление* — функция организованной системы, поддерживающая определенный режим ее деятельности. Экспертные системы данного типа предназначены для управления поведением сложных систем в соответствии с заданными спецификациями;

- *поддержка принятия решений* — совокупность процедур, обеспечивающая лицо, принимающее решения, необходимой информацией и рекомендациями, облегчающими процесс принятия решения. Такого рода ЭС оказывают помощь специалистам в выборе и/или генерации наиболее рациональной альтернативы из множества возможных при принятии ответственных решений.

Задачи интерпретации данных, диагностики, поддержки принятия решений относятся к задачам анализа, задачи проектирования, планирования и управления — к задачам синтеза. К комбинированному типу задач относятся обучение, мониторинг и прогнозирование. Примеры ЭС, решающих различные прикладные задачи, можно найти в [4, 11, 15].

1.3.2. ХАРАКТЕРИСТИКА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

Трудоемкость разработки ИИС в значительной степени зависит от используемых инструментальных средств. Инструментальные средства для разработки интеллектуальных приложений можно классифицировать по следующим основным параметрам:

- уровень используемого языка;
- парадигмы программирования и механизмы реализации;
- способ представления знаний;
- механизмы вывода и моделирования;
- средства приобретения знаний;
- технологии разработки приложений.

Уровень используемого языка. Мощность и универсальность языка программирования определяет трудоемкость разработки ЭС.

1. Традиционные (в том числе объектно-ориентированные) языки программирования типа С, С++ (как правило, они используются не для создания ЭС, а для создания инструментальных средств).

2. Специальные языки программирования (например, язык LISP, ориентированный на обработку списков; язык логического программирования PROLOG; язык рекурсивных функций РЕФАЛ и т.д.). Их недостатком является слабая приспособленность к объединению с программами, написанными на языках традиционного программирования.

3. Инструментальные средства, содержащие многие, но не все компоненты ЭС (например, система OPS 5, которая поддерживает продукционный подход к представлению знаний; языки KRL и FRL, используемые для разработки ЭС с фреймовым представлением знаний). Такое программное обеспечение предназначено для разработчиков, владеющих технологиями программирования и умеющих интегрировать разнородные компоненты в программный комплекс.

4. Оболочки ЭС общего назначения, содержащие все программные компоненты, но не имеющие знаний о конкретных предметных средах. Средства этого типа и последующего не требуют от разработчика приложения знания программирования. Примерами являются ЭКО, Leonardo, Nexpert Object, Каппа,

EXSYS, GURU, ART, KEE и др. В последнее время все реже употребляется термин «оболочка», его заменяют более широким термином «среда разработки». Если хотят подчеркнуть, что средство используется не только на стадии разработки приложения, но и на стадиях использования и сопровождения, то употребляют термин «полная среда» (complete environment). Для поддержания всего цикла создания и сопровождения программ используются интегрированные инструментальные системы типа Work Bench, например KEATS [18], Shelly [16], VITAL [19]. Основными компонентами системы KEATS являются: ACQUIST — средства фрагментирования текстовых источников знаний, позволяющие разбивать текст или протокол беседы с экспертом на множество взаимосвязанных, аннотированных фрагментов и создавать понятия (концепты); FLIK — язык представления знаний средствами фреймовой модели; GIS — графический интерфейс, используемый для создания гипертекстов и концептуальных моделей, а также для проектирования фреймовых систем; ERI — интерпретатор правил, реализующий процедуры прямого и обратного вывода; TRI — инструмент визуализации логического вывода, демонстрирующий последовательность выполнения правил; Tables — интерфейс манипулирования таблицами, используемый для хранения знаний в БЗ; CS — язык описания и распространения ограничений; TMS — немонотонная система поддержания истинности.

При использовании инструментария данного типа могут возникнуть следующие трудности:

а) управляющие стратегии, заложенные в механизм вывода, могут не соответствовать методам решения, которые использует эксперт, взаимодействующий с данной системой, что может привести к неэффективным, а возможно, и неправильным решениям;

б) способ представления знаний, используемый в инструментарии, мало подходит для описания знаний конкретной предметной области.

Большая часть этих трудностей разрешена в проблемно/предметно-ориентированных средствах разработки ИИС.

5. Проблемно/предметно-ориентированные оболочки и среды (не требуют знания программирования):

• проблемно-ориентированные средства — предназначены для решения задач определенного класса (задачи поиска, управ-

ления, планирования, прогнозирования и др.) и содержат соответствующие этому классу альтернативные функциональные модули;

- предметно-ориентированные средства — включают знания о типах предметных областей, что сокращает время разработки БЗ.

При использовании оболочек и сред разработчик приложения полностью освобождается от программирования, его основные трудозатраты связаны с формированием базы знаний.

Парадигмы программирования и механизмы реализации. Способы реализации механизма исполняемых утверждений часто называют *парадигмами программирования*. К основным парадигмам относят следующие:

- процедурное программирование;
- программирование, ориентированное на данные;
- программирование, ориентированное на правила;
- объектно-ориентированное программирование.

Парадигма *процедурного программирования* является самой распространенной среди существующих языков программирования (например, С и Паскаль). В процедурной парадигме активная роль отводится процедурам, а не данным; причем любая процедура активизируется вызовом. Подобные способы задания поведения удобны для описаний детерминированной последовательности действий одного процесса или нескольких взаимосвязанных процессов.

При использовании *программирования, ориентированного на данные*, активная роль принадлежит данным, а не процедурам. Здесь со структурами активных данных связывают некоторые действия (процедуры), которые активизируются тогда, когда осуществляется обращение к этим данным.

В парадигме, *ориентированной на правила*, поведение определяется множеством правил вида «условие-действие». Условие задает образ данных, при возникновении которого действие правила может быть выполнено. Правила в данной парадигме играют такую же роль, как и операторы в процедурной парадигме. Однако если в процедурной парадигме поведение задается детерминированной последовательностью операторов, не зависящей от значений обрабатываемых данных, то в парадигме, ориентированной на правила, поведение не задается заранее предписанной последовательностью правил, а формируется на основе значений данных, которые в текущий момент обрабатываются програм-

мой. Подход, ориентированный на правила, удобен для описания поведения, гибко и разнообразно реагирующего на большое многообразие состояний данных.

Парадигма *объектного программирования* в отличие от процедурной парадигмы не разделяет программу на процедуры и данные. Здесь программа организуется вокруг сущностей, называемых объектами, которые включают локальные процедуры (методы) и локальные данные (переменные). Поведение (функционирование) в этой парадигме организуется путем пересылки сообщений между объектами. Объект, получив сообщение, осуществляет его локальную интерпретацию, основываясь на локальных процедурах и данных. Такой подход позволяет описывать сложные системы наиболее естественным образом. Он особенно удобен для интегрированных ЭС.

Способ представления знаний. Наличие многих способов представления знаний вызвано стремлением представить различные типы проблемных сред с наибольшей эффективностью. Обычно способ представления знаний в ЭС характеризуют моделью представления знаний. Типичными моделями представления знаний являются правила (продукции), фреймы (или объекты), семантические сети, логические формулы (см. главу 2). Инструментальные средства, имеющие в своем составе более одной модели представления знаний, называют гибридными. Большинство современных средств, как правило, использует объектно-ориентированную парадигму, объединенную с парадигмой, ориентированной на правила.

Механизмы вывода и моделирования. В статических ЭС единственным активным агентом, изменяющим информацию, является механизм вывода экспертной системы. В динамических ЭС изменение данных происходит не только вследствие функционирования механизма исполняемых утверждений, но также в связи с изменениями окружения задачи, которые моделируются специальной подсистемой или поступают извне. Механизмы вывода в различных средах могут отличаться способами реализации следующих процедур.

1. Структура процесса получения решения:

- построение дерева вывода на основе обучающей выборки (индуктивные методы приобретения знаний) и выбор маршрута на дереве вывода в режиме решения задачи;

- компиляция сети вывода из специфических правил в режиме приобретения знаний и поиск решения на сети вывода в режиме решения задачи;

- генерация сети вывода и поиск решения в режиме решения задачи, при этом генерация сети вывода осуществляется в ходе выполнения операции сопоставления, определяющей пары «правило — совокупность данных», на которых условия этого правила удовлетворяются;

- в режиме решения задач ЭС осуществляет выработку правдоподобных предположений (при отсутствии достаточной информации для решения); выполнение рассуждений по обоснованию (опровержению) предположений; генерацию альтернативных сетей вывода; поиск решения в сетях вывода.

2. Поиск (выбор) решения:

- направление поиска — от данных к цели, от целей к данным, двунаправленный поиск;

- порядок перебора вершин в сети вывода — «поиск в ширину», при котором сначала обрабатываются все вершины, непосредственно связанные с текущей обрабатываемой вершиной G ; «поиск в глубину», когда сначала раскрывается одна наиболее значимая вершина — G_1 , связанная с текущей G , затем вершина G_1 делается текущей, и для нее раскрывается одна наиболее значимая вершина G_2 и т. д.

3. Процесс генерации предположений и сети вывода:

- режим — генерация в режиме приобретения знаний, генерация в режиме решения задачи;

- полнота генерируемой сети вывода — операция сопоставления применяется ко всем правилам и ко всем типам указанных в правилах сущностей в каждом цикле работы механизма вывода (обеспечивается полнота генерируемой сети); используются различные средства для сокращения количества правил и (или) сущностей, участвующих в операции сопоставления; например, применяется алгоритм сопоставления или используются знания более общего характера (метазнания).

Механизм вывода для динамических проблемных сред дополнительно содержит: планировщик, управляющий деятельностью ЭС в соответствии с приоритетами; средства, гарантирующие получение лучшего решения в условиях ограниченности ресурсов; систему поддержания истинности значений переменных, изменяющихся во времени.

В динамических инструментальных средствах могут быть реализованы следующие варианты подсистемы моделирования:

- система моделирования отсутствует;
- существует система моделирования общего назначения, являющаяся частью инструментальной среды;
- существует специализированная система моделирования, являющаяся внешней по отношению к программному обеспечению, на котором реализуется ЭС.

Средства приобретения знаний. В инструментальных системах они характеризуются следующими признаками:

1. Уровень языка приобретения знаний:
 - формальный язык;
 - ограниченный естественный язык;
 - язык пиктограмм и изображений;
 - ЕЯ и язык изображений.
2. Тип приобретаемых знаний:
 - данные в виде таблиц, содержащих значения входных и выходных атрибутов, по которым индуктивными методами строится дерево вывода;
 - специализированные правила;
 - общие и специализированные правила.
3. Тип приобретаемых данных:
 - атрибуты и значения;
 - объекты;
 - классы структурированных объектов и их экземпляры, получающие значения атрибутов путем наследования.

1.3.3.

ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ЭКСПЕРТНЫХ СИСТЕМ

Промышленная технология создания интеллектуальных систем включает следующие этапы:

- исследование выполнимости проекта;
- разработку общей концепции системы;
- разработку и тестирование серии прототипов;
- разработку и испытание головного образца;
- разработку и проверку расширенных версий системы;
- привязку системы к реальной рабочей среде.

Проектирование ЭС основано на трех главных принципах:

1. Мощность экспертной системы обусловлена прежде всего мощностью БЗ и возможностями ее пополнения и только затем — используемыми методами (процедурами) обработки информации.

2. Знания, позволяющие эксперту (или экспертной системе) получить качественные и эффективные решения задач, являются в основном эвристическими, эмпирическими, неопределенными, правдоподобными.

3. Неформальный характер решаемых задач и используемых знаний делает необходимым обеспечение активного диалога пользователя с ЭС в процессе ее работы.

Перед тем как приступить к разработке ЭС, инженер по знаниям должен рассмотреть вопрос, следует ли разрабатывать ЭС для данного приложения. Положительное решение принимается тогда, когда разработка ЭС *возможна, оправданна* и методы инженерии знаний *соответствуют* решаемой задаче.

Чтобы разработка ЭС была *возможной* для данного приложения, необходимо выполнение, по крайней мере, следующих требований:

- существуют эксперты в данной области, которые решают задачу значительно лучше, чем начинающие специалисты;

- эксперты сходятся в оценке предлагаемого решения, так как в противном случае будет невозможно оценить качество разработанной ЭС;

- эксперты способны вербализовать (выразить на естественном языке) и объяснить используемые ими методы, иначе трудно рассчитывать на то, что знания экспертов будут «извлечены» и заложены в ЭС;

- решение задачи требует только рассуждений, а не действий;

- задача не должна быть слишком трудной (т.е. ее решение должно занимать у эксперта несколько часов или дней, а не недель или лет);

- задача хотя и не должна быть выражена в формальном виде, но все же должна относиться к достаточно «понятной» и структурированной области, т.е. должна существовать возможность выделения основных понятий, отношений и способов получения решения задачи;

• решение задачи не должно в значительной степени опираться на «здравый смысл» (т.е. широкий спектр общих сведений о мире и о способе его функционирования, которые знает и умеет использовать любой нормальный человек), так как подобные знания пока не удастся в достаточном количестве заложить в системы искусственного интеллекта.

Приложение *соответствует* методам ЭС, если решаемая задача обладает совокупностью следующих характеристик:

• задача может быть естественным образом решена посредством манипулирования символами (с помощью символических рассуждений), а не манипулирования числами, как принято в математических методах и в традиционном программировании;

• задача должна иметь эвристическую, а не алгоритмическую природу, т. е. ее решение должно требовать применения эвристических правил. Для задач, которые могут быть гарантированно решены (при соблюдении заданных ограничений) с помощью формальных процедур, существуют более эффективные подходы, чем технологии ЭС.

При разработке ЭС, как правило, используется концепция *быстрого прототипа*, суть которой заключается в том, что разработчики не пытаются сразу построить конечный продукт. На начальном этапе они создают прототип (возможно, не единственный) ЭС, удовлетворяющий двум противоречивым требованиям: умение решать типичные задачи конкретного приложения и незначительные время и трудоемкость его разработки. При выполнении этих условий становится возможным параллельно вести процесс накопления и отладки знаний, осуществляемый экспертом, и процесс выбора (разработки) программных средств, выполняемый инженером по знаниям и программистами. Для удовлетворения указанным требованиям при создании прототипа используются разнообразные инструментальные средства, ускоряющие процесс проектирования.

Традиционная технология реализации ЭС включает шесть основных этапов (рис. 1.4): идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию [11].

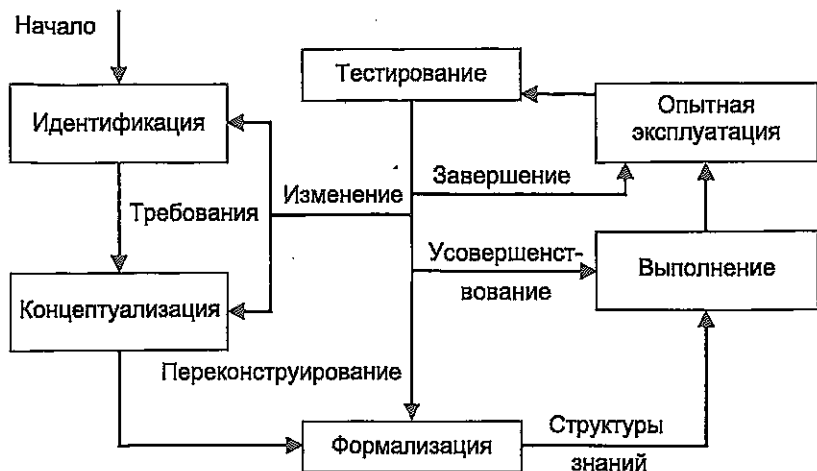


Рис. 1.4. Этапы разработки экспертных систем

На этапе *идентификации* определяются задачи, подлежащие решению, цели разработки, эксперты и типы пользователей.

На этапе *концептуализации* проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

На этапе *формализации* выбираются инструментальные средства и способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, моделируется работа системы, оценивается адекватность системы зафиксированных понятий, методов решения, средств представления и манипулирования знаниями рассматриваемой предметной области.

На этапе *выполнения* осуществляется заполнение базы знаний. В связи с тем, что основой ЭС являются знания, данный этап является одним из самых важных и самых трудоемких. Процесс приобретения знаний разделяют на извлечение знаний в диалоге с экспертами; организацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, «понятном» ЭС. Процесс приобретения знаний осуществляется инженером по знаниям на основе анализа деятельности эксперта по решению реальных задач. Проблемы приобретения знаний подробно описаны в главе 4.

На этапе *тестирования* эксперт и инженер по знаниям в интерактивном режиме с использованием диалоговых и объяснительных средств проверяют компетентность ЭС. Процесс тестирования продолжается до тех пор, пока эксперт не решит, что система достигла требуемого уровня компетентности.

На этапе *опытной эксплуатации* проверяется пригодность ЭС для конечных пользователей. Полученные результаты могут показать необходимость существенной модификации ЭС.

Процесс создания ЭС не сводится к строгой последовательности перечисленных выше этапов. В ходе разработки приходится неоднократно возвращаться на более ранние этапы и пересматривать принятые там решения.

Инструментальные средства различаются в зависимости от того, какую технологию разработки ЭС они допускают. Можно выделить, по крайней мере, четыре подхода к разработке ЭС:

- подход, базирующийся на поверхностных знаниях;
- структурный подход;
- подход, основанный на глубинных знаниях;
- смешанный подход, опирающийся на использование поверхностных и глубинных знаний.

Поверхностный подход применяется для сложных задач, которые не могут быть точно описаны. Его сущность состоит в получении от экспертов фрагментов знаний, релевантных решаемой задаче. При этом не предпринимается попыток систематического или глубинного изучения области, что предопределяет использование поиска в пространстве состояний в качестве универсального механизма вывода. Обычно в ЭС, использующих данный подход, в качестве способа представления знаний выбираются правила. Условие каждого правила определяет образец некоторой ситуации, в которой правило может быть выполнено. Поиск решения состоит в выполнении тех правил, образцы которых сопоставляются с текущими данными. При этом предполагается, что в процессе поиска решения последовательность формируемых таким образом ситуаций не оборвется до получения решения, т.е. не возникнет неизвестной ситуации, которая не соответствует ни одному правилу. Данный подход с успехом применяется к широкому классу приложений, но оказывается неэффективным в тех случаях, когда задача может структурироваться или для ее решения может использоваться некоторая модель.

Структурный подход к построению ЭС предусматривает структуризацию знаний проблемной области. Его появление обусловлено тем, что для ряда приложений применение техники поверхностных знаний не обеспечивает решения задачи. Структурный подход к построению ЭС во многом похож на структурное программирование. Однако применительно к ЭС речь идет не о том, чтобы структурирование задачи было доведено до точного алгоритма (как в традиционном программировании), а предполагается, что часть задачи решается с помощью эвристического поиска. Структурный подход в различных приложениях целесообразно сочетать с поверхностным или глубинным.

При *глубинном подходе* компетентность ЭС базируется на модели той проблемной среды, в которой она работает. Модель может быть определена различными способами (декларативно, процедурно). Необходимость использования моделей в ряде приложений вызвана стремлением исправить недостаток поверхностного подхода, связанный с возникновением ситуаций, не описанных правилами, хранящимися в БЗ. Экспертные системы, разработанные с применением глубинных знаний, при возникновении неизвестной ситуации способны самостоятельно определить, какие действия следует выполнить, с помощью некоторых общих принципов, справедливых для данной области экспертизы.

Глубинный подход требует явного описания структуры и взаимоотношений между различными сущностями проблемной области. В этом подходе необходимо использовать инструментальные средства, обладающие возможностями моделирования: объекты с присоединенными процедурами, иерархическое наследование свойств, активные знания (программирование, управляемое данными), механизм передачи сообщений объектам (объектно-ориентированное программирование) и т. п.

Смешанный подход в общем случае может сочетать поверхностный, структурный и глубинный подходы. Например, поверхностный подход может применяться для поиска адекватных знаний, которые затем используются некоторой глубинной моделью.

1. Амамия М., Танака Ю. Архитектура ЭВМ и ИИ. — М.: Мир, 1993.
2. Андрейчиков А. В., Андрейчикова О. Н. Компьютерная поддержка изобретательства (методы, системы, примеры, применения). — М.: Машиностроение, 1998.
3. Буров К. Обнаружение знаний в хранилищах данных // Открытые системы. — 1999. — № 5 — 6.
4. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. — СПб.: Питер, 2000.
5. Заринов Р. Х. Машинный поиск вариантов при моделировании творческого процесса. — М.: Наука, 1983.
6. Искусственный интеллект: Справочник. В 3-х кн. Кн. 1. Системы общения и экспертные системы/Под ред. Э.В. Попова. — М.: Радио и связь, 1990.
7. Искусственный интеллект: Справочник. В 3-х кн. Кн. 2. Модели и методы/Под ред. Д.А. Поспелова. — М.: Радио и связь, 1990.
8. Искусственный интеллект: Справочник. В 3-х кн. Кн. 3. Программные и аппаратные средства/Под ред. В.А. Захарова, В.Ф. Хорошевского. — М.: Радио и связь, 1990.
9. Ларичев О. И. Системы, основанные на экспертных знаниях: история, современное состояние и некоторые перспективы: Сб. науч. тр. Седьмой национальной конференции по искусственному интеллекту с международным участием. — М.: Изд-во физико-математической литературы, 2000.
10. Масалович А. И. От нейрона к компьютеру // Журнал доктора Добба. — 1992. — № 1.
11. Статические и динамические экспертные системы: Учеб. пособие / Э. В. Попов, И. Б. Фоминых, Е. Б. Кисель, М. Д. Шапот. — М.: Финансы и статистика, 1996.
12. Тельнов Ю. Ф. Интеллектуальные информационные системы в экономике: Учеб. пособие. — М.: СИНТЕГ, 1998.
13. Финн В. К. Правдоподобные рассуждения в интеллектуальных системах типа ДСМ // Итоги науки и техники. Сер. «Информатика». Т. 15. «Интеллектуальные информационные системы». — М.: ВИНТИ, 1991.
14. Цуриков В. М. Проект «Изобретающая машина» — интеллектуальная среда поддержки инженерной деятельности // Журнал ТРИЗ. — 1991. — № 21.

15. *Элти Дж., Кумбс М.* Экспертные системы: концепции и примеры: Пер. с англ. — М.: Финансы и статистика, 1987.
16. *Bouchet C., Brunet C., Anjewierden A.* SHELLY: An integrated workbench for KBS development // Proc. of 9th Int. Workshop Expert Syst. and their Appl. — France, Avignon, 1989. — No. 1.
17. *Durkin J.* Expert Systems: a view of the field // IEEE Expert. — 1996. — No. 2.
18. *Motta E., Rajan T., Dominique J., Eisenstadt M.* Methodological foundations of KEATS, the knowledge Engineer's Assistant // Knowledge Acquisition. — 1991. — No. 3.
19. VITAL: A methodology — based workbench for KBS life cycle Support // ESPRIT — II Project 5365, 1990.

Контрольные вопросы и задания

1. Охарактеризуйте основные направления исследований, проводимые в области искусственного интеллекта. Приведите известные вам примеры интеллектуальных систем.
2. Назовите основные функции, присущие ИИС. На чем основана их реализация?
3. Дайте краткую характеристику систем с интеллектуальным интерфейсом, экспертных систем, самообучающихся систем и адаптивных информационных систем.
4. Сформулируйте основные отличия систем искусственного интеллекта от обычных программных средств.
5. Перечислите и охарактеризуйте основные компоненты статических экспертных систем.
6. Какого профиля специалисты привлекаются для разработки экспертных систем? Каковы их функции?
7. Чем отличаются динамические экспертные системы от статических?
8. Охарактеризуйте экспертную систему по следующим параметрам: типу приложения, стадии существования, масштабу, типу проблемной среды, типу решаемой задачи.
9. Расскажите об основных характеристиках инструментальных средств, предназначенных для разработки интеллектуальных информационных систем (уровень используемого языка, парадигма программирования; способ представления знаний, механизм вывода и моделирования, средства приобретения знаний, технологии разработки приложений).

10. Опишите основные технологические этапы разработки экспертных систем: идентификацию, концептуализацию, формализацию, выполнение, тестирование, опытную эксплуатацию.
11. Расскажите о принципах самообучения, которые применяются в ИИС.
12. Какими способами осуществляются заполнение и модификация баз знаний ИИС?
13. Какие типы задач решаются с применением ЭС? Приведите примеры.
14. Подготовьте сообщение или реферат на тему «Применение интеллектуальных систем в экономике и бизнесе (другой проблемной области)».
15. Подготовьте сообщение или реферат, посвященный конкретной экспертной системе, используя для ее характеристики признаки, которые приведены в данной главе.

ТРАДИЦИОННЫЕ СПОСОБЫ ПРЕДСТАВЛЕНИЯ И ОБРАБОТКИ ЗНАНИЙ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

Я должен ... рассмотреть тысячи различных вещей для того, чтобы способ, с помощью которого я мог бы выразить истину, не поражал бы ничего воображения и не противоречил общепринятым мнениям.

Рене Декарт

Необходимой частью любой интеллектуальной системы являются знания. Теоретическими и практическими вопросами представления и обработки знаний в компьютерных системах активно занимаются исследователи, работающие в области *инженерии знаний*. Это понятие в 1977 г. ввел Э. Фейгенбаум, который писал: «По опыту нам известно, что большая часть знаний в конкретной предметной области остается личной собственностью эксперта. И это происходит не потому, что он не хочет разглашать своих секретов, а потому, что он не в состоянии сделать этого — ведь эксперт знает гораздо больше, чем сам осознает». Данное направление ИИ связано с развитием теоретических и прикладных аспектов приобретения и формализации знаний специалистов, с проектированием и разработкой баз знаний.

2.1. ОТЛИЧИЯ ЗНАНИЙ ОТ ДАННЫХ

Характерным признаком интеллектуальных систем является наличие знаний, необходимых для решения задач конкретной предметной области. При этом возникает естественный вопрос, что такое знания и чем они отличаются от обычных данных, обрабатываемых ЭВМ.

Данными называют информацию фактического характера, описывающую объекты, процессы и явления предметной области, а также их свойства. В процессах компьютерной обработки данные проходят следующие этапы преобразований:

- исходная форма существования данных (результаты наблюдений и измерений, таблицы, справочники, диаграммы, графики и т.д.);

- представление на специальных языках описания данных, предназначенных для ввода и обработки исходных данных в ЭВМ;

- базы данных на машинных носителях информации.

Знания являются более сложной категорией информации по сравнению с данными. Знания описывают не только отдельные факты, но и взаимосвязи между ними, поэтому знания иногда называют структурированными данными. Знания могут быть получены на основе обработки эмпирических данных. Они представляют собой результат мыслительной деятельности человека, направленной на обобщение его опыта, полученного в результате практической деятельности.

Для того чтобы наделить ИИС знаниями, их необходимо представить в определенной форме. Существуют два основных способа наделения знаниями программных систем. *Первый* — поместить знания в программу, написанную на обычном языке программирования. Такая система будет представлять собой единый программный код, в котором знания не вынесены в отдельную категорию. Несмотря на то что основная задача будет решена, в этом случае трудно оценить роль знаний и понять, каким образом они используются в процессе решения задач. Нелегким делом являются модификация и сопровождение подобных программ, а проблема пополнения знаний может стать неразрешимой.

Второй способ базируется на концепции баз данных и заключается в вынесении знаний в отдельную категорию, т.е. знания представляются в определенном формате и помещаются в БЗ. База знаний легко пополняется и модифицируется. Она является автономной частью интеллектуальной системы, хотя механизм логического вывода, реализованный в логическом блоке, а также средства ведения диалога накладывают определенные ограничения на структуру БЗ и операции с нею. В современных ИИС принят этот способ.

Следует заметить, что для того, чтобы поместить знания в компьютер, их необходимо представить определенными структу-

рами данных, соответствующих выбранной среде разработки интеллектуальной системы. Следовательно, при разработке ИИС сначала осуществляются накопление и представление знаний, причем на этом этапе обязательно участие человека, а затем знания представляются определенными структурами данных, удобными для хранения и обработки в ЭВМ. Знания в ИИС существуют в следующих формах:

- исходные знания (правила, выведенные на основе практического опыта, математические и эмпирические зависимости, отражающие взаимные связи между фактами; закономерности и тенденции, описывающие изменение фактов с течением времени; функции, диаграммы, графы и т. д.);

- описание исходных знаний средствами выбранной модели представления знаний (множество логических формул или продукционных правил, семантическая сеть, иерархии фреймов и т. п.);

- представление знаний структурами данных, которые предназначены для хранения и обработки в ЭВМ;

- базы знаний на машинных носителях информации.

Что же такое знания? Приведем несколько определений.

Из толкового словаря С. И. Ожегова: 1) «Знание — постижение действительности сознанием, наука»; 2) «Знание — это совокупность сведений, познаний в какой-либо области».

Из японского толкового словаря: «Знания — это результат, полученный познанием», или, более подробно, «система суждений с принципиальной и единой организацией, основанная на объективной закономерности».

Исследователями в области ИИ даются более конкретные определения знаний.

«Знания — это закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области» [1].

«Знания — это хорошо структурированные данные или данные о данных, или метаданные» [1].

«Знания — формализованная информация, на которую ссылаются или используют в процессе логического вывода» [8].

Существует множество классификаций знаний. Как правило, с помощью классификаций систематизируют знания конкретных предметных областей. На абстрактном уровне рассмотрения

можно говорить о признаках, по которым подразделяются знания, а не о классификациях. По своей природе знания можно разделить на декларативные и процедурные.

Декларативные знания представляют собой описания фактов и явлений, фиксируют наличие или отсутствие таких фактов, а также включают описания основных связей и закономерностей, в которые эти факты и явления входят.

Процедурные знания — это описания действий, которые возможны при манипулировании фактами и явлениями для достижения намеченных целей.

Для описания знаний на абстрактном уровне разработаны специальные языки — языки описания знаний. Эти языки также делятся на языки процедурного типа и декларативного. Все языки описания знаний, ориентированные на использование традиционных компьютеров фон-неймановской архитектуры, являются языками процедурного типа. Разработка языков декларативного типа, удобных для представления знаний, является актуальной проблемой сегодняшнего дня.

По способу приобретения знания можно разделить на *факты* и *эвристику* (правила, которые позволяют сделать выбор при отсутствии точных теоретических обоснований). Первая категория знаний обычно указывает на хорошо известные в данной предметной области обстоятельства. Вторая категория знаний основана на собственном опыте эксперта, работающего в конкретной предметной области, накопленном в результате многолетней практики.

По типу представления знания делятся на *факты* и *правила*. Факты — это знания типа «*А — это А*», такие знания характерны для баз данных и сетевых моделей. Правила, или продукции, — это знания типа «*ЕСЛИ А, ТО В*».

Кроме фактов и правил существуют еще *метазнания* — знания о знаниях. Они необходимы для управления БЗ и для эффективной организации процедур логического вывода.

Форма представления знаний оказывает существенное влияние на характеристики ИИС. Базы знаний являются моделями человеческих знаний. Однако все знания, которые привлекает человек в процессе решения сложных задач, смоделировать невозможно. Поэтому в интеллектуальных системах требуется четко разделить знания на те, которые предназначены для обработки компьютером, и знания, используемые человеком. Очевидно,

что для решения сложных задач БЗ должна иметь достаточно большой объем, в связи с чем неизбежно возникают проблемы управления такой базой. Поэтому при выборе модели представления знаний следует учитывать такие факторы, как однородность представления и простота понимания. Однородность представления приводит к упрощению механизма управления знаниями. Простота понимания важна для пользователей интеллектуальных систем и экспертов, чьи знания закладываются в ИИС. Если форма представления знаний будет трудна для понимания, то усложняются процессы приобретения и интерпретации знаний. Следует заметить, что одновременно выполнить эти требования довольно сложно, особенно в больших системах, где неизбежным становится структурирование и модульное представление знаний.

2.2. ТИПИЧНЫЕ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

К типичным моделям представления знаний относятся логическая, продукционная, фреймовая и модель семантической сети.

Каждой модели отвечает свой язык представления знаний. Однако на практике редко удастся обойтись рамками одной модели при разработке ИИС за исключением самых простых случаев, поэтому представление знаний получается сложным. Кроме комбинированного представления с помощью различных моделей, обычно используются специальные средства, позволяющие отразить особенности конкретных знаний о предметной области, а также различные способы устранения и учета нечеткости и неполноты знаний.

2.2.1. ЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Логическая модель основана на системе исчисления предикатов первого порядка. Знакомство с логикой предикатов начнем с исчисления высказываний.

Высказыванием называется предложение, смысл которого можно выразить значениями: истина (Т) или ложь (F). Напри-

мер, предложения «лебедь белый» и «лебедь черный» будут высказываниями. Из простых высказываний можно составить более сложные:

- «лебедь белый или лебедь черный»,
- «лебедь белый и лебедь черный»,
- «если лебедь не белый, то лебедь черный».

В свою очередь, сложные высказывания можно разделить на частичные, которые связаны между собой с помощью слов: и, или, не, если — то. Элементарными называются высказывания, которые нельзя разделить на части. Логика высказываний оперирует логическими связями между высказываниями, т. е. она решает вопросы типа: «Можно ли на основе высказывания A получить высказывание B ?»; «Истинно ли B при истинности A ?» и т.п. При этом семантика высказываний не имеет значения. Элементарные высказывания рассматриваются как переменные логического типа, над которыми разрешены следующие логические операции:

- $\neg$ отрицание (унарная операция);
- $\wedge$ конъюнкция (логическое умножение);
- $\vee$ дизъюнкция (логическое сложение);
- $\rightarrow$ импликация (если — то);
- $\leftrightarrow$ эквивалентность.

Операция импликации должна удовлетворять следующим требованиям.

1. Значение результата импликации зависит от двух операндов.
2. Если первый операнд (A) — истинный, то значение результата совпадает со значением второго операнда (B).
3. Операция импликации не коммутативна.
4. Результат импликации совпадает с результатом выражения $\neg A \vee B$.

Значения результатов логических операций над переменными A и B , являющимися элементарными высказываниями, приведены в табл. 2.1.

Исчисление высказываний позволяет формализовать лишь малую часть множества рассуждений, поскольку этот аппарат не позволяет учитывать внутреннюю структуру высказывания, ко-

Результаты вычисления логических операций

A	$\neg A$	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
Т	Ф	Т	Т	Т	Т	Т
Т	Ф	Ф	Ф	Т	Ф	Ф
Ф	Т	Т	Ф	Т	Т	Ф
Ф	Т	Ф	Ф	Ф	Т	Т

торая существует в естественных языках. Рассмотрим ставший классическим пример рассуждения о Сократе:

P : «Все люди смертны»

Q : «Сократ — человек»

R : «Сократ — смертен»

Используя для обозначения высказываний логические переменные P , Q , R , можно составить формулу: $(P \wedge Q) \rightarrow R$, которая может быть интерпретирована как «Если все люди смертны и Сократ является человеком, то Сократ является смертным». Однако эта формула не является общезначимой, поскольку относится только к одному объекту (Сократу). Кроме того, высказывание R не выводится из P и Q , т.е., если бы мы не сформулировали R заранее, мы не смогли бы записать приведенную выше формулу.

Чтобы осуществить этот примитивный логический вывод, высказывание Q следует разделить на две части: «Сократ» (субъект) и «человек» (свойство субъекта) и представить в виде отношения «субъект — свойство», которое можно записать с помощью функции *человек* (Сократ).

Очевидно, что свойство конкретного субъекта с именем «Сократ» быть «человеком» может быть присуще и ряду других субъектов, что позволяет заменить константу «Сократ» на некоторую переменную, например X . Тогда получим запись *человек* (X), которая обладает внутренней структурой, т.е. значение такого высказывания будет зависеть от его компонент. Записанная функция уже не является элементарным высказыванием, она называется предикатом.

Приведем объяснение понятия предиката, данное Д. А. Поспеловым [6]: «Под предикатом будем понимать некоторую связь, которая задана на наборе из констант или переменных.

Пример предиката: « P больше Q ».

Если семантика P и Q не задана, то о предикате сказать особенно нечего. Пожалуй, только то, что он является антирефлексивным, антисимметричным и транзитивным. Но при задании семантики (т.е. областей определения переменных P и Q) о предикате можно будет сказать существенно больше. Например, если P и Q — площади городов в России и Японии, то при задании списков городов и подстановке значений из этих списков в переменные мы получим отношение между двумя сущностями и сможем судить о его истинности, например:

«Площадь Волгограда больше площади Хиросимы» = T ;

«Площадь Вологды больше площади Токио» = F .

Иногда для утверждения об истинности или ложности предиката можно обойтись без подстановки. Например, если областью определения переменной X являются целые положительные числа, то предикат « X больше -5 » будет тождественно истинен».

Основными синтаксическими единицами логики предикатов являются константы, переменные, функции, предикаты, кванторы и логические операторы. Формальный синтаксис исчисления предикатов первого порядка удобно представить в нормальной форме Бэкуса—Наура, которая традиционно применяется для записи грамматик языков программирования.

```
<константа> → <идентификатор1>
<переменная> → <идентификатор2>
<функция> → <идентификатор3>
<предикат> → <идентификатор4>
<терм> → <константа> | <переменная> |
        | <функция> (<список термов>)
<список термов> → <терм> | <терм>, <список термов>
<атом> → <предикат> | <предикат> (<список термов>)
<литера> → <атом> | ¬ <атом>
<оператор> → ∧ | ∨ | → | ↔
<список переменных> → <переменная> | <переменная>,
                        <список переменных>
<квантор> → <(∃<список переменных>)> |
            | <(∀<список переменных>)>
<формула> → <литера> | ¬ <формула> | <квантор> (<формула>) |
            (<формула>) <оператор> (<формула>)
```

В данной записи любое имя в угловых скобках представляет собой тип синтаксического объекта. Определение каждого типа начинается с появления его имени в левой части каждой записи, т. е. слева от знака $\rightarrow$. В правой части каждой записи приводятся возможные способы организации синтаксически корректных объектов определяемого типа. Альтернативные варианты разделены знаком $|$, который можно интерпретировать как ИЛИ. Номера идентификаторов следует трактовать в том смысле, что идентификаторы, используемые для обозначения объектов разных типов, должны быть различимыми. Например, константы обозначаются именами $\langle \text{идентификатор1} \rangle$, которые формируются из строчных букв, причем первым символом должен быть один из следующих: $a, b, c, d, e, k, l, m, n, x, y, z, v, w, u$. Имена переменных $\langle \text{идентификатор2} \rangle$ должны начинаться, например, с заглавной буквы. Идентификаторы функций $\langle \text{идентификатор3} \rangle$ состоят из строчных букв, при этом первой является f, g, h, p или q . Имена предикатов $\langle \text{идентификатор4} \rangle$ должны состоять из прописных букв. Функции, как и предикаты, задают некоторую связь между переменными или константами. Но эта связь или отношение не характеризуются истинностным значением. С помощью функции можно представить сложный объект, например, функция $f_{\text{book}}(\text{Author}, \text{Title}, \text{Publisher}, \text{Year})$ представляет набор информации, характеризующей книгу. Предикат и функция отличаются также на синтаксическом уровне, а именно: функции могут являться аргументами предикатов (т.е. термами), а предикаты — нет. Следует заметить, что в логике предикатов более высоких порядков по сравнению с первым аргументами предикатов могут быть другие предикаты. Функции с нулевым числом мест (аргументов) являются аналогами констант. Предикат без аргументов эквивалентен высказыванию.

Кванторы в логике предикатов необходимы для определения области действия переменных. Так, в логическом выводе о Сократе высказывание «Все люди смертны» можно уточнить следующим образом:

«Для всех X , если X является человеком, то X является смертным».

Введя предикаты $\text{ЧЕЛОВЕК}(X)$ и $\text{СМЕРТЕН}(X)$, можем составить логическую формулу $\text{ЧЕЛОВЕК}(X) \rightarrow \text{СМЕРТЕН}(X)$. Чтобы показать справедливость этой формулы для любого X , используется квантор общности:

$\forall X$ — «для любого X ».

Тогда рассматриваемое утверждение запишется в виде $\forall(X) \text{ЧЕЛОВЕК}(X) \rightarrow \text{СМЕРТЕН}(X)$.

Кроме квантора общности в логике предикатов есть квантор существования: $\exists X$ — «существует хотя бы один такой X , что ...» или «найдется хотя бы один X , такой, что ...»

Переменные, находящиеся в сфере действия кванторов, называются связанными, остальные переменные в логических формулах называются свободными. Для того чтобы можно было говорить об истинности какого-либо утверждения без подстановки значений в переменные, все входящие в него переменные должны быть связаны кванторами.

Если в логическую формулу входит несколько кванторов, необходимо учитывать их взаимное расположение. Рассмотрим возможные интерпретации логической формулы $\text{ЛЮБИТ}(X, Y)$ с квантифицированными переменными. При этом существует несколько вариантов размещения кванторов, один из которых $\forall X \exists Y \text{ЛЮБИТ}(X, Y)$. Эту формулу можно интерпретировать двояко:

- для любого X существует хотя бы один человек Y , которого любит X ;

- существует по крайней мере один человек Y , которого любят все X ;

Для устранения этой неопределенности введем скобки и порядок применения кванторов — слева направо, тогда получим следующие формулы, соответствующие интерпретациям:

1. $(\forall X)(\exists Y) \text{ЛЮБИТ}(X, Y)$.

2. $(\exists Y)(\forall X) \text{ЛЮБИТ}(X, Y)$.

Рассмотрим остальные варианты расположения кванторов и их интерпретации.

3. $(\forall X)(\forall Y) \text{ЛЮБИТ}(X, Y)$ и $(\forall Y)(\forall X) \text{ЛЮБИТ}(X, Y)$ — «Всеобщее человеколюбие».

4. $(\exists X)(\forall Y) \text{ЛЮБИТ}(X, Y)$ — «Существует хотя бы один человек, который любит всех людей».

5. $(\forall Y)(\exists X) \text{ЛЮБИТ}(X, Y)$ — «Каждого человека кто-нибудь любит».

6. $(\exists X)(\exists Y) \text{ЛЮБИТ}(X, Y)$ и $(\exists Y)(\exists X) \text{ЛЮБИТ}(X, Y)$ — «Существует хотя бы один человек, который не утратил чувства любви».

В одной логической формуле не допускается применение разных кванторов к одной переменной, например выражение $(\exists X)(\forall X)P(X)$ является недопустимым.

Отрицание кванторных выражений выполняется в соответствии со следующими правилами:

$$\neg(\forall X)P(X)=(\exists X)\neg P(X),$$

$$\neg(\exists X)P(X)=(\forall X)\neg P(X).$$

Справедливость приведенных выражений вытекает из смысла кванторов. Эти соотношения позволяют любую формулу в логике предикатов представить в виде предваренной нормальной формы (ПНФ), в которой сначала выписываются все кванторы, а затем — предикатные выражения.

Пример ПНФ: $(\forall X)(\forall Y)(\exists Z)(P(X) \rightarrow Q(Y, Z, W))$.

В логике предикатов первого порядка не разрешается применение кванторов к предикатам (более высокие порядки это позволяют).

Формула, в которой все переменные связаны, называется предложением. Каждому предложению можно поставить в соответствие определенное значение — «истина» или «ложь».

Пример: пусть $f(X)$ — функция, задающая отношение «отец»; $P(X)$ — предикат, задающий отношение «человек». Тогда логическая формула $(\forall X)(P(f(X)) \rightarrow P(X))$ будет интерпретироваться как «Все существа, отцом которых является человек, — люди».

Операции в логике предикатов имеют неодинаковые приоритеты. Самый высокий приоритет имеет квантор общности, самый низкий — операция эквивалентности.

$$\frac{\forall, \exists, \neg, \wedge, \vee, \rightarrow, \leftrightarrow}{\text{убывание приоритета}} \rightarrow$$

Сложные формулы в логике предикатов получаются путем комбинирования атомарных формул с помощью логических операций. Такие формулы называются правильно построенными логическими формулами (ППФ). Интерпретация ППФ возможна только с учетом конкретной области интерпретации, которая представляет собой множество всех возможных значений термов, входящих в ППФ. Для представления знаний конкретной предметной области в виде ППФ необходимо прежде всего установить область интерпретации (мир Хербранда), т.е. выбрать константы, которые определяют объекты в данной области, а также функции и предикаты, которые определяют зависимости и отношения между объектами. После этого можно построить логические формулы, описывающие закономерности данной предмет-

ной области. Записать знания с помощью логической модели не удается в тех случаях, когда затруднен выбор указанных трех групп элементов (констант, функций и предикатов) или когда для описания этих знаний не хватает возможностей представления с помощью ППФ, например когда знания являются неполными, ненадежными, нечеткими и т.д.

Логическая модель применяется в основном в исследовательских системах, так как предъявляет очень высокие требования к качеству и полноте знаний предметной области.

2.2.2. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ПРАВИЛАМИ ПРОДУКЦИЙ

Продукционная модель в силу своей простоты получила наиболее широкое распространение. В этой модели знания представляются в виде совокупности правил типа «ЕСЛИ — ТО». Системы обработки знаний, использующие такое представление, получили название продукционных систем. В состав экспертной системы продукционного типа входят база правил, база фактических данных (рабочая память) и интерпретатор правил, реализующий определенный механизм логического вывода. Любое продукционное правило, содержащееся в БЗ, состоит из двух частей: антецедента и консеквента. Антецедент представляет собой посылку правила (условную часть) и состоит из элементарных предложений, соединенных логическими связками И, ИЛИ. Консеквент (заключение) включает одно или несколько предложений, которые выражают либо некоторый факт, либо указание на определенное действие, подлежащее исполнению. Продукционные правила принято записывать в виде АНТЕЦЕДЕНТ → КОНСЕКВЕНТ.

Примеры продукционных правил:

ЕСЛИ «двигатель не заводится» И «стартер двигателя не работает», ТО «неполадки в системе электропитания стартера»;

ЕСЛИ «животное имеет перья», ТО «животное — птица».

Антецеденты и консеквенты правил формируются из атрибутов и значений, например:

<i>Атрибут</i>	<i>Значение</i>
Двигатель	Не заводится
Стартер двигателя	Не работает
Животное	Имеет перья
Животное	Птица

Любое правило состоит из одной (или нескольких) пары *атрибут — значение*. В рабочей памяти продукционной системы хранятся пары атрибут — значение, истинность которых установлена в процессе решения конкретной задачи к некоторому текущему моменту времени. Содержимое рабочей памяти изменяется в процессе решения задачи. Это происходит по мере срабатывания правил. Правило *срабатывает*, если при сопоставлении фактов, содержащихся в рабочей памяти, с antecedентом анализируемого правила имеет место совпадение, при этом заключение сработавшего правила заносится в рабочую память. Поэтому в процессе логического вывода объем фактов в рабочей памяти, как правило, увеличивается (уменьшаться он может в том случае, если действие какого-нибудь правила состоит в удалении фактов из рабочей памяти). В процессе логического вывода каждое правило из базы правил может сработать только один раз.

При описании реальных знаний конкретной предметной области может оказаться недостаточным представление фактов с помощью пар *атрибут—значение*. Более широкие возможности имеет способ описания с помощью триплетов *объект—атрибут—значение*. В этом случае отдельная сущность предметной области рассматривается как объект, а данные, хранящиеся в рабочей памяти, показывают значения, которые принимают атрибуты этого объекта.

Примеры триплетов:

собака — кличка — Граф;
собака — порода — ризеншнауцер;
собака — окрас — черный.

Одним из преимуществ такого представления знаний является уточнение контекста, в котором применяются правила. Например, правило, относящееся к объекту «собака», должно быть применимо для собак с любыми кличками, всех пород и окрасок. С введением триплетов правила из базы правил могут срабатывать более одного раза в процессе одного логического вывода, поскольку одно правило может применяться к различным экземплярам объекта (но не более одного раза к каждому экземпляру).

Существуют два типа продукционных систем — с прямыми и обратными выводами. *Прямые* выводы реализуют стратегию «от

фактов к заключениям». При *обратных* выводах выдвигаются гипотезы вероятных заключений, которые могут быть подтверждены или опровергнуты на основании фактов, поступающих в рабочую память. Существуют также системы с двунаправленными выводами.

Основные достоинства продукционных систем связаны с простотой представления знаний и организации логического вывода. К недостаткам систем продукций можно отнести следующие:

- отличие от структур знаний, свойственных человеку;
- неясность взаимных отношений правил;
- сложность оценки целостного образа знаний;
- низкая эффективность обработки знаний.

При разработке небольших систем (десятки правил) проявляются в основном положительные стороны систем продукций, однако при увеличении объема знаний более заметными становятся слабые стороны.

2.2.3. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРЕДСТАВЛЕНИЕ ЗНАНИЙ ФРЕЙМАМИ

Фреймовая модель представления знаний основана на теории фреймов М. Минского, которая представляет собой систематизированную психологическую модель памяти человека и его сознания. Эта теория имеет весьма абстрактный характер, поэтому только на ее основе невозможно создание конкретных языков представления знаний.

Фреймом называется структура данных для представления некоторого концептуального объекта.

Фрейм имеет имя, служащее для идентификации описываемого им понятия, и содержит ряд описаний — *слов*, с помощью которых определяются основные структурные элементы этого понятия. За слотами следуют *шпации*, в которые помещают данные, представляющие текущие значения слотов. Слот может содержать не только конкретное значение, но также имя процедуры, позволяющей вычислить это значение по заданному алгоритму. Например, слот с именем *возраст* может содержать имя процедуры, которая вычисляет возраст человека по дате рождения,

записанной в другом слоте, и текущей дате. Процедуры, располагающиеся в слотах, называются связанными или присоединенными процедурами. Вызов связанной процедуры осуществляется при обращении к слоту, в котором она помещена. Заполнителями слота могут быть также правила продукций, используемые для определения конкретного значения. В слоте может содержаться не одно, а несколько значений, т. е. в качестве структурных составляющих фреймов могут использоваться данные сложных типов, а именно: массивы, списки, множества, фреймы и т. д. Например, в слоте с именем *брат* может содержаться список имен, если объект, описываемый данным фреймом, имеет нескольких братьев. Значение слота может представлять собой некоторый диапазон или перечень возможных значений, арифметическое выражение, фрагмент текста и т. д.

Совокупность данных предметной области может быть представлена множеством взаимосвязанных фреймов, образующих единую фреймовую систему, в которой объединяются декларативные и процедурные знания. Такая система имеет, как правило, иерархическую структуру, в которой фреймы соединены друг с другом с помощью родо-видовых связей. На верхнем уровне иерархии находится фрейм, содержащий наиболее общую информацию, истинную для всех остальных фреймов. Фреймы обладают способностью наследовать значения характеристик своих родителей. Например, фрейм АФРИКАНСКИЙ_СЛОН наследует от фрейма СЛОН значение характеристики *цвет*=«серый». Значение характеристики в дочернем фрейме может отличаться от родительского, например, значением данного слота для фрейма АЗИАТСКИЙ_СЛОН является *цвет*=«коричневый».

Над фреймами можно совершать некоторые теоретико-множественные операции, например объединение и пересечение. При объединении фреймов в результирующем фрейме будут присутствовать все слоты, которые встречались в исходных фреймах. В слотах, не являющихся общими, будут сохранены исходные значения. Если в объединяемых фреймах были одноименные слоты, в результирующем фрейме останется один слот с таким именем, значение его определится в результате объединения значений одноименных слотов. При пересечении фреймов в результирующем фрейме будут присутствовать только те слоты, которые имелись во всех исходных фреймах. Вычислить результирующие значения можно двумя способами. Первый способ состоит в

том, что в результирующем фрейме присутствуют только те значения, которые совпадали в исходных фреймах. Во втором способе результирующие значения находят путем пересечения значений из исходных фреймов.

Фреймовые системы подразделяются на статические и динамические, последние допускают изменение фреймов в процессе решения задачи.

Пример фрейма РУКОВОДИТЕЛЬ

Имя слота	Значение слота	Тип значения слота
Имя	Иванов И. И.	Строка символов
Рожден	01.01.1965	Дата
Возраст	<i>age(дата, рожден)</i>	Процедура
Специальность	Юрист	Строка символов
Отдел	Отдел кадров	Строка символов
Зарплата	80000	Число
Адрес	ДОМ_АДРЕС	Фрейм

В общем случае структура данных фрейма может содержать более широкий набор информации, в который входят следующие атрибуты.

Имя фрейма. Оно служит для идентификации фрейма в системе и должно быть уникальным. Фрейм представляет собой совокупность слотов, число которых может быть произвольным. Число слотов в каждом фрейме устанавливается проектировщиком системы, при этом часть слотов определяется самой системой для выполнения специфических функций (системные слоты), примерами которых являются: слот-указатель родителя данного фрейма (IS-A), слот-указатель дочерних фреймов, слот для ввода имени пользователя, слот для ввода даты определения фрейма, слот для ввода даты изменения фрейма и т.д.

Имя слота. Оно должно быть уникальным в пределах фрейма. Обычно имя слота представляет собой идентификатор, который наделен определенной семантикой. В качестве имени слота может выступать произвольный текст. Например, <Имя слота>= Главный герой романа Ф. М. Достоевского «Идиот», <Значение слота>= Князь Мышкин. Имена системных слотов обычно зарезервированы, в различных системах они могут иметь различные значения. Примеры имен системных слотов: IS-A, HASPART,

RELATIONS и т.д. Системные слоты служат для редактирования базы знаний и управления выводом во фреймовой системе.

Указатели наследования. Они показывают, какую информацию об атрибутах слотов из фрейма верхнего уровня наследуют слоты с аналогичными именами в данном фрейме. Указатели наследования характерны для фреймовых систем иерархического типа, основанных на отношениях типа «абстрактное — конкретное». В конкретных системах указатели наследования могут быть организованы различными способами и иметь разные обозначения:

U (Unique) — значение слота не наследуется;

S (Same) — значение слота наследуется;

R (Range) — значения слота должны находиться в пределах интервала значений, указанных в одноименном слоте родительского фрейма;

O (Override) — при отсутствии значения в текущем слоте оно наследуется из фрейма верхнего уровня, однако в случае определения значения текущего слота оно может быть уникальным. Этот тип указателя выполняет одновременно функции указателей U и S.

Указатель типа данных. Он показывает тип значения слота. Наиболее употребляемые типы: *frame* — указатель на фрейм; *real* — вещественное число; *integer* — целое число; *boolean* — логический тип; *text* — фрагмент текста; *list* — список; *table* — таблица; *expression* — выражение; *lisp* — связанная процедура и т.д.

Значение слота. Оно должно соответствовать указанному типу данных и условию наследования.

Демоны. Демоном называется процедура, автоматически запускаемая при выполнении некоторого условия. Демоны автоматически запускаются при обращении к соответствующему слоту. Типы демонов связаны с условием запуска процедуры. Демон с условием *IF-NEEDED* запускается, если в момент обращения к слоту его значение не было установлено. Демон типа *IF-ADDED* запускается при попытке изменения значения слота. Демон *IF-REMOVED* запускается при попытке удаления значения слота. Возможны также другие типы демонов. Демон является разновидностью связанной процедуры.

Присоединенная процедура. В качестве значения слота может использоваться процедура, называемая служебной в языке Лисп

или методом в языках объектно-ориентированного программирования. Присоединенная процедура запускается по сообщению, переданному из другого фрейма. Демоны и присоединенные процедуры являются процедурными знаниями, объединенными вместе с декларативными в единую систему. Эти процедурные знания являются средствами управления выводом во фреймовых системах, причем с их помощью можно реализовать любой механизм вывода. Представление таких знаний и заполнение ими интеллектуальных систем — весьма нелегкое дело, которое требует дополнительных затрат труда и времени разработчиков ИИС. Поэтому проектирование фреймовых систем выполняется, как правило, специалистами, имеющими высокий уровень квалификации в области искусственного интеллекта.

Пример сети фреймов приведен на рис. 2.1 [1]. На нем понятие **УЧЕНИК** наследует свойства фреймов **РЕБЕНОК** и **ЧЕЛОВЕК**, которые находятся на более высоких уровнях иерархии. Если будет задан вопрос «Любят ли ученики сладкое?», то следует ответить «да», так как этим свойством обладают все дети, что указано во фрейме **РЕБЕНОК**. Наследование свойств может быть частичным, например «возраст» для учеников не наследуется из фрейма «ребенок», так как явно указан в собственном фрейме.

В последние годы термин «фреймовый» часто заменяют термином «объектно-ориентированный». Шаблон фрейма можно рассматривать как класс, экземпляр фрейма — как объект. Языки объектно-ориентированного программирования (ООП) предоставляют средства создания классов и объектов, а также средства для описания процедур обработки объектов (методы). Языки ООП, не содержащие средств реализации присоединенных процедур, не позволяют организовать гибкий механизм логического вывода, поэтому разработанные на них программы либо представляют собой объектно-ориентированные базы данных, либо требуют интеграции с другими средствами обработки знаний (например, с языком PROLOG).

Существуют также специализированные языки представления знаний на основе фреймовой модели, примерами которых являются: FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймовая «оболочка» Кappa и др. Известны также экспертные системы фреймового типа: ANALYST, TRISTAN, ALTERID, МОДИС [8, 9, 12, 13].

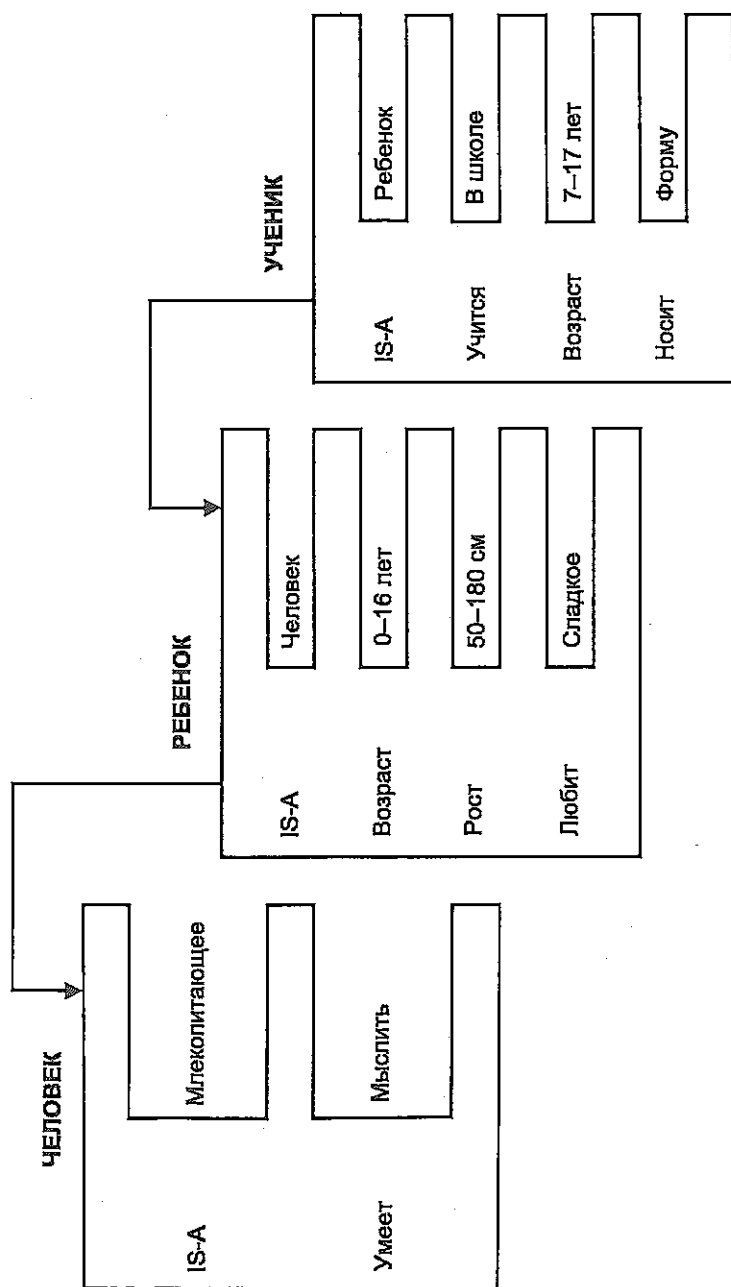


Рис.2.1. Пример иерархии фреймов

Общепринятого определения семантической сети не существует. Обычно под ней подразумевают систему знаний некоторой предметной области, имеющую определенный смысл в виде целостного образа сети, узлы которой соответствуют понятиям и объектам, а дуги — отношениям между объектами. При построении семантической сети отсутствуют ограничения на число связей и на сложность сети. Для того чтобы формализация оказалась возможной, семантическую сеть необходимо систематизировать. Семантические сети Куиллиана систематизируют функции отношений между понятиями с помощью следующих признаков:

- множество — подмножество (типы отношений «абстрактное — конкретное», «целое — часть», «род — вид»);
- индексы (свойства, имена прилагательные в языке и т.п.);
- конъюнктивные связи (логическое И);
- дизъюнктивные связи (логическое ИЛИ);
- связи по ИСКЛЮЧАЮЩЕМУ ИЛИ;
- отношения «близости»;
- отношения «сходства — различия»;
- отношения «причина — следствие» и др.

При построении семантической сети отсутствуют ограничения на число элементов и связей. Поэтому систематизация отношений между объектами в сети необходима для дальнейшей формализации. Пример семантической сети представлен на рис. 2.2.

Для реализации семантических сетей существуют специальные сетевые языки: NET, язык реализации систем SIMER+MIR и др. [4, 10]. Широко известны экспертные системы, использующие семантические сети в качестве языка представления знаний: PROSPECTOR, CASNET, TORUS [7, 9, 11 — 13].

Систематизация отношений конкретной семантической сети зависит от специфики знаний предметной области и является сложной задачей. Особого внимания заслуживают общезначимые отношения, присутствующие во многих предметных областях. Именно на таких отношениях основана концепция семантической сети. В семантических сетях, так же как при фреймовом представлении знаний, декларативные и процедурные знания не разделены, следовательно, база знаний не отделена от механизма

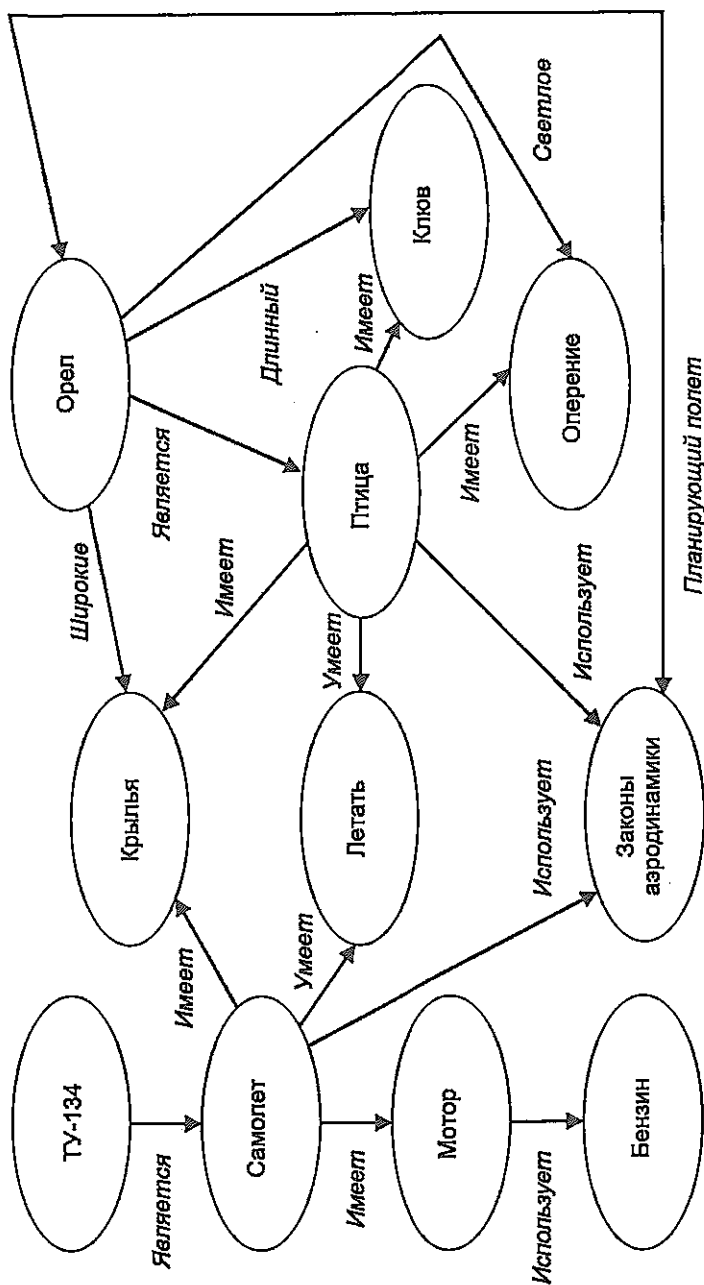


Рис. 2.2. Семантическая сеть, показывающая взаимоотношения птицы и самолета

вывода. Процедура логического вывода обычно представляет совокупность процедур обработки сети. Семантические сети получили широкое применение в системах распознавания речи и экспертных системах.

2.3. ТРАДИЦИОННЫЕ СПОСОБЫ ОБРАБОТКИ ЗНАНИЙ

Компьютерная обработка знаний является одной из областей обработки информации. Традиционная технология обработки информации ориентирована на вычисления, основанные на теории Тьюринга, которая реализована в ЭВМ с архитектурой фон Неймана. Этот подход основан на концепции последовательной обработки информации во времени, поэтому в таких ЭВМ используются в основном языки процедурного типа, а в качестве операционных механизмов — устройства управления и арифметические устройства. Процессы обработки знаний моделируют такие сферы человеческой деятельности, как рассуждения, систематизация, обучение и т.д. Если попытаться «переложить подобные задачи на плечи компьютеров», последним будет явно недостаточно жесткого набора инструкций в виде программы на процедурном языке программирования, даже если эта программа очень сложная. Адекватное представление знаний требует разработки и применения языков декларативного типа в интеллектуальных системах, а в качестве операционных механизмов — различных способов реализации логического вывода. Очевидно, будущее в этом направлении принадлежит компьютерам с архитектурой, отличной от фон-неймановской. Какими должны быть компьютеры, ориентированные на выполнение интеллектуальных задач, покажут дальнейшие теоретические и практические исследования. В настоящее время эксперименты проводятся в основном на машинах Тьюринга—фон Неймана и на машинах с параллельной обработкой данных.

Характерная черта ИИС, отличающая их от традиционных систем обработки информации, — использование знаний. Выбор способа представления знаний в интеллектуальной системе является ключевым моментом разработки. С точки зрения человека, желательно, чтобы описательные возможности используемой модели были как можно выше. С другой стороны, сложное пред-

ставление знаний требует специальных способов обработки (усложняется механизм вывода), что не только затрудняет проектирование и реализацию экспертной системы, но может привести к потере достоверности результатов или невозможности их интерпретации. В конечном итоге неизбежен компромисс между важнейшими условиями и требованиями, предъявленными к ИИС на этапе проектирования. Вопросы проектирования и реализации интеллектуальных программных средств — отдельная область исследований, а здесь рассмотрим способы реализации логического вывода в системах с классическими моделями представления знаний.

2.3.1. СПОСОБЫ ДОКАЗАТЕЛЬСТВА И ВЫВОДА В ЛОГИКЕ

Традиционной логикой называют формальную систему, предложенную Аристотелем более 2500 лет назад. Аристотель стремился установить и формально записать способы, с помощью которых можно было бы установить правильность рассуждений в разумной полемике. Множества правил, определяющих, какие умозаключения могут быть получены из множества суждений, он назвал *силлогизмом*. В данном случае под *суждением* понимается законченная мысль, которую можно выразить на естественном языке в одной из следующих четырех форм:

- все X являются P — $(\forall X)P(X)$;
- никакой X не является P — $(\forall X)\neg P(X)$;
- некоторые из X являются P — $(\exists X)P(X)$;
- некоторые из X не являются P — $(\exists X)\neg P(X)$.

Каждое правило силлогизма определяет переход от предпосылок к заключению, являющемуся интуитивно очевидным. Силлогизм изначально предназначен для управления разумной дискуссией на естественном языке. Как формальная теория, он чрезмерно сложен и неполон. Силлогизм можно назвать логикой классов, которая не содержит понятия дополнения класса. Например, определив класс A (допустим, субъекты, являющиеся водителями автомобилей), мы не можем построить его дополнение (т.е. установить всех лиц, которые не являются водителями).

Будучи тесно связанным с естественным языком, силлогизм иногда приводит к абсурдным результатам, в частности, не допу-

ская логическую эквивалентность двух способов выражения одной и той же мысли. Силлогизм неполон в том смысле, что он не позволяет осуществлять логические выводы, в которых затрагиваются вопросы существования элемента некоторого класса.

Последователи Аристотеля, основываясь на силлогизме, сформулировали принципы дедуктивного вывода для высказываний, которые находятся на более высоком уровне абстракции по сравнению с суждениями. Наиболее известными из этих правил являются следующие:

1. *Modus Ponendo Ponens*: «Если истинна импликация $A \rightarrow B$ и A истинно, то B истинно».

2. *Modus Tollendo Tollens*: «Если истинна импликация $A \rightarrow B$ и B ложно, то A ложно».

3. *Modus Ponendo Tollens*: «Если A истинно и конъюнкция $A \wedge B$ имеет результатом ложь, то B ложно».

4. *Modus Tollendo Ponens*: «Если A ложно и дизъюнкция $A \vee B$ истинна, то B является истиной».

На основе этих правил сформулировано правило «цепного заключения», весьма удобное для вывода в системе исчисления высказываний.

5. *Цепное заключение*: «Если истинна импликация $A \rightarrow B$ и истинна импликация $B \rightarrow C$, то импликация $A \rightarrow C$ является истинной».

Рассмотрим пример вывода с применением этих правил.

Пусть заданы следующие посылки:

1. $P \rightarrow Q$ /* Если растут мировые цены на топливно-энергетические ресурсы, то увеличиваются поступления в бюджет */.

2. $(R \vee Q) \rightarrow (R \vee S)$ /* Если наблюдается рост производства или увеличиваются поступления в бюджет, то следует увеличение производства или укрепление рубля */.

3. $(P \rightarrow Q) \rightarrow ((P \vee Q) \rightarrow (R \vee Q))$ /* Если растут мировые цены на топливно-энергетические ресурсы, то увеличиваются поступления в бюджет, из чего следует, что при росте цен на сырье или при увеличении поступлений в бюджет происходит рост производства или увеличение бюджета */.

Используя *Modus Ponendo Ponens*, из посылок 1 и 3 можно вывести следующее заключение.

4. $(P \vee Q) \rightarrow (R \vee Q)$ /* Если растут мировые цены на топливно-энергетические ресурсы или увеличиваются поступления в бюджет, то происходит рост производства или увеличение бюджета */.

Из посылок 4 и 2 с помощью цепного заключения можно получить посылку 5.

5. $(P \vee Q) \rightarrow (R \vee S)$ /* Если растут мировые цены на топливно-энергетические ресурсы или увеличиваются поступления в бюджет, то происходит рост производства или укрепление рубля */.

Таким образом, применяя правила логического вывода, получаем новые логические формулы на основании исходных. При значительном количестве исходных данных возможно получение большого количества цепочек вывода, результаты которых могут противоречить друг другу. Подобные проблемы должны быть корректно решены в конкретных интеллектуальных системах при организации управления стратегией логического вывода.

Проблема доказательства в логике состоит в нахождении истинностного значения заключения B , если предполагается истинность исходных посылок $A_1, A_2, \dots, A_n$, что записывается в виде: $A_1, A_2, \dots, A_n \vdash B$. Знак $\vdash$ в доказательствах и выводах следует читать «верно, что» или «можно вывести».

Существуют два основных метода решения проблемы доказательства в логике: семантический и синтаксический.

Семантический метод состоит в следующем. Можно перечислить все атомы, входящие в формулы $A_1, A_2, \dots, A_n, B$, и составить таблицу истинности для всевозможных комбинаций значений этих атомов. Затем следует осуществить просмотр полученной таблицы, чтобы проверить, во всех ли ее строках, где формулы $A_1, A_2, \dots, A_n$ имеют значения «истина», формула B также имеет значение «истина». Этот метод применим всегда, но может оказаться слишком трудоемким.

При *синтаксическом* методе доказательства сначала записывают посылки и, применяя к ним правила вывода, стараются получить из них другие истинные формулы. Из этих формул и исходных посылок выводят последующие формулы, и этот процесс продолжают до тех пор, пока не будет получено требуемое заключение (заметим, что это не всегда возможно). Этот процесс, по сути дела, и является логическим выводом, он часто применяется при доказательстве теорем в математике.

Иногда значение конкретной логической формулы не зависит от значений входящих в них атомов. Правильно построенные логические формулы, значением которых будет «истина» при любых значениях входящих в них атомов, называются *тавтологиями*. Тавтологии, или теоремы логики, обладают следующим свой-

ством: если вместо всех вхождений некоторого атома в тавтологию подставить произвольную логическую формулу, то снова будет получена истинная формула. Эта новая формула называется частным случаем исходной формулы, или результатом подстановки.

Правило подстановки. Если $C(A)$ — тавтология и вместо всех вхождений формулы A в C подставить формулу B , то $C(B)$ — тавтология. Для обозначения тавтологий используется символ $\models$, который читается «общезначимо» или «всегда истинно».

Примеры тавтологий:

$$\models A \wedge (A \rightarrow B) \rightarrow B;$$

$$\models \neg B \wedge (A \rightarrow B) \rightarrow \neg A;$$

$$\models (A \rightarrow B) \rightarrow ((A \vee C) \rightarrow (B \vee C)).$$

Докажем то, что первая приведенная тавтология (Modus Ponendo Ponens) всегда будет иметь значение «истина», для чего построим сокращенную таблицу истинности (табл. 2.2).

Таблица 2.2

Таблица истинности для доказательства тавтологии

A	$A \rightarrow B$	$A \wedge (A \rightarrow B)$	$A \wedge (A \rightarrow B) \rightarrow B$
T	B	B	T
T	B	B	T
F	T	F	T
F	T	F	T

В приведенной таблице T — истина, F — ложь. Значение импликации совпадает со значением второго аргумента в том случае, если первый аргумент имеет значение T, поэтому в первых двух строках второго столбца присутствует второй аргумент — B , значение которого может быть произвольным. Конъюнкция $A \wedge (A \rightarrow B)$ при истинном A также будет иметь результат, совпадающий со значением B . Последний столбец таблицы комментариев не требует, так как приведенные в нем результаты очевидны.

Попробуем установить тавтологичность этой же формулы синтаксическим методом. Для этого раскроем все скобки по распределительному закону и убедимся, что результат упрощается до формул с очевидными значениями истинности:

$$\begin{aligned}
 (A \wedge (A \rightarrow B)) \rightarrow B &= \neg(A \wedge (\neg A \vee B)) \vee B = (\neg A \vee (A \wedge \neg B)) \vee B = \\
 ((\neg A \vee A) \wedge (\neg A \vee \neg B)) \vee B &= (T \wedge (\neg A \vee \neg B)) \vee B = \neg A \vee \neg B \vee B = \\
 \neg A \vee (\neg B \vee B) &= \neg A \vee T = T.
 \end{aligned}$$

Помимо использования тавтологий и подстановок полезным средством для вывода является эквивалентность. Необходимо уметь правильно осуществлять замену взаимно эквивалентных формул. Например, можно подставить $Q \vee P$ вместо $P \vee Q$, так как $Q \vee P \leftrightarrow P \vee Q$.

Эквивалентность $A \leftrightarrow B$ можно заменить двусторонней импликацией $(A \rightarrow B) \wedge (B \rightarrow A)$, так как эти выражения имеют одну и ту же таблицу истинности. Отсюда можно сделать вывод, что логическая эквивалентность — это импликация в обоих направлениях. Эквивалентность можно привести в конъюнктивную нормальную форму, которая имеет вид: $(\neg A \vee B) \wedge (\neg B \vee A)$. Если в этом выражении раскрыть скобки, получим дизъюнктивную нормальную форму: $(\neg A \wedge \neg B) \vee (A \wedge B)$. Другими словами, если A и B эквивалентны, то они либо оба истинны, либо оба ложны.

Примеры тавтологий с эквивалентностями:

$$\begin{aligned}
 \models (A \rightarrow B) &\leftrightarrow (\neg B \rightarrow \neg A); \\
 \models (A \vee B) &\leftrightarrow (B \vee A); \\
 \models A \wedge (A \vee B) &\leftrightarrow A; \\
 \models (X \rightarrow (Y \rightarrow Z)) &\leftrightarrow (X \wedge Y \rightarrow Z).
 \end{aligned}$$

В процедурах логического вывода эквивалентности можно использовать двумя способами:

- 1) расписывать в виде двух отдельных импликаций;
- 2) использовать при заменах.

При этом важно понимать отличие замены от подстановки, которое состоит в следующем: если $A \leftrightarrow B$, то A можно заменить на B в любом вхождении в формулу C , не меняя ее значения, причем замену не обязательно осуществлять во всех вхождениях.

В противовес тавтологиям в логике существуют противоречия — формулы, значением которых всегда будет «ложь» независимо от значений входящих в них атомов. Примером является выражение $B \wedge \neg B = F$ при любом значении B .

Множество ППФ для некоторой предметной области называется теорией заданной области знаний, а каждая отдельная ППФ именуется аксиомой. Цель построения теории заключается в описании нужных знаний наиболее экономичным способом. Если удастся построить теорию, которая адекватно описывает за-

данную область знаний, то все истинные факты из области интерпретации будут следствиями аксиом этой теории, другими словами, их можно будет вывести из множества ППФ. Соответственно ложные факты не будут следствиями теории, следовательно, их нельзя будет получить путем логического вывода на основании аксиом данной теории. Теорию называют *полной*, если все истинные факты являются следствиями этой теории. Это означает, что каждая истинная для данной теории ППФ может быть доказана на основании ее аксиом.

Про теорию говорят, что она является *синтаксически последовательной (непротиворечивой)*, если из аксиом теории невозможно вывести противоречие. Теория, в которой можно доказать и P , и $\neg P$, непоследовательна.

В качестве примера рассмотрим построение теории и ее проверку на полноту и непротиворечивость для следующего фрагмента знаний:

«Если у Вас нету дома,
Пожары ему не страшны,
И жена не уйдет к другому,
Если у Вас нет жены».

Постараемся обойтись средствами логики высказываний и начнем с формирования области интерпретации. Для этого введем обозначения:

A — «У Вас есть дом»
 B — «Дом может сгореть»
 C — «У Вас есть жена»
 D — «Жена может уйти к другому»

На базе этих четырех высказываний построим логические формулы:

$\neg A \rightarrow \neg B$;
 $\neg C \rightarrow \neg D$.

Кроме логических формул, выражающих связь фактов, любая теория содержит истинные факты, на основе которых становится возможным конкретная интерпретация ППФ. Допустим, что в нашей теории такими фактами являются A , B , $\neg C$, $\neg D$.

Полученная теория является полной и последовательной, поскольку каждый факт этой теории выводится из остальных акси-

ом, при этом не возникнет противоречия. Доказать это несложно как семантическим, так и синтаксическим способом, однако для более сложных областей знаний необходимо использовать определенные стратегии доказательства, позволяющие преодолеть хаотичность процессов логического вывода.

Кратко рассмотрим три основные стратегии доказательства:

- доказательство с введением допущения;
- доказательство методом «от противного» (приведение к противоречию);
- доказательство методом резолюции.

Доказательство с введением допущения. Для доказательства импликации вида $A \rightarrow B$ допускается, что левая часть импликации истинна, т.е. A принимается в качестве дополнительной посылки, после чего делают попытки доказать правую часть B . Такая стратегия доказательства часто применяется в геометрии при доказательстве теорем.

Теорема 1. $A \vdash B$ тогда и только тогда, когда $\vdash A \rightarrow B$.

Эта теорема утверждает, что доказуемость заключения B при допущении истинности A эквивалентна доказуемости импликации $A \rightarrow B$ без дополнительных допущений. Справедливость данной теоремы следует из приведенной ранее таблицы истинности (см. табл. 2.2).

Теорема 2. $A_1, A_2, \dots, A_n \vdash B$ тогда и только тогда, когда $\vdash (A_1 \wedge A_2 \wedge \dots \wedge A_n) \rightarrow B$.

Эта теорема выводится из предыдущей и того факта, что посылки $A_1, A_2, \dots, A_n$ истинны только тогда, когда истинна их конъюнкция.

Пример: требуется доказать $A_1, A_2, \dots, A_n, P \vdash Q$. В соответствии с теоремой 2 получим $\vdash (A_1 \wedge A_2 \wedge \dots \wedge P) \rightarrow Q$. Применив эквивалентность вида $(A \rightarrow (P \rightarrow Q)) \leftrightarrow (A \wedge P \rightarrow Q)$, имеем $\vdash (A_1 \wedge A_2 \wedge \dots \wedge A_n) \rightarrow (P \rightarrow Q)$, откуда $\vdash A_1, A_2, \dots, A_n \vdash (P \rightarrow Q)$ в силу теоремы 2.

Приведение к противоречию. Этот метод доказательства, предложенный К. Робинсоном в 1960-х гг., вывел исследования в области искусственного интеллекта на новый уровень. Метод обусловил появление *обратных выводов* и эффективных способов выявления противоречий. Суть его состоит в следующем. Например, требуется доказать $A \rightarrow B$. Вместо этого можно попытаться доказать $\neg B \rightarrow \neg A$, используя эквивалентность. Такое доказательство можно провести двояко, а именно: или допустить A и дока-

затем B (это будет прямой вывод), или сделать допущение о том, что B — ложно, после чего сделать попытку опровергнуть посылку A (обратный вывод). Приведение к противоречию — комбинация прямого и обратного вывода, т. е. для доказательства $A \rightarrow B$ можно одновременно допустить A и $\neg B$ (посылка истинна, а заключение — ложно):

$$\neg(A \rightarrow B) = \neg(\neg A \vee B) = A \wedge \neg B.$$

В процессе доказательства можно двигаться по пути, который начинается от A или от $\neg B$. Если B выводимо из A , то, допустив истинность A , мы доказали бы B . Поэтому, сделав допущение $\neg B$, получим противоречие. Если вывод приведет к успеху (т. е. противоречие не будет получено), это будет свидетельствовать о несовместимости либо противоречивости исходных посылок. Мы также не получим противоречия, если доказываемое предложение $A \rightarrow B$ является ложным.

Доказательство методом резолюции. Этот метод считается более трудным для понимания, однако имеет важное преимущество перед другими: он легко формализуем. В основе метода лежит тавтология, получившая название «правило резолюции»:

$$\models (X \vee A) \wedge (Y \vee \neg A) \rightarrow (X \vee Y).$$

Теоремы 1 и 2 позволяют записать это правило в следующем виде:

$$(X \vee A), (Y \vee \neg A) \vdash (X \vee Y),$$

что дает основания утверждать: из посылок $(X \vee A)$ и $(Y \vee \neg A)$ можно вывести $(X \vee Y)$.

Докажем эту тавтологию с помощью тождеств булевой алгебры:

$$\begin{aligned} \neg((X \vee A) \wedge (Y \vee \neg A)) \vee X \vee Y &= (\neg X \wedge \neg A) \vee (\neg Y \wedge A) \vee X \vee Y = \\ (X \vee (\neg X \wedge \neg A)) \vee (Y \vee (\neg Y \wedge A)) &= (X \vee \neg A) \vee (Y \vee A) = X \vee Y \vee A \vee \neg A = \\ X \vee Y \vee T &= T. \end{aligned}$$

Для наглядности построим сокращенную таблицу истинности (табл. 2.3), из которой можно убедиться в справедливости правила резолюции, результат которого обозначен в таблице через R .

Легко убедиться, что все значения в столбце R будут истинными и при значении $A=F$.

В процессе логического вывода с применением правила резолюции выполняются следующие шаги.

Доказательство правила резолюции

X	Y	$X \vee Y$	A	$X \vee A$	$Y \vee \neg A$	$(X \vee A) \wedge (Y \vee \neg A)$	R
T	T	T	T	T	T	T	T
T	F	T	T	T	F	F	T
F	T	T	T	T	T	T	T
F	F	F	T	T	F	F	T

1. Устраняются операции эквивалентности и импликации:

$$A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A);$$

$$A \rightarrow B = \neg A \vee B.$$

2. Операция отрицания продвигается внутрь формул с помощью законов де Моргана:

$$\neg(A \wedge B) = \neg A \vee \neg B;$$

$$\neg(A \vee B) = \neg A \wedge \neg B.$$

3. Логические формулы приводятся к дизъюнктивной форме:

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C).$$

Правило резолюции содержит в левой части конъюнкцию дизъюнктов, поэтому приведение посылок, используемых для доказательства, к виду, представляющему собой конъюнкцию дизъюнктов, является необходимым этапом практически любого алгоритма, реализующего логический вывод на базе метода резолюции. Метод резолюции легко программируется, это одно из важнейших его достоинств.

Предположим, нужно доказать, что если истинны соотношения $P \rightarrow R$, $P \vee Q$ и $Q \rightarrow S$, то можно вывести формулу $R \vee S$. Для этого нужно выполнить следующие шаги.

1. Приведение посылок к дизъюнктивной форме: $\neg P \vee R$, $P \vee Q$, $\neg Q \vee S$.

2. Построение отрицания выводимого заключения $\neg(R \vee S) = \neg R \wedge \neg S$. Полученная конъюнкция справедлива, когда $\neg R$ и $\neg S$ одновременно истинны.

3. Применение правила резолюции:

$$(\neg P \vee R) \wedge \neg R \Rightarrow \neg P;$$

$$(P \vee Q) \wedge \neg P \Rightarrow Q;$$

$$(\neg Q \vee S) \wedge \neg S \Rightarrow \neg Q;$$

$$Q \wedge \neg Q \Rightarrow F \text{ (противоречие или «пустой дизъюнкт»)}.$$

Итак, предположив ложность выводимого заключения, получаем противоречие, следовательно, выводимое заключение является истинным, т.е. $R \vee S$ выводимо из исходных посылок.

Именно правило резолюции послужило базой для создания языка логического программирования PROLOG. По сути дела, интерпретатор языка PROLOG самостоятельно реализует вывод, подобный вышеописанному, формируя ответ на вопрос пользователя, обращенный к базе знаний.

В логике предикатов для применения правила резолюции предстоит осуществить более сложную унификацию логических формул в целях их приведения к системе дизъюнктов. Это связано с наличием дополнительных элементов синтаксиса, в основном кванторов, переменных, предикатов и функций.

Алгоритм унификации предикатных логических формул включает следующие шаги.

1. Исключение операций эквивалентности.
2. Исключение операций импликации.
3. Внесение операций отрицания внутрь формул.
4. Исключение кванторов существования. Это может произойти на шаге 3 вследствие применения законов де Моргана, а именно: в результате отрицания $\exists$ меняется на $\forall$, но при этом может произойти и обратная замена. Тогда для исключения $\exists$ поступают следующим образом: все вхождения некоторой переменной, связанной квантором существования, например $(\exists X)$, заменяются в формуле на новую константу, например a . Эта константа представляет собой некоторое (неизвестное) значение переменной X , для которого утверждение, записанное данной формулой, истинно. При этом важно то, что на все места, где присутствует X , будет подставлено одно и то же значение a , пусть оно и является неизвестным в данный момент. Такие константы называются *сколемовскими*, а операция — *сколемизацией* (по имени известного математика Сколема).

5. Кванторы общности выносятся на первые места в формулах. Это также не всегда является простой операцией, иногда при этом приходится делать переименование переменных.

6. Раскрытие конъюнкций, попавших внутрь дизъюнкций.

После выполнения всех шагов описанного алгоритма унификации можно применять правило резолюции. Обычно при этом осуществляется отрицание выводимого заключения, и алгоритм вывода можно кратко описать следующим образом: Если задано

несколько аксиом (теория Th) и предстоит сделать заключение о том, выводима ли некоторая формула P из аксиом теории Th , строится отрицание P и добавляется к Th , при этом получают новую теорию Th_1 . После приведения $\neg P$ и аксиом теории к системе дизъюнктов можно построить конъюнкцию $\neg P$ и аксиом теории Th . При этом существует возможность выводить из исходных дизъюнктов дизъюнкты-следствия. Если P выводимо из аксиом теории Th , то в процессе вывода можно получить некоторый дизъюнкт Q , состоящий из одной литеры, и противоположный ему дизъюнкт $\neg Q$. Это противоречие свидетельствует о том, что P выводимо из аксиом Th . Вообще говоря, существует множество стратегий доказательства, нами рассмотрена лишь одна из возможных — нисходящая.

Пример: представим средствами логики предикатов следующий текст:

«Если студент умеет хорошо программировать, то он может стать специалистом в области прикладной информатики».

«Если студент хорошо сдал экзамен по информационным системам, значит, он умеет хорошо программировать».

Представим этот текст средствами логики предикатов первого порядка. Введем обозначения: X — переменная для обозначения студента; *хорошо* — константа, соответствующая уровню квалификации; $P(X)$ — предикат, выражающий возможность субъекта X стать специалистом по прикладной информатике; $Q(X, \text{хорошо})$ — предикат, обозначающий умение субъекта X программировать с оценкой *хорошо*; $R(X, \text{хорошо})$ — предикат, задающий связь студента X с экзаменационной оценкой по информационным системам.

Теперь построим множество ППФ:

$$(\forall X)Q(X, \text{хорошо}) \rightarrow P(X).$$

$$(\forall X)R(X, \text{хорошо}) \rightarrow Q(X, \text{хорошо}).$$

Дополним полученную теорию конкретным фактом $R(\text{Иванов}, \text{хорошо})$.

Выполним логический вывод с применением правила резолюции, чтобы установить, является ли формула $P(\text{Иванов})$ следствием вышеприведенной теории. Другими словами, можно ли вывести из этой теории факт, что студент Иванов станет специалистом в прикладной информатике, если он хорошо сдал экзамен по информационным системам.

Доказательство

1. Выполним преобразование исходных формул теории в целях приведения к дизъюнктивной форме:

$(\exists X) \neg Q(X, \text{хорошо}) \vee P(X); (\exists X) \neg R(X, \text{хорошо}) \vee Q(X, \text{хорошо}); R(\text{иванов}, \text{хорошо}).$

2. Добавим к имеющимся аксиомам отрицание выводимого заключения

$\neg P(\text{иванов}).$

3. Построим конъюнкцию дизъюнктов

$(\exists X) \neg Q(X, \text{хорошо}) \vee P(X) \wedge \neg P(\text{иванов}) \Rightarrow \neg Q(\text{иванов}, \text{хорошо}),$

заменяя переменную X на константу иванов .

Результат применения правила резолюции называют *резольвентой*. В данном случае резольвентой является $\neg Q(\text{иванов})$.

4. Построим конъюнкцию дизъюнктов с использованием резольвенты, полученной на шаге 3:

$(\exists X) \neg R(X, \text{хорошо}) \vee Q(X, \text{хорошо}) \wedge \neg Q(\text{иванов}, \text{хорошо}) \Rightarrow \neg R(\text{иванов}, \text{хорошо}).$

5. Запишем конъюнкцию полученной резольвенты с последним дизъюнктом теории:

$\neg R(\text{иванов}, \text{хорошо}) \wedge R(\text{иванов}, \text{хорошо}) \rightarrow F$ (противоречие).

Следовательно, факт $P(\text{иванов})$ выводим из аксиом данной теории.

Для определения порядка применения аксиом в процессе вывода существуют следующие эвристические правила:

1. На первом шаге вывода используется отрицание выводимого заключения.

2. В каждом последующем шаге вывода участвует резольвента, полученная на предыдущем шаге.

2.3.2. ПРЯМОЙ И ОБРАТНЫЙ ВЫВОД В ЭКСПЕРТНЫХ СИСТЕМАХ ПРОДУКЦИОННОГО ТИПА

Любая экспертная система продукционного типа должна содержать три основные компоненты: базу правил, рабочую память и механизм вывода.

База правил (БП) — формализованные с помощью правил продукций знания о конкретной предметной области.

Рабочая память (РП) — область памяти, в которой хранится множество фактов, описывающих текущую ситуацию, и все пары *атрибут-значение*, которые были установлены к определенному моменту. Содержимое РП в процессе решения задачи изменяется обычно, увеличиваясь в объеме по мере применения правил. Другими словами, РП — это динамическая часть базы знаний, содержимое которой зависит от окружения решаемой задачи. В простейших ЭС хранимые в РП факты не изменяются в процессе решения задачи, однако существуют системы, в которых допускается изменение и удаление фактов из РП. Это системы с немотонным выводом, работающие в условиях неполноты информации.

Механизм вывода выполняет две основные функции:

- просмотр существующих в рабочей памяти фактов и правил из БП, а также добавление в РП новых фактов;
- определение порядка просмотра и применения правил. Порядок может быть прямым или обратным.

Прямой порядок — от фактов к заключениям. В экспертных системах с прямыми выводами по известным фактам отыскивается заключение, которое из этих фактов следует. Если такое заключение удастся найти, оно заносится в рабочую память. Прямые выводы часто применяются в системах диагностики, их называют выводами, управляемыми данными.

Обратный порядок вывода — от заключений к фактам. В системах с обратным выводом вначале выдвигается некоторая гипотеза о конечном суждении, а затем механизм вывода пытается найти в рабочей памяти факты, которые могли бы подтвердить или опровергнуть выдвинутую гипотезу. Процесс отыскания необходимых фактов может включать достаточно большое число шагов, при этом возможно выдвижение новых гипотез (целей). Обратные выводы управляются целями.

Для выполнения указанных функций механизм вывода включает компоненту вывода и управляющую компоненту.

Компонента вывода. Ее действие основано на применении правила логического вывода *Modus Ponendo Ponens*. Суть применения этого правила в продукционных системах состоит в следующем. Если в РП присутствует истинный факт *A* и в БП существует правило вида «ЕСЛИ *A*, ТО *B*», то факт *B* признается истинным и заносится в РП. Такой вывод легко реализуется на ЭВМ,

однако при этом часто возникают проблемы, связанные с распознаванием значений слов, а также с тем, что факты могут иметь внутреннюю структуру и между элементами этой структуры возможны различного рода связи. Например, пусть имеется факт *A* — «Автомобиль Иванова — белый» и правило «ЕСЛИ Автомобиль — белый, ТО Автомобиль легко заметить ночью». Человек легко выведет заключение «Автомобиль Иванова легко заметить ночью», но это не под силу ЭС чисто продукционного типа. Она не сможет сформировать такое заключение, потому что *A* не совпадает точно с антецедентом правила. Подобная проблема уже затрагивалась, когда рассматривались различия логики высказываний и логики предикатов. Кроме того, невысокая интеллектуальная мощность продукционных систем обусловлена тем, что человек выводит заключения, имея в своем распоряжении все свои знания, т.е. БЗ огромного объема, в то время как ЭС способны вывести сравнительно небольшое количество заключений, используя заданное множество правил. Из сказанного можно сделать вывод о том, что компонента вывода в ЭС должна быть организована так, чтобы быть способной функционировать в условиях недостатка информации.

Управляющая компонента. Она определяет порядок применения правил, а также устанавливает, имеются ли еще факты, которые могут быть изменены в случае продолжения работы (при немономтонном выводе). Механизм вывода работает циклически, при этом в одном цикле может сработать только одно правило. Схема цикла приведена на рис. 2.3. В цикле выполняются следующие основные операции:

- сопоставление — образец (антецедент) правила сравнивается с имеющимися в РП фактами;

- разрешение конфликтного набора — выбор одного из нескольких правил в том случае, если их можно применить одновременно;

- срабатывание правила — в случае совпадения образца некоторого правила из базы правил с фактами, имеющимися в рабочей памяти, происходит срабатывание правила, при этом оно отмечается в БП;

- действие — изменение содержимого РП путем добавления туда заключения сработавшего правила. Если в заключении содержится директива на выполнение некоторой процедуры, последняя выполняется.

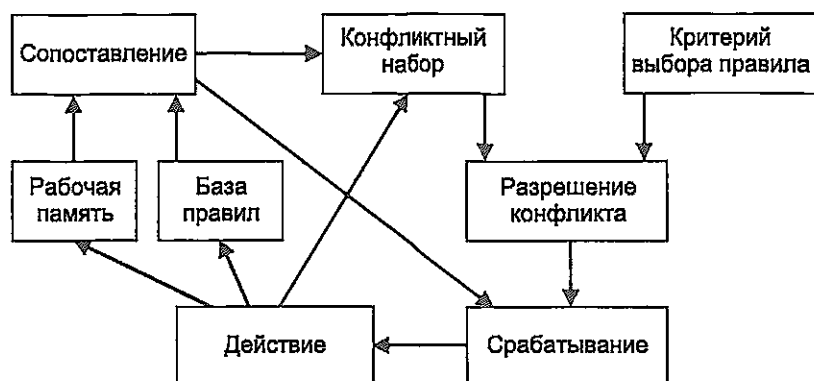


Рис.2.3. Схема цикла работы механизма вывода

Поскольку механизм вывода работает циклически, следует знать о способах завершения цикла. Традиционными способами являются либо исчерпание всех правил из БП, либо выполнение некоторого условия, которому удовлетворяет содержимое рабочей памяти (например, появление в ней какого-то образца), либо комбинация этих способов.

Особенностью ЭС является то, что они не располагают процедурами, которые могли бы построить в пространстве состояний сразу весь путь решения задачи. Траектория поиска решения полностью определяется данными, получаемыми от пользователя в процессе логического вывода.

Рассмотрим простейшие примеры прямого и обратного вывода в системах продукционного типа.

Пример прямого вывода. Пусть в БП имеются следующие правила:

Правило 1. «ЕСЛИ *Двигатель не заводится* И *Фары не горят*, ТО *Сел аккумулятор*».

Правило 2. «ЕСЛИ *Указатель бензина находится на нуле*, ТО *Двигатель не заводится*».

Предположим, что в рабочую память от пользователя ЭС поступили факты: *Фары не горят* и *Указатель бензина находится на нуле*.

Рассмотрим основные шаги алгоритма прямого вывода.

1. Сопоставление фактов из РП с образцами правил из БП. Правило 1 не может сработать, а Правило 2 срабатывает, так как образец, совпадающий с его антецедентом, присутствует в РП.

2. Действие сработавшего Правила 2. В РП заносится заключение этого правила — образец *Двигатель не заводится*.

3. Второй цикл сопоставления фактов в РП с образцами правил. Теперь срабатывает Правило 1, так как конъюнкция условий в его антецеденте становится истинной.

4. Действие Правила 1, которое заключается в выдаче пользователю окончательного диагноза — *Сел аккумулятор*.

5. Конец работы (БП исчерпана).

Пример прямого вывода с конфликтным набором. Теперь допустим, что в БП кроме Правила 1 и Правила 2 присутствует Правило 3:

«ЕСЛИ *Указатель бензина находится на нуле*, ТО *Нет бензина*».

В РП находятся те же факты, что в предыдущем примере.

В результате сопоставления в первом же цикле возможно применение двух правил — Правила 2 и Правила 3, т.е. возникает конфликтный набор и встает задача выбора: какое из этих правил применить первым. Если выберем Правило 2, то в РП добавится факт *Двигатель не заводится* и на следующем шаге опять возникнет конфликтный набор, так как можно будет применить Правило 1 и Правило 3. Если будет выбрано Правило 1, то к заключению *Сел аккумулятор* придем за два шага. При любом другом выборе порядка применения правил к этому же заключению приходим за три шага. Если завершение цикла работы ЭС наступает после просмотра всех правил, то число шагов будет равно трем, причем порядок применения правил не будет иметь какого-либо значения.

Пример обратного вывода. Предположим, что в БП имеется два правила (Правило 1 и Правило 2), а в РП — те же факты, что в предыдущих примерах с прямым выводом.

Алгоритм обратного вывода содержит следующие шаги.

1. Выдвигается гипотеза окончательного диагноза — *Сел аккумулятор*.

2. Отыскивается правило, заключение которого соответствует выдвинутой гипотезе, в нашем примере — это Правило 1.

3. Исследуется возможность применения Правила 1, т.е. решается вопрос о том, может ли оно сработать. Для этого в рабочей памяти должны присутствовать факты, совпадающие с образцом этого правила. В рассматриваемом примере Правило 1 не может сработать из-за отсутствия в РП образца *Двигатель не заводится*. Этот факт становится новой целью на следующем шаге вывода.

4. Поиск правила, заключение которого соответствует новой цели. Такое правило есть — Правило 2.

5. Исследуется возможность применения Правила 2 (сопоставление). Оно срабатывает, так как в РП присутствует факт, совпадающий с его образцом.

6. Действие Правила 2, состоящее в занесении заключения *Двигатель не заводится* в РП.

7. Условная часть Правила 1 теперь подтверждена фактами, следовательно, оно срабатывает, и выдвинутая начальная гипотеза подтверждается.

8. Конец работы.

При сравнении этого примера с примером прямого вывода нельзя заметить преимуществ обратных выводов перед прямыми.

Пример обратного вывода с конфликтным набором. Предположим, что в БП записаны Правило 1, Правило 2, Правило 3 и Правило 4:

«ЕСЛИ *Засорился бензонасос*, ТО *Двигатель не заводится*».

В РП присутствуют те же самые факты: *Фары не горят* и *Указатель бензина находится на нуле*.

В данном случае алгоритм обратного вывода с конфликтным набором включает следующие шаги.

1. Выдвигается гипотеза *Сел аккумулятор*.

2. Поиск правила, заключение которого совпадает с поставленной целью. Это Правило 1.

3. Исследуется возможность применения Правила 1. Оно не может сработать, выдвигается новая подцель *Двигатель не заводится*, соответствующая недостающему образцу.

4. Поиск правил, заключения которых совпадают с новой подцелью. Таких правил два — Правило 2 и Правило 4. Если выберем Правило 2, то дальнейшие шаги совпадают с примером без конфликтного набора. Если выберем Правило 4, то оно не работает, так как в РП нет образца *Засорился бензонасос*. После этого будет применено Правило 2, что приведет к успеху, но путь окажется длиннее на один шаг.

Следует обратить внимание на то, что Правило 3, не связанное с поставленной целью, вообще не затрагивалось в процессе вывода. Этот факт свидетельствует о более высокой эффективности обратных выводов по сравнению с прямыми, так как при обратных выводах существует тенденция исключения из рассмотрения правил, не имеющих отношения к поставленной цели.

В экспертных системах процедуры управления логическим выводом закрыты не только для пользователя, но и для инженера по знаниям, однако о них необходимо иметь представление, чтобы корректно интерпретировать результаты. Для этого нужно знать, в каком виде хранятся знания и как выбираются начальная точка поиска, правила разрешения конфликтов, структуру, с помощью которой хранятся знания. Например, в известном семействе ЭС *OPS* применяется стратегия прямых выводов, эффективность которых существенно повышается благодаря использованию алгоритма согласования *RETE* при генерации конфликтного набора. Суть этого алгоритма сводится к следующему: каждый раз при добавлении в РП нового образца проверяется правило, в котором он используется, и если образец удовлетворяет антецеденту некоторого правила, то он запоминается именно в этом качестве. В конфликтный набор правило включается только в том случае, если добавление образца удовлетворяет всем условиям. Для разрешения конфликтов в системах семейства *OPS*, а также в других системах с прямыми выводами широкое распространение получил метод разрешения конфликтов *LEX*, в котором предпочтение отдается правилам со ссылкой на самый последний сгенерированный образец. Если таких правил несколько, то среди них выбирается правило с наибольшим числом условий в антецеденте.

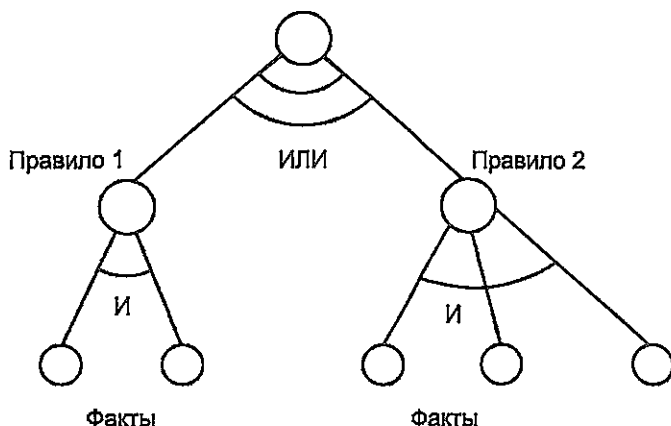


Рис. 2.4. Простейший фрагмент структуры И-ИЛИ-графа

В больших ЭС продукционного типа все множество знаний обычно хранится в виде древовидной структуры, называемой И-ИЛИ-графом. Фрагменты такой структуры приведены на рис. 2.4 и 2.5. Классическая форма produkcji предполагает наличие в антецеденте только связки И. На практике классическая форма может быть расширена, например, введением связки ИЛИ в условную часть либо включением в антецедент вычислений на основании содержимого рабочей памяти и т.п. Если существует множество правил, из которых выводится одно и то же заключение

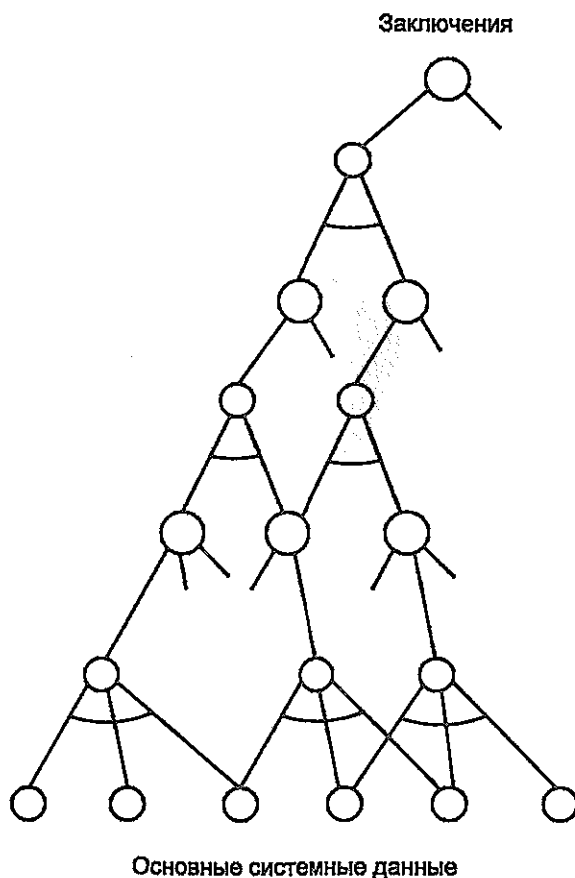


Рис. 2.5. Фрагмент структуры И-ИЛИ-графа
продукционной экспертной системы

ние, то, выполнив операцию дизъюнкции над всеми заключениями, полученными с помощью этих правил, можно показать отношение между результатом отдельного вывода и данными, на основании которых делается вывод.

С помощью И-ИЛИ-графа обратный вывод в ЭС продукционного типа можно представить как проблему поиска определенного пути на графе. Выбор одной из связок ИЛИ соответствует разрешению конфликтного набора, при этом не безразличен порядок оценки условий в антецеденте, соединенных связкой И. Задачи и стратегии поиска на И-ИЛИ-графах широко освещены в литературе [4, 5] и не будут рассматриваться здесь подробно. Однако следует остановиться на способах повышения эффективности поиска, так как в системах, имеющих практическую ценность, насчитываются сотни правил, и следует знать, с помощью каких стратегий управления выводом можно минимизировать время решения задач.

Стратегия поиска в глубину. При выборе очередной подцели в процессе обратного вывода предпочтение всегда, когда возможно, отдается той, которая соответствует следующему, более детальному уровню описания задачи. Например, система диагностики, сделав на основании известных симптомов предположение о причинах неисправности, будет запрашивать уточняющие признаки и симптомы до тех пор, пока полностью не подтвердит (опровергнет) выдвинутую гипотезу. Пример организации поиска в глубину показан на рис. 2.6, где цифрами обозначены номера шагов поиска.

Стратегия поиска в ширину. При поиске в ширину сначала анализируются все симптомы (факты), находящиеся на одном уровне пространства состояний задачи, даже если они относятся к разным целям (подцелям), и только после этого происходит переход к поиску симптомов следующего уровня. На рис. 2.7 показаны шаги поиска в ширину, обозначенные номерами, указанными в вершинах. На рисунке представлена стратегия обратного вывода на том же И-ИЛИ-графе, который приведен и на рис. 2.6. Алгоритм поиска в глубину более эффективен в отношении времени поиска и обработки знаний, однако он характеризуется более высоким риском потери перспективных решений по сравнению с поиском в ширину.

Разбиение на подзадачи. Декомпозиция дает положительный эффект только для хорошо структурированных областей знаний,

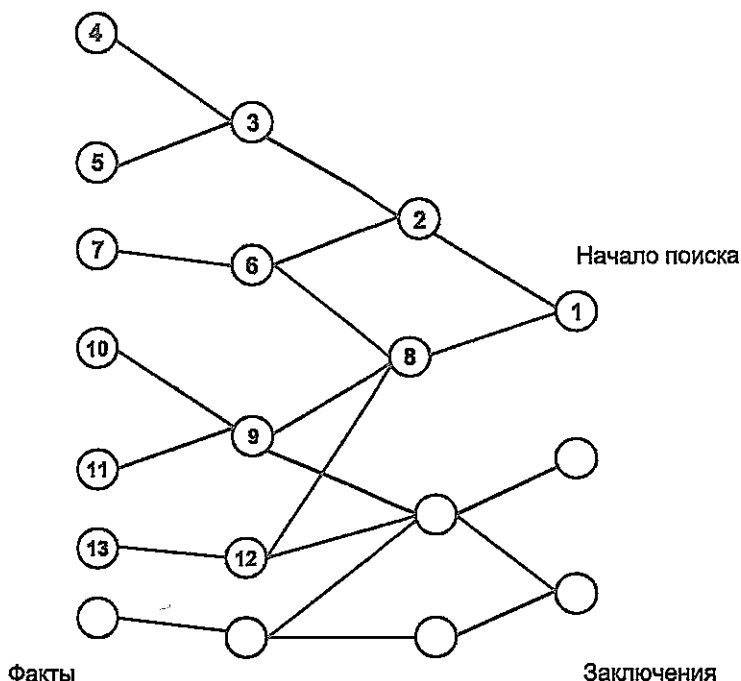


Рис. 2.6. Поиск в глубину при обратном выводе

так как применение этой стратегии основано на правильном понимании сущности задачи и возможности ее представления в виде системы иерархически связанных целей-подцелей, причем разбиение на подзадачи необходимо выполнить оптимальным способом.

α - β -алгоритм. С помощью этого алгоритма исходная задача сводится к уменьшению пространства состояний путем удаления в нем ветвей, неперспективных для поиска успешного решения, т.е. просматриваются только те вершины, в которые можно попасть в результате следующего шага, после чего неперспективные направления исключаются. Например, в БЗ продукционной системы, заполненной знаниями о животном мире, не следует искать животных, не относящихся к млекопитающим, в направлении, берущем начало от вершины, определяющей млекопитающих. Данная стратегия является определенным компромиссом между поиском в ширину и поиском в глубину.

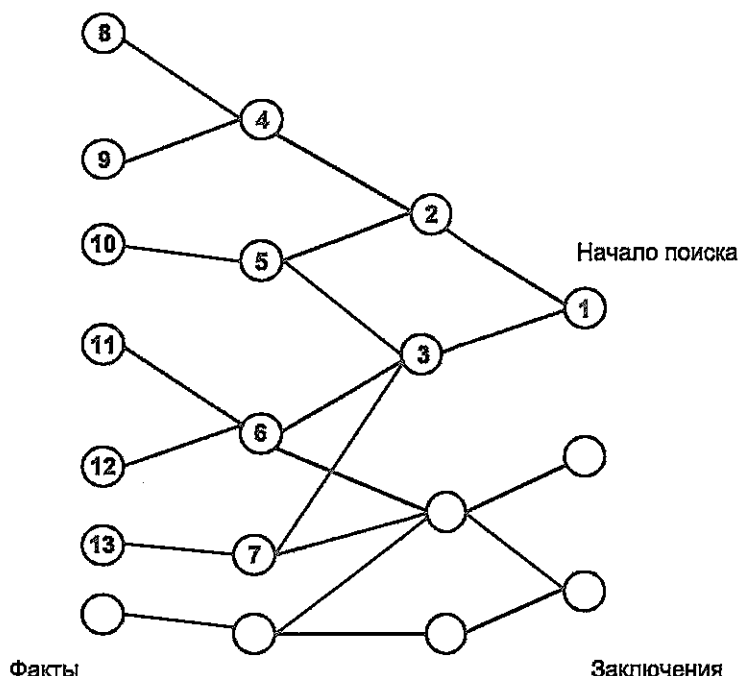


Рис. 2.7. Поиск в ширину при обратном выводе

Для ее успешной реализации следует располагать дополнительными эвристическими знаниями, которые используются при выборе перспективных направлений. Впечатляющий пример применения варианта этой стратегии продемонстрирован разработчиками системы *Deer Blue*, сумевшей обыграть лучшего шахматиста планеты.

2.3.3. ОБРАБОТКА ЗНАНИЙ В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ С ФРЕЙМОВЫМ ПРЕДСТАВЛЕНИЕМ

В интеллектуальных системах с фреймовым представлением знаний используются три способа управления логическим выводом: демоны, присоединенные процедуры и механизм наследования. Последний можно назвать единственным основным меха-

низмом вывода, которым оснащены фреймовые (объектно-ориентированные) системы.

Управленческие функции механизма наследования заключаются в автоматическом поиске и определении значений слотов фреймов нижележащих уровней по значениям слотов фреймов верхних уровней, а также в запуске присоединенных процедур и демонов.

Присоединенные процедуры и демоны позволяют реализовать любой механизм вывода в системах с фреймовым представлением знаний. Однако эта реализация имеет конкретный характер и требует значительных затрат труда проектировщиков и программистов. Рассмотрим простой пример. В табл. 2.4 показана структура фрейма «Научная конференция».

Таблица 2.4

Фрейм «Научная конференция»

Имя слота	Значение слота	If-needed	If-added	If-removed
Дата	1.02. 10 : 10		ЗАКАЗ	
Место проведения	Аудитория 209			
Тема доклада	Прогнозирование тенденций в экономике			
Докладчик	Иванов И.И.			
		КТО?		

Демон ЗАКАЗ — это процедура, которая автоматически запускается при попытке подстановки значения в слот с именем *Место проведения*. Ее главное назначение состоит в проверке возможности заказа аудитории на нужное время. Такая процедура на языке LISP может выглядеть примерно так:

```
LISP proc ЗАКАЗ(Название конференции, Место проведения, Дата)
if возможно(Место проведения, Дата)
then заказать (Название конференции, Место проведения, Дата)
else сообщение («Заказ невозможен», Название конференции)
end.
```

Демон КТО? автоматически запускается при обращении к слоту Докладчик, если значение этого слота не определено. Основное содержание данной процедуры — генерация запроса к пользователю типа «Кто выступает?», получение ответа и его запись в качестве значения слота.

Реализация вывода с помощью присоединенных процедур требует наличия механизма обмена информацией между фреймами. В качестве такого механизма обычно используется механизм сообщений.

На рис. 2.8 схематично показан обмен информацией между фреймами **AA** и **BB** во время исполнения присоединенной процедуры **CALC**, при этом вызывается процедура **MEAN**, расположенная в фрейме **BB**.

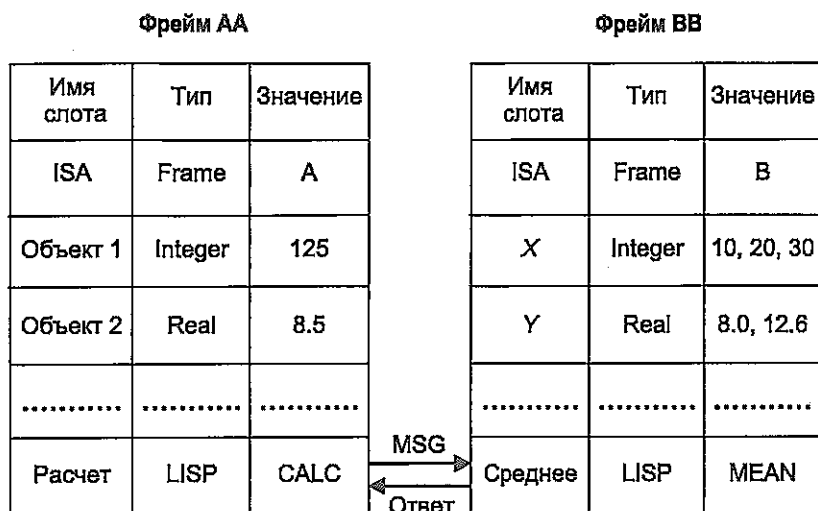


Рис 2.8. Обмен информацией между фреймами

Допустим, что процедура *CALC(result)* выполняет расчет, в процессе которого происходит обращение к фрейму **BB** с использованием команды **MSG**, реализующей передачу сообщения в другой фрейм.

```
LISP proc CALC(result)
.....
MSG(Среднее, BB, X)
.....
end.
```

Команда **MSG** имеет три параметра: 1 — имя слота, к которому происходит обращение (в данном случае значением слота

Среднее является присоединенная процедура *MEAN*); 2 — имя фрейма, в котором содержится необходимая информация (*BB*); 3 — имя слота-параметра, в котором находятся данные для расчета (*X*). Таким образом, запуск процедуры *CALC* вызовет исполнение следующих действий: передача сообщения во фрейм *BB* на запуск процедуры *MEAN*, которая найдет среднее арифметическое чисел, записанных в слоте *X*; вычисленное значение будет записано в переменную *result* и передано в *CALC* как ответ на сообщение *MSG*.

Итак, в интеллектуальных системах с фреймовым представлением знаний невозможно четко отделить процедурные знания от декларативных, поскольку присоединенные процедуры и демоны одновременно являются и знаниями, и средствами управления логическим выводом. На рис. 2.9 схематично показаны средства управления выводом во фреймовой системе. Возможность организации выводов любого типа является существенным преимуществом фреймовых систем по сравнению с продукционными и логическими. Не менее важным достоинством является большее сходство этой модели представления знаний со структурой знаний в памяти человека. Вместе с тем практиче-

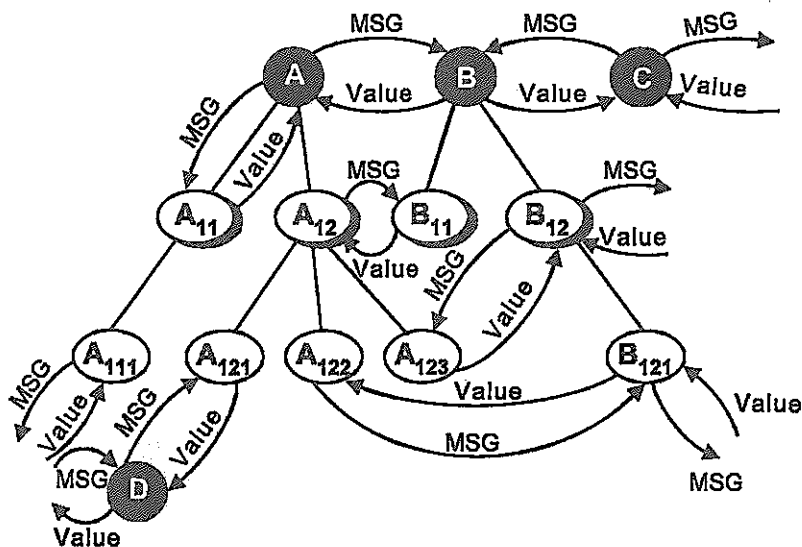


Рис. 2.9. Средства управления выводом в интеллектуальной системе фреймового типа

кая реализация фреймовых систем сопряжена со значительной трудоемкостью как на этапе проектирования, так и при реализации. Поэтому стоимость промышленных экспертных систем фреймового типа на порядок превосходит стоимость продукционных систем.

2.4.

ПРИМЕРЫ РАЗРАБОТКИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ С ПРИМЕНЕНИЕМ ТИПИЧНЫХ МОДЕЛЕЙ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Рассмотрим примеры применения традиционных способов представления и обработки знаний в небольших интеллектуальных системах, которые могут разрабатываться студентами во время курсового проектирования по предмету «Интеллектуальные информационные системы». Опускаем все, что связано с освоением и применением готовых промышленных оболочек ЭС, а также задачи, которые могут быть решены с использованием языка PROLOG. Такие приложения требуют знакомства со специальной информацией о соответствующих программных продуктах, которая широко представлена в литературе по программированию и использованию готовых программных средств. Студенты старших курсов высших учебных заведений, владеющие программированием, могут самостоятельно разрабатывать оболочки небольших ЭС логического и продукционного типа, а также писать компьютерные игры. Разработка игр пользуется большой популярностью, так как позволяет увидеть и оценить результаты своего труда, причем подобные задачи не требуют обширных знаний предметной области, без которых невозможно проектировать ЭС. Набор знаний, используемых в играх, четко ограничен и обычно хорошо известен разработчику компьютерной реализации.

Опыт показывает, что помимо написания игр для одного участника («пятнашки», головоломки, прохождение лабиринтов и т.п.) вполне посильной и интересной задачей является написание игр со многими участниками, в частности карточных и позиционных игр, в которых кроме компьютерного игрока участвует человек.

2.4.1. ОБОЛОЧКА ЭКСПЕРТНОЙ СИСТЕМЫ ПРОДУКЦИОННОГО ТИПА

Разрабатывать оболочки небольших ЭС можно практически на любом языке процедурного или объектно-ориентированного программирования, а также на языках логического типа, наиболее известным представителем которых является PROLOG. В последнем случае программист освобождается от реализации механизма логического вывода, который является частью PROLOG-систем. В его задачи входят поиск, структурирование и представление знаний средствами логики предикатов, создание БЗ и написание программы, управляющей процессом обработки знаний.

При разработке оболочек ЭС продукционного типа на процедурных языках программирования кроме создания БЗ необходимо реализовать механизм прямого или обратного логического вывода на основе правила *Modus Ponendo Ponens* и организацию интерфейса с пользователем. В базе знаний должны храниться список фактов в виде текстовых знаний и набор правил, показывающих связи между фактами в рассматриваемой предметной области.

На рис. 2.10 и 2.11 приведены экранные формы оболочки ЭС, ориентированной на задачи диагностики, где показано взаимо-

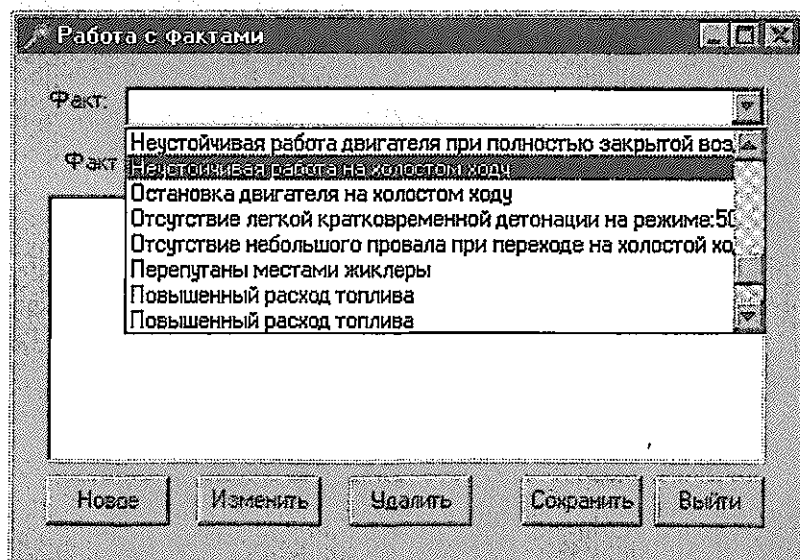


Рис. 2.10. Окно ввода и изменения фактов в ЭС продукционного типа

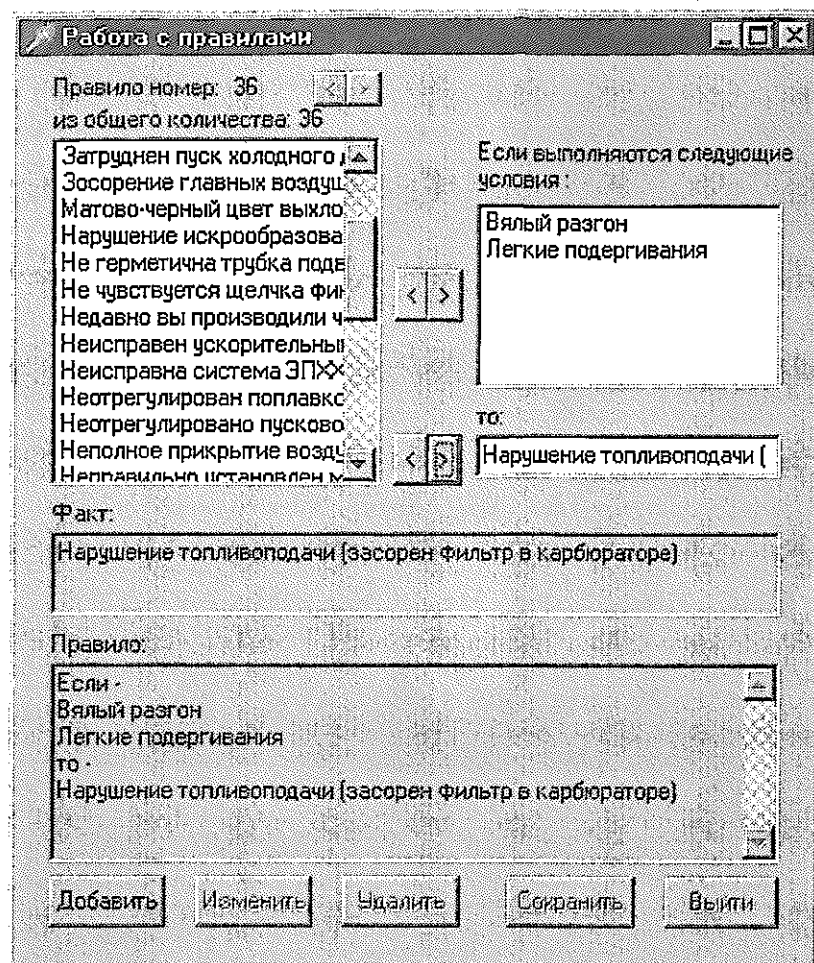


Рис. 2.11. Окно ввода и редактирования правил
в ЭС для решения задач диагностики

действие ЭС с пользователем на этапе ввода и редактирования знаний.

В разработанной системе реализована стратегия прямого логического вывода, в процессе которого осуществляется горизонтальный поиск фактов на заданной в виде И-ИЛИ-графа структуре знаний предметной области. При этом предусмотрено созда-

ние специального файла, в котором хранятся вопросы, задаваемые пользователю в процессе решения задач диагностики.

Процесс функционирования системы диагностики, реализованной на языке TurboPascal 7.0 и работающей под управлением MS DOS, проиллюстрирован на рис. 2.12. В данной версии системы не предусмотрено формирование файла вопросов, пользователь просто указывает на наличие или отсутствие предъявляемого симптома. Система предоставляет пользователю объяснение полученных результатов, показывая наборы фактов, на основании которых получены те или иные заключения.

Программная реализация процедур поиска и обработки знаний не вызывает принципиальных трудностей. В простейшем случае БЗ представляет собой два файла, в одном из которых хра-

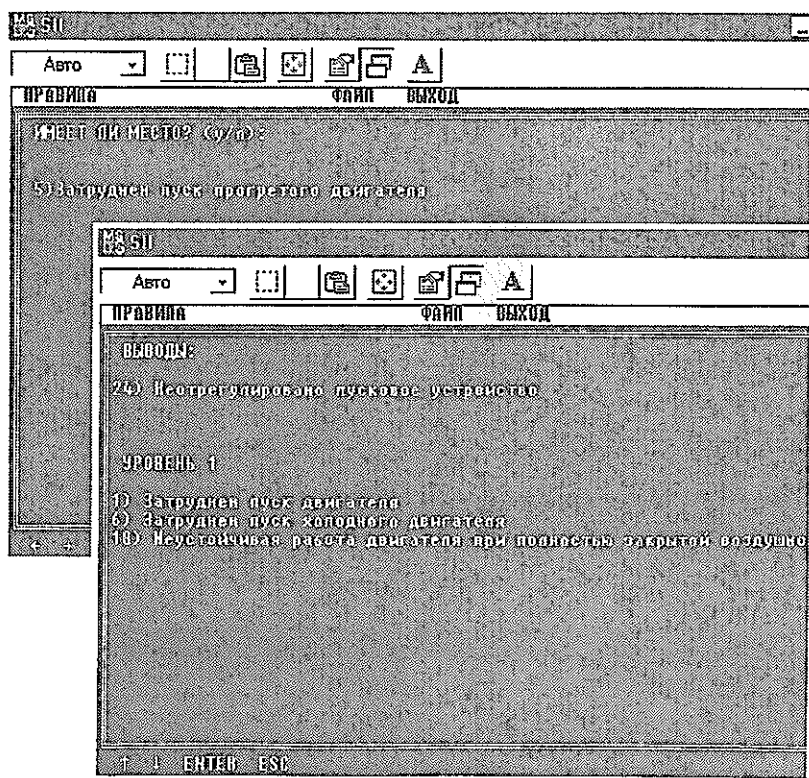


Рис. 2.12. Пример решения конкретной задачи диагностики

няются текстовые описания фактов, используемых в антецедентах и консеквентах правил, а в другом — правила, устанавливающие взаимосвязи фактов. Пользователи подобных систем должны подготовить конкретную информацию и создать необходимые файлы БЗ. При этом порядок записи правил может оказывать влияние на порядок вопросов, которые ЭС будет задавать пользователю. Поэтому целесообразно построить И-ИЛИ-граф, отражающий взаимосвязи фактов, и с его помощью определить *первичные* факты (симптомы), которые встречаются только в антецедентах правил, *вторичные*, присутствующие в условных частях одних правил и в заключениях других, и *диагнозы* — факты, являющиеся окончательными заключениями. Совершая обход И-ИЛИ-графа в направлении от первичных фактов к заключениям, ЭС задает пользователю вопросы, на основании полученных ответов формирует содержимое рабочей памяти и делает соответствующие выводы (см. разд. 2.3.2).

Для разрешения конфликтов в разработанных системах используются алгоритм *LEX* (см. разд. 2.3.2), а также другие эвристические способы. Применение программ подобного типа приносит пользу в процессах извлечения знаний и в экспериментальных исследованиях возможностей их представления средствами продукционной модели. При выборе адекватной модели представления знаний в создаваемых ИИС часто возникает необходимость рассмотрения альтернативных вариантов представления, при этом использование простых в обращении инструментальных программных средств позволяет с минимальными затратами осуществить экспериментальную проверку продукционной модели.

2.4.2. ПРОГРАММИРОВАНИЕ КАРТОЧНОЙ ИГРЫ СО МНОГИМИ УЧАСТНИКАМИ

Карточные игры отличаются большим разнообразием правил и обычно включают более двух участников. Это делает их сложнее многих позиционных игр и требует построения правил принятия решений, которые могут изменяться в течение игры. То обстоятельство, что правила принятия решений в играх являются эмпирическими, могут отличаться у разных игроков и меняться по ходу игры, делает увлекательным процесс разработки подоб-

ных программ. При разработке компьютерных игр практически не удастся обойтись без отладочных экспериментов. Несмотря на то что разработчики хорошо знакомы с играми, которые они программируют, первоначально сформулированные ими правила принятия решений или оценочные функции, как правило, не позволяют компьютерному игроку играть на уровне человека. Следует отметить, что большинство разработчиков игр считает делом чести довести квалификацию своего компьютерного игрока хотя бы до собственного уровня. Для этого в процессе тестирования и доводки экспериментально подбираются структура и параметры оценочных функций, вырабатываются оригинальные стратегии и формулируются различные наборы правил.

В качестве примера разработки рассмотрим старинную русскую карточную игру в «Подкидного дурачка». Правила этой игры хорошо известны, поэтому не будем их описывать. Так как количество возможных в игре ситуаций достаточно велико, а возможные действия противников плохо предсказуемы, попытки определить путь достижения цели компьютерного игрока в пространстве состояний задачи не имеют смысла. В данном случае желательно выработать принципы оценивания текущей ситуации, позволяющие судить о степени приближения к конечной цели, которая заключается в том, чтобы первому среди всех игроков остаться без карт на руках. Стремление к минимизации количества карт в процессе игры далеко не всегда обеспечивает конечный успех игрока, поэтому представляется целесообразным ввести понятие *относительного потенциала игрока*, который следует поддерживать максимально высоким в течение игры. Потенциал можно вычислить по следующей формуле:
$$M_b = \sum_{i=1}^n (\text{Rank}_i + 9K_i) / n,$$

где n — количество карт на руках; Rank_i — ранг i -й карты, принимающий значения от 1 до 9 в соответствии с достоинством карты; K_i — признак козырной карты, $K_i = 1$, если i -я карта принадлежит к козырной масти, и $K_i = 0$ — в противном случае.

В течение игры ряд событий происходит по нормативным правилам, т.е. без учета мнения игроков. К таким событиям относятся первоначальная раздача карт игрокам, взятие новых карт из колоды, а также наступление событий *Ходить* и *Биться*. Событие *Подкидывать* наступает по воле игрока. Активные действия игрока могут быть реализованы различными способами. Для выбора способа действия из всех возможных альтернатив необходимо

сформулировать правила принятия решений, в которых можно использовать значение относительного потенциала M_b .

В качестве возможных альтернативных вариантов основных событий были приняты следующие: *Ходить*={*ходить младшей; ходить парными; ходить козырем; ходить неудобной для противника мастью*}; *Подкидывать*={*подкидывать; не подкидывать*}; *Биться*={*биться младшей по масти; биться младшим козырем; биться козырем того же ранга; принять*}.

Если флаг текущего события сигнализирует о необходимости совершения хода, то начинается анализ возможных вариантов решений в соответствии со сформированным набором правил. Следует подчеркнуть, что вычисление и сравнение значений относительного потенциала выполняются для альтернативных вариантов множества карт, которое останется на руках у игрока после совершения анализируемых действий. Лучшей альтернативой является та, которая обеспечивает максимальное значение относительного потенциала остающихся на руках карт. Таким образом, правило выбора решения A_i при наступлении события *Ходить* можно записать в виде:

$$A^* = \max_{A_i \in \text{Ходить}} M_b(A_i).$$

Решение о *подкидывании* карт принимается на основе сравнения возможных альтернативных действий с текущим значением потенциала игрока. Правило выбора действия в этом случае может учитывать близость финала игры и склонность игрока к риску. При наступлении события *Биться* альтернативных вариантов бывает меньше, чем при заходе. Решение о принятии карт может оказаться неизбежным, если игрок не имеет на руках карт, которыми можно отбиться. В этом случае не требуется вычисление значения потенциала. Кроме того, такое решение может быть принято сознательно в том случае, если игрок способен побить карты противника, но значение потенциала резко снижается и состояние игры не предвещает скорого финала.

Формула вычисления относительного потенциала свидетельствует о невысокой вероятности выбора альтернатив *ходить козырем* или *биться козырем*, так как это часто приводит к существенному уменьшению потенциала. Поэтому реализация описанной тактики позволяла программе достаточно хорошо играть на начальной стадии, но часто приводила к проигрышу в конце игры,

так как *компьютерный игрок* не хотел расставаться с козырными картами, стремясь сохранить высокий потенциал. Этот факт был обнаружен в процессе тестирования и послужил причиной модификации правил принятия решений таким образом, чтобы тактика изменялась по ходу игры. В данной версии программы основанием для изменения тактики является мера близости к финалу игры. В качестве оценки такой меры можно использовать количество карт в колоде или количество вышедших из игры карт. В рассматриваемой реализации программы сигналом для изменения тактики являлось исчерпание карт в колоде. При наступлении этого события компьютерный игрок начинает принимать решения по другим правилам, ориентированным на минимизацию количества карт на руках. Программа с изменяющейся тактикой уже не совершает упомянутых выше ошибок.

Несмотря на кажущуюся несерьезность, разработка игровых программ является хорошим тренингом для будущих и уже состоявшихся специалистов в области разработки интеллектуальных информационных систем.

Литература

1. *Гаврилова Т. А., Хорошевский В. Ф.* Базы знаний интеллектуальных систем. — СПб.: Питер, 2000.
2. Логический подход к искусственному интеллекту: от классической логики к логическому программированию: Пер. с фр./А. Тейз, П. Грибомон, Ж. Луи и др. — М.: Мир, 1990.
3. *Нильсон Н. Дж.* Искусственный интеллект. Методы поиска решений. — М.: Мир, 1973.
4. *Осинов Г. С.* Приобретение знаний интеллектуальными системами. — М.: Наука, 1997.
5. *Осуга С.* Обработка знаний: Пер. с яп. — М.: Мир, 1989.
6. *Поспелов Д. А.* Моделирование рассуждений. Опыт анализа мыслительных актов. — М.: Радио и связь, 1989.
7. Построение экспертных систем: Пер. с англ. / Под ред. Ф. Хейес-Рота, Д. Уотермена, Д. Лената. — М.: Мир, 1987.
8. Представление и использование знаний: Пер. с яп./Под ред. Х. Уэно, М. Исидзука. — М.: Мир, 1989.
9. *Таунсенд К., Фохт Д.* Проектирование и программная реализация экспертных систем на персональных ЭВМ: Пер. с англ. — М.: Финансы и статистика, 1990.

10. Цейтин Г. С. Программирование на ассоциативных сетях // ЭВМ в проектировании и производстве. — Л.: Машиностроение. — 1985. — Вып. 2.
11. Элти Дж., Кумбс М. Экспертные системы: концепции и примеры: Пер. с англ. — М.: Финансы и статистика, 1987.
12. Durkin J. Expert System: Catalog of applications. — ICS, USA, 1998.
13. Sisdia R., War Kentin M. AI in business and management // PC AI, Jan-Feb, 1992.

Контрольные вопросы и задания

1. Чем отличаются знания от данных? Приведите определения знаний.
2. Дайте характеристику основных признаков, по которым классифицируются знания (природа знаний, способ приобретения знаний, тип представления знаний).
3. Расскажите о логических способах представления знаний. Укажите преимущественную область применения логической модели.
4. Проведите формализацию небольшого фрагмента знаний средствами логики высказываний (логики предикатов).
5. Охарактеризуйте продукционную модель представления знаний. Приведите примеры представления знаний правилами. В чем отличия между продукционными системами с прямыми, обратными и двунаправленными выводами?
6. Опишите фреймовую модель представления знаний. Приведите пример фреймового представления.
7. Охарактеризуйте модель представления знаний в виде семантической сети. Расскажите об основных видах используемых в этой модели отношений.
8. Приведите примеры логического вывода с использованием правил *Modus Ponendo Ponens* и *Ценное заключение*.
9. Докажите предложенную тавтологию семантическим (синтаксическим) методом.
10. Расскажите о теоремах логики и их использовании в ИИС. Приведите примеры.
11. Опишите возможности применения в логическом выводе операции эквивалентности. Приведите примеры тавтологий с эквивалентностями.
12. Опишите стратегию доказательства с введением допущения. Приведите пример.

13. Рассмотрите пример доказательства путем приведения к противоречию.
14. Расскажите о стратегии доказательства методом резолюции. Приведите пример.
15. Опишите функционирование механизма вывода продукционной ЭС и охарактеризуйте его составляющие: компоненту вывода и управляющую компоненту.
16. Сформулируйте собственные примеры прямого и обратного вывода в ЭС продукционного типа.
17. Приведите пример представления знаний в виде И-ИЛИ-графа.
18. Опишите и представьте в графическом виде стратегии поиска решений: в глубину, ширину, разбиением на подзадачи, на основе α - β -алгоритма.
19. Расскажите о способах организации логического вывода в интеллектуальных системах с фреймовым представлением знаний.
20. Разработайте программную реализацию интеллектуальной системы с продукционным представлением знаний и механизмом вывода на базе правила *Modus Ponendo Ponens*.
21. Выполните формализацию знаний средствами продукционной модели, которые могут использоваться в интеллектуальной системе для поддержки задач диагностики экономического и финансового состояния предприятия (других задач).
22. Выполните представление знаний средствами описанных моделей для известной Вам игры.

НЕЧЕТКИЕ ЗНАНИЯ И СПОСОБЫ ИХ ОБРАБОТКИ

Абсолютная ясность — одна из форм
полного тумана.

*Из фильма «Семнадцать
мгновений весны»*

При разработке интеллектуальных систем знания о конкретной предметной области, для которой создается система, редко бывают полными и абсолютно достоверными. Даже количественные данные, полученные путем достаточно точных экспериментов, имеют статистические оценки достоверности, надежности, значимости и т.д. Информация, которой заполняются экспертные системы, получается в результате опроса экспертов, мнения которых субъективны и могут расходиться. Наряду с количественными характеристиками в базах знаний ИИС должны храниться качественные показатели, эвристические правила, текстовые знания и т.д. При обработке знаний с применением жестких механизмов формальной логики возникает противоречие между нечеткими знаниями и четкими методами логического вывода. Разрешить это противоречие можно или путем преодоления нечеткости знаний (когда это возможно), или с использованием специальных методов представления и обработки нечетких знаний.

3.1. ВИДЫ НЕЧЕТКОСТИ ЗНАНИЙ, СПОСОБЫ ИХ УСТРАНЕНИЯ И/ИЛИ УЧЕТА В ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

Некорректные задачи существуют практически в любой проблемной области. Там, где есть некорректные и слабоструктурированные задачи, можно ожидать эффекта от применения интеллектуальных систем. В области некорректных задач точные знания о проблеме получить невозможно или нельзя это сделать сразу, поэтому там применяется подход, суть которого состоит в по-

степенном приближении к полному набору необходимых знаний. При этом используются методы представления нечетких знаний и механизмы вывода, работающие в их среде.

Смысл термина *нечеткость* многозначен. Трудно претендовать на исчерпывающее определение этого понятия, поэтому рассмотрим лишь основные его компоненты, к которым относятся следующие:

- недетерминированность выводов;
- многозначность;
- ненадежность;
- неполнота;
- неточность.

Недетерминированность выводов. Это характерная черта большинства систем искусственного интеллекта. Недетерминированность означает, что заранее путь решения конкретной задачи в пространстве ее состояний определить невозможно. Поэтому в большинстве случаев методом проб и ошибок выбирается некоторая цепочка логических заключений, согласующихся с имеющимися знаниями, а в случае если она не приводит к успеху, организуется перебор с возвратом для поиска другой цепочки и т.д. Такой подход предполагает определение некоторого первоначального пути. Для решения подобных задач предложено множество эвристических алгоритмов [6, 14]. Рассмотрим один из них — классический алгоритм A^* , разработанный на этапах становления искусственного интеллекта.

В алгоритме A^* используются оценочные функции, построенные на основе априорных оценок стоимости пути до целевого состояния. Такие оценки, по сути дела, тоже представляют собой эвристические знания. Для поиска в пространстве состояний используются дерево поиска и методы горизонтального (в ширину) и вертикального (в глубину) поиска на этом дереве. Основные шаги и понятия алгоритма рассмотрим на примере игры в «8», являющейся усеченной версией игры в «15». Целью игры является переход из некоторого начального состояния в конечное, как показано на рис. 3.1.

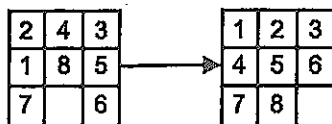


Рис. 3.1. Переход из начального состояния в конечное в игре «8»

В этой игре в качестве основного объекта удобнее рассматривать не передвигаемые шашки, а перемещение пустого квадрата. При этом можно определить четыре основных оператора, выполняемых над пустым квадратом:

- перемещение пустого квадрата влево;
- перемещение пустого квадрата вверх;
- перемещение пустого квадрата вниз;
- перемещение пустого квадрата вправо.

Оценочная функция $f(n)$ будет формироваться как стоимость оптимального пути к цели из начального состояния через n вершин дерева поиска. Дерево поиска для данного примера показано на рис. 3.2. Значение оценочной функции в n -й вершине можно представить как сумму двух составляющих $f(n) = g(n) + h(n)$, где $g(n)$ — стоимость оптимального пути от первой вершины до n -й, а $h(n)$ — стоимость оптимального пути от n -й вершины до цели. Для простоты будем считать, что стоимость перемещения одной шашки (или пустого квадрата) равна 1. Оптимальным будет путь, имеющий минимальную стоимость. Точное значение $f(n)$ невозможно знать в процессе поиска, поэтому введем априорную оценку значения функции: $\hat{f}(n) = g(n) + \hat{h}(n)$, где $g(n)$ — глубина пройденного пути на дереве поиска от 1-й до n -й вершины; $\hat{h}(n)$ — априорное значение $h(n)$.

Основная проблема заключается в определении второй компоненты $h(n)$, так как этот путь еще не пройден. В качестве априорной оценки $\hat{h}(n)$ можно, например, взять число шашек, находящихся не на своих местах на n -м шаге поиска. Сформировав таким образом оценочную функцию, определим стратегию выбора вершин (применения операторов), в которых значения функции минимальны. Результат поиска показан на рис. 3.2, где цифры в кружках показывают последовательность переходов из начального состояния в конечное. Проанализируем основные шаги алгоритма.

1. Рассматриваем все возможные операторы над пустым квадратом в начальном состоянии и выбираем вариант с наименьшим значением $\hat{h}(n)$.

2. Применяем выбранный оператор, в результате получаем новое состояние.

3. Создаем вершины следующего уровня иерархии, анализируя применение всех возможных операторов для перехода в новое состояние.

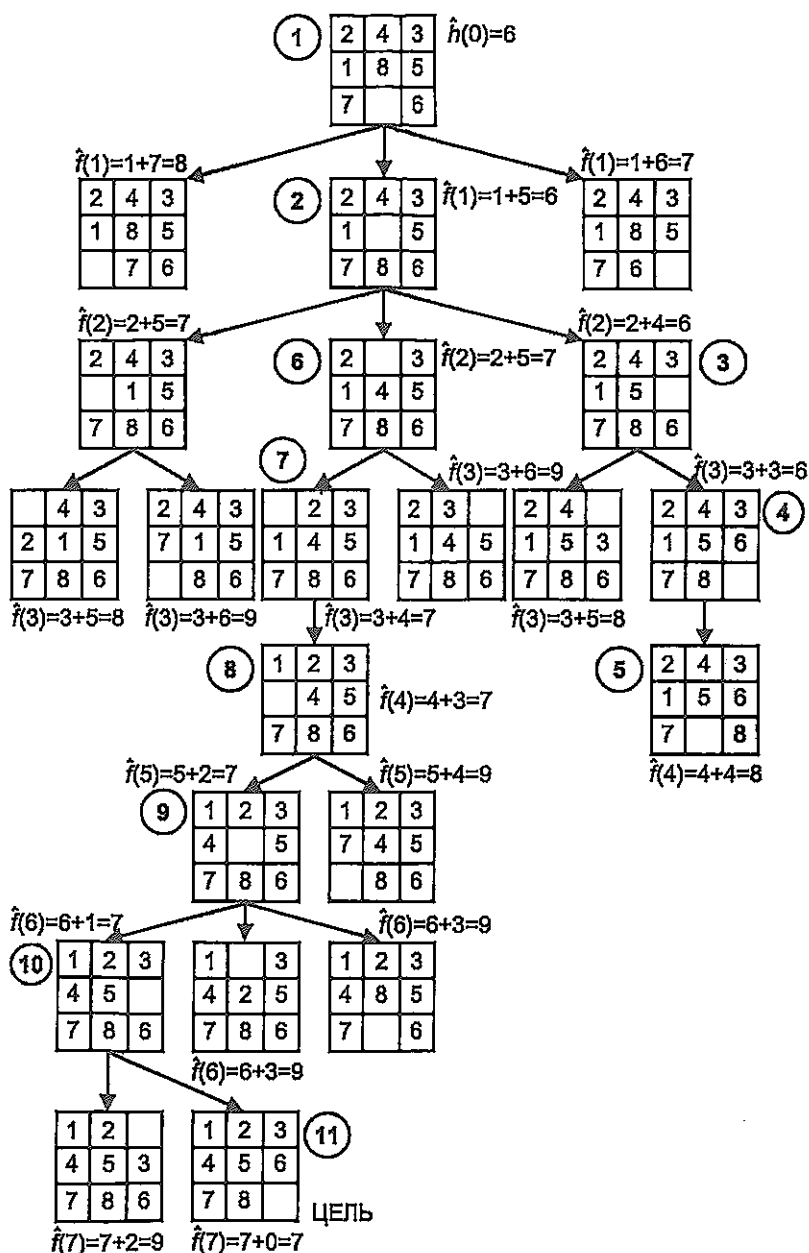


Рис. 3.2. Дерево поиска для игры в «8»

4. Выбираем состояние с наименьшим значением $\hat{h}(n)$.

5. Повторяем перечисленные действия до тех пор, пока не достигнем цели.

В данном случае важно, что $\hat{h}(n) \leq h(n)$, так как если априорная оценка стоимости оптимального пути не превышает истинной стоимости, то нахождение оптимального пути гарантировано. Это условие можно интерпретировать следующим образом: цель поиска не будет достигнута, пока число перемещений меньше числа пашек, находящихся не на своих местах.

Если $\hat{h}(n)$ выбрать по-другому, например $\hat{h}(n)=0$, то будем осуществлять горизонтальный поиск на дереве состояний задачи, при котором раскрываются все вершины нижележащего уровня.

Таким образом, *недетерминированность выводов* — черта, органически присущая интеллектуальным системам, т.е. неустранимая компонента нечеткости знаний. Ее следует учитывать при выработке эффективных способов представления и хранения знаний, а также при построении алгоритмов поиска и обработки знаний, которые позволяют получить решение задачи за наименьшее число шагов. Для построения таких алгоритмов обычно применяются эвристические метазнания (знания о знаниях).

Многозначность. Многозначность интерпретации — обычное явление в задачах распознавания. При понимании естественного языка серьезными проблемами становятся многозначность смысла слов, их подчиненности, порядка слов в предложении и т. п. Проблемы понимания смысла возникают в любой системе, взаимодействующей с пользователем на естественном языке. Распознавание графических образов также связано с решением проблемы многозначной интерпретации. При компьютерной обработке знаний многозначность необходимо устранять путем выбора правильной интерпретации, для чего разработаны специ-

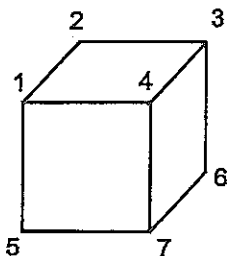


Рис. 3.3. Распознаваемое изображение

альные методы. Рассмотрим один из таких методов — *метод релаксации*, предназначенный для систематического устранения многозначности при интерпретации изображений [11].

Устранение многозначности достигается с помощью циклических операций фильтрации. Рассмотрим пример: Пусть имеется черно-белое изображение (рис 3.3), которое должен распознать компьютер.

Одним из этапов распознавания объекта является интерпретация смысла линий. Для идентификации граней введем следующие метки:

- (+) — выпуклая грань;
- (-) — вогнутая грань;
- (→) — справа от стрелки находится видимая поверхность.

Первый цикл метода релаксации осуществляет «локальный взгляд» на каждую вершину, при этом будем иметь интерпретации вершин 1–7, показанные на рис. 3.4.

Если при интерпретации заданного контура рассматривать сразу две соседние вершины, то они должны иметь общую грань с одной и той же меткой, при этом число вариантов интерпретации уменьшается. Интерпретация вершин, для которых число вариантов изменилось, приведена на рис. 3.5.

Подобную операцию называют *фильтрацией*. Ее можно применять неоднократно в целях устранения многозначности. В рассматриваемом примере после однократной фильтрации число интерпретаций вершин сократилось, поэтому фильтрацию можно применить еще раз (критерием окончания итерационного цикла в данном методе является неизменность вариантов интерпретации на двух последовательных итерациях). Результат второго применения фильтрации показан на рис. 3.6.

Дальнейшее применение фильтрации не приводит к сокращению числа вершин-кандидатов для интерпретации. Оставшиеся кандидаты позволяют составить четыре варианта прямоугольного параллелепипеда:

- 1 — не прилегающий ни к одной из поверхностей (висячий);
- 2, 3, 4 — прилегающий к поверхности одной из трех невидимых граней.

С другими методами устранения многозначности можно более подробно познакомиться в [11, 12].

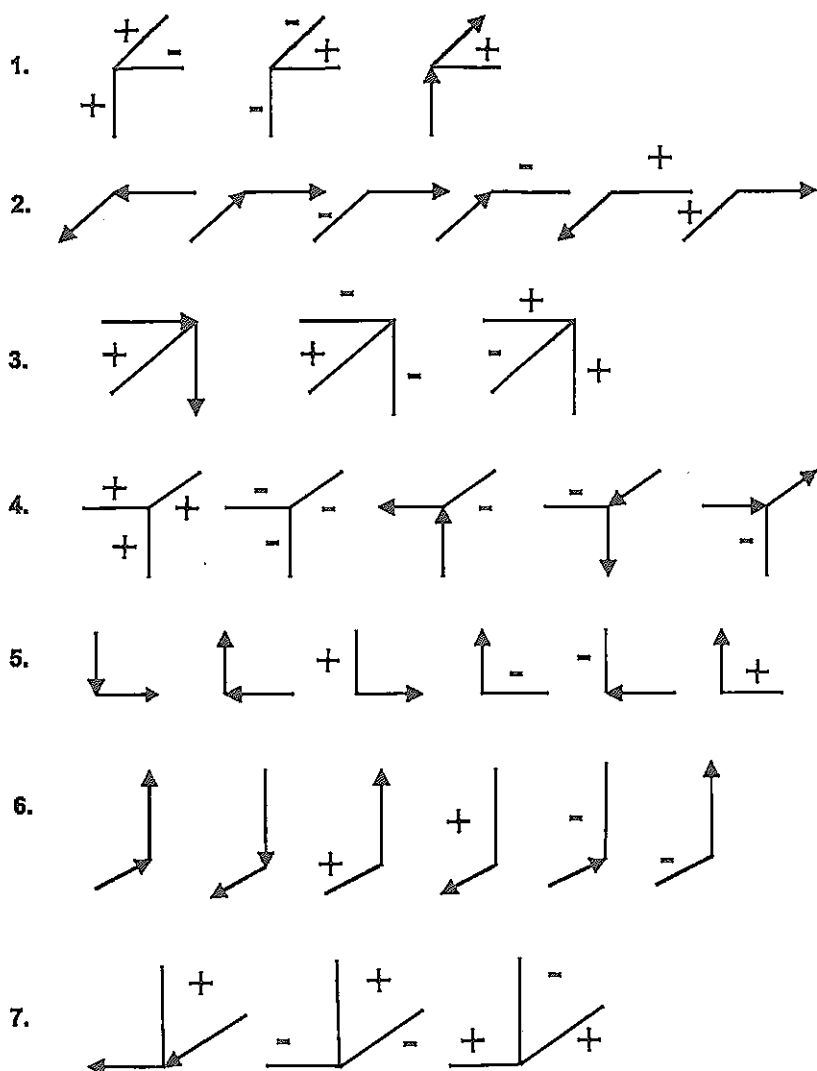


Рис. 3.4. Локальные интерпретации вершин изображения

Ненадежность знаний и выводов. Ненадежность знаний означает, что для оценки их достоверности нельзя применить двух-балльную шкалу (1 — абсолютно достоверные; 0 — недостоверные знания). Для более тонкой оценки достоверности знаний приме-

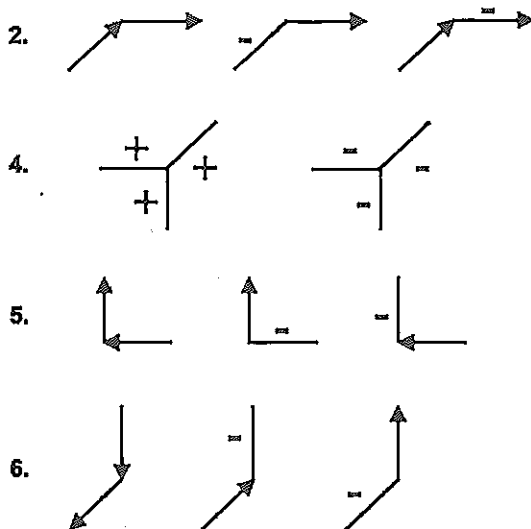


Рис. 3.5. Локальные интерпретации вершин изображения с учетом связей между ними

няется вероятностный подход, основанный на теореме Байеса [7, 11, 15], и другие методы. Например, в экспертной системе *MYSIN*, предназначенной для диагностики и выбора метода лечения инфекционных заболеваний, разработан метод вывода с использованием коэффициентов уверенности [7, 10, 11]. Широкое применение на практике получили нечеткие выводы, строящиеся на базе нечеткой логики, ведущей свое происхождение от теории нечетких множеств [2, 4, 8, 9].

В системах с ненадежными знаниями на И-ИЛИ-графе появляется еще один тип связи КОМБ — комбинированная связь. Фрагмент структуры такого графа показан на рис. 3.7. Если ненадежные знания объединены связкой И, то степень надежности заключения традиционно выбирается как минимальное значение, при объединении связкой ИЛИ — как максимальное значение степени надежности. Связь КОМБ означает, что заключение, основанное на фактах, объединяемых этим видом связи, будет получено с оценкой достоверности, вычисляемой тем или иным способом. Одним из видов оценки достоверности знаний являются коэффициенты уверенности, используемые в экспертной

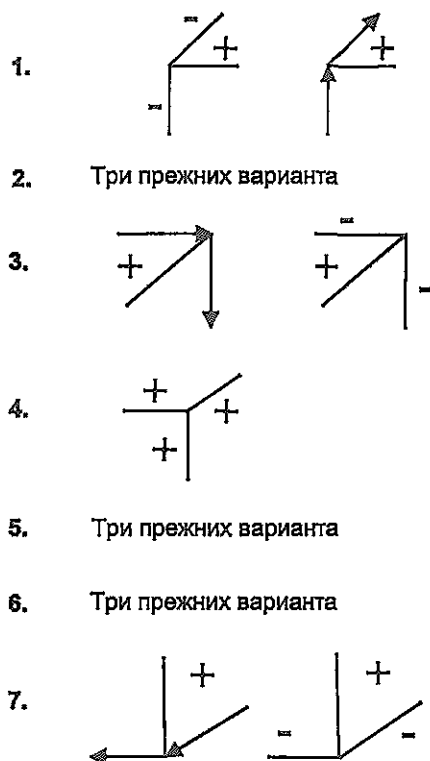


Рис. 3.6. Результат вторичной операции фильтрации

системе *MYSIN*, которые могут принимать значения, принадлежащие отрезку $[-1, 1]$, при этом 1 соответствует истинному, а (-1) — ложному утверждению.

Коэффициент уверенности (*CF*), характеризующий связь КОМБ при выводе заключения *A* на основе посылок *X* и *Y*, вычисляется по одной из следующих формул:

$$CF[A, (X, Y)] = 1, \text{ если } CF[A, X] = 1 \text{ или } CF[A, Y] = 1;$$

$$CF[A, (X, Y)] = CF[A, X] + CF[A, Y] - CF[A, X]CF[A, Y],$$

$$\text{если } CF[A, X] > 0 \text{ и } CF[A, Y] > 0;$$

$$CF[A, (X, Y)] = \frac{CF[A, X] + CF[A, Y]}{1 - \min\{|CF[A, X]|, |CF[A, Y]|\}},$$

$$\text{если } CF[A, X]CF[A, Y] \leq 0, CF[A, X] \neq \pm 1 \text{ и } CF[A, Y] \neq \pm 1;$$

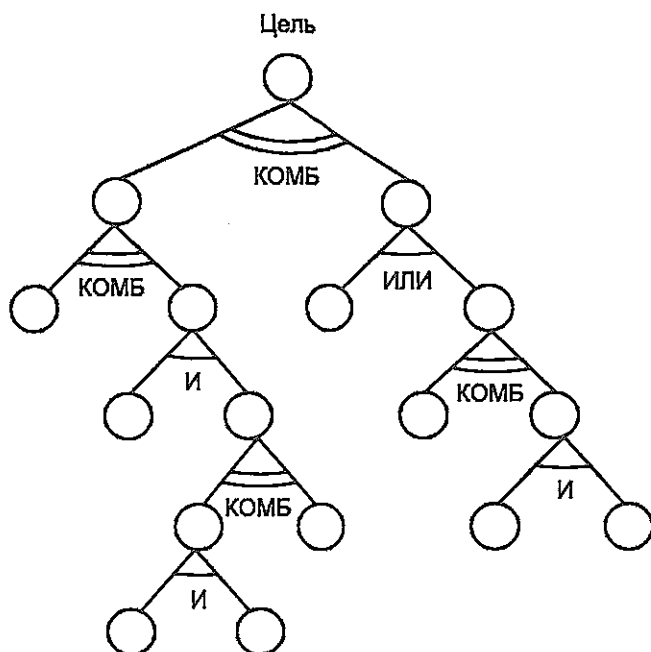


Рис. 3.7. Фрагмент структуры И-ИЛИ-графа для системы с ненадежными знаниями

$$CF[A, (X, Y)] = CF[A, X] + CF[A, Y] + CF[A, X]CF[A, Y],$$

если $CF[A, X] < 0$ и $CF[A, Y] < 0$;

$$CF[A, (X, Y)] = -1, \text{ если } CF[A, X] = -1 \text{ или } CF[A, Y] = -1.$$

Несмотря на отсутствие строгого теоретического обоснования, коэффициенты уверенности находят широкое применение в экспертных системах продукционного типа благодаря простоте восприятия и интерпретации получаемых результатов, которые неплохо согласуются с реальностью.

Вероятностный метод оценки надежности знаний получил в инженерии знаний широкое развитие. Рассмотрим один из таких подходов, *субъективный байесовский метод*. В данном подходе связи между элементами знаний не разделяются на типы; вместо этого каждому элементарному фрагменту знаний (факту, представленному парой *атрибут — значение* или утверждением) ставится в соответствие минимальное или максимальное значение

байесовской вероятности, после чего степени надежности выводимых заключений рассчитываются как апостериорные (условные) вероятности по формулам, полученным на базе формулы Байеса.

Пусть $P(H)$ — вероятность некоторой гипотезы (заключения) при отсутствии каких-либо свидетельств (т.е. априорная вероятность, назначаемая экспертом) и пусть $P(H:E)$ — апостериорная вероятность гипотезы H при наличии свидетельства (факта) E , вычисляемая по формуле

$$P(H:E) = \frac{P(E:H)P(H)}{P(E)},$$

где $P(E)$ — вероятность свидетельства E , которая вычисляется по формуле $P(E) = P(E:H)P(H) + P(E:\neg H)P(\neg H)$; $P(E:H)$ — вероятность наличия свидетельства E при условии истинности гипотезы H (например, вероятность наличия высокой температуры при заболевании гриппом); $P(E:\neg H)$ — вероятность свидетельства E при условии ложности заключения H (вероятность высокой температуры у пациента, не болеющего гриппом).

Значения этих вероятностей, как правило, определяются экспертами и хранятся в базе знаний.

Субъективный байесовский подход получил широкое распространение благодаря своей простоте. К тому же он достаточно обоснован теоретически. Однако этот подход, как и предыдущий, имеет нерешенные проблемы: например, сумма вероятностей опровергающих друг друга событий может оказаться больше 1. Сложной задачей для экспертов является назначение априорных вероятностей условных событий. Теоретические исследования в данном направлении активно продолжаются, и в распоряжении проектировщиков экспертных систем уже имеются такие мощные средства, как вероятностная логика [7, 10], нечеткая логика [2, 4, 8, 9], теория Демпстера—Шафера [5, 10, 11] и т.п.

Неполнота знаний и немонотонная логика. Абсолютно полных знаний не бывает, поскольку процесс познания бесконечен. В связи с этим состояние базы знаний должно изменяться с течением времени. В отличие от простого добавления информации, как в базах данных, при добавлении новых знаний возникает опасность получения противоречивых выводов, т.е. выводы, полученные с использованием новых знаний, могут опровергать те, что были получены ранее. Еще хуже, если новые знания будут нахо-

даться в противоречии со «старыми», тогда механизм вывода может стать неработоспособным. Многие экспертные системы первого поколения были основаны на модели закрытого мира, обусловленной применением аппарата формальной логики для обработки знаний. *Модель закрытого мира* предполагает жесткий отбор знаний, включаемых в базу, а именно: БЗ заполняется исключительно верными понятиями, а все, что ненадежно или неопределенно, заведомо считается ложным. Другими словами, все, что известно базе знаний, является истиной, а остальное — ложью. Такая модель имеет ограниченные возможности представления знаний и таит в себе опасность получения противоречий при добавлении новой информации. Тем не менее эта модель достаточно распространена; например, на ней базируется язык PROLOG. Недостатки модели закрытого мира связаны с тем, что формальная логика исходит из предпосылки, согласно которой набор определенных в системе аксиом (знаний) является *полным* (теория является полной, если каждый ее факт можно доказать, исходя из аксиом этой теории). Для полного набора знаний справедливость ранее полученных выводов не нарушается с добавлением новых фактов. Это свойство логических выводов называется *монотонностью*. К сожалению, реальные знания, закладываемые в экспертные системы, крайне редко бывают полными.

Рассмотрим простой пример. Допустим, в БЗ содержатся следующие утверждения:

«Птицы летают».

«Пингвин не летает».

«Лоло — птица».

На основе этих знаний можно получить заключение «Лоло летает» и сделать вывод о том, что «Пингвин не является птицей».

Если в БЗ добавить факт «Лоло — пингвин», то получим противоречащие предыдущим заключения: «Лоло не летает» и «Лоло не является птицей».

В качестве средств формальной обработки неполных знаний, для которых необходимы *немонотонные выводы*, разрабатываются методы немонотонной логики: немонотонная логика Макдермотта и Доула, в которой вводятся условные логические операции, логика умолчания Рейтера, немонотонная логика Маккарти и т.п. [5, 10, 11]. Многие из этих теорий еще не полностью отработаны, но предложенные в них элементы уже нашли применение

ние в практических разработках (проверка и учет непротиворечивости элементов знаний, установление значений по умолчанию во фреймовых системах и т.п.). Так, пример с пингвином не вызывает противоречий при использовании фреймового представления знаний (рис. 3.8).

Для организации логических выводов в интеллектуальных системах с неполными знаниями вместо традиционной дедукции применяется абдукция [3]. *Абдукцией* называется процесс формирования объясняющей гипотезы на основе заданной теории и имеющихся наблюдений (фактов). Рассмотрим простейший пример абдуктивного вывода. Предположим, теория содержит правило: «ЕСЛИ студент отлично знает математику, ТО он может стать хорошим инженером» и факт: «Студент Иванов отлично знает математику». Кроме того, имеется наблюдение: «Студент Иванов стал хорошим экономистом», которое не выводится из заданной теории. Для того чтобы его вывести, необходимо сформировать абдуктивную (объясняющую) гипотезу, которая не будет противоречить вышеприведенной теории. Такой гипотезой может быть, например, следующая: «Хороший математик может стать хорошим экономистом».

Целью абдуктивного вывода является формирование одного (или более) объяснения Δ наблюдаемого факта G на основе информации, хранящейся в БЗ интеллектуальной системы (теория T). Объяснение Δ должно быть таким, чтобы $T \cup \Delta \vdash G$ и чтобы $T \cup \Delta$ было непротиворечиво. Другими словами, наблюдение G можно вывести из теории T лишь при ее расширении некоторым множеством гипотез Δ . В большинстве случаев абдуктивные гипотезы выбираются из заранее определенного множества предложений, отражающих определенный аспект знаний конкретной предметной области. Теорию T можно рассматривать как основу всех возможных расширений $T \cup \Delta$ для каждой абдуктивной гипотезы Δ .

Абдуктивные выводы используются в задачах диагностики для обнаружения причин наблюдаемого неправильного поведения систем, в задачах, связанных с пониманием естественного языка, для решения проблем накопления и усвоения знаний и т.д.

Для работы с неполными знаниями предназначена также *система поддержания значений истинности*, в которой все знания делятся на достоверные и недостоверные, при этом предусматривается систематическое упорядочение БЗ в целях устранения недостоверных знаний. Достоверные на данный момент знания отно-

Фрейм ПТИЦА

Имя слота	Указатель наследования	Тип значения	Значение
Летает	O	boolean	Да
Имеет	O	string	Крылья
.....			

Фрейм ЛАСТОЧКА

Имя слота	Указатель наследования	Тип значения	Значение
IS-A	—	frame	ПТИЦА
Летает	S	boolean	
Имеет	S	string	
.....			

Фрейм ПИНГВИН

Имя слота	Указатель наследования	Тип значения	Значение
IS-A	—	frame	ПТИЦА
Летает	U	boolean	Нет
Имеет	S	string	
.....			

Фрейм ЛОЛО

Имя слота	Указатель наследования	Тип значения	Значение
IS-A	—	frame	ПИНГВИН
Летает	S	boolean	
Имеет	S	string	
.....			

Рис. 3.8. Установка значений по умолчанию в системах фреймов

сят к классу IN, а сомнительные и недостоверные — к классу OUT. Если при добавлении новых знаний возникает противоречие, то выполняется проверка классов знаний, при этом возможны миграции из класса в класс [10, 11, 15].

Исследования в области немонотонных выводов — это попытки расширить границы формальной логики, в которые не вписываются реальные знания, необходимые интеллектуальным системам.

Неточность знаний. Известно, что количественные данные (знания) могут быть неточными, при этом существуют количественные оценки такой неточности (доверительный интервал, уровень значимости, степень адекватности и т.д.). Лингвистические знания также могут быть неточными. Для учета неточности лингвистических знаний используется теория нечетких множеств, предложенная Л. Заде в 1965 г. Этому ученому принадлежат слова: «Фактически нечеткость может быть ключом к пониманию способности человека справляться с задачами, которые слишком сложны для решения на ЭВМ». Развитие исследований в области нечеткой математики привело к появлению нечеткой логики и нечетких выводов, которые выполняются с использованием знаний, представленных нечеткими множествами, нечеткими отношениями, нечеткими соответствиями и т. д.

3.2. НЕЧЕТКИЕ МНОЖЕСТВА И НЕЧЕТКИЕ ВЫВОДЫ

Как и количественные измерения, лингвистические оценки могут быть неточными. Например, понятию «старик» могут соответствовать люди разного возраста в зависимости от того, кто определяет это понятие, а также от цели определения. Семнадцатилетние юноши могут называть стариками сорокалетних родителей, а люди пенсионного возраста — тех, кому за 80.

Пусть U — полное множество объектов некоторого класса. Нечеткое подмножество F множества U , которое в дальнейшем будем называть нечетким множеством, определяется через функцию принадлежности $\mu_F(u)$, $u \in U$. Эта функция отображает элементы u множества U на множество вещественных чисел отрезка $[0,1]$, которые указывают степень принадлежности каждого элемента нечеткому множеству F .

Если полное множество U состоит из конечного числа элементов $u_i, i = 1, 2, \dots, n$, то нечеткое множество F можно представить в следующем виде:

$$F = \mu_F(u_1)/u_1 + \mu_F(u_2)/u_2 + \dots + \mu_F(u_n)/u_n,$$

где знак $+$ означает не сложение, а скорее объединение, а символ $/$ показывает, что значение $\mu_F(u_i)$ относится к элементу u_i , а не означает деление на u_i .

В случае если множество U является непрерывным, F можно записать как интеграл: $F = \int \mu_F(u) / u.$

Пусть U — множество людей в возрасте от 0 до 100 лет и пусть понятия «молодой», «среднего возраста» и «старый» представлены нечеткими множествами F_1, F_2 и F_3 соответственно. Эти нечеткие множества являются подмножествами множества U . Функции принадлежности элементов множества U к понятиям, представленным нечеткими множествами F_1, F_2, F_3 , показаны на рис. 3.9 и имеют следующий вид:

$$F_1 = \mu_{F_1}(u) = 1/0 + 1/10 + 0.8/20 + 0.3/30;$$

$$F_2 = \mu_{F_2}(u) = 0.5/30 + 1/40 + 0.5/50;$$

$$F_3 = \mu_{F_3}(u) = 0.4/50 + 0.8/60 + 1/70 + 1/80 + 1/90.$$

Принадлежность к нечеткому множеству F_1 , соответствующему понятию «молодой», составляет 1 для детей; двадцатилет-

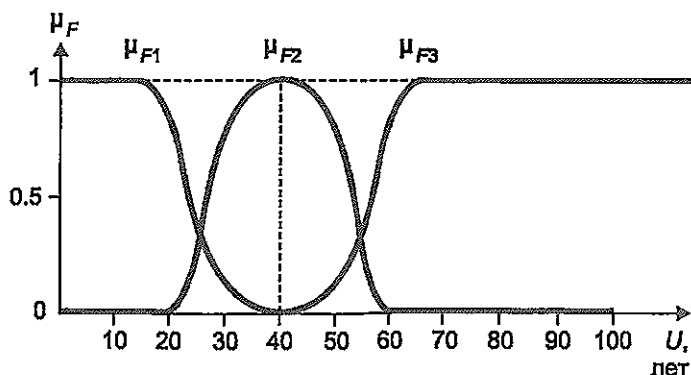


Рис. 3.9. Функции принадлежности нечетких множеств, соответствующих понятиям:

F_1 — «молодой», F_2 — «средний», F_3 — «старый»

ний человек принадлежит множеству $F1$ со степенью 0.8, а степень принадлежности тридцатилетнего равна 0.3. При записи функций принадлежности элементы нечеткого множества со значениями $\mu_F(u_i) = 0$ не включаются, поэтому функция принадлежности $F1$ содержит всего четыре терма, так как степень принадлежности людей старше 30 лет к понятию «молодой» равна нулю. Аналогично построены функции принадлежности нечетких множеств $F2$ и $F3$.

Операции над нечеткими множествами. Над нечеткими множествами, как и над обычными, можно выполнять математические операции. Рассмотрим важнейшие из них: дополнение множества, объединение и пересечение множеств.

Операция дополнения:

$$\bar{F} = \sum_{i=1}^n (1 - \mu_F(u_i)) / u_i, \quad \mu_{\bar{F}}(u) = 1 - \mu_F(u).$$

Если нечеткое множество $F1$ соответствует лингвистической переменной «молодой», то какому понятию будет соответствовать дополнение этого множества? Нетрудно догадаться, что это будет понятие «немолодой», функция принадлежности которого показана на рис. 3.10, а математическая запись имеет вид:

$$\mu_{\bar{F}1} = 0.2/20 + 0.7/30 + 1/40 + 1/50 + 1/60 + 1/70 + 1/80 + 1/90.$$

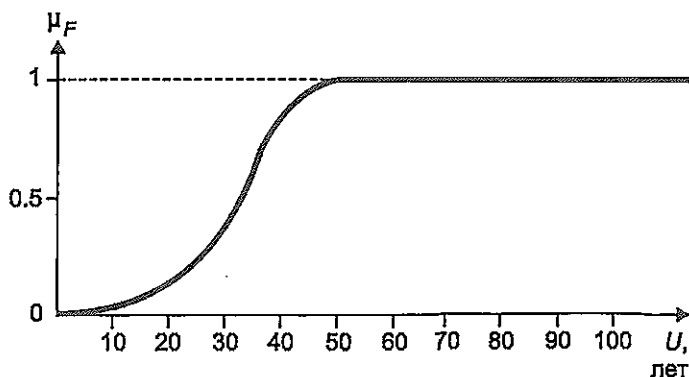


Рис. 3.10. Функция принадлежности дополнения нечеткого множества $F1$

Операция объединения:

$$F \cup G = \sum_{i=1}^n (\mu_F(u_i) \vee \mu_G(u_i)) / u_i, \quad \mu_{F \cup G}(u) = \mu_F(u) \vee \mu_G(u),$$

где операция $\vee$ соответствует взятию максимума. Какому понятию будет соответствовать объединение нечетких множеств $F1$ и $F2$? Вычислим функцию принадлежности

$$\mu_{F1 \cup F2}(u) = 1/0 + 1/10 + 0.8/20 + 0.5/30 + 1/40 + 0.5/50.$$

График этой функции приведен на рис. 3.11, а ее лингвистической интерпретацией является понятие «человек молодого или среднего возраста».

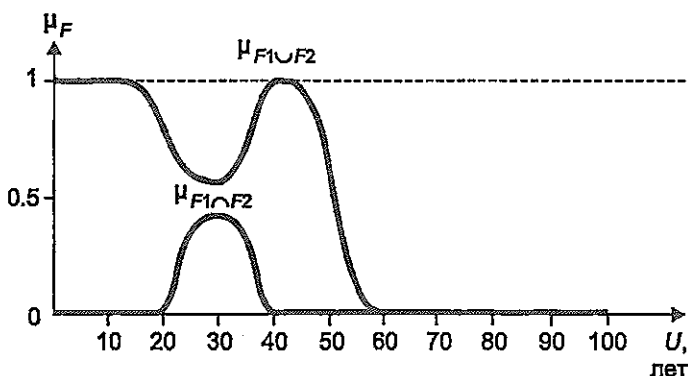


Рис. 3.11. Результаты объединения и пересечения нечетких множеств $F1$ и $F2$

Операция пересечения:

$$F \cap G = \sum_{i=1}^n (\mu_F(u_i) \wedge \mu_G(u_i)) / u_i, \quad \mu_{F \cap G}(u) = \mu_F(u) \wedge \mu_G(u),$$

где символ $\wedge$ обозначает взятие минимума. Вычислим функцию принадлежности пересечения множеств $F1$ и $F2$:

$$\mu_{F1 \cap F2}(u) = 0.3/30.$$

Остальные члены этой функции, соответствующие значениям аргумента, кратным 10, равны нулю. Возможные лингвисти-

ческие интерпретации: «уже не молодой, но еще не средний возраст», «одновременно молодой и средний возраст».

Симметрическая разность:

$$FDG = (F \cup G) \cap (\bar{F} \cup \bar{G}) = D(\mu_F(u), \mu_G(u));$$

$$D(\mu_F(u), \mu_G(u)) = \sum_{i=1}^n \{ \min_i [\max(\mu_F(u_i), \mu_G(u_i)),$$

$$\max(1 - \mu_F(u_i), 1 - \mu_G(u_i))] \} / u_i.$$

В рассматриваемом примере функция принадлежности нечеткого множества, соответствующая $F1 \Delta F2$, будет интерпретироваться как «не молодой и не средний возраст» и иметь два минимума.

Нечеткие отношения. Элементы знаний связаны друг с другом отношениями различного рода. Часто эти отношения заданы в виде текстовых описаний или правил, которые необходимо формализовать для реализации нечетких выводов.

Нечетким отношением R между полным множеством U и другим полным множеством V называется нечеткое подмножество прямого декартова произведения $U \times V$, определяемое следующим образом:

$$R = \sum_{i=1}^l \sum_{j=1}^m \mu_R(u_i, v_j) / (u_i, v_j),$$

где $U = \{u_1, u_2, \dots, u_l\}$, $V = \{v_1, v_2, \dots, v_m\}$.

Предположим, что между элементами знаний, представленными нечеткими множествами F и G , существует связь, заданная правилом: «ЕСЛИ F , ТО G », при этом $F \subseteq U$, $G \subseteq V$. Один из способов построения нечеткого отношения между F и G , реализующий импликацию $F \rightarrow G$, состоит в следующем (заметим, что возможны другие способы, см. разд. 3.4):

$$R = F \times G = \sum_{i=1}^l \sum_{j=1}^m (\mu_F(u_i) \wedge \mu_G(v_j)) / (u_i, v_j), \quad \mu_R(u, v) = \mu_F(u) \wedge \mu_G(v).$$

Пусть U и V — множества натуральных чисел от 1 до 4. Определим понятия «малые числа» и «большие числа» с помощью нечетких множеств F и G соответственно:

$$U = V = \{1, 2, 3, 4\};$$

$$F = 1/1 + 0.6/2 + 0.1/3;$$

$$G = 0.1/2 + 0.6/3 + 1/4.$$

Пусть задано правило: «ЕСЛИ u — малое число, ТО v — большое». Построим соответствующее нечеткое отношение $R = F \times G$:

$$R = \begin{array}{c|cccc} & \xrightarrow{\quad} u_i & & & \\ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \end{array} & \begin{array}{cccc} 0.1 & 0.6 & 1 & \\ 0.1 & 0.6 & 0.6 & \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 \end{array} & \begin{array}{c} \downarrow \\ \\ \\ \end{array} \\ & v_j & & & \end{array}$$

Свойства нечетких отношений:

- объединение $(R \cup S)(u, v) = R(u, v) \vee S(u, v)$, $u \in U$, $v \in V$;
- пересечение $(R \cap S)(u, v) = R(u, v) \wedge S(u, v)$, $u \in U$, $v \in V$;
- операция включения $R \subseteq S \Leftrightarrow R(u, v) \leq S(u, v)$, $u \in U$, $v \in V$;
- свойство поглощения (идемпотентности) $R \cap R = R$, $R \cup R = R$;
- коммутативность $R \cap S = S \cap R$, $R \cup S = S \cup R$;
- ассоциативность $R \cap (S \cap Q) = (R \cap S) \cap Q$, $R \cup (S \cup Q) = (R \cup S) \cup Q$;
- дистрибутивность $R \cap (S \cup Q) = (R \cap S) \cup (R \cap Q)$, $R \cup (S \cap Q) = (R \cup S) \cap (R \cup Q)$;
- рефлексивность:

- если $\mu_R(u, u) = 1$, отношение R — рефлексивное;
- если $\mu_R(u, u) < 1$, отношение R — слаборефлексивное;
- если $\mu_R(u, u) = 0$, отношение R — антирефлексивное;
- если $\mu_R(u, u) > 0$, отношение R — слабоантирефлексивное;

- симметричность $\mu_R(u, v) = \mu_R(v, u)$, $u, v \in U$;
- транзитивность $\mu_R(u, v) \geq \mu_R(u, z) \wedge \mu_R(z, v)$, $u, v, z \in U$.

Композиция нечетких отношений. Если знания представлены с помощью нечетких множеств и нечетких отношений, то для реализации логических выводов в нечеткой среде необходимо иметь возможность применения совокупности правил. Поскольку знания в виде правил формализуются нечеткими отношениями, нужно уметь осуществлять их композицию, которая может выполняться с помощью операции максиминной свертки.

Пусть R — нечеткое отношение из области U в область V , а S — нечеткое отношение из области V в область W , тогда нечеткое отношение из области U в область W определим как свертку следующего вида:

$$R \circ S = \sum_{i=1}^l \sum_{j=1}^m \vee (\mu_R(u_i, v_j) \wedge \mu_S(v_j, w_k)) / (u_i, w_k).$$

Поясним применение максиминной свертки на примере.

Пусть R — нечеткое отношение между множествами U и V , которые представляют собой совокупности натуральных чисел от 1 до 4. Семантика отношения R соответствует правилу: «ЕСЛИ u — малые числа, ТО v — большие». Конкретное значение этого отношения возьмем из предыдущего примера. Определим отношение S из V в W . С этой целью на V определим понятие «немалые числа», которое будет дополнением введенного ранее нечеткого множества F , и обозначим его $F1$. Введем множество $W = \{1, 2, 3, 4\}$ и определим на нем понятие «очень большие числа», которое обозначим H . К этому понятию числа 1 и 2 имеют степень принадлежности, равную 0, число 3 имеет значение принадлежности 0.5, и только 4 принадлежит со степенью, равной 1:

$$\mu_{F1}(v) = 0/1 + 0.4/2 + 0.9/3 + 1/4;$$

$$\mu_H(w) = 0.5/3 + 1/4.$$

Отношение между полными множествами V и W сформулируем в виде правила: «ЕСЛИ v — немалые числа, ТО w — очень большие числа». Построим нечеткое отношение S , соответствующее этому правилу и являющееся подмножеством декартова произведения $F1$ и H :

$$S = \begin{array}{c} \xrightarrow{\quad} v_j \\ \left| \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.4 \\ 0 & 0 & 0.5 & 0.9 \\ 0 & 0 & 0.5 & 1 \end{array} \right| \downarrow \\ w_k \end{array}$$

Вычислим максиминную свертку нечетких отношений $R \circ S$, результат которой должен соответствовать последовательному применению двух правил: «ЕСЛИ u — малое число, ТО v — большое»; «ЕСЛИ v — немалое число, ТО w — очень большое»:

$$R \circ S = \begin{array}{c} \xrightarrow{\quad} u_i \\ \left| \begin{array}{cccc} 0 & 0.1 & 0.6 & 1 \\ 0 & 0.1 & 0.6 & 0.6 \\ 0 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 \end{array} \right| \circ \left| \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.4 \\ 0 & 0 & 0.5 & 0.9 \\ 0 & 0 & 0.5 & 1 \end{array} \right| = \left| \begin{array}{cccc} 0 & 0 & 0.5 & 1 \\ 0 & 0 & 0.5 & 0.6 \\ 0 & 0 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 \end{array} \right| \downarrow \\ w_k \end{array}$$

При парных сравнениях элементов i -й строки и j -го столбца из них выбирается наименьший, затем из четырех минимальных элементов выбирается максимум, который является результатом, и записывается в ячейку с координатами i, j . Результирующее нечеткое отношение показывает взаимосвязь областей U и V . Вообще над нечеткими отношениями можно выполнять операции свертки других видов (минимаксная, максимумультипликативная и т. п.) [4].

Нечеткие выводы. Рассмотрим традиционный дедуктивный вывод, основанный на применении правила вывода *Modus Ponendo Ponens*, в среде нечетких знаний. Вспомним его формулировку: «ЕСЛИ A — истина, И импликация $A \rightarrow B$ — истина, ТО B — истина», т. е. из факта A и правила «ЕСЛИ A , ТО B », можно вывести B . В среде нечетких знаний факт A и образец правила A^* не обязательно всегда и везде совпадают, так как факты представлены нечеткими множествами, являющимися подмножествами полных знаний, а правила — нечеткими отношениями, которые есть подмножества декартовых произведений полных множеств. Поэтому если A и A^* близки друг к другу, то их можно сопоставить и получить вывод B^* в сфере их совпадения. Композиционное правило вывода в среде нечетких знаний базируется на операции максиминной свертки и имеет вид: $B^* = A^* \circ R$, где R — нечеткое отношение, соответствующее импликации $A \rightarrow B$, а B^* — приближенное заключение, выраженное нечетким множеством

$$\mu_B(v) = \sum_{i=1}^m \vee (\mu_A(u_i) \wedge \mu_R(u_i, v_j)) / (v_j).$$

Пусть F и G — нечеткие множества, соответствующие понятиям «малые числа» и «большие числа» и являющиеся подмножествами полных множеств $U = V = \{1, 2, 3, 4\}$. Функции принадлежности множеств F и G имеют вид:

$$\mu_F(u) = 1/1 + 0.6/2 + 0.1/3;$$

$$\mu_G(v) = 0.1/2 + 0.6/3 + 1/4.$$

Пусть также задано правило $F \rightarrow G$: «ЕСЛИ u — малые числа, ТО v — большие», формализованное нечетким отношением R

$$R = \begin{vmatrix} 0 & 0.1 & 0.6 & 1 \\ 0 & 0.1 & 0.6 & 0.6 \\ 0 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 \end{vmatrix}.$$

В качестве исходной посылки для вывода задан факт: « u — число около 2», представленный нечетким множеством $F1 \in U$ с функцией принадлежности $\mu_{F1}(u) = 0.3/1 + 1/2 + 0.3/3$.

Используя композиционное правило вывода, попробуем дать ответ на вопрос: «Что представляет собой v , если u — число около 2, и, если области U и V связаны отношением R »?

$$G1 = F1 \circ R = [0.3 \ 1 \ 0.3 \ 0] \circ \begin{vmatrix} 0 & 0.1 & 0.6 & 1 \\ 0 & 0.1 & 0.6 & 0.6 \\ 0 & 0.1 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 \end{vmatrix} = [0 \ 0.1 \ 0.6 \ 0.6].$$

График функции принадлежности результата нечеткого вывода показан на рис. 3.12. Для нее можно предложить следующую лингвистическую интерпретацию: « v — не очень большое число» или « v — до некоторой степени большое число».

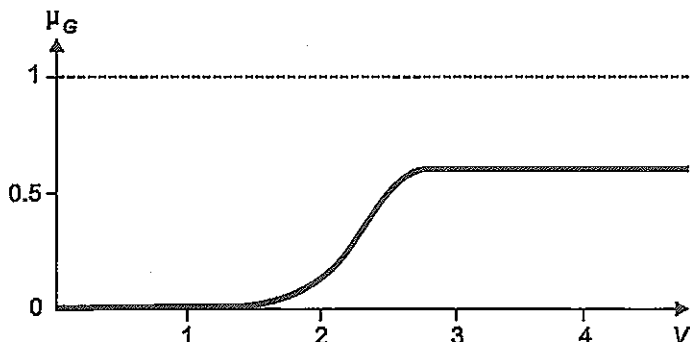


Рис. 3.12. Функция принадлежности результата нечеткого вывода

3.3. ПРИМЕР РАЗРАБОТКИ КОМПЬЮТЕРНОЙ ИГРЫ ПОЗИЦИОННОГО ТИПА

Рассмотрим пример обработки нечетких знаний, в частности учет неопределенности пути решения задачи в пространстве состояний на примере разработки программы, способной играть с человеком в позиционную игру.

РЕВЕРСИ — логическая игра, рассчитанная на двух участников, которые играют на доске размером 8×8 фишками разного цвета. Участники начинают игру, имея по две фишки, стоящие в центре доски по диагонали друг от друга. В процессе игры совершаются ходы, в результате которых число фишек увеличивается. Цель каждого игрока заключается в том, чтобы к моменту окончания игры число его фишек преобладало над числом фишек противника. В игре приняты следующие нормативные правила:

- при очередном ходе фишку можно ставить на свободную клетку в любом направлении, но обязательно рядом хотя бы с одной из фишек противника;

- фишка должна ставиться так, чтобы хотя бы одна из фишек противника оказалась *замкнутой* своими фишками. При этом замкнутые фишки противника меняют цвет и становятся своими;

- фишки могут неоднократно менять цвет, но не могут переставляться на доске;

- игра заканчивается, если доска заполнена, или на ней присутствуют фишки только одного цвета, или ни один из игроков не может сделать очередной ход.

Примеры возможных позиций в игре показаны на рис. 3.13.

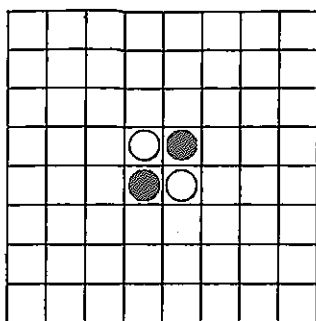
При написании *компьютерного игрока* возникают следующие задачи:

1. Выделение и представление объектов игры.
2. Реализация операций, соответствующих правилам перемещения и изменения объектов.
3. Построение правил принятия решений при выборе хода.
4. Управление последовательностью игры.

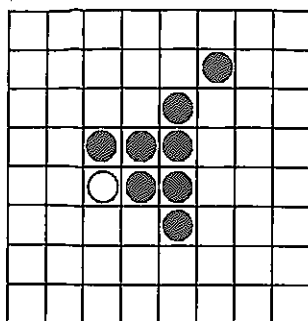
В данном случае *объектами* игры являются свободная клетка и занятые клетки двух цветов (белая и черная). Над объектами совершаются операции — *ходы*, которые заключаются в занятии свободной клетки в соответствии с заданными правилами. Изменение цвета одной или нескольких клеток является результатом хода.

Интеллект компьютерного игрока проявляется в процессах принятия решений. Для выбора хода можно применять набор правил или оценочную функцию, значение которой будет отражать его целесообразность. В данной задаче разработаны две оценочные функции, соответствующие различным уровням квалификации игрока.

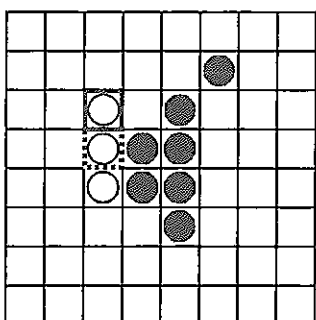
На уровне *новичка* компьютер выбирает оптимальный для себя ход, не рассматривая возможные ответные ходы противника.



Начальная позиция в игре



Текущая позиция Q



Возможные ходы белой фишкой из позиции Q

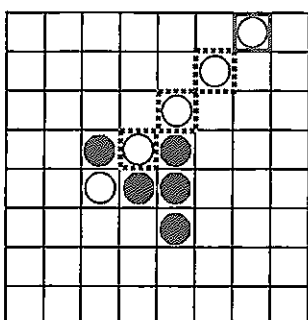


Рис. 3.13. Позиции игры РЕВЕРСИ

Оценочная функция имеет вид

$$\max_{Q \in X \times Y} R(x, y) = \sum_{i=1}^n s_i + ss, \quad (3.1)$$

- где $x, y = 1, \dots, 8$ – координаты клетки, в которую можно совершить ход;
 $Q \in X \times Y$ – подмножество полного множества клеток $X \times Y$, на которые можно поставить очередную фишку;
 n – число замыкаемых клеток;
 s_i – ценность i -й замыкаемой клетки;
 ss – ценность клетки, на которую совершается ход.

Значения s_i и ss вычисляются на основе следующих эмпирически подобранных констант:

• $s_i = 2$ для кромочной клетки поля. Такие клетки могут быть замкнуты только вдоль кромки доски, поэтому они предпочтительнее внутренних клеток;

• $s_i = 1$ для остальных клеток;

• угловые клетки поля не могут быть замкнуты, они являются наиболее ценными для захвата;

• $ss = 0.8$, если ход делается на угловую клетку;

• $ss = 0.4$, если ход делается на кромочную клетку;

• $ss = 0$ для всех остальных клеток.

Анализ возможности хода осуществляется следующим образом. Определяются все свободные клетки, соседствующие с фишками противника, и для них вычисляется значение $R(x, y)$. Клетки, для которых $R(x, y) < 1$, отбрасываются, так как не соответствуют условиям игры. Ход совершается на клетку с максимальным значением $R(x, y)$. Если таких клеток несколько, выбирается первая попавшаяся.

На уровне *профессионала* компьютер анализирует ход, который может сделать противник. Оценочная функция возможного хода R в данном случае вычисляется как разность между возможным выигрышем очередного хода и возможным проигрышем при следующем ходе противника:

$$\max_{Q \in X \times Y} R(dep = 0, x, y) = \left\{ \sum_{i=1}^n s_i + ss - \max_{Q1 \in X \times Y} [R(dep = 1, x_1, y_1)] \right\},$$

где x_1, y_1 — координаты клеток, на которые может поставить свою фишку человек при ходе компьютера в клетку с координатами x, y ; параметр dep указывает на игрока, для которого вычисляется оценка эффективности хода. Нулевое значение соответствует ходу компьютера; $dep=1$ соответствует вычислению эффективности хода, который может совершить противник (человек), по формуле (3.1).

Управление игрой заключается в инициализации исходной позиции и отслеживании текущей ситуации, включающей ряд проверок на возможность совершения текущего хода, на передачу хода, на наличие свободных клеток, а также изменение цветов фишек и выдачу текущих сообщений, в том числе об окончании игры и ее результатах.

Проведенные эксперименты показали существенное преимущество применения тактики, учитывающей вероятные ходы противника. Эксперименты с увеличением глубины просмотра веро-

ятных ходов компьютера и человека показали, что «заглядывать» дальше чем на два хода не имеет смысла, так как опытные игроки сознательно препятствуют разыгрываемым компьютером комбинациям, а неопытные делают то же самое, не понимая целей компьютерного игрока.

3.4. ОПИСАНИЕ КОМПЬЮТЕРНОЙ СИСТЕМЫ «НЕЧЕТКАЯ ЛОГИКА»

Программная система «Нечеткая логика» разрабатывалась для исследования применения нечеткого логического вывода в задачах принятия решений, диагностики и прогнозирования [1]. Система «Нечеткая логика» способна осуществлять нечеткий логический вывод на множестве правил, заданных пользователем. Вычисления в процессе логического вывода могут быть реализованы различными способами.

Разработанная система выполняет логические выводы на основе нечеткой композиции — аналога *Modus Ponendo Ponens* в среде нечетких знаний:

$$B^* = A^* \circ R,$$

где B^* — приближенное заключение;

A^* — нечеткое представление исходного факта;

R — нечеткое отношение, соответствующее импликации вида $A \rightarrow B$.

Мера близости выводимого заключения B^* к эталонному образцу B определяется визуально или с помощью скалярного индекса сходства нечетких множеств B и B^* , вычисляемого по формуле [8]:

$$S_k(B, B^*) = \frac{|B \cap B^*|}{|B \cup B^*|},$$

где операция $|B|$ обозначает скалярную мощность нечеткого множества B ,

заданного на базовом множестве U : $|B| = \sum_{u \in U} \mu_B(u)$.

В системе допускается организация правил с одним выходом «ЕСЛИ A , ТО B » и правил с двумя выходами «ЕСЛИ A , ТО B , ИНАЧЕ C ». При этом антецедент может быть сложным логическим вы-

ражением, включающим операции отрицания, дизъюнкции и конъюнкции. Для вычисления нечеткого отношения R на нечетких множествах $A = \int_U \mu_A(u) / u$; $B = \int_V \mu_B(v) / v$; и $C = \int_V \mu_C(v) / v$

используются следующие способы [4, 8, 9]:

1) импликация Мамдани для правил с одним выходом:

$$R = \int_{U \times V} (\mu_A(u) \wedge \mu_B(v)) / (u, v);$$

2) максиминное правило с одним выходом:

$$R = \int_{U \times V} ((\mu_A(u) \wedge \mu_B(v)) \vee (1 - \mu_A(u))) / (u, v);$$

3) максиминное правило с двумя выходами:

$$R = \int_{U \times V} ((\mu_A(u) \wedge \mu_B(v)) \vee ((1 - \mu_A(u)) \wedge \mu_C(v))) / (u, v);$$

4) бинарное правило с одним выходом:

$$R = \int_{U \times V} (((1 - \mu_A(u)) \vee \mu_B(v)) \wedge (\mu_A(u))) / (u, v);$$

5) бинарное правило с двумя выходами:

$$R = \int_{U \times V} (((1 - \mu_A(u)) \vee \mu_B(v)) \wedge (\mu_A(u) \vee \mu_C(v))) / (u, v);$$

6) импликация Лукасевича для правил с одним выходом:

$$R = \int_{U \times V} (1 \wedge (1 - \mu_A(u) + \mu_B(v))) / (u, v);$$

7) импликация Лукасевича для правил с двумя выходами:

$$R = \int_{U \times V} ((1 \wedge (1 - \mu_A(u) + \mu_B(v))) \wedge (\mu_A(u) + \mu_C(v))) / (u, v);$$

8) импликация Геделя для правил с одним выходом:

$$R = \int_{U \times V} (\mu_A(u) \rightarrow_g \mu_B(v)) / (u, v),$$

$$\text{где } \mu_A(u) \rightarrow_g \mu_B(v) = \begin{cases} 1, & \text{если } \mu_A(u) \leq \mu_B(v); \\ 0, & \text{если } \mu_A(u) > \mu_B(v); \end{cases}$$

9) импликация Геделя для правил с двумя выходами:

$$R = \int_{U \times V} ((\mu_A(u) \rightarrow_g \mu_B(v)) \wedge (1 - (\mu_A(u) \rightarrow_g \mu_C(v)))) / (u, v).$$

Обобщенный алгоритм нечеткого вывода включает следующие шаги.

1. Пользователь заполняет базу знаний, вводя лингвистические переменные (ЛП), образцы (эталоны) значений ЛП в виде функций принадлежности и правила (рис. 3.14).

2. Из БЗ выбираются правила, участвующие в цепочке выводов.

3. Для очередного правила в цепочке из БЗ извлекаются образцы значений ЛП, присутствующих в антецеденте правила. Пользователю предоставляется возможность в графическом режиме ввести функции принадлежности, соответствующие исходным фактам, или изменить эталонные множества из БЗ.

4. Результат применения правила выводится на экран вместе с эталонными значениями возможных заключений (рис. 3.15).

5. Пользователю предоставляется возможность сохранения полученного результата в БЗ для того, чтобы оно могло участвовать в выводе на последующих этапах.

6. Для обрабатываемого правила вычисляется нечеткое отношение с использованием эталонных функций принадлежности. При этом в системе предусмотрены два способа вычисления конъюнкции: минимум и произведение. Бинарные нечеткие операции в антецеденте выполняются после операции композиции над аргументами, т. е. для правила вида «ЕСЛИ A_1 И A_2 , ТО B » сначала вычисляется нечеткое множество F_1 как композиция поступившего в систему факта A_1^* и правила «ЕСЛИ A_1 , ТО B », затем множество F_2 , соответствующее композиции факта A_2^* с правилом «ЕСЛИ A_2 , ТО B ». Множество B^* , которое соответствует полученному заключению, определяется как результат объединения нечетких множеств F_1 и F_2 , т.е. $\mu_{B^*}(v) = \max\{\mu_{F_1}(v), \mu_{F_2}(v)\}$. Если в антецеденте правила присутствует операция ИЛИ, нечеткое множество B^* определяется как результат пересечения нечетких множеств F_1 и F_2 , т.е. $\mu_{B^*}(v) = \min\{\mu_{F_1}(v), \mu_{F_2}(v)\}$. Вычисление нечетких отношений, соответствующих импликациям, производится способом, который выбрал пользователь.

7. Если в цепочке еще есть правила, то осуществляется переход на шаг 3, иначе вывод завершен.

Информация о ЛП и их значениях, хранимая в БЗ, включает ссылки на массивы точек, по которым строятся функции принадлежности, а также сведения об используемом типе интерполяции

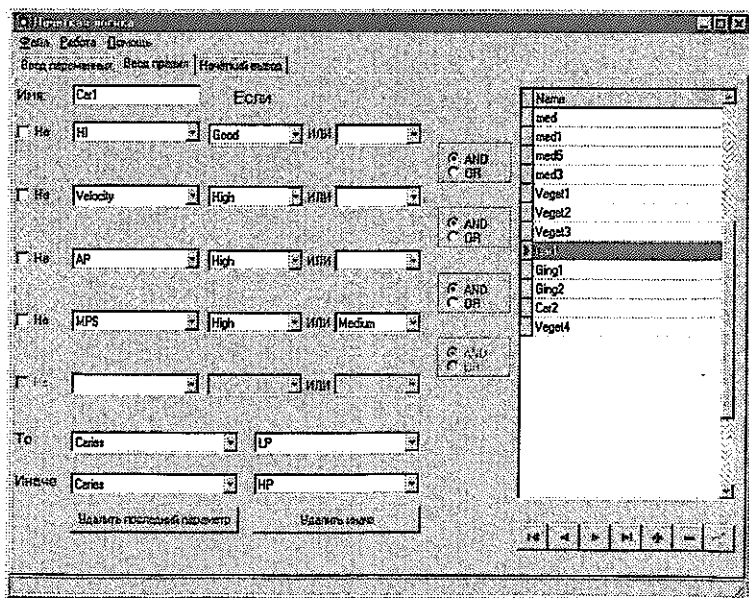
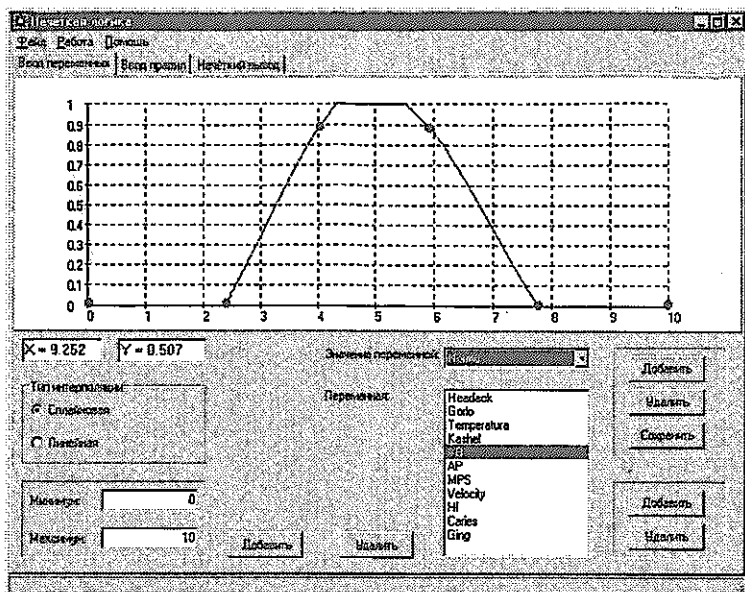


Рис. 3.14. Окно ввода правил и эталонных функций принадлежности в базу знаний

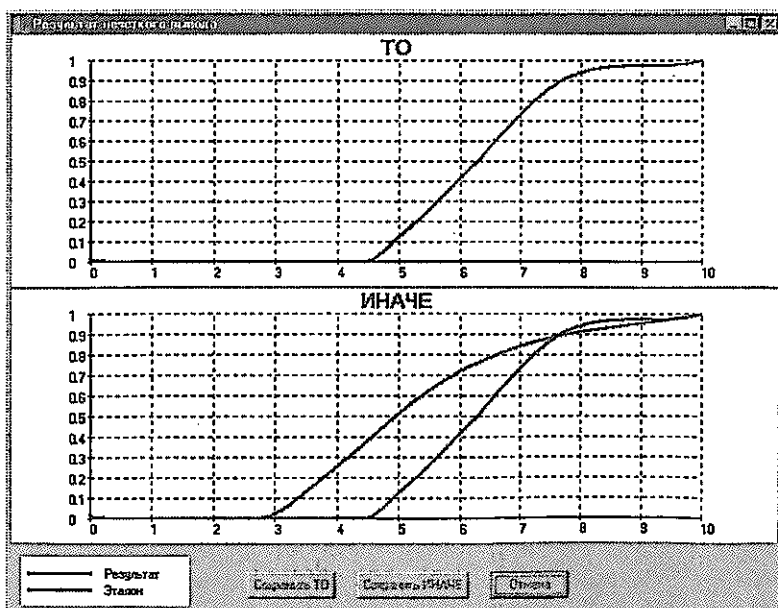
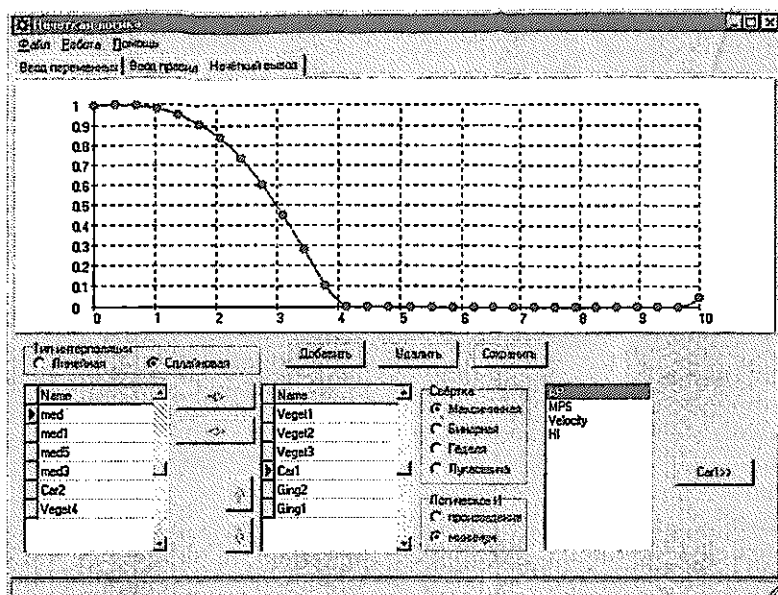


Рис. 3.15. Окно представления результатов нечеткого вывода

(линейная или сплайновая). В базе знаний предусмотрено хранение не только эталонных образцов значений ЛП, но и последних версий образцов реальных фактов, поступивших на вход системы. Информация о правилах содержит ссылки на используемые ЛП и их значения, а также на логические операции, применяемые в антецеденте.

Исследование различных способов вычисления импликации показало, что в подавляющем большинстве случаев полученные разными способами результаты позволяют правильно судить о приближенной истинности или ложности выведенных понятий. Однако практически во всех примерах имеют место отличия в результатах, которые усиливаются тем больше, чем больше образцы реальных фактов отличаются от эталонных. Невозможно рекомендовать один способ вычисления как более правильный. Вопрос о потерях точности в процессах многошаговых логических выводов требует дальнейшего исследования, так как в нечеткой логике результат вывода на каждом правиле требует соотнесения с образцом истинного или противоположного ему суждения. Эта задача возлагается либо на пользователя, либо решается самой системой на основе оценки степени сходства. В ситуациях, когда меры сходства с обоими образцами примерно одинаковы, существует высокая вероятность ошибки, которая может фатально повлиять на окончательный результат. В подобных ситуациях целесообразно отнесение полученного результата к обоим классам понятий и реализация двух версий рассуждения.

Программная система «Нечеткая логика» не предусматривает структуризации закладываемых в нее знаний. Это приводит к ограничению размерности решаемых задач и создает определенные неудобства для пользователя. Ему приходится самому подбирать группу правил для решения конкретной задачи, он участвует в процессе принятия решения, связанного с интерпретацией полученных результатов. Тем не менее такая организация системы имеет свои преимущества, позволяющие решить ряд проблем, в том числе:

1. Снимается проблема несовместимости противоречивых знаний в рамках единой системы логического типа.
2. В процессе вывода не участвует и не обрабатывается лишняя информация.
3. Система лояльна к восприятию новых знаний.
4. Увеличивается число возможных цепочек правил, участвующих в процессе одного вывода.

Эти положительные свойства позволяют выдвинуть гипотезу о целесообразности организации БЗ с переменной структурой в промышленных ЭС, которые должны включать диалоговую компоненту, предназначенную для различных вариантов структурирования знаний, а также средства проверки на непротиворечивость сформированных структур знаний.

3.5. ПРИМЕР ПРОГНОЗИРОВАНИЯ РАЗВИТИЯ ПРЕДПРИЯТИЯ С ПРИМЕНЕНИЕМ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ НА БАЗЕ НЕЧЕТКОЙ ЛОГИКИ

Рассмотрим задачу анализа текущего состояния предприятия и прогнозирования его развития на ближайшее будущее. Крупные зарубежные предприятия и фирмы ведут постоянный мониторинг изменяющегося окружения, отслеживают тенденции и вырабатывают политики достижения желаемого будущего. При управлении стратегического типа предприятия справляются с возможными кризисными ситуациями до их наступления, принимая упреждающие меры. Этот путь (стратегический) является наиболее перспективным, однако он требует солидных затрат, которые не могут себе позволить малые предприятия. Другой подход состоит в проведении оперативного анализа состояния предприятия и основных факторов окружения, а также в своевременном принятии адекватных решений. Этот способ активного управления приемлем при высокой динамичности факторов внешнего окружения, но требует от руководителей способности действовать быстро и решительно. Антикризисные меры принимаются при первых признаках ухудшения состояния предприятия. Третий, наиболее рискованный способ управления не требует применения специальных мер для анализа и прогнозирования. При этом управление предприятием осуществляется либо на основе интуиции руководителя, либо реактивно, т.е. антикризисные меры принимаются после того, как состояние предприятия существенно ухудшилось.

Рассмотрим фрагмент ЭС с нечетким представлением знаний, которая может оказаться полезной при активном управле-

нии предприятием. Экспертная система содержит базу знаний, которая используется для решения следующих задач:

- 1) диагностика текущего состояния предприятия;
- 2) прогнозирование состояния предприятия на ближайшее будущее;
- 3) выработка рекомендаций для достижения желаемого состояния в будущем.

Знания в ЭС представлены в виде правил, содержащих выражения с ЛП, значениям которых соответствуют нечеткие множества. Для решения каждого вида задач сформированы наборы правил, которые могут быть скорректированы пользователем. Правила диагностики состояния предприятия включают ЛП, которые отражают тенденции, значения или экспертные оценки основных экономических показателей:

- стадия жизненного цикла;
- форма собственности;
- технологическая оснащенность;
- направленность потоков сырья, комплектующих и готовой продукции (экспорт — импорт);
- относительная прибыльность;
- рентабельность;
- ликвидность;
- стоимость капитала;
- соотношение собственных и заемных средств;
- оборачиваемость средств;
- диверсификация продукции и т.д.

На основе анализа нечетких значений показателей выводится нечеткая оценка текущего состояния предприятия, которая используется в процессе прогнозирования будущего состояния. При этом используется другой набор правил прогнозирования, включающих информацию о ресурсах, политиках и стратегиях предприятия, а также о наблюдаемых и предполагаемых изменениях внешних факторов. Результаты прогнозирования могут иметь условный характер (т.е. возможность перехода в определенное состояние зависит от ряда условий). Задача выбора мероприятий, направленных на достижение благоприятного состояния в будущем, решается с использованием третьего набора правил, включающих значения оценок текущего и желаемого состояний предприятия.

Рассмотрим возможный набор правил для диагностики состояния предприятия:

1. ЕСЛИ относительная прибыльность предприятия (RP) падает до критического уровня ($Critical$), ТО его цена ($EntCost$) уменьшается ($Fall$), ИНАЧЕ цена ($EntCost$) остается стабильной ($Constant$).
2. ЕСЛИ RP возрастает ($Raise$), ТО $EntCost$ увеличивается ($Grow$).
3. ЕСЛИ RP падает ($Reduce, Critical$), ТО стоимость акций ($ActCost$) понижается ($Fall$), ИНАЧЕ $ActCost=Constant$.
4. ЕСЛИ RP падает ($Reduce, Critical$), ТО риск потери капитала ($Risk$) возрастает (Up), ИНАЧЕ $Risk=Constant$.
5. ЕСЛИ цена предприятия падает ($EntCost=Fall$) И суммарные обязательства кредиторам ($CreditPay$) велики ($Large$), ТО капитал акционеров уменьшается ($Capital=Fall$), ИНАЧЕ $Capital=NotFall$.
6. ЕСЛИ стоимость акций падает ($ActCost=Fall$) И риск возрастает ($Risk=Up$), ТО возможности кредитования ($CreditTake$) уменьшаются ($Fall$).
7. ЕСЛИ прибыльность предприятия падает ($RP=Reduce, Critical$) И уменьшается кредитование ($CreditTake=Fall$), ТО возможен дефицит средств ($Deficit=Yes$), ИНАЧЕ $Deficit=No$.
8. ЕСЛИ есть дефицит средств ($Deficit=Yes$) И акционерный капитал уменьшается ($Capital=Fall$), ТО предприятие находится в состоянии технической неплатежеспособности ($StateEnt=TechCrisis$).
9. ЕСЛИ $EntCost=Constant$ И $Capital=NotFall$, ТО $StateEnt=Norma$, ИНАЧЕ $StateEnt=TechCrisis$.
10. ЕСЛИ уменьшился акционерный капитал предприятия ($Capital=Fall$) И снизилось кредитование ($CreditTake=Fall$), И доля заемных средств велика ($CreditMeans=Large$), ТО предприятие находится в состоянии кризиса ($StateEnt=Crisis$).
11. ЕСЛИ $EntCost=Constant$ и $Deficit=No$, ТО $StateEnt=Norma$.
12. ЕСЛИ $EntCost=Grow$ и $Deficit=No$ И $CreditPay=NotLarge$, ТО $StateEnt=Growth$.

В процессе логического вывода на заданной цепочке правил получаются нечеткие образцы значений ЛП, которые по выбору пользователя записываются в БЗ для осуществления последующих шагов вывода, а также для прогнозирования. При этом могут

использоваться правила с одним выходом или с двумя. Простые правила содержат антецедент, включающий только одно значение ЛП. Сложные правила могут содержать нечеткие операции отрицания, конъюнкции и дизъюнкции.

На рис. 3.16 показаны функции принадлежности нечетких множеств, соответствующие значениям ЛП *StateEnt*, которая характеризует возможные состояния предприятия (*Crisis*; *TechCrisis*; *Norma*; *Growth*).

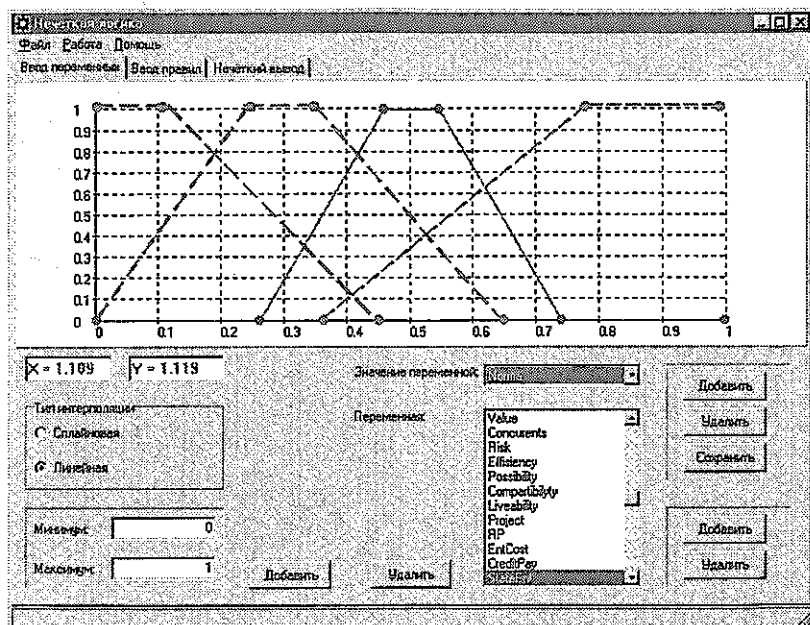


Рис. 3.16. Функции принадлежности нечетких множеств, соответствующих значениям переменной *StateEnt*

Результат обработки первого правила показан на рис. 3.17. Он больше свидетельствует в пользу заключения о падении цены предприятия, чем о ее постоянстве. В то же время принадлежность к значению *Constant* ЛП *EntCost* в данном случае является достаточно высокой (>0.4).

На следующем шаге вывода используются полученное значение ЛП *EntCost* (*Fall*) и нечеткая оценка величины аукционного капитала (ЛП *Capital*). Нечеткая оценка состояния предприятия,

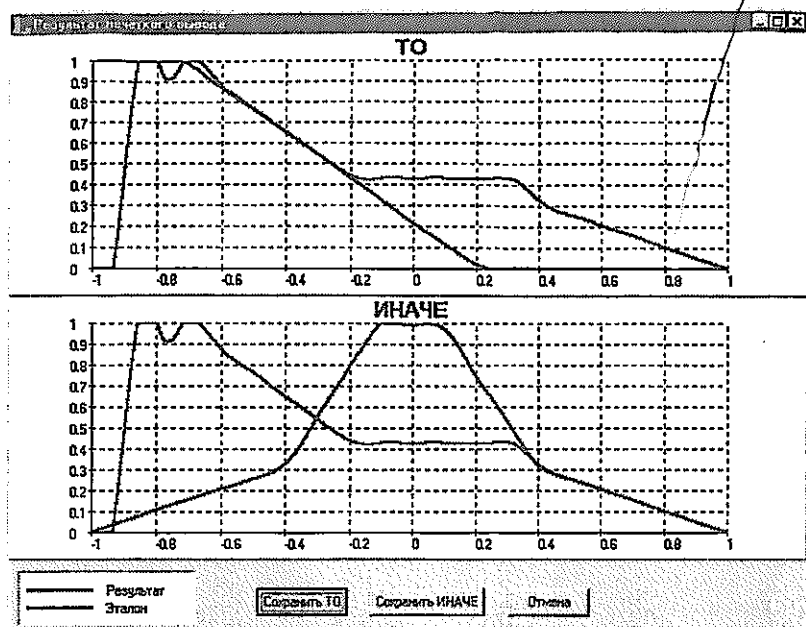


Рис. 3.17. Результат нечеткого вывода значения переменной *EntCost*

полученная в результате вывода, представлена на рис. 3.18. Очевидно, что полученный результат более сходен со значением *TechCrisis*, выводимым по ветке ИНАЧЕ, чем со значением *Norma*, выводимым по ветке ТО. Однако степень принадлежности к предкризисному состоянию приблизительно равна ≈ 0.7 , что можно интерпретировать скорее как угрозу предкризисного состояния.

Оценка состояния предприятия может проводиться по различным характеристикам, в том числе по показателям бухгалтерской отчетности. Количественный характер этих показателей позволяет применять другие, более точные методы оценки. Следует заметить, что адекватные результаты можно получить только на основе статистического анализа динамики упомянутых количественных показателей, позволяющего нивелировать помехи. При использовании нечеткого подхода важно умение устанавливать правильное соответствие между конкретными числами и значениями ЛП. Существенное влияние на результаты имеют выбор ЛП и формулировка правил.

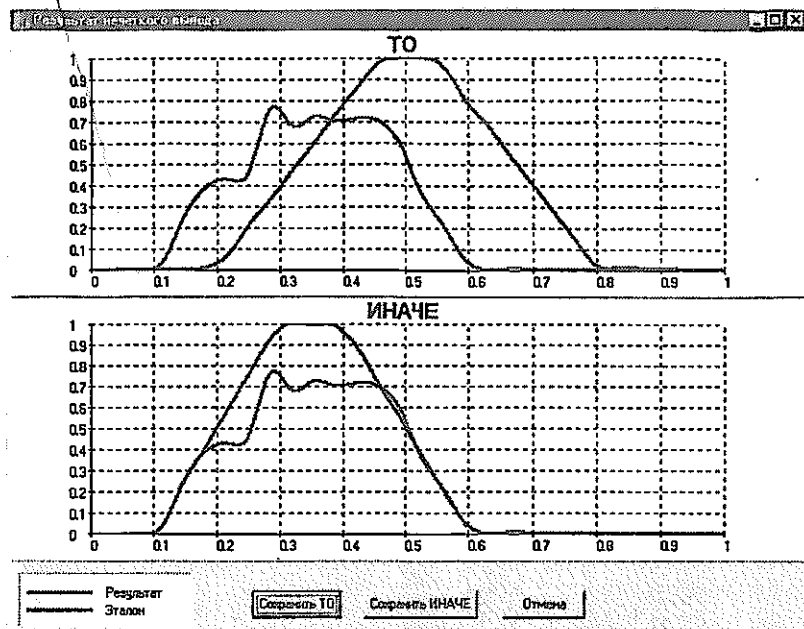


Рис. 3.18. Нечеткая оценка состояния предприятия

Задача краткосрочного прогнозирования состояния предприятия решается с использованием оценки текущего состояния на новом наборе правил, содержащих в качестве условий предположения о возможных параметрах окружения предприятия и возможных действиях руководства. Например, для предприятий, находящихся в нормальном состоянии, рассматривая вероятные изменения внешних факторов (падение курса рубля, удорожание кредитов, увеличение налогов, рост цен на энергоносители и т.п.), можно получить неблагоприятный прогноз на будущее. Тогда возникает необходимость решения следующей задачи — разработки политики развития предприятия, которая обеспечит его выживание и процветание. Предприятия с положительной динамикой экономических показателей также должны учитывать возможные изменения окружающей среды, чтобы принимать своевременные меры для обеспечения хорошего состояния в будущем. Для предприятий в состоянии кризиса отпадает необходимость прогнозирования, так как в этом случае можно сразу пере-

ходить к разработке антикризисных мер. Особую актуальность задачи прогнозирования и выработки политик имеют для предприятий, находящихся в предкризисном состоянии. В этом случае указанные задачи объединяются в одну, т.е. составляется один набор правил, элементами которых становятся прогнозируемые значения ЛП, описывающих внешние факторы, а также значения ЛП, соответствующих действиям по управлению предприятием. Наиболее важными внутренними факторами, влияющими на будущее предприятия, которое находится в предкризисном состоянии, являются следующие:

- стадия жизненного цикла;
- отрасль деятельности;
- направленность потоков сырья и продукции;
- форма собственности;
- соотношение собственных и заемных средств;
- доля рынка;
- диверсификация производства;
- прогрессивность используемых технологий;
- конкурентоспособность продукции;
- уровень запасов;
- оборачиваемость средств;
- ликвидационная стоимость активов;
- уровень издержек производства;
- ценовая политика;
- профессионализм кадров.

Представляя перечисленные факторы с помощью лингвистических переменных, можно сформировать множество правил, например, следующего вида:

1. ЕСЛИ *StateEnt*=*TechCrisis* И *Отрасль*=*Добывающая* И *Направленность_продукции*=*Экспорт* И *Курс_рубля*=*Падает* И *Доля_рынка*=*Большая*, ТО *Платежеспособность*=*Растет*.
2. ЕСЛИ *StateEnt*=*TechCrisis* И *CreditPay*=*Small* И *Платежеспособность*=*Растет* И *Конкурентоспособность*=*Низкая* И *Технологии*=*Старые* И *Спрос*=*Высокий*, ТО *Действие*=*Обновить_технологии*.
3. ЕСЛИ *Действие*=*Обновить_технологии*, ТО *Конкурентоспособность*=*Высокая*.
4. ЕСЛИ *StateEnt*=*TechCrisis* И *CreditPay*=*Large* И *Платежеспособность*=*Растет*, ТО *Действие*=*Уменьшить_кредитные_обязательства*.

5. ЕСЛИ Действие=Уменьшить_кредитные_обязательства, ТО CreditPay=Small.
6. ЕСЛИ StateEnt=TechCrisis И CreditPay=Small И Платежеспособность=Растет И Конкурентоспособность=Низкая И Доля_рынка=Малая И Спрос=Низкий, ТО Действие=Диверсификация_производства.
7. ЕСЛИ StateEnt=TechCrisis И Конкурентоспособность=Высокая И Спрос=Высокий, ТО Действие=Снизить_издержки.
8. ЕСЛИ StateEnt=TechCrisis И CreditPay=Small И Платежеспособность=Растет И Конкурентоспособность=Высокая И Спрос=Высокий, ТО Действие=Увеличить_мощность.
9. ЕСЛИ Действие=Диверсификация_производства, ТО Доля_рынка=Высокая.
10. ЕСЛИ Действие=Снизить_издержки, ТО Конкурентоспособность=Высокая.
11. ЕСЛИ Действие=Увеличить_мощность, ТО Рентабельность=Высокая.
12. ЕСЛИ Платежеспособность=Растет И Конкурентоспособность=Высокая И Доля_рынка=(Большая ИЛИ Средняя), ТО StateEnt=Norma.
13. ЕСЛИ Платежеспособность=Растет И Рентабельность=Высокая, ТО StateEnt=Norma.
14. ЕСЛИ Платежеспособность=Не_растет И Конкурентоспособность=Низкая И Спрос=Низкий И Рентабельность=Низкая, ТО StateEnt=Crisis.
15.

Логический анализ ситуаций с нечеткими переменными обладает большей гибкостью, чем традиционная логика, и не требует полного набора аксиом для вывода. Эти же качества являются источниками основных недостатков нечеткого логического подхода к задачам диагностики и прогнозирования. Они заключаются в отсутствии обоснованных процедур построения функций принадлежности, в трудностях представления знаний и интерпретации результатов. Проведенные исследования методов нечеткой логики в задачах принятия решений и прогнозирования дают основания полагать, что эти подходы наиболее плодотворны в тех случаях, когда для построения функций принадлежности можно использовать объективную информацию, например, в нечетком управлении или в нечеткой диагностике по данным измерений [1, 8, 9].

Практическое применение методов обработки нечетких суждений позволяет сделать следующие выводы:

1. Большое количество правил, ЛП и их значений приводит к противоречиям и/или к плохо интерпретируемым результатам нечеткого вывода. Лингвистический подход уместно применять для описания закономерностей общего характера, свойственных рассматриваемой предметной области. При этом задача определения набора важнейших ЛП и формирования соответствующих правил может быть решена только с помощью экспертов высшей квалификации. Для более точного описания состояния предприятия следует применять другие модели, основанные на количественных методах описания множества характеристик.

2. Главной проблемой нечетких методов является сложность построения функций принадлежности и еще более высокая сложность интерпретации получаемых результатов. Методы выполнения свертки нечетких отношений влияют на результат в гораздо меньшей степени, чем вид используемых функций принадлежности.

3. Приемлемое представление знаний получается, как правило, в результате серии вычислительных экспериментов, проводимых с помощью специалиста в области инженерии знаний.

3.6. ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ РАБОТЫ С НЕЧЕТКИМИ ЗНАНИЯМИ

Коммерческое программное обеспечение, основанное на нечетких моделях представления знаний, широко представлено на мировом рынке. Большая часть программных продуктов ориентирована на решение научно-технических задач и продвигается на рынок вместе с соответствующей аппаратурой. В табл. 3.1 приведены наименования и краткие характеристики некоторых известных в России программных средств, основанных на нечетких моделях [13].

Пакет прикладных программ *FuziCalc* предназначен для выполнения быстрых прикидочных расчетов в задачах принятия решений, возникающих в различных областях бизнеса. Пакет обес-

Программные средства, основанные на нечетких моделях

Продукт	Фирма	Область применения
AIS	Adaptive Information Systems	Извлечение нечеткой информации и гибкие опрашивающие системы
FuziCalc	FuziWare	Программная среда, ориентированная на анализ финансовых проблем
Бизнес-прогноз	Тора-Инфо-Центр (дистрибьютер)	Поддержка процессов принятия управленческих решений
CubiCalc	HyperLogic's	Система для поддержки задач нечеткого моделирования и управления
FMC	Transfer-Tech	Оболочка для разработки нечетких ЭС
FRIL	FrilSystems Ltd	Нечеткий язык логических выводов
FuzzySoft	Fuzzy Soft AG	Нечеткая система моделирования
Metus Fuzzy Library	Metus Systems	Библиотека программ для обработки нечетких представлений на языке C/C++
NeuFuz	National Semiconductor	Нейронная сеть и нечеткая логика
Fide	Aptronics	Инструментальная среда для разработки нечетких систем, использующая микропроцессоры на нечеткой логике
FuzzyTECH	Inform Software	Программная среда для разработки нечетких ЭС
TILShell	Togai Infra Logic	Совокупность программ и аппаратных средств для создания нечетких ЭС

печивает вычисления в формате электронных таблиц и удобен для анализа финансовых показателей. Он содержит большое число стандартных функций, используемых в задачах анализа финансово-хозяйственной деятельности предприятий. С помощью пакета *FuziCalc* можно определять показатели прибыльности и окупаемости бизнеса, оценки задолженности клиентов и оборачиваемости товарно-материальных запасов, а также ускорять составление оперативных финансовых отчетов и других документов.

Российский программный продукт *Бизнес-прогноз* также ориентирован на поддержку управленческих решений в бизнесе. В нем предусмотрено представление бизнес-процесса деревом решений, при этом оценка возможных исходов осуществляется с помощью функций принадлежности. Вычисление нечетких оценок альтернативных решений проводится на основе наблюдаемых или субъективно назначенных вероятностей событий. Пакет позволяет решать задачи прогнозирования качества сделок с учетом ряда вычисляемых показателей, в том числе вероятного дохода от сделки; степени риска (вероятность безубыточного исхода сделки); степени неопределенности результата; доли средневероятной прибыли в общей сумме сделки и т. п.

К известным алгоритмически-ориентированным программным продуктам относятся пакеты *CubiCalc* и *NeuFuz*. Пакет *CubiCalc* предоставляет набор средств для построения нечетких моделей, в том числе нечеткий редактор для ввода и редактирования значений ЛП, соответствующих им нечетких множеств и правил; операторы, позволяющие изменять значения ЛП с помощью специальных модификаторов (например, «несколько», «очень»); весовые коэффициенты правил. *CubiCalc* содержит нечеткий словарь и редактируемую базу знаний. Результаты нечеткого вывода представляются в виде двумерной решетки.

Система *NeuFuz* сочетает в себе нечеткий вывод и нейронную сеть, которая используется для автоматического формирования правил и функций принадлежности по примерам. Нейронная сеть состоит из семи слоев и обучается по методу обратного распространения ошибки (см. главу 5).

Инструментальная среда *Fide* позволяет строить нечеткие интеллектуальные системы путем интеграции модулей нечеткого вывода, каждый из которых определен в текстовом файле, содержащем определения входных и выходных переменных, связанных с нечеткими множествами и правилами. Отличительной чертой системы *Fide* является креативный модуль, включающий подсистему анализа и подсистему моделирования.

Среда для разработки нечетких ЭС *FuzzyTECH* содержит множество редакторов, предназначенных для создания входных и выходных переменных, функций принадлежности и правил.

Данная среда порождает программные коды интеллектуальных систем на языке C/C++ и других языках, ориентированных на специальные микроконтроллеры.

Инструментальная система *TILShell* является средством CASE-технологии, предназначенным для создания нечетких экспертных систем. Ядро системы содержит редакторы правил и функций принадлежности, а также развитые средства для быстрой разработки конкретных приложений, которые легко могут быть состыкованы с другими программными разработками, сделанными с использованием инструментария TIL.

Литература

1. *Андрейчиков А. В., Андрейчикова О. Н.* Анализ, синтез, планирование решений в экономике. — М.: Финансы и статистика, 2000.
2. *Борисов А. Н., Крумберг О. А., Федоров И. П.* Принятие решений на основе нечетких моделей. — Рига, Зинатне, 1990.
3. *Вагин В. Н., Загорянская А. А.* Организация абдуктивного вывода средствами теории аргументации // Тр. конгресса «Искусственный интеллект в 21 веке». — М.: Изд-во физико-математической литературы, 2001.
4. *Заде Л.* Понятие лингвистической переменной и его применение к принятию приближенных решений: Пер. с англ. — М.: Мир, 1976.
5. Логический подход к искусственному интеллекту: Пер. с фр. / *А. Тейз, П. Грибомон, Ж. Луи и др.* — М.: Мир, 1990.
6. *Лорьер Ж.-Л.* Системы искусственного интеллекта: Пер. с фр. / Под ред. В. Л. Стефанюка. — М.: Мир, 1991.
7. *Нейлор К.* Как построить свою экспертную систему: Пер. с англ. — М.: Энергоатомиздат, 1991.
8. Нечеткие множества и теория возможностей. Последние достижения: Пер. с англ. / Под ред. Р. Р. Ягера. — М.: Радио и связь, 1986.
9. Нечеткие множества в моделях управления и искусственного интеллекта / Под ред. Д.А. Поспелова. — М.: Физматлит, 1986.
10. *Осуга С.* Обработка знаний: Пер. с яп. — М.: Мир, 1989.
11. Представление и использование знаний: Пер. с яп. / Под ред. Х.Уэно, М. Исидзука. — М.: Мир, 1989.
12. *Фор А.* Восприятие и распознавание образов: Пер. с фр. / Под ред. Г. П. Катыга. — М.: Машиностроение, 1989.

13. *Фролов Ю. В.* Интеллектуальные системы и управленческие решения. – М.: МГПУ, 2000.
14. *Хант Э.* Искусственный интеллект: Пер. с англ./ Под ред. В. Л. Стефанюка. – М.: Мир, 1978.
15. Экспертные системы. Принципы работы и примеры: Пер. с англ. / Под ред. Р. Форсайта – М.: Радио и связь, 1987.

Контрольные вопросы и задания

1. Поясните смысл понятия «нечеткость» знаний. Дайте характеристику компонентам нечеткости.
2. Что такое недетерминированность выводов? Какие средства следует использовать в системах, обладающих этим свойством?
3. Проанализируйте какую-либо известную вам игру на предмет ее машинной реализации. Какими приемами необходимо обеспечить компьютерного игрока в этой игре?
4. Расскажите о способах устранения многозначности. Почему ее необходимо устранять? Приведите примеры.
5. Какими способами можно представлять и обрабатывать ненадежные знания? Приведите примеры.
6. Охарактеризуйте способы обработки неполных знаний в интеллектуальных системах. Приведите собственный пример появления противоречия в логической ЭС при добавлении нового знания. Какие преимущества по сравнению с логическими имеют фреймовые системы?
7. Что такое абдукция? Дайте формальное определение и объясните, чем она отличается от дедукции. Приведите примеры.
8. Дайте определение понятий «лингвистическая переменная» и «нечеткое множество», поясните их на примере. Какие операции можно выполнять над нечеткими множествами? Рассмотрите математические операции над нечеткими множествами: дополнение, объединение и пересечение множеств.
9. Дайте определение нечеткого отношения и расскажите о свойствах нечетких отношений. Каким образом и с какой целью используются нечеткие отношения в ИИС?
10. Что такое нечеткая импликация? Какими способами ее можно реализовать для правил с одним выходом и двумя выходами? Приведите примеры.

11. Расскажите о нечетком логическом выводе. Чем он отличается от четкого логического вывода по правилу *Modus Ponendo Ponens*. Дайте определение композиции нечетких отношений. Приведите пример нечеткой композиции.
12. Расскажите об организации ЭС с нечетким логическим выводом. Какие способы используются в таких системах для представления и обработки знаний?
13. Разработайте БЗ небольшого объема с нечетким представлением знаний.
14. Подготовьте собственные наборы нечетких правил для реализации в системе с нечетким выводом.
15. Разработайте интеллектуальную систему с нечетким представлением знаний для решения небольшой прикладной экономической или управленческой задачи.

Самое непостижимое в мире то, что он постижим.

Альберт Эйнштейн

Проблема приобретения знаний включает две основные задачи: получение информации и ее систематизацию. Процесс получения знаний от экспертов является ключевым при разработке интеллектуальных информационных систем, которые постоянно нуждаются в новых знаниях, поэтому на стадии разработки необходимо рассматривать проблемы их перманентного обучения. С развитием средств информатизации появились новые задачи, связанные с управлением знаниями в ИИС.

4.1. СТРАТЕГИИ ПОЛУЧЕНИЯ ЗНАНИЙ

Извлечением знаний называют процесс получения знаний от экспертов. Извлечение знаний — сложная и трудоемкая процедура, в результате которой инженеру по знаниям (когнитологу, аналитику) необходимо создать собственную модель предметной области на основе информации, полученной от экспертов. Попытки получить знания, необходимые для разработки ИИС, непосредственно от экспертов и обойтись без когнитологов обычно не приводят к успеху, так как в этом случае предъявляются очень высокие требования к эксперту, который, являясь специалистом в предметной области, будет вынужден приобрести квалификацию инженера по знаниям. Кроме того, существует еще несколько причин, вызывающих необходимость участия аналитиков в процессах извлечения знаний, а именно:

- лучшим способом для вербализации знаний эксперта является диалог;
- опытный аналитик, вооруженный современной методологией системного анализа, может помочь эксперту в структуризации знаний предметной области;

• инженер по знаниям помогает эксперту осознать «скрытые» знания, предлагая ему установить причинно-следственные связи (а также связи другой природы) на множестве выделенных понятий.

Успех на этапе извлечения знаний во многом зависит от квалификации аналитика, который должен иметь образование, включающее знания из разных областей, в том числе из когнитивной психологии, системного анализа, математической логики, искусственного интеллекта и т.д.

Как правило, каждый когнитолог сам изобретает язык для описания полученных от эксперта знаний путем пополнения сложившегося языка конкретной науки специальными терминами и знаками. Стандарта таких языков пока не существует. Тем не менее желательно, чтобы такие языки были понятными и содержали как можно меньше неточностей. Разработка языков инженерии знаний ведется в различных направлениях, в частности известны языки-классификации, логико-конструктивные языки, структурно-логические и др. Перспективный подход к созданию подобных языков открывает *семиотика* — наука о знаковых системах. Классическая семиотика является чисто гуманитарной наукой, основные интересы которой сосредоточены в области культуры человеческого поведения, искусства и языка. Область исследований *прикладной семиотики* связана с применением знаковых систем для представления и обработки знаний в практических приложениях ИИ [36].

4.1.1. АСПЕКТЫ ИЗВЛЕЧЕНИЯ ЗНАНИЙ

Процессы извлечения знаний рассматривают в трех основных аспектах: психологическом, лингвистическом и гносеологическом (рис. 4.1).

Психологический аспект. Это самый важный из всех аспектов, так как извлечение знаний происходит в процессе *общения* когнитологов с экспертами, где психология играет доминирующую роль.

Процесс извлечения знаний для интеллектуальных систем необходимо организовать не как однонаправленный процесс передачи сообщений от эксперта аналитику, а как совместный поиск истины [11, 17].

В процессе разговорного общения много информации теряется [28], поэтому важной проблемой является увеличение ин-

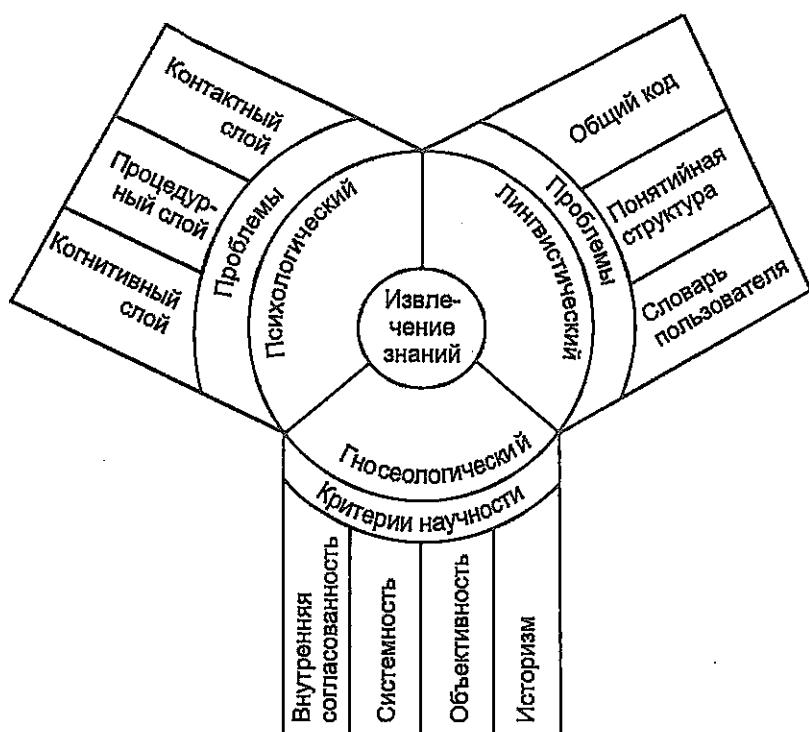


Рис. 4.1. Аспекты извлечения знаний

формативности общения аналитика и эксперта за счет использования методик, выработанных в психологии (рис. 4.2).

Модель общения включает участников общения, средства общения и предмет общения (знания). В соответствии с этими компонентами выделяются три слоя психологических проблем: контактный, процедурный, когнитивный.

Степень информативности общения аналитика и эксперта на уровне *контактного* слоя зависит в основном от пола, возраста, темперамента личности и мотивации участников общения. Установлено, что хорошие результаты дают гетерогенные пары (мужчина/женщина) и соотношение возрастов:

$$5 < (B_3 - B_a) < 20,$$

где B_3 — возраст эксперта;

B_a — возраст аналитика.

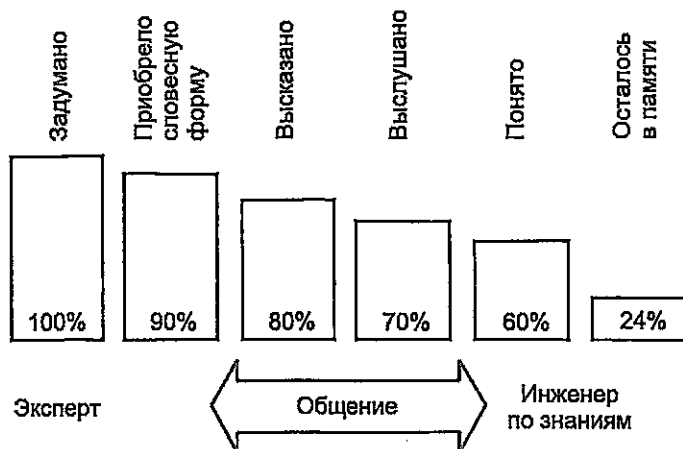


Рис. 4.2. Потери информации при разговорном общении

Желательно, чтобы участники процесса общения обладали такими качествами, как доброжелательность, хорошая память, внимание, наблюдательность, воображение, впечатлительность, собранность, настойчивость, общительность и находчивость [11].

В рамках контактного слоя наиболее предпочтительными из четырех классических типов темперамента являются сангвиники и холерики.

Параметры *процедурного слоя* обеспечивают эффективность извлечения знаний. К этим параметрам относятся: ситуация общения (место, время, продолжительность); оборудование (вспомогательные средства, освещенность, мебель); профессиональные приемы (темп, стиль, методы и др.). Для повышения эффективности процесса извлечения знаний инженер по знаниям должен подобрать значимые для эксперта стимулы, поскольку последний передает аналитику один из самых ценных ресурсов — знания.

Когнитивный слой связан с изучением семантического пространства памяти эксперта и с воссозданием его понятийной структуры и модели рассуждений. Когнитивный слой характеризуется когнитивным стилем и семантической репрезентативностью.

Под *когнитивным стилем* человека понимается специфическая совокупность критериев, используемых им в процессе познания мира для решения различных задач. Когнитивный стиль —

это система средств и индивидуальных приемов, к которым прибегает человек для организации своей деятельности, обеспечивающей достижение желаемых результатов. Для повышения эффективности извлечения знаний целесообразно подбирать экспертов и аналитиков, обладающих вполне определенными характеристиками когнитивного стиля. Наиболее важными из них являются следующие: полнезависимость (независимость от шумового поля), импульсивность — рефлексивность, жесткость — гибкость, когнитивная эквивалентность.

Желательно, чтобы и аналитики, и эксперты обладали следующими когнитивными характеристиками:

- *высокой полнезависимостью*, которая подразумевает способность выделять главные аспекты рассматриваемой проблемы и отбрасывать все лишнее, что не относится к поставленной задаче. Это качество желательно иметь и аналитику, и эксперту, однако следует учитывать тот факт, что полнезависимые люди более контактны и склонны к общению;

- *рефлексивностью*, характеризующей склонность к рассудительности и самоанализу (в то время как импульсивность характеризуется быстрым, зачастую недостаточно обоснованным принятием решений);

- *когнитивной эквивалентностью*, определяющей способность человека к различению понятий и разбиению их на классы и подклассы;

- эксперты — устойчивостью представлений, т. е. *жесткостью* структуры восприятия, а аналитики — *гибкостью*, т. е. умением легко приспосабливаться к новой обстановке.

Для эффективного построения ИИС инженер по знаниям должен владеть специальными неформальными методами и математическим аппаратом, позволяющими ему воссоздавать полученные от эксперта знания с помощью различных моделей, например, таких, как продукционная или фреймовая. Не навязывая эксперту какой-либо модели, аналитик должен подобрать средства представления знаний, имеющие максимально высокую семантическую репрезентативность.

Лингвистический аспект. Актуальность исследования этого аспекта определяется тем, что язык является основным средством общения в процессе извлечения знаний. В области лингвистических проблем наиболее важными являются понятия: общий код, понятийная структура, словарь пользователя.

Общим кодом называют специальный промежуточный язык общения между экспертом и инженером по знаниям. Этот язык включает совокупность общенаучных и специальных понятий из профессиональной литературы, элементов базового языка, неологизмов и т.п. (рис. 4.3). Общий код позволяет преодолеть языковой барьер в процессе общения когнитологов с экспертами. Выработка общего кода для партнеров осуществляется в соответствии с информационными потоками, представленными на рис. 4.3.

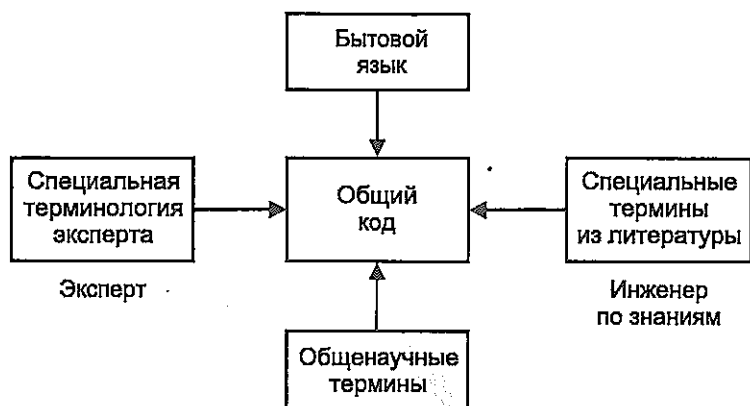


Рис 4.3. Структура формирования общего кода

В дальнейшем общий код преобразуется в *понятийную структуру*, или семантическую сеть, которая связывает понятия, хранящиеся в памяти человека. Выявление отношений между элементами знаний, представленных понятиями, является одной из самых сложных проблем в процессах извлечения знаний. Хорошо известно, что естественные знания человека представляют собой связанные структуры, а не разрозненные фрагменты. Однако до настоящего времени при разработке БЗ учитывается весьма ограниченный набор связей между понятиями, в то время как в действительности существует большое разнообразие таких отношений. Д. А. Поспелов [35] выделяет более 200 базовых видов отношений между понятиями. Такое многообразие делает невозможным однозначное определение набора признаков, описывающих конкретное понятие, и, как следствие, однозначную классификацию понятий. Очевидно, сложность данной проблемы яв-

ляется главной причиной того, что на сегодняшний день отсутствуют надежные методики формирования понятийных структур.

Тем не менее построение той или иной иерархии понятий входит в задачи концептуального анализа структуры знаний любой предметной области. В последнее время в ИИ стал широко употребляться термин *онтология*, имеющий многозначную интерпретацию [63], в частности в [11] приводятся следующие его определения.

1. Онтология как философская дисциплина представляет собой систему категорий, являющихся следствием определенного взгляда на мир.

2. Онтология как неформальная система концептуализации знаний предполагает построение описания множества выделенных объектов, понятий, связей и отношений в заданной области знаний. При этом могут использоваться формальные или неформальные средства. В простейшем случае онтология может описывать только иерархию понятий, связанных отношениями «элемент-класс». Помимо этого она может содержать набор аксиом и правил вывода, позволяющих выразить другие отношения между понятиями и ограничить область интерпретации понятий.

3. Онтология как представление концептуальной системы в виде логической теории означает использование определенного синтаксиса для представления знаний.

Разработка *словаря пользователя* необходима в связи с тем, что конечный пользователь не обязан владеть профессиональным языком предметной области, который использовался при построении БЗ. Интерфейс пользователя создается, как правило, путем доработки словаря общего кода.

Гносеологический аспект. Он объединяет методологические проблемы получения нового научного знания, так как процесс познания часто сопровождается появлением новых понятий и теорий. В процессе разработки БЗ эксперты довольно часто впервые формулируют некоторые закономерности на основе накопленного эмпирического опыта. Последовательность *факт*→*обобщенный факт*→*эмпирический закон*→*теоретический закон* называется гносеологической цепочкой. Теория — это не только система обобщения накопленных знаний, но и способ получения нового знания.

Основными критериями качества новых знаний являются внутренняя согласованность, системность, объективность, историзм.

В процессе извлечения знаний аналитиков прежде всего интересуют эмпирические знания экспертов, представляющие собой результаты наблюдений, которые могут оказаться несогласованными. *Внутренняя согласованность* эмпирических знаний характеризуется понятиями модальности, противоречивости, неполноты. Под модальностью знания понимается возможность его существования в различных категориях. Противоречивость является естественным свойством эмпирических знаний и не всегда может и должна быть устранена. Напротив, противоречивость может служить отправной точкой в рассуждениях экспертов. Неполнота знаний связана с невозможностью исчерпывающего описания любой предметной области.

Системность знаний основана на определении места новых знаний в многоуровневой иерархической организации. При этом необходимо найти ответы на вопросы: какие понятия детализируют или обобщают новые знания и в каких отношениях они состоят с известными фактами и закономерностями?

Объективность знаний определить практически невозможно. Процессы накопления, описания, представления, обработки, интерпретации и оценивания качества знаний выполняются конкретными людьми, поэтому их результаты имеют субъективный характер. Объективность некоторых закономерностей часто связывают с широтой области их применимости. Границы этой области можно установить экспериментальным или теоретическим путем, но далеко не всегда. В качестве косвенных свидетельств объективности иногда допускают совпадение представлений различных экспертов и подтверждение выдвинутых гипотез известными фактами.

Историзм знаний связан с развитием и изменением представлений о предметной области с течением времени.

Процесс познания можно представить следующими этапами:

- описание и обобщение фактов;
- выявление связей между фактами, формулирование правил и закономерностей;
- построение модели знаний предметной области;
- объяснение и прогнозирование явлений на основе модели.

На начальных этапах инженер по знаниям, исследуя структуру умозаключений эксперта, может использовать разные теории и подходы для построения формальной модели знаний предметной области. Наиболее известными и часто применяемыми при-

емами являются математическая логика, теория ассоциаций и гештальт-психология.

Математическая логика формирует критерии, которые гарантируют точность, значимость и непротиворечивость общих понятий, рассуждений и выводов. Применяя логический подход, когнитолог выполняет следующие операции: определение понятий, выявление сходства и различия, анализ, абстрагирование, обобщение, классификацию, образование суждений, составление силлогизмов и т.д. Проблема применения логического подхода к представлению знаний заключается в том, что человек не всегда мыслит категориями строгой классической логики, а естественная система знаний не является полной, допускает противоречия и многозначные оценки истинности.

В *теории ассоциаций* мышление представляется в виде цепочки идей, связанных общими понятиями. Здесь применяются следующие приемы:

- ассоциации, приобретенные на основе связей различной природы;

- привлечение прошлого опыта;
- метод проб и ошибок со случайным успехом;
- привычные («автоматические») реакции и пр.

Гештальт-психология [8] ориентирует аналитика на выделение целостного образа или структуры знаний (*гештальта*) как основы для понимания процессов и явлений окружающего мира. Понятие гештальта во многом согласуется с понятием фрейма. Применение данной теории ориентирует эксперта на формирование модели знаний в соответствии с критериями простоты, связности и гармонии.

Идеализированная модель знаний предметной области строится на основе установленных логических связей между понятиями. Модель формализуется с помощью категориального аппарата, формально-знаковых средств математики и логики. Для адекватного отображения в модели реальной картины мира инженеру по знаниям необходимо владеть такими приемами, как идеализация, абстрагирование, огрубление. Критерием качества построенной модели является способность ИИС делать прогнозы и объяснять множество явлений из заданной предметной области. Инженер по знаниям должен стремиться, чтобы результирующая модель знаний была достаточно полной, связной и непротиворечивой.

4.1.2. ПРОБЛЕМЫ СТРУКТУРИРОВАНИЯ ЗНАНИЙ

В настоящее время для структурирования знаний используются структурный и объектный подходы. *Структурный* подход основан на идее алгоритмической декомпозиции, где каждый модуль системы выполняет один из важных этапов общего процесса. В рамках структурного подхода разработано большое число выразительных средств: диаграммы потоков данных, структурированные словари (тезаурусы), языки спецификаций систем, таблицы решений, стрелочные диаграммы, диаграммы переходов, деревья целей, средства управления проектом (PERT-диаграммы) и др. [56, 62, 75].

Объектный (объектно-ориентированный) подход связан с объектной декомпозицией, при которой каждый объект рассматривается как экземпляр определенного класса. К базовым понятиям этого подхода относятся следующие [6, 64, 73]:

- *Абстрагирование*, которое М. Шоу [73] определил как упрощенное описание системы, где выделяются наиболее существенные для рассмотрения свойства и детали, а незначительные аспекты опускаются. Абстрактное представление реальности отражено моделью сущности (*объект*) и моделью поведения (*метод*). Объекты соответствуют понятиям предметной области. Методы представляют собой операции, которые можно выполнять над объектами.

- *Класс* — множество объектов, связанных общностью структуры и свойств.

- *Иерархия* — упорядоченная система абстракций (классов).

- *Наследование* — такое соотношение между классами, когда один класс использует структурную или функциональную часть другого класса (или нескольких других).

- *Типизация* — ограничение, накладываемое на класс, которое препятствует взаимозаменяемости объектов, принадлежащих различным классам.

- *Модульность* — свойство системы, связанное с возможностью ее декомпозиции на ряд взаимосвязанных частей (модулей).

- *Инкапсуляция* — ограничение доступа к внутренней структуре и механизмам функционирования объекта.

- *Полиморфизм* — возможность наделения объекта различными свойствами и стратегиями поведения. Иными словами, одно имя может соответствовать различным классам объектов, входя-

щим в один суперкласс. Следовательно, объект, обозначенный этим именем, может по-разному реагировать на некоторое множество действий.

В настоящее время развивается *объектно-структурный* подход к структурированию знаний на основе обобщения существующих подходов [11]. Основная идея объектно-структурного подхода связана с проведением последовательного объектно-структурного анализа информации о рассматриваемой предметной области, для представления которой используется стратифицированная модель (табл. 4.1). В процессе объектно-структурного анализа происходят выделение и структуризация понятий с применением разнообразных методов анализа знаний.

На стадии структурирования знаний предметной области необходимо решить следующие задачи: составление словаря используемых терминов; выявление понятий и их атрибутов; выявление связей и определение отношений между понятиями; детализация и обобщение понятий; построение обобщенной структуры знаний предметной области.

Первым шагом структурирования знаний является определение входных и выходных данных, которые в дальнейшем будут детализироваться и уточняться. На основании этих данных, а также по результатам анализа протоколов сеансов извлечения

Таблица 4.1

Стратификация знаний предметной области

Уровень страты	Категория знаний	Вид анализа знаний
1	ЗАЧЕМ	Стратегический анализ: назначение и функции системы
2	КТО	Организационный анализ: коллектив разработчиков системы
3	ЧТО	Концептуальный анализ: основные принципы, понятийная структура
4	КАК	Функциональный анализ: гипотезы и модели принятия решений
5	ГДЕ	Пространственный анализ: окружение, оборудование, коммуникации
6	КОГДА	Временной анализ: временные параметры и ограничения
7	ПОЧЕМУ	Причинно-следственный (каузальный) анализ: формирование подсистемы объяснений
8	СКОЛЬКО	Экономический анализ: ресурсы, затраты, прибыль, окупаемость

знаний составляется набор ключевых слов (терминов), в процессе обработки которого выявляются объекты, понятия и их атрибуты. Под *понятием* подразумевается обобщение предметов некоторого класса по специфическим признакам. Формирование понятий — серьезная проблема. Для выявления понятий используются традиционные методы распознавания образов и классификации, а также нетрадиционные методы, базирующиеся на методологии инженерии знаний. Наиболее распространенными методиками выявления объектов и понятий являются:

- методика формирования перечня понятий;
- интервьюирование специалистов;
- составление списка элементарных действий;
- составление оглавления учебника.

Практическое использование этих методик показало, что наиболее результативными из них являются методики интервьюирования и составления оглавления учебника.

Существует мнение [44], что теория понятий есть только в одной науке — математике, где возможно их строгое определение. В гуманитарных науках определения понятий чаще всего отсутствуют, следовательно, там вместо понятий приходится иметь дело с идеями. Для строго определенных понятий существуют языки, способные выразить связи между ними, но для описания связей между идеями таких языков пока нет.

Отношения между понятиями могут иметь различную природу: «целое—часть», «причина—следствие», доминирование, временные и пространственные отношения, ситуативные, ассоциативные, функциональные и др. Выявление связей и отношений между понятиями является сложной задачей, для решения которой используются разнообразные средства. Многие инженеры по знаниям сами изобретают различные методы и приемы в процессе работы с экспертом. К таким методам можно отнести «сортировку карточек» и построение замкнутых кривых [11]. Один из современных подходов к представлению взаимосвязанных структур знаний основан на использовании *сценариев*, которые строятся по аналогии с организацией человеческой памяти, где все знания объединены связями различных типов. Сценарии состоят из фрагментов (сцен), связанных пространственными или временными отношениями. Элементы знаний из фрагментов могут быть связаны отношениями различной природы: функциональными, ассоциативными, ситуативными, причинно-следственными и др. [49].

Процедуры обобщения и детализации понятий плохо формализуемы и требуют от экспертов и аналитиков высокой квалификации. Иерархическое представление знаний требует установления отношений между понятиями внутри каждого уровня иерархии и между ними. Заключительным этапом структурирования знаний являются анализ возможных цепочек рассуждения и выработка правил принятия решений, которые позволяют объединить сформированные понятия и отношения в динамическую модель знаний предметной области.

Последовательность структурирования знаний зависит от особенностей конкретной области и от наработанных для рассматриваемой категории знаний предметно-ориентированных методов структурирования.

4.1.3. СЕМИОТИЧЕСКИЙ ПОДХОД К ПРИОБРЕТЕНИЮ ЗНАНИЙ

Семиотический подход к моделированию человеческих знаний считается в настоящее время одним из самых перспективных [36]. На рис. 4.4 показана схема, известная в семиотике как треугольник Фреге. Объекты реального мира, называемые *денотатами*, отражаются в сознании человека (ментальном мире), и в результате этого отражения возникают представления о денотатах.

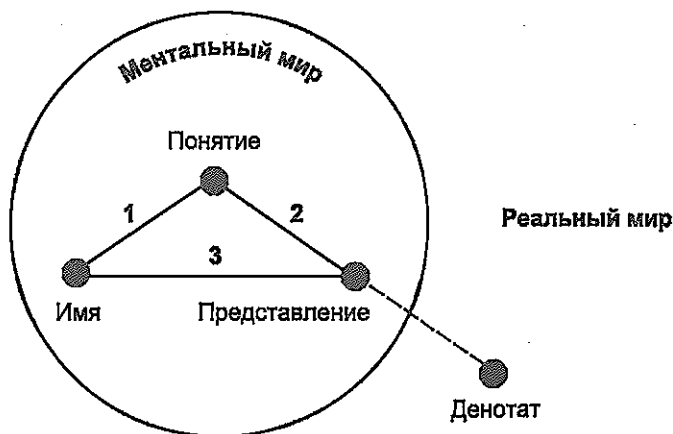


Рис. 4.4. Треугольник Фреге

Представление — это интегрированный образ денотата (называемый в психологии гештальтом), полученный на основе ощущений и других источников информации.

Для различения денотатов человек использует имена, связанные с представлениями о сущностях, и формирует соответствующие понятия, используя процедуры выявления сходства и различия представлений. Между ментальными объектами (представление, имя, понятие), образующими треугольник Фреге, существуют связи, соответствующие определенным механизмам мышления. Связь 1 позволяет по имени сущности активизировать в памяти все сведения о ее свойствах (понятие). Эта же связь в обратном направлении позволяет по описанию сущности определить ее имя («Не лает, не кусает, а в дом не пускает. Что это?»). Связь 2 позволяет по представлению о сущности найти информацию о ее свойствах или сформировать представление о сущности по совокупности свойств. Связь 3 соединяет представление о денотате с его именем, т.е. имя может активизировать представление и наоборот.

Информационная структура, соответствующая треугольнику Фреге, в котором вершины отождествляются с именем, понятием и представлением, называется *знаком* или *семой*.

Между знаками существуют различного рода отношения. Особое значение в приобретении знаний имеют отношения наследования, к которым относятся следующие типы: «элемент — класс», «часть — целое», «вид — род». Важным свойством отношений «элемент — класс» ($x \in X$) является то, что между элементами класса может не быть никакого сходства, кроме их принадлежности к одному классу. Определение класса в подобных случаях возможно путем перечисления имен входящих в него элементов. В отношениях «часть — целое» ($y \subset X$) наследуется часть свойств сущности, которой приписана роль «целого» (X). При этом части одного целого могут быть совершенно не похожи друг на друга. Отношения «вид — род» базируются на наследовании всех свойств рода X видом z (например, любой вид млекопитающих наследует признак выкармливания детенышей молоком), поэтому между видами одного рода обязательно существует сходство.

Отношения наследования образуют иерархические структуры в системах знаков. На рис. 4.5 показан пример подобной иерархии, где первый уровень образован отношением типа «элемент — класс», второй — отношением «вид — род», а третий — отношени-

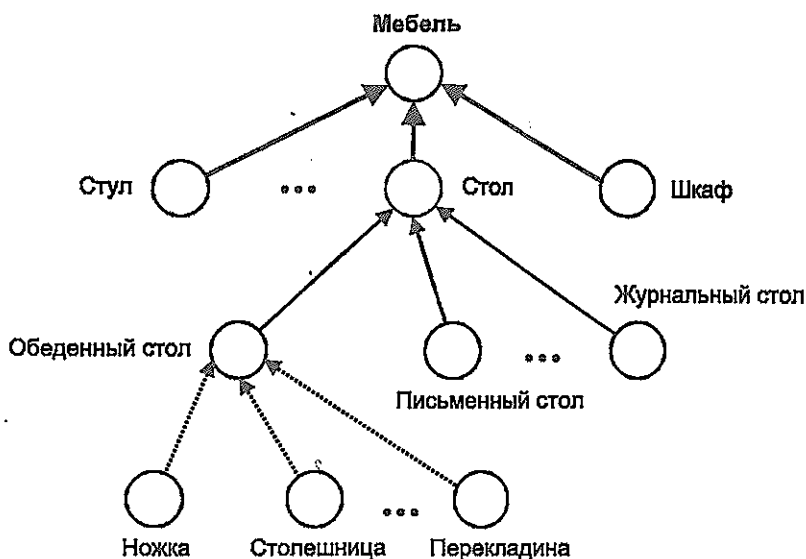


Рис. 4.5. Пример иерархии знаков с отношениями наследования разных типов

ем «часть — целое». В отношении «элемент — класс» иерархическая связь устанавливается между именами знаков, а в отношениях «часть — целое» и «вид — род» — между понятиями или представлениями.

Между знаками, находящимися на одном иерархическом уровне, тоже могут существовать связи. Типичным примером таких связей являются отношения типа «причина — следствие». Наличие отношений внутри иерархических уровней приводит к образованию сетей, в которых вершины соответствуют знакам, а дуги — отношениям между знаками. По сути дела, это и есть семантические сети, о которых говорилось в главе 2. Кроме этой аналогии следует отметить сходство понятий знака и фрейма, породившее концепцию *знака-фрейма*, который представляет собой треугольник с вершинами:

- *Имя* (имя фрейма=имя знака);
- *Протофрейм* (понятие, представленное набором слотов «пустого» фрейма-прототипа);
- *Экзофрейм* (представление, реализованное конкретным фреймом-экземпляром).

Для исследования операций над знаками-фреймами необходимо ввести метауровень, на котором в качестве денотатов выступают знаки-фреймы, а ментальными объектами являются метазнаки, которые характеризуются синтаксисом (именем или другим способом кодирования), семантикой (понятием, конкретным смыслом для субъекта) и прагматикой (процедурами, связанными со знаками). Взаимосвязь этих аспектов также можно представить треугольником Фреге, в котором существуют связи между синтаксисом и семантикой, позволяющие получать по имени всю информацию о сущности, и, наоборот, по описанию метазнака определить его синтаксис. Связи между семантикой и прагматикой позволяют формировать определенные действия на основе анализа ситуации, которая характеризуется данным знаком. Связь между этими аспектами в противоположном направлении можно интерпретировать как объяснение причин произошедших событий (процедур). Связь синтаксиса с прагматикой предполагает переход к действиям при упоминании имени без анализа семантики (нажать на тормоз при красном сигнале светофора), а также позволяет восстановить синтаксис (имя) ситуации по совершенным действиям.

Синтаксис, семантика и прагматика знака (метазнака) не привязаны жестко и однозначно к той сущности, которую они характеризуют. Рассмотрим пример [36], подтверждающий договорный характер использования знаков. Допустим, в поездке за грибами участники договорились зажечь костер в том случае, если будет обнаружено место с большим количеством грибов. Результат этой договоренности можно рассматривать как создание знака, обозначением (синтаксисом) которого является дым от костра, семантика соответствует факту обнаружения грибов, а прагматика (действия) очевидна: если кто-то не нашел грибов, нужно идти в сторону костра. Однако участники поездки могут принять другие соглашения. Например, при обнаружении грибов не разжигать костер, а громко крикнуть. В этом случае изменится только синтаксис. Если они договорятся о том, что дым костра является сигналом наступления обеда, то изменится семантика, а прагматика останется прежней — идти в сторону костра. Наконец, если принято соглашение о том, что дым от костра означает сигнал возвращения домой, то изменится не только семантика, но и прагматика, так как в этом случае нужно идти не в сторону костра, а к автобусной остановке.

Структура семиотической модели знаний некоторой предметной области, или *поля знаний*, на котором создается БЗ интеллектуальной системы, показана на рис. 4.6 [11].

Синтаксис семиотической модели поля знаний можно описать следующими признаками:

$$P = \{I, O, M\},$$

где I — структура исходных данных, подлежащих обработке и интерпретации в ИИС;

O — структура выходных данных;

M — модель предметной области, описывающая преобразование I в O , которая может быть представлена двумя составляющими $M = \{S_k, S_f\}$.

Компонента S_k отражает понятийную структуру предметной области в виде статической концептуальной структуры, а S_f — ди-

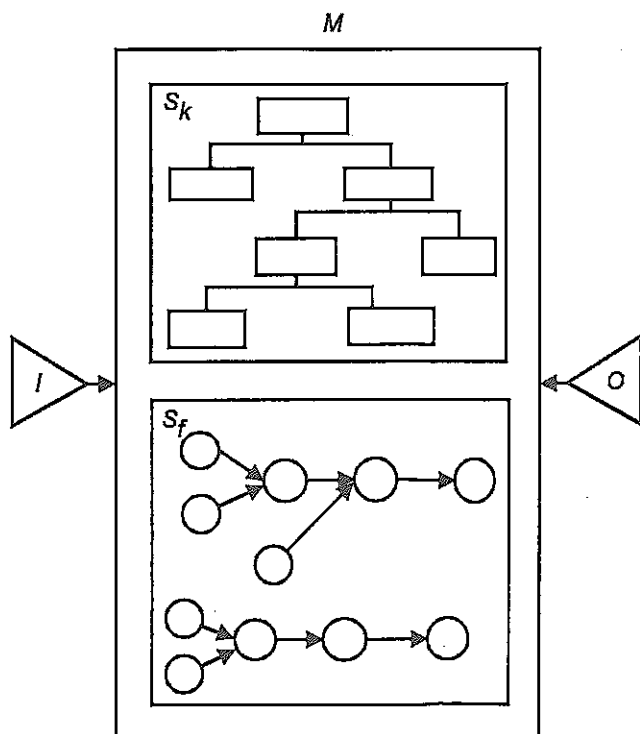


Рис 4.6. Структура поля знаний

намическую функциональную структуру, моделирующую возможные схемы рассуждений эксперта на основе функциональных связей или отношений между понятиями, образующими S_k .

Семантика модели определяется конкретным содержанием знаний проблемной области. Смысл используемых в модели понятий и отношений определяется и постепенно уточняется в процессе получения знаний, в котором можно выделить следующие этапы.

1. Построение первичной модели, которая отражает представление эксперта о предметной области. Эта модель является семантической репрезентацией реальности и личного опыта эксперта.

2. Эксперт вербализует свой опыт, объясняя способы рассуждений в процессе передачи знаний аналитику. Семантику знаний предметной области теперь можно представить текстом.

3. Инженер по знаниям интерпретирует вербализованный опыт эксперта и формирует собственное представление о знаниях данной предметной области.

4. Формируется модель поля знаний.

Под прагматической составляющей семиотической модели специалисты в области искусственного интеллекта понимают практические аспекты разработки поля знаний, связанные с созданием и применением технологий проведения структурного анализа, согласованием отдельных фрагментов знаний, разрешением противоречий и т.п. Прагматика в данном случае определяет выбор технологий, которые использует инженер по знаниям для преобразования хаотического опыта эксперта в стройную модель поля знаний проблемной области.

4.1.4. МЕТОДЫ ИЗВЛЕЧЕНИЯ ЗНАНИЙ

Многообразие задач, ситуаций и источников знаний обусловило появление большого количества методов извлечения, приобретения и формирования знаний. Одна из возможных классификаций методов извлечения знаний приведена на рис. 4.7, на первом уровне которой выделены два больших класса. Первый класс образуют коммуникативные методы, которые ориентиро-

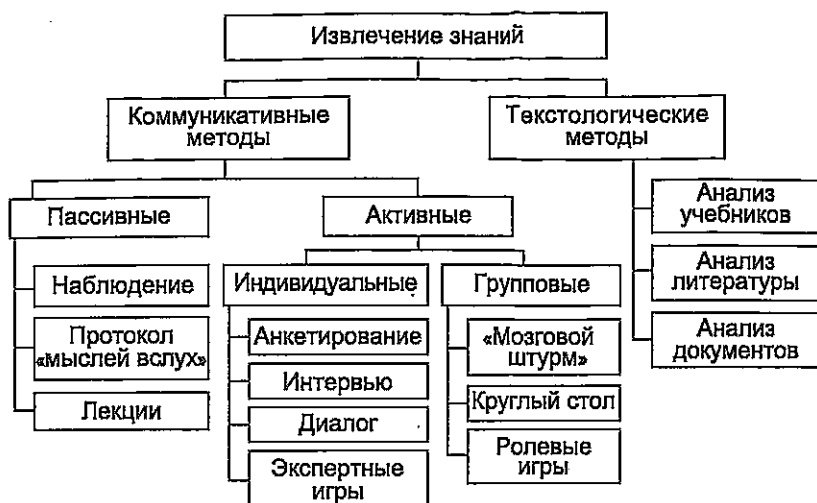


Рис. 4.7. Классификация методов извлечения знаний

ваны на непосредственный контакт инженера по знаниям с экспертом (источником знаний), второй класс — текстологические методы, основанные на приобретении знаний из документов и специальной литературы.

Коммуникативные методы. Они разделяются на пассивные и активные. В пассивных методах ведущую роль играет эксперт, в активных — инженер по знаниям. При решении конкретных задач, как правило, используются как пассивные, так и активные методы. Активные методы делятся на индивидуальные и групповые. В групповых методах знания получают от множества экспертов, в индивидуальных — от единственного эксперта. Индивидуальные методы получили более широкое применение на практике по сравнению с групповыми.

Пассивные коммуникативные методы включают наблюдение, анализ протоколов «мыслей вслух», процедуры извлечения знаний из лекций.

Метод наблюдения является одним из наиболее применяемых на начальных этапах разработки экспертных систем. Его суть заключается в фиксировании всех действий эксперта, его реплик и объяснений. При этом аналитик не вмешивается в работу эксперта, а только наблюдает за процессом решения реальных задач

либо за решением проблем, имитирующих реальные задачи. Наблюдения за процессом решения реальных задач позволяют инженеру по знаниям глубже понять предметную область. Однако эксперт в этом случае испытывает большое психологическое напряжение, понимая, что осуществляет не только свою профессиональную деятельность, но и демонстрирует ее инженеру по знаниям. Наблюдение за имитацией процесса снимает это напряжение, но приводит к снижению полноты и качества извлекаемых данных. Наблюдения за имитацией незаменимы в тех случаях, когда наблюдения за реальным процессом невозможны из-за специфики изучаемой предметной области.

Метод анализа протоколов «мыслей вслух» отличается от метода наблюдения тем, что эксперт не только комментирует свои действия, но и объясняет цепочку своих рассуждений, приводящих к решению. Основной проблемой, возникающей при использовании этого метода, является принципиальная сложность для любого человека словесного описания собственных мыслей и действий. Повысить полноту и качество извлекаемых знаний можно за счет многократного уточняющего протоколирования рассуждений эксперта.

Метод извлечения знаний из лекций предполагает, что эксперт передает свой опыт инженеру по знаниям в форме лекций. При этом инженер по знаниям может заранее сформулировать темы лекций. Если этого не удастся сделать, то когнитолог конспектирует лекции и задает вопросы. Качество информации, предоставленной экспертом в ходе лекции, определяется четкостью сформулированной темы, а также способностями лектора в структурировании и изложении своих знаний и рассуждений.

Сравнительные характеристики пассивных методов извлечения знаний приведены в табл. 4.2.

Одна из возможных классификаций людей по психологическим характеристикам [32] делит всех на три типа:

- мыслитель (познавательный тип);
- собеседник (эмоционально-коммуникативный тип);
- практик (практический тип).

Мыслители ориентированы на интеллектуальную работу, учебу, теоретические обобщения и обладают свойствами полнезависимости и рефлексивности. *Собеседники* — общительные, открытые люди, готовые к сотрудничеству. *Практики* предпочитают действия разговорам, хорошо реализуют замыслы других.

Сравнительные характеристики пассивных методов
извлечения знаний

Показатели	Наблюдения	«Мысли вслух»	Лекции
Достоинства	Отсутствие влияния аналитика и его субъективной позиции Максимальное приближение аналитика к предметной области	Свобода самовыражения для эксперта Вербализация рассуждений Отсутствие влияния аналитика и его субъективной позиции	Свобода самовыражения для эксперта Структурированное изложение Высокая концентрация Отсутствие влияния аналитика и его субъективной позиции
Недостатки	Отсутствие обратной связи Фрагментарность полученных комментариев	Отсутствие обратной связи Возможность ухода «в сторону» в рассуждениях эксперта	«Зашумленность» деталями Слабая обратная связь Недостаток хороших лекторов среди экспертов-практиков
Требования к эксперту (типы и основные качества)	Собеседник или мыслитель (способность к вербализации мыслей, аналитический склад ума, открытость, рефлексивность)		Мыслитель (лекторские способности)
Требования к аналитику (типы и основные качества)	Мыслитель (наблюдательность, полнезависимость)	Мыслитель или собеседник (полнезависимость, способность к общению)	Мыслитель (полнезависимость, способность к общению)
Характеристика предметной области	Слабо- и среднеструктурированные, слабо- и среднедокументированные		Слабоструктурированные, слабодокументированные

Предметные области отличаются уровнем документированности и структурированности. Для характеристики предметной области по уровню документированности выделяют три класса: хорошо документированные, среднедокументированные и сла-

бодокументированные области (рис. 4.8). По степени структурированности знаний предметные области могут быть:

- хорошо структурированными (с четкой аксиоматизацией, широким применением математического аппарата, устоявшейся терминологией);
- среднеструктурированными (с определившейся терминологией, развивающейся теорией, явными взаимосвязями между явлениями);
- слабоструктурированными (с размытыми определениями, богатым эмпирическим материалом, скрытыми взаимосвязями).

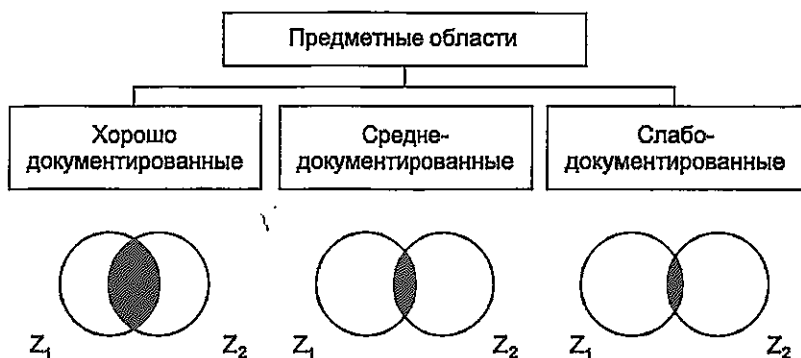


Рис. 4.8. Классификация предметных областей по уровню документированности:

Z_1 — экспертное «личное» знание;
 Z_2 — материализованное в книгах «общее» знание;
 $Z_{по} = Z_1 \cup Z_2$ — знания предметной области

Активные индивидуальные методы включают методы анкетирования, интервьюирования, свободного диалога и игры с экспертом. Сравнительный анализ методов данной группы приведен в табл. 4.3.

Преимуществом *методов анкетирования* является то, что анкета или вопросник составляются инженером по знаниям заранее и используются для опроса экспертов. Составление анкеты следует проводить с учетом рекомендаций, выработанных в социологии и психологии. Основными требованиями к анкетам являются следующие [11]:

1. Анкета не должна быть монотонной и однообразной, чтобы не вызывать скуку или усталость. Для этого необходимо разнообразить

Сравнение активных индивидуальных методов

Показатели	Анкетирование	Интервьюи- рование	Свободный диалог
Достоинства	Возможность стандартизированного опроса нескольких экспертов Не требует особенного напряжения от аналитика во время процедуры анкетирования	Наличие обратной связи (возможность уточнения контекста и разрешения противоречий)	Гибкость Обратная связь Возможность изменения сценария и формы сеанса
Недостатки	Требуется умения и опыта составления анкет Отсутствие контекста и обратной связи. Вопросы анкеты могут быть неправильно поняты экспертом	Требуется значительного времени на подготовку вопросов интервью	Требуется от аналитика высочайшего напряжения Отсутствие формальных методик проведения Трудность протоколирования результатов
Требования к эксперту	Практик и мыслитель	Собеседник и мыслитель	
Требования к аналитику	Мыслитель (педантизм в составлении анкет, внимательность)	Собеседник (журналистские навыки, умение слушать)	Собеседник (наблюдательность, умение слушать, обаяние)
Характеристика предметной области	Слабоструктурированные, слабо- и среднедокументированные		

разить тематику и форму задания вопросов, включить вопросы-шутки и применить стиль игры.

2. Анкета должна быть приспособлена к языку эксперта.

3. Следует учитывать, что вопросы влияют друг на друга, поэтому важно расположить их в правильной последовательности.

4. В анкете должно содержаться оптимальное число избыточных вопросов, часть которых предназначена для контроля правильности ответов, а другая часть — для снятия напряжения.

Метод интервьюирования отличается от метода анкетирования тем, что позволяет аналитику опускать ряд вопросов в зависимости от ситуации, вставлять новые вопросы в анкету, изменять темы и разнообразить ситуацию общения. Важную роль в методе интервьюирования играют вопросы, классификация которых показана на рис. 4.9.

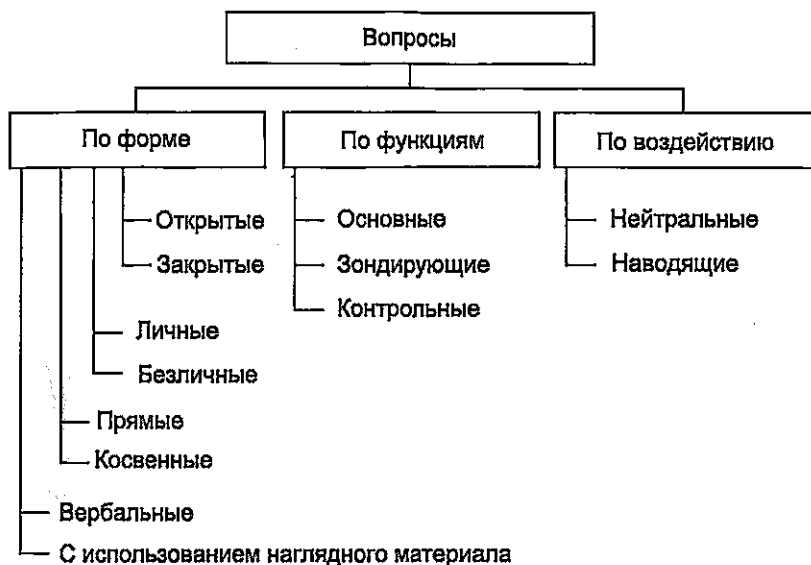


Рис. 4.9. Классификация вопросов при интервьюировании

Открытый вопрос называет тему или предмет, оставляя эксперту полную свободу в отношении формы и содержания ответа. *Закрытый* вопрос предлагает эксперту выбрать ответ из предложенного набора.

Личный вопрос непосредственно касается личного опыта эксперта. *Безличный* вопрос направлен на выявление наиболее распространенных закономерностей предметной области.

Прямой вопрос непосредственно указывает на интересующий предмет или тему (используется при «зажатости» эксперта). *Косвенный* вопрос исподволь затрагивает рассматриваемую проблему.

Вербальный вопрос — традиционный устный вопрос. *Вопрос с использованием наглядного материала* позволяет разнообразить

интервью и снять усталость эксперта (используются фотографии, рисунки, карточки).

Основной вопрос направлен на выявление знаний. *Зондирующий* вопрос направляет рассуждения эксперта в нужную сторону. *Контрольный* вопрос проверяет достоверность и объективность информации, полученной в интервью ранее.

Нейтральный вопрос подчеркивает беспристрастность инженера по знаниям к предмету исследования. *Наводящий* вопрос ориентирует эксперта принять во внимание позицию инженера по знаниям.

Дополнительно в интервью рекомендуется включать следующие вопросы: *контактные* (снимающие психологический барьер между аналитиком и экспертом), *буферные* (разграничивающие отдельные темы интервью), *оживляющие память* экспертов (реконструирующие отдельные случаи из практики), *«провоцирующие»* (способствующие получению неподготовленных ответов).

При использовании метода интервьюирования следует иметь в виду, что его эффективность во многом определяется языком вопросов (понятностью, лаконичностью, терминологией); порядком вопросов (логическая последовательность); уместностью вопросов (этичностью и вежливостью).

Прежде чем готовить вопросы, аналитик должен овладеть ключевым набором знаний исследуемой предметной области, поскольку любой вопрос имеет смысл только в контексте.

Метод свободного диалога позволяет извлекать знания в форме беседы с экспертом, поэтому здесь не предусматривается использование жесткого вопросника или плана. В то же время подготовка к свободному диалогу должна проводиться по специальной методике, в которую входит общая, специальная, конкретная и психологическая подготовка. Общая подготовка направлена на повышение научной эрудиции, овладение общей культурой, знакомство с системной методологией. Специальная подготовка сводится к овладению теорией и навыками интервьюирования. Конкретная подготовка предполагает изучение предметной области, подготовку ситуации общения, знакомство с экспертом, тестирование эксперта. Психологическая подготовка включает знакомство с теорией общения и с когнитивной психологией.

Игры с экспертом существенно отличаются от приведенных выше индивидуальных активных методов извлечения знаний и рассматриваются в классе групповых активных методов, где особое место принадлежит ролевым и экспертным методам.

Активные групповые методы включают «мозговой штурм», дискуссии за круглым столом и ролевые игры. Групповые методы позволяют творчески интегрировать знания множества экспертов. Сравнительный анализ методов приведен в табл. 4.4.

Метод «мозгового штурма» — один из наиболее известных и широко применяемых методов генерирования новых идей путем творческого сотрудничества группы специалистов. Являясь в некотором смысле единым мозгом, группа пытается штурмом пре-

Таблица 4.4

Сравнение активных групповых методов извлечения знаний

Показатели	«Мозговой штурм»	Дискуссия за круглым столом
Достоинства	Позволяет выявлять глубинные пласты зна- ний (на уровне бессоз- нательного) Активизирует экспер- тов Позволяет получать новые знания	Позволяет получать более объективные фрагменты знаний Оживляет процедуру извлечения знаний Позволяет участникам обмениваться знания- ми
Недостатки	Возможен только для новых интересных ис- следовательских про- блем Не всегда эффективен (довольно низкий про- цент продуктивных идей)	Требует больших орга- низационных затрат Отличается сложнос- тью проведения
Требования к эксперту	Мыслитель (способ- ность к творчеству)	Собеседник или мыс- литель (искусство по- лемики)
Требования к анали- тику	Собеседник или мыс- литель (быстрая реак- ция и чувство юмора)	Собеседник (диплома- тические способности)
Характеристика пред- метной области	Слабоструктурирован- ные и слабодокумен- тированные с наличи- ем перспективных «бе- лых пятен»	Слабоструктурирован- ные и слабодокумен- тированные с наличи- ем спорных проблем

одолеть трудности, мешающие разрешить рассматриваемую проблему. В процессе такого штурма участники выдвигают и развивают собственные идеи, стимулируя появление новых и комбинируя их. Для обеспечения максимального эффекта «мозговой шторм» должен подчиняться определенным правилам и основываться на строгом разделении во времени процесса выдвижения идей и процесса их обсуждения и оценки. На первой стадии штурма запрещается осуждать выдвинутые идеи и предложения (считается, что критические замечания уведут к частностям, прерывают творческий процесс, мешают выдвижению идей). Роль аналитика состоит в том, чтобы активизировать творческое мышление участников заседания и обеспечить выдвижение возможно большего числа идей.

После выдвижения идей выполняются тщательное их обсуждение, оценка и отбор лучших. На стадии обсуждения участники «мозгового штурма» должны сконцентрироваться на положительных сторонах идей, найти в них рациональные зерна и предложить направления их развития. Выдвигаемые в процессе обсуждения дополнительные идеи могут базироваться на идеях других участников или, наоборот, служить для них фундаментом, катализатором. Значительный эффект дает комбинирование идей при одновременном выявлении преимуществ и недостатков синтезируемых при этом вариантов.

Метод «мозгового штурма» эффективен при решении не слишком сложных задач общего организационного характера, когда проблема хорошо знакома всем участникам заседания и по рассматриваемому вопросу имеется достаточная информация. Существует ряд модификаций этого метода [2, 4, 13].

Индивидуальный «мозговой шторм» проводится по тем же правилам, что и коллективный, но выполняется одним экспертом, который одновременно генерирует идеи, дает им объективную оценку и критикует их.

Массовый «мозговой шторм» проводится в массовой аудитории (до нескольких десятков человек). Отбор идей проводится на промежуточных этапах. Эксперты группируются по 6 — 8 человек, при этом важно, чтобы непосредственное отношение к задаче имел лишь руководитель группы, а остальные были лишь знакомы с нею (иначе амбиции могут сыграть негативную роль). Шторм проводится в два этапа. На первом этапе оперативные группы осуществляют прямой коллективный «мозговой шторм».

При этом желательно, чтобы каждая группа работала над задачей, наиболее близкой по тематике к профилю вошедших в нее специалистов. На втором этапе руководители каждой группы в течение нескольких минут оценивают выдвинутые идеи, отбирают из них наиболее интересные и сообщают их на «пленарном заседании».

Двойной «мозговой штурм» органически соединяет в себе процессы генерирования идей и их доброжелательной позитивной критики.

Обратный «мозговой штурм» отличается от прямого тем, что в нем больше внимания уделяется критике высказанных идей.

Метод дискуссии за круглым столом предполагает равноправное обсуждение экспертами поставленной проблемы. Отличительной особенностью метода дискуссии является коллективное рассмотрение предметной области с разных точек зрения и исследование спорных гипотез.

Экспертные игры предназначены для извлечения знаний и базируются на деловых, диагностических и компьютерных играх (табл. 4.5).

По числу участников игры подразделяют на индивидуальные (игры с экспертом) и групповые (ролевые игры в группе). По применению специального оборудования — игры с тренажерами и игры без реквизита. Особый класс представляют собой компьютерные игры.

В *играх с экспертом* инженер по знаниям берет на себя чью-нибудь роль в моделируемой ситуации. *Ролевые игры* в группе предусматривают участие в игре нескольких специалистов. Участники игры наделяются определенными ролями, а собственно игра проводится по составленному когнитологом сценарию. В целях повышения эффективности ролевых игр в них необходимо вводить элементы состязательности.

Игры с применением тренажеров позволяют зафиксировать трудно уловимые знания, которые возникают в реальных ситуациях и могут быть потеряны при выходе из них.

Компьютерные экспертные игры в настоящее время используются в основном в целях обучения. Они полезны для «разминки» экспертов перед сеансом извлечения знаний.

Текстологические методы включают методы извлечения знаний, основанные на изучении текстов учебников, специальной литературы и документов. Текстология — это наука, целью кото-

Сравнительные характеристики экспертных игр

Показатели	Экспертные игры		
	индивидуальные	групповые	компьютерные
Достоинства	Дают возможность сравнительно быстро получить качественную картину принятия решения Позволяют выяснить, какую информацию и как использует эксперт	Реалистично воссоздают атмосферу конкретной задачи Раскрепощают экспертов Групповые занятия более объективны Выявляют логику и аргументацию экспертов	Вызывают интерес у эксперта Привлекают дизайном и динамикой
Недостатки	Отсутствие методик и стандартного набора игр Высокие профессиональные требования к аналитику	Требуют от аналитика знания основ игротехники Сложность создания игр для конкретных предметных областей	Сложность и высокая стоимость создания специализированных игр в конкретной предметной области
Требования к эксперту	Собеседник или практик (раскованность и актерское мастерство)		Практик (без психологического барьера к ЭВМ)
Требования к аналитику	Собеседник (режиссерские способности, умение создавать сценарии, актерское мастерство)	Собеседник (способность к ведению конференса, режиссерские способности, актерское мастерство)	Мыслитель (контакт с программистом)
Характеристика предметной области	Средне- и слабоструктурированные и слабодокументированные		

рой является практическое прочтение текстов, изучение и интерпретация литературных источников, а также рассмотрение семиотических, психолингвистических и других аспектов извлечения знаний из текстов. Схема извлечения знаний из специальных текстов приведена на рис. 4.10, где M_1 — смысл, заложенный автором и основанный на его собственной модели мира; M_2 — смысл, который постигает инженер по знаниям; I — интерпретация текста, изложенного словесно; T — словесное изложение знаний; V — результат вербализации.

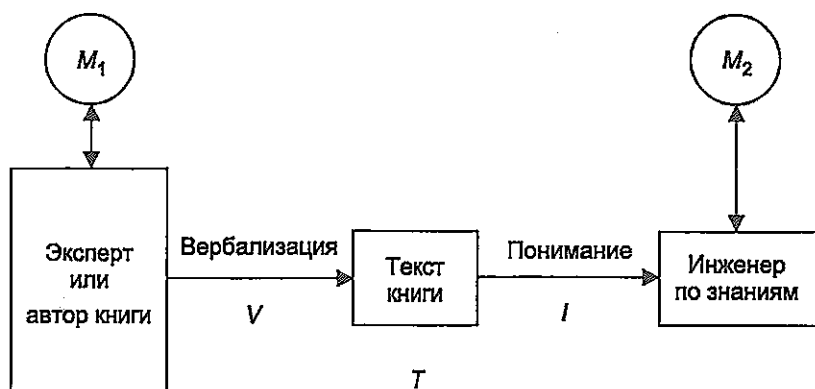


Рис. 4.10. Схема извлечения знаний из текста

Научный текст T , который формируется автором на основе модели M_1 , включает в себя пять компонентов, погруженных в языковую среду L : $T = (\alpha, \beta, \gamma, \delta, \theta)_L$, где α — первичный материал наблюдений; β — система научных понятий; γ — субъективный взгляд автора; δ — дополнительная информация, не касающаяся научного знания (общие места); θ — заимствования. На процесс понимания (интерпретации) I и на модель M_2 влияют следующие компоненты: $M_2 = [(\alpha, \beta, \gamma, \theta)', \varpi, \epsilon, \phi]$, где $(\alpha, \beta, \gamma, \theta)'$ — экстракт компонентов, почерпнутый из текста T ; ϖ — предварительные знания аналитика о предметной области; ϵ — общенаучная эрудиция инженера по знаниям; ϕ — личный опыт аналитика.

Автор книги, являющийся в данном случае экспертом, на основании собственной модели мира M_1 пишет книгу, представляя свои знания в виде текста (см. рис. 4.10). Затем инженер по знаниям на основании своей модели знаний M_2 и информации, по-

черпнутой из текста T , интерпретирует знания эксперта. Схема процесса извлечения знаний из текста показывает, что смысл M_1 , заложенный автором книги, отличается от смысла M_2 , который постигает читатель (инженер по знаниям). Эти различия определяются тем, насколько развиты факторы ω , ϵ и ϕ у конкретного аналитика (читателя).

Немецкий философ и языковед В. фон Гумбольдт так охарактеризовал сложную и трудно формализуемую процедуру взаимного понимания людей: «Люди понимают друг друга не потому, что передают собеседнику знаки предметов, и даже не потому, что взаимно настраивают друг друга на точное и полное воспроизведение идентичного понятия, а потому, что взаимно затрагивают друг в друге одно и то же звено цепи чувственных представлений и зачатков внутренних понятий, прикасаются к одним и тем же клавишам инструмента своего духа, благодаря чему у каждого вспыхивают в сознании соответствующие, но не тождественные смыслы».

Особую сложность представляет извлечение знаний из специальной литературы и методик, поскольку в них очень высока степень концентрации специальных знаний.

Простейший алгоритм извлечения знаний из текстов [11] включает следующие шаги.

1. Составить «базовый» список литературы для ознакомления с предметной областью.
2. Выбрать текст для извлечения знаний.
3. Беглое знакомство с текстом. Провести консультации со специалистами для определения значений незнакомых слов.
4. Сформировать первую гипотезу о макроструктуре текста.
5. Внимательно прочитать текст и выписать ключевые слова и выражения, определив тем самым «смысловые вехи».
6. Определить связи между ключевыми словами, разработать макроструктуры текста в форме графа или реферата.
7. Сформировать новое представление знаний на основании макроструктуры текста.

4.1.5. ВЫЯВЛЕНИЕ «СКРЫТЫХ» СТРУКТУР ЗНАНИЙ

Знания, полученные от эксперта в прямом диалоге, имеют поверхностный характер и не отражают сложных механизмов мышления специалиста. Альберт Эйнштейн говорил: «Слова, на-

писанные или произнесенные, не играют, видимо, ни малейшей роли в механизме моего мышления» [1]. Известный психолог Л.С. Выготский научно обосновал, что мышление и речь имеют разные корни [9]. В настоящее время известно, что язык выступает в качестве материального носителя результатов мышления на предварительной и завершающей его стадиях [39]. Ранее многие ученые придерживались иной точки зрения, например древнегреческий мыслитель Платон полагал, что если человек не может представить свое знание в виде правил, то это уже не знание, а вера. Зигмунд Фрейд, отводя главную роль в человеческом поведении *бессознательному*, считал, что оно ни при каких обстоятельствах не может стать *сознательным*, тем самым отвергая возможность извлечения скрытых знаний.

Скрытые (имплицитные) знания служат основой интуитивного мышления. Интуиция позволяет человеку быстро принимать правильные решения в сложных ситуациях при недостаточной информации. При этом полагают, что человек может неявно использовать ряд посылок в своих рассуждениях, а также обходиться без применения строгих правил логического вывода. Роль интуиции в принятии решений трудно переоценить. Поэтому проблемы извлечения, вербализации и использования имплицитных знаний в ИИС всегда будут актуальными. В XX в. доминирующим направлением в теории познания было материалистическое, в рамках которого роль человека сводилась к сбору информации об окружающей его объективной реальности и к построению ее дискретных моделей. Эта парадигма положена в основу архитектуры современных компьютеров, главной чертой которых является дискретность обрабатываемой информации. Недостатки дискретных моделей — это низкая скорость обработки информации (по сравнению со способностями человека) и невозможность представления имплицитных знаний, которые существуют в невербальной форме.

Выдающийся когнитолог М. Хайдеггер предложил новый подход в теории познания, выдвинув гипотезу о том, что человек не может иметь объективных знаний об окружающей действительности, ибо она структурируется человеком в зависимости от его целей, конкретных обстоятельств, ценностей и т.п. [14, 51]. По мнению Хайдеггера, дискретные модели окружающего мира, претендующие на объективность, имеют небольшое значение в жизни человека. Главную роль в его деятельности играют импли-

цитные знания, навыки и живой опыт, которые существуют в невербальной, а следовательно, не в дискретной форме. Идеи Хайдеггера послужили основой коннекционистской (*connect* — связывать, англ.) теории познания, положенной в основу моделей нейронных сетей. Коннекционистская модель не является дискретной, она не строится в явном виде, а появляется в результате обучения на примерах. Нейронные сети успешно применяются для имитации *бессознательных* знаний.

Как и следовало ожидать, решение одних проблем породило другие. Избавление от дискретности сопровождалось существенным снижением возможности структурного представления знаний. Поэтому перспективы развития теории познания и ее прикладной ветви — методологии приобретения знаний интеллектуальными системами связаны с интеграцией обоих подходов. Хорошей аналогией являются дифференциальные уравнения, являющиеся непрерывными моделями реальных процессов, для решения которых используются дискретные методы. Ниже будут кратко рассмотрены методы выявления скрытых знаний, разработанные в психосемантике, а знакомство с моделями нейронных сетей состоится в главе 5.

Психосемантика позволяет исследовать структуры сознания через моделирование индивидуальной системы знаний человека и выявлять элементы знаний, которые могут им не осознаваться (латентные, скрытые, имплицитные). Эта наука объединяет методы когнитивной психологии, психолингвистики, психологии восприятия и исследования индивидуального сознания.

Основным методом психосемантики является построение субъективных семантических пространств, для чего, как правило, применяются статистические процедуры, а именно: многомерное шкалирование [26, 41, 42], репертуарные решетки [45, 66, 68], факторный [18, 27] и кластерный анализ [15, 16, 27]. Эти методы позволяют сгруппировать отдельные описательные признаки в более емкие категории-факторы. Таким образом происходит переход к описанию предметной области на более высоком уровне абстракции с помощью метаязыка выделенных категорий. Исследование свойств семантических пространств позволяет выявлять закономерности в конкретной области знаний.

Расположение первичных понятий в семантическом пространстве существенно зависит от опыта и профессиональной компетентности испытуемых, что можно использовать для контроля знаний путем сопоставления семантических пространств хоро-

ших специалистов и новичков. Многочисленные исследования, проведенные в данном направлении, показали, что размерность семантического пространства уменьшается с повышением уровня квалификации специалистов. Этот факт согласуется с известным положением когнитивной психологии о том, что процесс познания сопровождается обобщением.

Алгоритм построения семантического пространства включает три главных этапа.

1. Выбор и применение метода оценки семантического сходства признаков, предъявляемых испытуемому.

2. Построение структуры семантического пространства на основании математического анализа полученной матрицы сходства. При этом происходит уменьшение числа исследуемых понятий за счет обобщения.

3. Идентификация и интерпретация выделенных факторных структур, кластеров, групп объектов, осей и т.д.

Рассмотрим пример построения семантического пространства, отражающего сходство распространенных английских предлогов [42]. Пространство строилось с применением девятибалльной шкалы для оценки степени затруднения, возникающего у испытуемого при выборе им предлога из рассматриваемой пары. Результатом обработки полученных данных методом многомерного шкалирования является семантическое пространство, отражающее сложность употребления английских предлогов для носителей русского языка. На рис. 4.11 показано пространство, построенное по данным, полученным от человека, хорошо владеющего английским языком. Семантическое пространство построено в двух координатных осях: ось абсцисс соответствует предлогам группы «направление — движение», а ось ординат — предлогам «цели — средства». Пространство хорошо структурировано и наглядно показывает сходство и различие предлогов (чем больше похожи предлоги, тем ближе соответствующие им точки на центральной окружности). Следует заметить, что пространство, построенное по данным «новичка», было совершенно не структурированным, т.е. в нем было практически невозможно выделить группы близких предлогов.

Методы многомерного шкалирования. Они основаны на статистических методах обработки экспертных оценок сходства между анализируемыми объектами, которые выбираются из определенной шкалы. Результаты обработки представляются в виде точек некоторого координатного пространства. Возможность визуализации

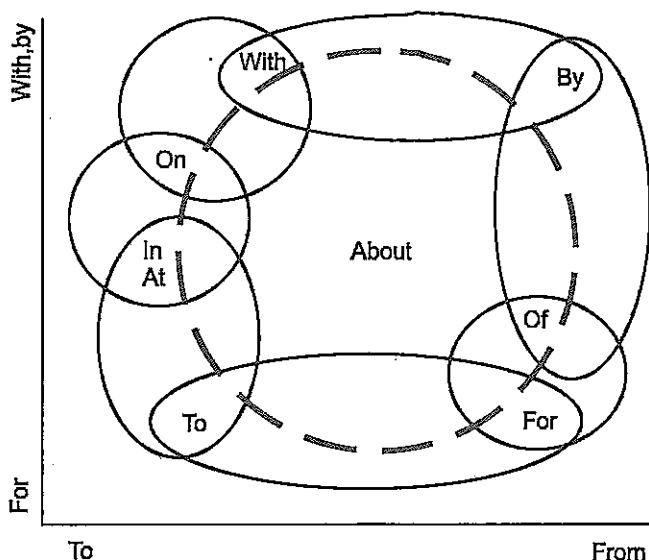


Рис. 4.11. Семантическое пространство сложности употребления английских предлогов для носителя русского языка

зации результатов является безусловным преимуществом метода, однако она быстро утрачивается с увеличением размерности пространства. Шкалированием называют поиск подпространства, для которого величина $|D^m - D^{m*}|$ имеет минимальное значение. Здесь D^m , D^{m*} — матрицы расстояний между объектами (признаками) в исходном пространстве D^m и в искомом подпространстве D^{m*} , m и m^* — размерности соответствующих пространств. Если $m^* = 2$, шкалирование превращается в проецирование на плоскость. Значения расстояний в матрицах D могут выбираться экспертом из предложенной шкалы либо вычисляться по совокупности признаков, описывающих объект. Во втором случае расстояния можно вычислить разными способами [26]. Одной из самых популярных метрик является евклидово расстояние

$$d_{i,j} = \left(\sum_{k=1}^K (x_i^k - x_j^k)^2 \right)^{1/2},$$

где x_i^k , x_j^k — значения k -го признака у i -го и j -го объектов соответственно;

K — общее число признаков.

Расстояния-метрики должны удовлетворять следующим условиям:

$$d(x, y) \geq 0; d(x, x) = 0; d(x, y) = d(y, x); d(x, y) + d(y, z) \geq d(x, z).$$

Метрическим шкалированием называют образование новых классов с использованием метрических расстояний. Этот тип обработки данных ориентирован на максимальное сближение числовых значений матриц D^m и D^{m*} . Существует также *неметрическое шкалирование*, которое не предъявляет жестких требований к сближению пространств и во многих случаях более оправдано в связи с условностью понятия «расстояние».

Важно отметить, что в шкалировании отыскиваются не новые признаки, а новые пространства, поэтому его результаты следует интерпретировать как восстановленную (на плоскости или в объеме) структуру расположения точек. Главными недостатками метода многомерного шкалирования являются:

- субъективные оценки сходства между объектами и признаками обрабатываются как расстояния в пространстве, а результаты анализируются на основе геометрической интерпретации. Это ограничивает размерность выявляемых пространств и требует серьезного упрощения реальных знаний эксперта, следствием которого могут стать неадекватные БЗ;

- выделенные подпространства не имеют иерархической организации, что затрудняет их интерпретацию;

- используется только один вид отношений между понятиями (отношение сходства).

Метафорический подход. Он ориентирован на выявление скрытых составляющих практического опыта эксперта и основан на сравнении объектов предметной области с абстрактными объектами из мира метафор, в результате чего можно выявить новые свойства анализируемых объектов и определить отношение эксперта к ним. Используя метафорические сравнения, эксперт выходит за рамки объективности и действует в соответствии со своими субъективными представлениями.

Метод репертуарных решеток. Предложен Дж. Келли в 1955 г. [66, 72], широко применяется в психологических исследованиях для выявления личностных свойств, которые проявляются через систему личностных конструкторов. Этот метод может применяться и для извлечения знаний.

Репертуарная решетка представляет собой матрицу, которая заполняется экспертом. Столбцам матрицы соответствуют определенные группы объектов (элементов), в качестве которых могут выступать люди, предметы, понятия, отношения, звуки и др. Строки матрицы соответствуют *конструктам*, которые представляют собой биполярные признаки, параметры, шкалы, отношения или способы поведения. Дж. Келли называл конструктом признак или свойство, определяющие сходство двух или нескольких объектов и их отличие от других объектов. Иными словами, *конструкты* — это признаки, которые могут использоваться для обобщения и разделения объектов на классы. Конструкты можно применить не к любым объектам, а только в некотором «диапазоне их пригодности» [45, 60].

Конструкты могут быть заданы аналитиком либо подбираются самим экспертом. В последнем случае выделяются личностные конструкты, отражающие идеи или мысли, которые человек использует для осознания, интерпретации, объяснения или предсказания действительности. Примерами личностных конструктов могут быть «умный — глупый», «мужской — женский», «хороший — плохой». Слово *репертуарная* означает, что анализируемые объекты выбираются по специальным правилам, так, чтобы они были связаны определенным контекстом аналогично репертуару ролей в пьесе. Вторым смыслом этого определения заключается в том, что в технике репертуарных решеток элементы часто задаются в виде обобщенных описаний, ролей, исполнителями которых каждый человек мысленно представляет знакомых ему людей или конкретные предметы.

Для выявления конструктов используются: последовательный метод, а также методы минимального контекста, самоидентификации и ролевой персонификации. В соответствии с методом минимального контекста эксперту предъявляются произвольные «тройки» объектов и предлагается определить свойства, отличающие один объект от двух других. В результате определяются не только значения характеристик, но и сами характеристики. Решетка формируется следующим образом. По одной из ее осей располагаются значимые конкретные объекты, а по другой — разряды (типы, роли), к которым они относятся. Распределив объекты по типам, эксперт заполняет первый ряд матрицы под колонками, обозначив три объекта кружочками. При этом он должен подобрать характеристику, которая обеспечивает сходство двух объектов и отличает их от третьего. Кружочки, соответст-

вующие сходным объектам, перечеркиваются. В столбец с именем «полюс конструкта» записывается наименование признака, обеспечивающего сходство двух объектов, в столбец «противоположный полюс» — имя признака, отличающего третий объект от двух сходных. Затем проводится анализ оставшихся в первом ряду объектов по выделенному положительному конструкту и галочками отмечаются объекты, обладающие этим свойством. Традиционная решетка должна быть квадратной, т.е. в матрице заполняют число строк, равное числу объектов (типов). В общем случае это условие не является обязательным. Пример репертуарной решетки для выявления знаний о качестве рекламной продукции показан на рис. 4.12.

Анализ репертуарных решеток позволяет выявлять значимые для специалиста признаки (конструкты), определять силу и направленность связей между конструктами и строить из них связанную систему.

Самым распространенным и простым методом анализа репертуарной решетки является кластерный анализ. Иерархическая кластеризация осуществляется на основе выбора элементов матриц, имеющих наибольшее число связей. Кроме того, конструкты могут быть представлены как точки многомерного пространства, плоскости которого определяются числом связанных с конструктами элементов. Факторный анализ пространства конструктов позволяет судить об их значимости, а корреляционный анализ — выявлять значимые связи между ними.

Слабым местом в теории Дж. Келли является предположение о том, что человек может точно описать конструкты, которые он использует, чтобы объяснить, чем сравниваемые объекты похожи друг на друга и чем отличаются. Процедура выявления и вербализации конструктов очень утомительна для экспертов, поэтому во многих методиках используются готовые наборы конструктов, релевантные рассматриваемым объектам. Решетки Келли являются удачным инструментом для выявления свойств личности через наборы *субъективных* параметров и предпочтений. О субъективности полученных знаний нужно помнить при их последующем использовании.

Известны программные средства для поддержки процессов извлечения знаний с использованием репертуарных решеток, среди них PLANET [61, 74], AQUINAS [53], KRITON [57], SIMER+MIR [33].

	Пиво "Клинское"	Телемагазин	Мазь "Биоптин"	Новые квартиры	"Снайп"	Сад и огород	"Алипа Бланка"	"Ербалайф"	Кухонная посуда	Полус конструкта	Противоположный полус
A Видеоролик	⊗	✓	✓		○				⊗	Эффективный	Неэффективный
B Телепередача		⊗		⊗		✓	○		✓	Информативный	Неинформативный
C Радиоролик	○		✓	✓		⊗			⊗	Нравственный	Безнравственный
D Статья			○					⊗	⊗	Оригинальный	Посредственный
E Стенд	⊗		✓		✓		⊗		○	Для широкого круга	Для узкого круга
F Радиопередача	✓	○	✓		⊗		⊗			Короткий	Длинный
G Реклама на транспорте		✓		⊗	⊗			○	✓	Ненавязчивый	Навязчивый
H Рекламные агенты	⊗				⊗		○		✓	Красивый	Неэстетичный
I Презентации	⊗	✓	✓	○		✓			⊗	Дорогой	Дешевый

Рис. 4.12. Пример репертуарной решетки

4.2. ПОСТРОЕНИЕ БАЗ ЗНАНИЙ ДЛЯ ЭКСПЕРТНЫХ СИСТЕМ ДИАГНОСТИКИ

Академик О. И. Ларичев выделяет два типа знаний, передаваемых от одного человеческого поколения к другому [22]. Знание первого типа определено как *декларативное*. К нему относятся факты, теории, задачи и другие сведения, которые описаны в книгах и учебниках, относящихся к различным областям науки. Ко второму типу знаний относится человеческое *умение* решать задачи, сочинять музыку, лечить больных, находить неисправности в машинах и аппаратах, изобретать новые устройства и т.п.

Передать умения значительно сложнее, чем получить знания в процессе обучения в школе и университете. Умение передается от учителя к ученику в результате длительного непосредственного обучения и преумножается в дальнейшем только путем самообразования при решении многочисленных задач. В результате обучения ученик становится экспертом, который обладает высоким уровнем профессионализма в определенной предметной области.

На уровень знаний эксперта определяющее влияние оказывают такие факторы, как время, затраченное на решение профессиональных задач, качество знаний, передаваемых учителем, и природные способности человека. Научные исследования показали [58], что в становлении эксперта большую роль играют время, затраченное на упражнения в процессе всей его профессиональной деятельности, а также консультации учителя.

Установлено, что в памяти человека существует так называемая *рабочая память*, которая обеспечивает быстрый доступ к *долговременной памяти*. Такая организация памяти способствует высокой эффективности решения сложных плохо формализованных задач.

Исследования, результаты которых приведены в работах [67, 69], показали, что в процессе длительного решения типовых задач в памяти экспертов формируются особым образом организованные иерархические структуры знаний в виде наборов признаков и правил. По этим признакам и правилам происходит группировка объектов предметной области, обеспечивающая удобное хранение информации и быстрый доступ к ней. Процесс решения сложных задач человеком невозможно формализовать в полном

объеме, поскольку экспертные знания имеют подсознательный характер. Тем не менее существуют задачи, для которых разработаны подходы к построению полных, непротиворечивых и значительных по объему баз экспертных знаний (умений). Примером является задача классификации с явно заданными признаками, характерная для следующих проблем:

- диагностика заболеваний по совокупности клинических и инструментальных признаков;
- диагностика неисправностей сложных машин и механизмов;
- диагностика состояния предприятия, региона или государства по экономическим, технологическим, социальным и другим признакам;
- поиск месторождений полезных ископаемых.

Отличительной особенностью перечисленных задач является их повторяемость. При многократном решении подобных задач специалисты накапливают опыт, который позволяет им находить эффективные решения за короткое время. Этот опыт целесообразно представить в явной форме и впоследствии хранить, преумножать, корректировать и использовать при решении аналогичных задач.

В задачах классификации с явно заданными признаками искусство эксперта проявляется в умении «увидеть» через заданную совокупность значений отдельных признаков целостный образ объекта [22, 23].

Чтобы применить на практике метод классификации для построения баз экспертных знаний, необходимо иметь полный набор признаков, их значений и классов возможных решений.

Постановка задачи экспертной классификации. Дано: N — число диагностических признаков; W_i — число упорядоченных вербальных оценок качества на шкале i -го диагностического признака; $X_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$ — множество оценок на шкале i -го признака; Q — количество диагностических классов ($P_1, P_2, \dots, P_Q$), к которым могут принадлежать классифицируемые объекты; $A = X_1 \times X_2 \times X_3 \times \dots \times X_n$ — множество всех гипотетически возможных состояний, описываемых диагностическими признаками; $a_i = (a_{i1}, a_{i2}, \dots, a_{in})$ — состояние, принадлежащее множеству A , описываемое вектором, компонентами которого являются значения признаков.

Требуется: на основе знаний экспертов классифицировать все векторы a_j , отнеся каждый из них к одному или нескольким классам решений.

Данная постановка задачи основана на предположении, что эксперт обладает всей полнотой знаний, и нужно найти подход, позволяющий их выявить.

Метод экспертной классификации содержит ряд процедур, направленных на структуризацию проблемы и обработку экспертной информации. Результатом *структуризации проблемы* является совокупность признаков, необходимых для полной классификации объектов определенного типа, все возможные значения этих признаков и перечень классов возможных решений. Для выявления множества признаков, адекватно представляющих состояние объекта исследования, применяется человеко-машинная процедура АРИАДНА [30], в которой использованы идеи «диагностических игр» Гельфанда [11]. Эта процедура имитирует диалог по телефону между экспертом и неопытным специалистом при условии, что первый находится далеко от объекта исследования, а второй — рядом с объектом. Диалог построен так, что ЭВМ задает вопросы эксперту, используя его ответы для постановки новых вопросов. Таким образом, работая в режиме диалога с компьютером, эксперт описывает специфические характеристики объекта исследования для каждого класса решений.

Практические результаты по исследованию метода экспертной классификации получены в области медицинской диагностики [23, 24]. В частности, там была решена задача дифференциальной диагностики тромбоэмболии легочной артерии (ТЭЛА) и острого инфаркта миокарда (ОИМ). Для диагностики этих заболеваний эксперты выделили следующие диагностические признаки: анамнез — факторы риска; боль; цвет кожи; дыхание; артериальное давление; электрокардиограмма; рентгенограмма грудной клетки; эхокардиограмма; ферменты крови.

Для каждого из признаков были определены соответствующие значения. В частности, для признака *анамнез — факторы риска* на шкале были заданы следующие значения:

- 1 — в анамнезе операция, травмы, роды, тромбофлебит, опухоли;
- 2 — в анамнезе стенокардия, ишемическая болезнь сердца;
- 3 — в анамнезе патологии нет.

Классификация состояний объекта исследования осуществляется экспертом на основании предъявляемых ему компьютером комбинаций значений диагностических признаков, характеризующих одно из возможных состояний объекта в виде клинической ситуации. Дополнительно эксперту предоставляется перечень классов решений, из которых он выбирает свой ответ. При этом удастся значительно уменьшить число задаваемых вопросов, принимая *гипотезу о различной характерности* значений диагностического признака по отношению к каждому из классов. Эта гипотеза означает, что эксперт может упорядочить все значения каждого диагностического признака по их характерности для каждого из классов решений, при этом упорядочение не зависит от значений других признаков.

Пусть значения диагностического признака X_i находятся в следующем отношении характерности для класса P_j : $\langle x_{ji}, x_{mi} \rangle \in D_{P_j}$, где D_{P_j} — отношение доминирования для класса P_j . Данная запись означает, что для класса P_j значение i -го признака x_{ji} более характерно, чем значение x_{mi} .

Например, для диагностического признака *цвет кожи* шкала значений выглядит следующим образом:

- 1 — резкий цианоз лица, шеи, верхней половины туловища;
- 2 — бледность кожных покровов, акроцианоз;
- 3 — нормальный цвет кожи.

Для упомянутых ранее заболеваний ТЭЛА и ОИМ характерность значений названных признаков различается. Значения признаков для ТЭЛА и для ОИМ упорядочиваются по характерности 3-2-1 и 2-3-1 соответственно.

На основании бинарных отношений характерности для отдельных признаков можно построить отношения доминирования по характерности для каждого класса на множестве состояний (векторов a_i). При этом $\langle a_i, a_k \rangle \in D_{P_j}$, если для каждого признака значение соответствующего компонента вектора a_i не менее характерно по отношению к классу P_j , чем значение компонента вектора a_k , и хотя бы для одного компонента данное значение более характерно, чем прочие.

Использование гипотезы доминирования по характерности позволяет построить на множестве состояний A так называемый «конус» доминирования по характерности, который помогает классифицировать группы состояний. Например, эксперт отнес к классу P_j состояние a_i . При этом значения отдельных призна-

ков, являющиеся компонентами вектора a_j , могут быть не самыми характерными для класса P_j . Опираясь на гипотезу о характерности, можно ожидать, что другие состояния a_k , описания которых совпадают с a_j , за исключением тех диагностических признаков, значения которых не являются *самыми характерными* для класса P_j и могут иметь более характерные для P_j оценки, также будут отнесены к классу P_j . В связи с этим один ответ эксперта во многих случаях позволяет классифицировать группу состояний.

Выводимые гипотезы о характерности признаков требуют проверки. Это можно сделать путем сравнения классификаций, полученных на основе опроса экспертов, с последующим распространением информации в соответствии с конусами доминирования по характерности. В общем случае конусы могут пересекаться, так как некоторые состояния могут принадлежать одновременно нескольким классам. При этом могут иметь место явные противоречия, свидетельствующие либо о допущенных экспертом ошибках, либо о взаимных зависимостях диагностических признаков. В любом случае экспертную информацию необходимо скорректировать. При наличии зависимостей между признаками обычно производят объединение зависимых классов.

Важное место в задачах экспертной классификации отводится специальным алгоритмам, которые позволяют определить последовательность состояний, предъявляемых экспертам в процессе определения их принадлежности к классам. Существуют различные стратегии предъявления состояний (объектов) [10, 23, 24, 46], в частности, первыми могут предъявляться более информативные состояния.

Построенная полная классификация может быть представлена граничными элементами классов. Эти элементы находятся «на границе» между двумя классами решений, т.е. имеют значения признаков, характерные для нескольких классов. На основе этих элементов строятся решающие правила экспертов, имеющие структуру дерева. В корне этого дерева находятся наиболее существенные для класса решений значения признаков, к которым добавляется совокупность сочетаний значений менее важных признаков.

Технология формирования баз экспертных знаний для задач классификации с явными признаками [22] хорошо согласуется с известными данными психологов, которые получены при исследовании процессов классификации объектов, описываемых мно-

гими признаками. Например, установлено, что на основании многолетней практики эксперт-классификатор формирует в своей долговременной памяти совокупность решающих правил, которые позволяют описывать границы классов решений. При этом количество таких правил невелико и свободно помещается в кратковременной памяти. Когда эксперту предъявляется очередной объект, который он должен отнести к соответствующему классу решений, решающие правила переносятся в его кратковременную память, где производится анализ объекта. Если объект не удастся классифицировать с помощью решающих правил, находящихся в оперативной памяти, то они заменяются другими правилами из базы знаний эксперта, выработанными за время его профессиональной деятельности.

4.3. ПРОБЛЕМЫ ОБУЧЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Рассмотрим проблемы приобретения знаний интеллектуальными информационными системами. Процесс приобретения знаний называется *обучением*. Большинство обучающихся систем, разработанных до настоящего времени, ориентировано на воспроизведение компьютером механизмов приобретения знаний, свойственных человеку. Несмотря на ряд успехов в этой области, проблема сложного эвристического обучения интеллектуальных систем до сих пор не решена. С ее решением ряд исследователей [37, 38] связывает надежды на то, что обученные компьютеры смогут создавать программы и генерировать решения лучше, чем это удастся человеку.

Смысл процесса машинного обучения можно пояснить следующим образом. В процессе повторения однотипных экспериментов происходит модификация программной системы, в результате которой система демонстрирует на следующем этапе экспериментов результаты лучшие, чем прежде.

Мы уже говорили о том, что приобретение знаний реализуется с помощью двух функций: получения информации извне и ее систематизации. В зависимости от интеллектуальных способностей ИИС возможны разные формы приобретения знаний и представления исходной информации.

В книге японских специалистов по инженерии знаний [38] предложена следующая классификация этапов обучения, соответствующих способностям компьютерных систем к формализации.

A. Получение информации без логических выводов

1. Ввод программ.
2. Ввод фактических данных.

B. Получение знаний извне

1. Получение готового набора знаний, представленных во внутреннем формате.
2. Получение знаний во внутреннем формате в режиме диалога.
3. Получение знаний во внешнем формате и их понимание.

C. Обучение по примерам

1. Параметрическое обучение.
2. Обучение на основе выводов по аналогии.
3. Индуктивное обучение.
4. Обучение нейронных сетей.

D. Приобретение знаний на метауровне

Методы категории *A* можно назвать обучением без выводов или механическим запоминанием. Здесь полученная информация используется для решения задач в том же виде, в котором поступает в систему.

В методах категории *B* внешняя информация поступает в компьютерную систему в форме знаний, следовательно, ее можно использовать в процессе логического вывода. Примером такой информации являются правила, поступающие в ЭС продукционного типа. В интеллектуальной системе необходимо иметь функцию преобразования поступающих знаний в формат, удобный для дальнейшего использования. Эту функцию выполняют редакторы знаний, которые, в отличие от редакторов данных в БД, не только корректируют поступающую информацию, но и выполняют проверку знаний, содержащихся в БЗ, на наличие конфликтов (противоречий) с новыми знаниями.

Если знания поступают в ИИС во внешнем формате, то кроме преобразования и редактирования возникает задача их понимания. Такие проблемы характерны для систем с ЕЯ-интерфейсом.

Одной из актуальных проблем ИИ является *операционализация знаний*, связанная с пониманием, преобразованием и исполь-

зованием советов и подсказок, которые человек сообщает интеллектуальной системе в ходе решения конкретных задач. Советы и подсказки, поступающие в систему на естественном языке в терминах предметной области, необходимо преобразовывать в процедуры, ориентированные на выполнение определенных действий.

Категория *С* принципиально отличается от *А* и *В* тем, что интеллектуальные системы приобретают знания самостоятельно, выполняя сбор отдельных фактов, их обобщение и систематизацию. В процессе решения задач определенного класса (примеров) компьютерная система выявляет понятия, выбирает формат их представления и проводит структуризацию. Подобные задачи вызывают сложности и у человека (см. разд. 4.1, 4.2).

Проблемы понимания и преобразования знаний характерны для категорий *В* и *С*, причем последняя отличается наличием процедур обобщения, примерами которых являются замена постоянных атрибутов языка (констант) на переменные, исключение описаний с ограниченным применением, индукция, абдукция и др. Способы обобщения тесно связаны с языком представления знаний в ИИС.

Параметрическое обучение — самая простая форма в категории *С*. Оно заключается в определении общего вида правила, формирующего результат вывода, и в последующей корректировке входящих в него параметров, зависящих от конкретных данных. Пример обучающейся системы такого типа — известная экспертная система *Meta-Dendral*, предназначенная для построения структурных формул химических соединений на основе данных масс-спектрального анализа [37, 50]. В ней выводятся новые правила путем коррекции уже заложенных в БЗ продукций.

Обучение по аналогии базируется на гипотезе о том, что «если две ситуации подобны по нескольким признакам, то они подобны еще по одному признаку». Подобие ситуаций распознается на основе обобщенной меры совпадения значений важнейших признаков, с помощью которых описаны ситуации.

Вопросы индуктивного обучения рассматриваются в разд. 4.3.1, а моделям нейронных сетей посвящена глава 5.

Вопросы *приобретения знаний на метауровне* актуальны на современном этапе развития ИИ, так как связаны с выработкой стратегий управления процессом решения задач в ИИС. Это направление активно развивается, но здесь пока не выработано ус-

тойчивых представлений и апробированных моделей. Вопросы приобретения метазнаний частично затрагиваются в новых направлениях *Data Mining* и *Knowledge Discovery*, которые связаны с извлечением знаний из данных и будут рассмотрены ниже.

4.3.1. ИНДУКТИВНЫЕ ВЫВОДЫ В ЛОГИКЕ

Выводы по индукции позволяют на основе обобщения частных фактов получить общие (для некоторого множества объектов) закономерности. В процессе индуктивного обучения формируются новые правила, теории и структуры. Индуктивные выводы возможны в том случае, когда представление результата частично определяется на основе входной информации. В дедуктивных выводах (см. главу 2) диапазон порождаемых представлений не может быть шире диапазона, заданного априори. Диапазон представлений, порождаемых в процессе индуктивного вывода, шире, чем диапазон, заданный изначально.

Пусть P — множество известных фактов, имеющих в БЗ, а H — некоторая гипотеза (направленная на обобщение этих фактов). Если P выводится из H , то будем считать гипотезу H истинной. Это можно записать в виде

$$\frac{P, H \rightarrow P}{H}.$$

Рассмотрим пример. Пусть множество P включает факты:

$$P = \{ДОМ(иванов), ДОМ(петров), ДОМ(сидоров)\}.$$

Предикат $ДОМ(X)$ имеет интерпретацию «Объект X имеет дом». На основе заданного множества фактов P выдвигаем гипотезу $H: (\forall X) ДОМ(X)$, которая соответствует обобщению следующего вида: «Любой объект X имеет дом». Гипотеза H является истинной для любого факта из множества P , следовательно, P выводимо из H , и мы вправе считать гипотезу H истинной. Замена констант «иванов», «петров» и «сидоров» обобщающей переменной $(\forall X)$ расширяет область действия предиката $ДОМ(X)$ за пределы множества P . Например, при появлении нового объекта «березкин» можно получить вывод $ДОМ(березкин)$, хотя из P этого не следует. Следовательно, диапазон объектов расширился по

сравнению с исходным, а гипотеза H может быть помещена в БЗ как новый элемент знаний. За счет расширения множества P появляется возможность вывести новые заключения, которые также можно включить в БЗ. Однако при расширении класса объектов всегда есть возможность совершить ошибку. Например, если в рассмотренном примере появляется объект «береза», то приходим к странному выводу — $ДОМ(береза)$. Очевидно, обобщение H является слишком широким в данном случае. Попробуем сузить его, ограничив количество объектов. Пусть

$$\frac{P = \{P_1, P_2\}, \{P_1, H\} \rightarrow P_2}{H}, \text{ что можно интерпретировать фразой}$$

«гипотеза H истинна, если подмножество P_2 множества P можно вывести из гипотезы H и оставшегося подмножества P_1 ».

Допустим, подмножества P_1 и P_2 имеют вид:

$$P_1 = \{\forall X[ДОМ(X) \rightarrow ЧЕЛОВЕК(X)]\};$$

$$P_2 = \{ДОМ(иванов); ДОМ(петров); ДОМ(сидоров)\}.$$

На основании нового множества P можно выдвинуть другую гипотезу $H_1: (\forall X)[ЧЕЛОВЕК(X) \rightarrow ДОМ(X)]$, которая накладывает ограничение на область подстановки объектов в переменную X , так как в данном случае этот объект должен быть человеком. Теперь при появлении объекта «березкин» можно вывести факт $ДОМ(березкин)$, но при появлении объекта «береза» значением предиката $ЧЕЛОВЕК(береза)$ будет «ложь» и, следовательно, факт $ДОМ(береза)$ не будет выведен.

Если в множество P добавить информацию о существовании людей, не имеющих дома, т.е. в множество P_1 добавить формулу

$$(\exists X)[ЧЕЛОВЕК(X) \wedge \neg ДОМ(X)],$$

то при подстановке объекта «березкин» не сможем вывести предикат $ДОМ(березкин)$, так как принадлежность к роду человеческому в данном случае не является достаточным условием для владения недвижимостью. Следовательно, диапазон объектов подстановки необходимо сделать еще уже. Модифицируем подмножество P_1 , добавив еще одно условие:

$$(\forall X)[ДОМ(X) \rightarrow ВЛАДЕЛЕЦ(X)].$$

Выдвинем новую гипотезу H_2 :

$$(\forall X)[ВЛАДЕЛЕЦ(X) \rightarrow ДОМ(X)].$$

Теперь будем получать правдоподобные выводы, так как множество объектов для возможных подстановок в приведенные формулы ограничено людьми — владельцами домов.

Традиционный метод обобщения состоит в выборе *гипотезы минимального обобщения* среди большого числа возможных гипотез, в которых объекты из заданного множества фактов (P_2) заменяются переменной и которые расширяют диапазон применения исходных логических формул. Гипотеза H_2 в рассмотренном примере называется минимальным обобщением. Для того чтобы формализовать процесс минимального обобщения, необходимо иметь правила, с помощью которых можно выбрать ту или иную гипотезу. Например, чтобы увидеть, какое из ограничений сильнее $ЧЕЛОВЕК(X)$ или $ВЛАДЕЛЕЦ(X)$, необходимо иметь правило вида

$$(\forall X)[ВЛАДЕЛЕЦ(X) \rightarrow ЧЕЛОВЕК(X)].$$

Если такие правила сформулированы в системе, то процесс замены констант на переменные не представляет особых трудностей. Удаление из БЗ фактов, противоречащих установленным правилам, обычно не вызывает осложнений. Трудной проблемой является создание новых предикатов, поскольку эта операция неформализуема.

Таким образом, *индуктивный вывод* — это построение объясняющего правила на основе заданных данных. В системах с индуктивными выводами на каждом шаге необходимо *объяснять* все данные, полученные к заданному моменту времени. Данные, полученные на последующих шагах, могут не удовлетворять ранее полученным объяснениям. В этом случае следует корректировать полученные ранее объясняющие правила (гипотезы). Следовательно, процесс индуктивного обучения может оказаться весьма длительным, что вполне согласуется с процессом обучения человека.

Для реализации индуктивного вывода необходимо:

- сформулировать множество правил — объектов вывода;
- выбрать формальный метод представления правил;
- определить способ получения информации извне (показ примеров);
- задать формальный метод вывода;
- сформулировать критерий правильности вывода.

Индуктивные выводы выполняются в процессе бесконечного повторения цикла, включающего процедуры запроса входных

данных, формирования гипотез, получения и проверки результатов. В настоящее время индуктивные выводы часто используются для порождения объясняющих гипотез в системах с правдоподобными рассуждениями абдуктивного типа, в которых могут быть построены различные обобщения базовой теории, соответствующие наблюдаемым фактам.

4.3.2. ДСМ-МЕТОД

Сокращение ДСМ расшифровывается Джон Стюарт Милль. Этот ученый в середине XIX в. предложил принципы индуктивного вывода, которые положены в основу метода автоматического порождения гипотез. Способы установления причинно-следственных отношений, предложенные Миллем, основываются на идеях выявления сходства и различия в наблюдаемых ситуациях. Способность улавливать сходство и выделять различия — фундаментальная способность, присущая, по-видимому, всем живым существам. Опираясь на эту способность, Милль сформулировал следующие принципы индукции.

1. *Принцип единственного различия*: «Если после введения какого-либо фактора появляется (или после его удаления исчезает) известное явление, причем мы не вводим и не удаляем никакого другого обстоятельства, которое могло бы иметь влияние, то указанный фактор составляет причину явления» [34]. Этот принцип можно проиллюстрировать схемой:

$$A, B, C \Rightarrow D$$

$$A, B, C \Rightarrow D$$

.....

$$A, B, C \Rightarrow D$$

$$B, C \not\Rightarrow D$$

Знак $\Rightarrow$ трактуется как появление D при наличии A, B, C . При достаточном количестве экспериментов принцип единственного различия позволяет утверждать, что A является причиной, а D — следствием.

2. *Принцип единственного сходства*, который гласит: «Если все обстоятельства явления, кроме одного, могут отсутствовать, не уничтожая этим явления, то это обстоятельство является причиной данного явления». Схема такова:

$$A, B, C \Rightarrow D$$

$$A, B, C \Rightarrow D$$

.....

$$A, B \Rightarrow D$$

$$A, C \Rightarrow D$$

.....

$$A \Rightarrow D$$

Из этой схемы следует, что A и D связаны причинно-следственным отношением.

3. *Принцип единственного остатка*: «Если вычестъ из какого-либо явления ту его часть, которая является следствием известных причин, то остаток явления есть следствие остальных причин». Рассмотрим схему:

$$A, B, C \Rightarrow D, E$$

$$A, B, C \Rightarrow D, E$$

.....

$$B, C \Rightarrow E$$

После того как из примеров $A, B, C \Rightarrow D, E$ было «вычтено» причинно-следственное отношение $A \Rightarrow D$, были получены наблюдения $B, C \Rightarrow E$, на основании которых можно предположить, что B и C являются возможными причинами явления E . Для дальнейшего уточнения нужно проверить, приводит ли исключение B к появлению E . Если так, то причиной явления E служит C , в противном случае — B . Возможно также, что явление E обусловлено одновременным наличием B и C , т.е. появление некоторого элемента ситуации может определяться не отдельными факторами, а их совокупностью.

Схемы Милля справедливы лишь при условии, что в описании ситуации присутствует *полное* множество наблюдаемых фактов и явлений.

Сущность ДСМ-метода заключается в следующем. Пусть задано множество причин $A = \{A_1, A_2, \dots, A_p\}$, множество следствий $B = \{B_1, B_2, \dots, B_m\}$ и множество оценок $Q = \{q_1, q_2, \dots, q_r\}$. Выражение вида $A_i \Rightarrow B_j$ называется положительной гипотезой, выражающей утверждение « A_i является причиной B_j с оценкой достоверности q_k ». Отрицательной гипотезой называется выражение $A_i \neq B_j$, которое формулируется « A_i не является причиной B_j с оценкой достоверности q_k ». Положительные гипотезы будем обо-

значать $h_{ij,k}^+$, отрицательные — $h_{ij,k}^-$. Среди значений выделим два специальных, которые можно интерпретировать как «ложь» (0) и «истина» (1). Гипотезы с этими оценками можно рассматривать как явления, истинность или ложность которых твердо установлена. Остальные значения между 0 и 1 будем обозначать рациональными числами k/n , где $k=1, \dots, n-1$, а n характеризует число примеров.

Обобщенный алгоритм ДСМ-метода включает следующие шаги.

1. На основе исходного множества положительных и отрицательных примеров (наблюдений) формируется набор гипотез, которые записываются в матрицы M^+ и M^- . Гипотезы формируются на основе выявления сходства и различия в примерах. Матрицы имеют вид:

$$M^+ = \begin{array}{c|ccc} & B_l & \dots & B_w \\ \hline A_l & h_{llk}^+ & \dots & h_{lwm}^+ \\ \dots & \dots & \dots & \dots \\ A_r & h_{rls}^+ & \dots & h_{rwd}^+ \end{array} ; \quad M^- = \begin{array}{c|ccc} & B_j & \dots & B_v \\ \hline A_x & h_{xjk}^- & \dots & h_{xvi}^- \\ \dots & \dots & \dots & \dots \\ A_z & h_{zjm}^- & \dots & h_{zvf}^- \end{array} .$$

2. К исходному множеству примеров добавляются новые наблюдения, которые могут либо подтверждать выдвинутые гипотезы, либо опровергать их, при этом оценки гипотез изменяются следующим образом. Если некоторая гипотеза h_{ijk} имела оценку $q_k=k/n$, то при появлении нового примера $(n+1)$ проводится проверка на подтверждение этой гипотезы. В случае положительного ответа оценка $q_k=(k+1)/(n+1)$, иначе $q_k=(k-1)/(n+1)$. В процессе накопления информации оценки выдвинутых гипотез могут приближаться к 1 или 0. Изменение оценок может также иметь колебательный характер, что, как правило, ведет к исключению таких гипотез из множеств M^+ или M^- .

3. Циклическое добавление примеров, сопровождающееся изменением оценок достоверности гипотез с периодическим изменением множеств M^+ и M^- .

4. Завершение процесса индуктивного вывода при выполнении условий окончания цикла. В качестве таких условий могут использоваться меры близости значений q_i к 0 или 1, а также до-

полнительные условия, которые могут быть связаны с ограничением времени (количества новых примеров) вывода и т.п.

В современных модификациях ДСМ-метода используются выводы по аналогии, проводится учет контекста реализации причинно-следственных отношений, применяются нечеткие описания фактов и т.д. [3, 29, 43].

4.4. **СРЕДСТВА КОМПЬЮТЕРНОЙ ПОДДЕРЖКИ ПРИБРЕТЕНИЯ ЗНАНИЙ**

Проблема автоматизированного приобретения знаний связана с разработкой специальных информационных технологий, обеспечивающих поддержку процедур извлечения и структурирования знаний. К настоящему времени автоматизированные системы приобретения знаний прошли в своем развитии три стадии.

На первой стадии в середине 1980-х гг. появилось первое поколение систем приобретения знаний на базе «оболочек» экспертных систем. Процессы извлечения и структурирования знаний выполнялись человеком. Подсистема приобретения знаний служила для ввода знаний в БЗ и ее корректировки. Экспертные системы заполнялись знаниями по следующей схеме:

- создание конкретной экспертной системы;
- опустошение базы знаний;
- разработка системы приобретения знаний для нового наполнения БЗ;
- формирование базы знаний для другой экспертной системы.

На второй стадии в конце 1980-х гг. появились системы приобретения знаний второго поколения, основанные на предварительном детальном анализе предметной области и моделях, позволяющих рассматривать процедуры извлечения, структурирования и формализации знаний как процесс преобразования лингвистических знаний в другие представления и структуры [11]. Существенное влияние на системы второго поколения оказала психосемантика, на базе которой были созданы инструментальные средства многомерного шкалирования, факторного анализа, репертуарных решеток, логического вывода.

Третья стадия развития систем приобретения знаний (с 1990-х гг.) связана с созданием автоматизированных средств

приобретения знаний. При этом структура БЗ формируется в процессе приобретения знаний, а не заранее.

Множество существующих и потенциально возможных систем приобретения знаний можно отобразить классификацией, предложенной в работе [11] (табл. 4.6).

Таблица 4.6

Методы и системы приобретения знаний

Метод приобретения знаний	Наименование системы и авторы	Характеристика
Структурированное интервью	RESIAS (Davis R.) [55] ROGET (Bennet J.) [52] SALT (Markus S.) [70] MOLE (Eshelman L.) [59] OPAL (Muzen M.) [71] МЕДИКС (Ларичев О.И.) [10]	Формирует новые понятия и правила Производит концептуальную организацию знаний для диагностических ЭС Формирует базы знаний в области конструирования методом пошагового расширения ограничений Обеспечивает контекстное приобретение знаний на основе структурированного интервью Обеспечивает формирование и наращивание БЗ экспертной системы, дающей советы по лечению онкологических больных Использует процедуры экспертной классификации для независимых свойств, признаков и их значений. Повышение эффективности экспертной классификации обеспечивается за счет применения априорно заданного отношения линейного порядка на множестве состояний
Имитация консультаций	АРИАДНА (Моргоев В.) [30] ЭСКИЗ (Андриенко Г.) [3]	Реализует метод многократного решения экспертом проблемы классификации в режиме последовательной вопросно-ответной консультации Включает набор игр для приобретения знаний, являющихся модификациями метода репертуарных решеток

Метод приобретения знаний	Наименование системы и авторы	Характеристика
Интегрированные среды приобретения знаний	AQUINAS (Boose J.) [53]	Содержит набор программных средств для извлечения экспертных знаний разными методами: средства анализа репертуарных решеток с последующим преобразованием системы конструкторов в базу продукционных правил; методы конструирования иерархических структур знаний; средства извлечения и представления неточных знаний; подсистемы тестирования, пополнения и коррекции базы знаний и др.
	KITTEN (Shaw M.) [73]	Основана на построении и анализе репертуарных решеток. В отличие от AQUINAS данная интегрированная среда обеспечивает извлечение элементов из тестов, анализирует примеры решения задач экспертом и генерирует продукционные правила
Приобретение знаний из текстов	KRITON (Diderich J.) [57] TAKT (Kaplan R.) [65]	Выявляет процедурные знания на основе метода протокольного анализа из книг, документов, описаний, инструкций Выделяет из предварительно подготовленного текста объекты, процессы и отношения каузального характера
Инструментарий прямого приобретения знаний	SIMER + MIR (Осипов Г.С.) [33]	Позволяет формировать модели и базы знаний предметной области с неясной структурой объектов, неполно описанным множеством свойств объектов, большим набором разнородных связей между объектами

4.5. МЕТОДЫ И СРЕДСТВА ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Интеллектуальный анализ данных — одно из новых направлений ИИ. Этот термин является кратким и весьма неточным переводом с английского языка терминов *Data Mining* и *Knowledge Discovery in Databases (DM&KDD)*. Более точный перевод — «добыча данных» и «выявление знаний в базах данных». Появление технологий *DM&KDD* обусловлено накоплением огромных объемов информации в компьютерных базах данных, которые стало невыгодно хранить и которыми стало трудно пользоваться традиционными способами. Последнее обстоятельство связано со стремительным развитием вычислительной техники и программных средств для представления и обработки данных. Большие объемы накопленных данных постоянно приходится модифицировать из-за быстрой смены аппаратного и программного обеспечения БД, при этом неизбежны потери и искажение информации. Одним из средств для преодоления подобных трудностей является создание информационных хранилищ данных, доступ к которым не будет сильно зависеть от изменения данных во времени и от используемого программного обеспечения. Другой подход ориентирован на сжатие больших объемов данных путем нахождения некоторых общих закономерностей (знаний) в накопленной информации. Оба направления актуальны с практической точки зрения. Второй подход более интересен для специалистов в области ИИ, так как связан с решением проблемы приобретения новых знаний. Следует заметить, что наиболее плодотворным является сочетание обоих направлений.

Наличие хранилища данных — необходимое условие для успешного проведения всего процесса *KDD*. Вопросы организации хранилищ данных рассматриваются в [19, 20, 25, 40, 48]. Хранилищем данных называют предметно-ориентированное, интегрированное, привязанное ко времени, неизменяемое собрание данных, используемых для поддержки процесса принятия управленческих решений. Предметная ориентация означает, что данные объединены в категории и хранятся в соответствии с теми областями, которые они описывают, а не в соответствии с приложениями, которые их используют. Такой принцип хранения гарантирует, что отчеты, сгенерированные различными аналитиками, бу-

дуг опираться на одну и ту же совокупность данных. Привязанность ко времени означает, что хранилище можно рассматривать как собрание *исторических* данных, т.е. конкретные значения данных однозначно связаны с определенными моментами времени. Атрибут времени всегда явно присутствует в структурах хранилищ данных. Данные, занесенные в хранилище, уже не изменяются в отличие от оперативных систем, где присутствуют только последние, постоянно изменяемые версии данных. Для хранилищ данных характерны операции добавления, а не модификации данных. Современные средства администрирования хранилищ данных обеспечивают эффективное взаимодействие с программным инструментарием *DM* и *KDD*. В качестве примера можно привести разработки компании SAS Institute: *SAS Warehouse Administrator* и *SAS Enterprise Miner*.

Рассмотрим простой пример, иллюстрирующий технологии *DM&KDD*. В базах данных можно хранить большую таблицу значений переменных X и Y , но если удалось установить зависимость между этими переменными, то без существенных потерь информации можно значительно сократить объем занимаемой памяти, поместив туда найденную зависимость, например $Y = \sin(kX)$. В общем случае зависимости, выявляемые в базах данных, могут быть представлены правилами, гипотезами, моделями нейронных сетей и т.п. Интеллектуальные средства извлечения информации позволяют почерпнуть из БД более глубокие сведения, чем традиционные системы оперативной обработки транзакций (*OLTP – On-Line Transaction Processing*) и оперативной аналитической обработки (*OLAP*) [40]. Выведенные из данных закономерности и правила можно применять для описания существующих отношений и закономерностей, а также для принятия решений и прогнозирования их последствий.

Извлечение знаний из БД является одной из разновидностей машинного обучения, специфика которой заключается в том, что реальные БД, как правило, проектируются без учета потребностей извлечения знаний и содержат ошибки [7].

В технологиях *DM&KDD* используются различные математические методы и алгоритмы: классификация, кластеризация, регрессия, прогнозирование временных рядов, ассоциация, последовательность.

Классификация — инструмент обобщения. Она позволяет перейти от рассмотрения единичных объектов к обобщенным по-

нениям, которые характеризуют некоторые совокупности объектов и являются достаточными для распознавания объектов, принадлежащих этим совокупностям (классам). Суть процесса формирования понятий заключается в нахождении закономерностей, свойственных классам. Для описания объектов используются множества различных признаков (атрибутов). Проблема формирования понятий по признаковым описаниям была сформулирована М. М. Бонгартом [5]. Ее решение базируется на применении двух основных процедур: обучения и проверки. В процедурах обучения строится классифицирующее правило на основе обработки обучающего множества объектов. Процедура проверки (экзамена) состоит в использовании полученного классифицирующего правила для распознавания объектов из новой (экзаменационной) выборки. Если результаты проверки признаны удовлетворительными, то процесс обучения заканчивается, в противном случае классифицирующее правило уточняется в процессе повторного обучения.

Кластеризация — это распределение информации (записей) из БД по группам (кластерам) или сегментам с одновременным определением этих групп. В отличие от классификации здесь для проведения анализа не требуется предварительного задания классов.

Регрессионный анализ используется в том случае, если отношения между атрибутами объектов в БД выражены количественными оценками. Построенные уравнения регрессии позволяют вычислять значения зависимых атрибутов по заданным значениям независимых признаков.

Прогнозирование временных рядов является инструментом для определения тенденций изменения атрибутов рассматриваемых объектов с течением времени. Анализ поведения временных рядов позволяет прогнозировать значения исследуемых характеристик.

Ассоциация позволяет выделить устойчивые группы объектов, между которыми существуют неявно заданные связи. Частота появления отдельного предмета или группы предметов, выраженная в процентах, называется *распространенностью*. Низкий уровень распространенности (менее одной тысячной процента) говорит о том, что такая ассоциация не существенна. Ассоциации записываются в виде правил: $A \Rightarrow B$, где A — посылка, B — следствие. Для определения важности каждого полученного ассоциа-

тивного правила необходимо вычислить величину, которую называют *доверительность* A к B (или взаимосвязь A и B). Доверительность $\delta(A/B)$ показывает, как часто при появлении A появляется B , и рассчитывается как $\delta(A/B) = \epsilon(A \cap B) / \epsilon(A)$, где $\epsilon(A \cap B)$ — распространенность совместного появления A и B ; $\epsilon(A)$ — распространенность A . Например, если $\delta(A/B) = 20\%$, то это значит, что при покупке товара A в каждом пятом случае приобретается и товар B . Необходимо отметить, что если $\epsilon(A) \neq \epsilon(B)$, то $\delta(A/B) \neq \delta(B/A)$. В самом деле, покупка компьютера влечет за собой покупку дисков, но покупка дисков не ведет к покупке компьютера. Важной характеристикой ассоциации является *мощность*, которая рассчитывается по формуле $M(A/B) = \delta(A/B) / \epsilon(B)$. Чем больше мощность, тем сильнее влияние, которое наличие A оказывает на появление B .

Типичным примером применения ассоциации является анализ структуры покупок. Например, при проведении исследования в супермаркете можно установить, что 65% купивших картофельные чипсы берут также и «кока-колу», а при наличии скидки за такой комплект «колу» приобретают в 85% случаев. Подобные результаты представляют ценность при формировании маркетинговых стратегий.

Последовательность — это метод выявления ассоциаций во времени. В данном случае определяются правила, которые описывают последовательное появление определенных групп событий. Такие правила необходимы для построения сценариев. Кроме того, их можно использовать, например, для формирования типичного набора предшествующих продаж, которые могут повлечь за собой последующие продажи конкретного товара.

К интеллектуальным средствам *DM&KDD* относятся нейронные сети, деревья решений, индуктивные выводы, методы рассуждения по аналогии, нечеткие логические выводы, генетические алгоритмы, алгоритмы определения ассоциаций и последовательностей, анализ с избирательным действием, логическая регрессия, эволюционное программирование, визуализация данных. Иногда перечисленные методы применяются в различных комбинациях.

Нейронные сети относятся к классу нелинейных адаптивных систем с архитектурой, условно имитирующей нервную ткань, состоящую из нейронов. Математическая модель нейрона представляет собой некий универсальный нелинейный элемент, до-

пускающий возможность изменения и настройки его характеристик. Подробнее вопросы построения моделей нейронных сетей рассмотрены в главе 5. Нейронные сети широко применяются для решения задач классификации. Построенную сеть сначала нужно «обучить» на примерах, для которых известны значения исходных данных и результаты. Процесс «обучения» сети заключается в подборе весов межнейронных связей и модификации внутренних параметров активационной функции нейронов. «Обученная» сеть способна классифицировать новые объекты (или решать другие примеры), однако правила классификации остаются не известными пользователю.

Деревья решений — метод структурирования задачи в виде древовидного графа, вершины которого соответствуют продукционным правилам, позволяющим классифицировать данные или осуществлять анализ последствий решений. Этот метод дает наглядное представление о системе классифицирующих правил, если их не очень много. Простые задачи решаются с помощью этого метода гораздо быстрее, чем с использованием нейронных сетей. Для сложных проблем и для некоторых типов данных деревья решений могут оказаться неприемлемыми. Кроме того, для этого метода характерна проблема значимости. Одним из последствий иерархической кластеризации данных является то, что для многих частных случаев отсутствует достаточное число обучающих примеров, в связи с чем классификацию нельзя считать надежной. Методы деревьев решений реализованы во многих программных средствах, а именно: *C5.0* (RuleQuest, Австралия), *Clementine* (Integral Solutions, Великобритания), *SIPINA* (University of Lyon, Франция), *IDIS* (Information Discovery, США).

Индуктивные выводы позволяют получить обобщения фактов, хранящихся в БД (см. разд. 4.3.1, 4.3.2). В процессе индуктивного обучения может участвовать специалист, поставляющий гипотезы. Такой способ называют *обучением с учителем*. Поиск правил обобщения может осуществляться *без учителя* путем автоматической генерации гипотез. В современных программных средствах, как правило, сочетаются оба способа, а для проверки гипотез используются статистические методы. Примером системы с применением индуктивных выводов является *XpertRule Miner*, разработанная фирмой Attar Software Ltd. (Великобритания).

Рассуждения на основе аналогичных случаев (*Case-based reasoning* — *CBR*) основаны на поиске в БД ситуаций, описания которых

сходны по ряду признаков с заданной ситуацией. Принцип аналогии позволяет предполагать, что результаты похожих ситуаций также будут близки между собой. Недостаток этого подхода заключается в том, что здесь не создается каких-либо моделей или правил, обобщающих предыдущий опыт. Кроме того, надежность выводимых результатов зависит от полноты описания ситуаций, как и в процессах индуктивного вывода. Примерами систем, использующих CBR, являются: *KATE Tools* (Acknosoft, Франция), *Pattern Recognition Workbench* (Unica, США).

Нечеткая логика применяется для обработки данных с размытыми значениями истинности (см. разд. 3.2, 3.4), которые могут быть представлены разнообразными лингвистическими переменными. Нечеткое представление знаний широко применяется в системах с логическими выводами (дедуктивными, индуктивными, абдуктивными) для решения задач классификации и прогнозирования, например в системе *XpertRule Miner* (Attar Software Ltd., Великобритания), а также в *AIS* и *NeuFuz* и др. (см. главу 3).

Генетические алгоритмы входят в инструментарий *DM&KDD* как мощное средство решения комбинаторных и оптимизационных задач. Они часто применяются в сочетании с нейронными сетями (см. главу 6). В задачах извлечения знаний применение генетических алгоритмов сопряжено со сложностью оценки статистической значимости полученных решений и с трудностями построения критериев отбора удачных решений. Представителем пакетов из этой категории является *GeneHunter* фирмы Ward Systems Group. Генетические алгоритмы используются также в пакете *XpertRule Miner* и др.

Логическая (логистическая) регрессия используется для предсказания вероятности появления того или иного значения дискретной целевой переменной. Дискретная зависимая (целевая) переменная не может быть смоделирована методами обычной многофакторной линейной регрессии. Тем не менее вероятность результата может быть представлена как функция входных переменных, что позволяет получить количественные оценки влияния этих параметров на зависимую переменную. Полученные вероятности могут использоваться и для оценки шансов. Логическая регрессия — это, с одной стороны, инструмент классификации, который используется для предсказания значений категориальных переменных, с другой стороны — регрессионный инструмент, позволяющий оценить степень влияния входных факторов на результат.

Эволюционное программирование — самая новая и наиболее перспективная ветвь *DM&KDD*. Суть метода заключается в том, что гипотезы о форме зависимости целевой переменной от других переменных формулируются компьютерной системой в виде программ на определенном внутреннем языке программирования. Если это универсальный язык, то теоретически он способен выразить зависимости произвольной формы. Процесс построения таких программ организован как эволюция в мире программ (см. главу 6). Когда система находит программу, достаточно точно выражающую искомую зависимость, она начинает вносить в нее небольшие модификации и отбирает среди построенных дочерних программ те, которые являются наиболее точными. Затем найденные зависимости переводятся с внутреннего языка системы на понятный пользователю язык (математические формулы, таблицы и т.п.). При этом активно используются средства визуализации. Методы эволюционного программирования реализованы в системе *PolyAnalyst* (Unica, США).

В современных средствах *DM&KDD* часто используются комбинированные методы. Например, продукт компании *SAS Enterprise Miner 3.0* содержит модуль автоматического построения результирующей гибридной модели, определенной на множестве моделей, которые предварительно были созданы различными методами: деревьях решений, нейронных сетей, обобщенной многофакторной регрессии. Программная система *Darwin*, разработанная компанией *Thinking Machines*, позволяет не только строить модели на основе нейронных сетей или деревьев решений, но также использовать визуализацию и системы рассуждений по аналогии. Кроме того, этот продукт включает своеобразный генетический алгоритм для оптимизации моделей. Активно работает в области интеллектуального анализа данных компания *IBM*. Многие из полученных в ее лабораториях результатов нашли применение в выпускаемых инструментальных пакетах, которые можно отнести к четырем из пяти стандартных типов приложений «глубокой переработки» информации: классификации, кластеризации, выявлению последовательностей и ассоциаций.

В настоящее время на рынке представлены разнообразные программные средства, реализующие технологии *DM&KDD*. Следует отметить, что большинство из них имеет очень высокую стоимость. Рассмотрим некоторые известные пакеты. Среди инструментальных средств создания интеллектуальных приложений для бизнеса определенный интерес вызывает семейство програм-

мных продуктов *Business Intelligence (BI)* компании Cognos, которое включает четыре взаимосвязанные и дополняющие друг друга системы: *Impromptu*, *PowerPlay*, *Scenario* и *4Thought*. Отличительной особенностью рассматриваемых средств является сочетание эффективности реализуемых в них методов с дружественным интерфейсом, что делает их легкодоступными для освоения непрограммирующими пользователями [48].

Система *Impromptu* обеспечивает доступ к базам данных, позволяя непрофессиональному пользователю формировать разнообразные запросы и отчеты. Система поддерживает работу с пространственными типами СУБД: Oracle, MS SQL Server, Sybase SQL Server, Sybase NetGateway, OmniSQL Gateway, MDI DB2 Gateway, Informix, CA-Ingres, Gupta SQLBase, а также обеспечивает доступ через ODBC. Отчеты *Impromptu* могут использоваться в качестве источников данных для остальных систем семейства BI, выступая при этом в роли информационных витрин.

Система *PowerPlay* является средством для проведения OLAP-анализа. Технология OLAP позволяет существенно повысить эффективность обработки информации в реляционных БД за счет многомерного представления данных в виде гиперкубов; привязки информации ко времени, дающей возможность анализа динамики данных; реализации сложной вычислительной обработки больших массивов данных [54].

PowerPlay обеспечивает многомерный просмотр данных с нисходящим и уровневый анализом, в процессе которого существует возможность выявления исключений и особых случаев, ранжирования и разнообразной обработки данных. Гиперкубы имеют неограниченную размерность и могут создаваться как на серверах, так и на клиентских компьютерах. Новые версии *PowerPlay* обеспечивают возможность работы с гиперкубами через Web-браузеры.

Система *Scenario* предназначена для выявления взаимосвязей в данных статистическими методами, в частности по критерию Хи-квадрат (метод CHAID) для нахождения однородных сегментов данных с «аналогичным» поведением относительно целевого показателя. Кроме того, в системе используется метод деревьев решений для построения классификаций. Результаты обработки данных наглядно отображаются средствами визуализации.

В системе реализованы следующие виды анализа данных:

- *ранжирование* — упорядочение факторов по степени их влияния на целевой показатель. С каждым фактором связывается ве-

совой коэффициент, дающий количественную оценку степени влияния;

- *сегментация* — разделение области значений фактора на сегменты для проведения дальнейшего нисходящего анализа;

- *профилирование лучших образцов* — выявление основных характеристик наиболее успешных результатов (регионов, филиалов, клиентов и т.д.);

- *выявление ассоциаций* — поиск ассоциированных групп значений факторов;

- *выявление исключений* — поиск элементов, выпадающих из общей картины. Появление подобных элементов может быть вызвано как ошибками в данных, которые следует исправить, так и необычными ситуациями в работе компании, требующими определенных действий со стороны руководства.

В системе *Scenario* реализованы три стратегии анализа: 1) режим исследования, предназначенный для предварительного анализа задачи; 2) режим тестирования, ориентированный на высокую точность и надежность результатов; 3) режим верификации, позволяющий проводить оценку достоверности и значимости полученных знаний.

Система *4Thought* осуществляет извлечение знаний из БД с применением нейронных сетей, предоставляя следующие возможности:

- моделирование сложных нелинейных зависимостей между факторами и целевыми показателями;

- выявление тенденций в данных (при наличии временных рядов);

- работа с неполными и зашумленными данными при относительно небольшом объеме исходной информации.

Анализируемые данные представляются в виде электронной таблицы, столбцам которой соответствуют атрибуты из таблиц базы данных, а строкам — записи. При подготовке данных для анализа пользователь может редактировать таблицы, а также включать в них вычисляемые столбцы. Как и в пакете *Scenario*, для постановки задачи моделирования должны быть указаны целевой (моделируемый) показатель и переменные-факторы. Данные в таблице рассматриваются системой как примеры для обучения нейронной сети. Совокупность всех данных разбивается на

два подмножества — модельное и тестовое. В каждом цикле обучения сеть сначала обучается на модельном подмножестве, а затем проверяется корректность ее работы на тестовых данных. Обучение сети проводится до тех пор, пока точность результатов, полученных на модельном и на тестовом подмножествах, растет. При работе с небольшими объемами данных, когда выбор тестового подмножества существенно влияет на результаты обучения, пользователь может проводить обучение последовательными циклами, выбирая для каждого цикла новое разбиение на тестовые и модельные данные.

Кроме рассмотренных программных продуктов фирмы Cognos можно привести длинный список программных средств, ориентированных на поддержку *DM&KDD*. Особенно активно в последние годы развиваются средства с использованием нейронных сетей, примерами которых являются: *BrainMaker* (CSS), *NeuroShell* (Ward Systems Group), *OWL* (HyperLogic).

Применение технологий *DM&KDD* имеет большие перспективы, так как существенно влияет на увеличение доходов предприятия путем выбора правильных стратегий деятельности. Лидерами в применении технологий интеллектуального анализа данных являются телекоммуникационные компании и компании, выпускающие кредитные карточки. Средства *DM&KDD* активно применяют страховые компании и фондовые биржи. Серьезные успехи связаны с применением этих подходов в медицине, где можно прогнозировать эффективность применения медикаментов, хирургических процедур и медицинских тестов. Прогнозирование в финансовой сфере всегда было одной из самых актуальных задач. В настоящее время компании, действующие на финансовом рынке, на основе *DM*-технологий определяют рыночные и отраслевые характеристики для предсказания индивидуальных и фондовых предпочтений в ближайшем будущем.

Ключом успеха в применении методов *DM&KDD* являются качество данных, мощность используемого программного обеспечения и мастерство аналитика, который участвует в процессе построения модели. На эффективность обработки данных положительно влияют следующие параметры программного обеспечения: развитые средства формирования запросов и визуализа-

ции результатов, наличие графического инструментария, возможность оперативной аналитической обработки, разнообразие и эффективность алгоритмов построения моделей.

К типовым задачам *DM&KDD* в области экономики и бизнеса относятся [19, 48]:

- прогнозирование — в частности, при планировании и составлении бюджета фирмы необходимо прогнозировать объемы продаж и другие параметры с учетом многочисленных взаимосвязанных факторов: сезонных, региональных, общеэкономических и т.п.;

- маркетинговый анализ, в результате которого определяется зависимость спроса от таких факторов, как стоимость товара, затраты на продвижение продукции и рекламу и т.д.;

- анализ работы персонала — производительность труда служащих зависит от уровня подготовки, от оплаты труда, опыта работы, взаимоотношений с руководством и т.д. Установив степень влияния этих факторов, можно выработать методику повышения производительности труда, а также предложить оптимальную стратегию подбора кадров в будущем;

- анализ эффективности продажи товаров по почте — выявляется круг потенциальных покупателей, оценивается вероятность возможных покупок и исследуются различные формы рекламной переписки;

- профилирование клиентов — формирование «портрета типичного клиента компании», т.е. определение группы клиентов, сотрудничество с которыми наиболее выгодно. Кроме того, важно выяснить, почему работа с некоторыми из заказчиков стала неэффективной, и выработать стратегию поиска подходящих клиентов в будущем;

- оценка потенциальных клиентов — выявление характерных особенностей заявок, которые закончились реальными продажами. Полученные знания используются в процессах планирования переговоров и сделок;

- анализ работы региональных отделений компании;

- сравнительный анализ конкурирующих фирм.

Перечисленные задачи актуальны практически для всех отраслей бизнеса: банковского дела, страхования, финансовых рынков, производства, торговли и т.д.

1. Адамар Ж. Исследования психологии процесса изобретения в области математики. — М.: Советское радио, 1974.
2. Андрейчиков А. В., Андрейчикова О. Н. Анализ, синтез, планирование решений в экономике. — М.: Финансы и статистика, 2000.
3. Андриенко Г. Л., Андриенко Н. В. Игровые процедуры сопоставления в инженерии знаний // Сб. тр. III конференции по искусственному интеллекту. — Тверь, 1992.
4. Бажин И. И. Информационные системы менеджмента. — М.: ГУ-ВШЭ, 2000.
5. Бонгарт М. М. Проблема узнавания. — М.: Наука, 1967.
6. Буч Г. Объектно-ориентированное проектирование с примерами применения. — М.: Конкорд, 1992.
7. Вагин В. Н., Федотов А. А., Фомина М. В. Методы извлечения и обобщения информации в больших базах данных // Известия Академии наук. Сер. Теория и системы управления. — 1999. — № 5.
8. Вертгеймер М. Продуктивное мышление: Пер. с нем. — М.: Прогресс, 1987.
9. Выготский Л. С. Мышление и речь: Собр. соч. Т. 2. — М.: Педагогика, 1982.
10. Выявление экспертных знаний / О. И. Ларичев, А. И. Мечитов, Е. М. Мошкович, Е. М. Фуремс. — М.: Наука, 1989.
11. Гаврилова Т. А., Хорошевский В. Ф. Базы знаний интеллектуальных систем. — СПб.: Питер, 2000.
12. Гельфанд И. М., Розенфельд Б. И., Шифрин М. А. Структурная организация данных в задачах медицинской диагностики и прогнозирования // Вопросы кибернетики. Задачи медицинской диагностики и прогнозирования с точки зрения врача. — М.: АН СССР, 1982.
13. Джонс Дж. К. Методы проектирования: Пер. с англ. — М.: Мир, 1986.
14. Дрейфус Х. Чего не могут вычислительные машины. — М.: Прогресс, 1978.
15. Дюран Б., Оддел П. Кластерный анализ: Пер. с фр. — М.: Статистика, 1977.
16. Жамбю М. Иерархический кластер-анализ и соответствия. — М.: Финансы и статистика, 1988.
17. Каган М. С. Мир общения: проблема межсубъектных отношений. — М.: Политиздат, 1988.

18. *Кендалл М., Стьюарт А.* Многомерный статистический анализ и временные ряды: Пер. с англ. — М.: Наука, главн. ред физ.-мат. литературы, 1976.
19. *Киселев М., Соломатин Е.* Средства добычи знаний в бизнесе и финансах // Открытые системы. — 1997. — № 4.
20. *Кречетов Н., Иванов П.* Продукты для интеллектуального анализа данных // Computer Week — Москва. — 1997. — № 14–15.
21. *Ларичев О. И.* Структура экспертных знаний // Психологический журнал. — 1995. — № 3.
22. *Ларичев О. И.* Теория и методы принятия решений, а также хроника событий в Волшебных странах. — М.: Логос, 2000.
23. *Ларичев О. И., Болотов А. А.* Система ДИФКЛАСС: построение полных непротиворечивых баз экспертных знаний в задачах дифференциальной классификации // Научно-техническая информация. Сер. 2. — 1996. — № 9.
24. *Ларичев О. И., Нарыжный Е. В.* Компьютерное обучение экспертным знаниям // Доклады Академии наук. — 1998. — Т. 332.
25. *Львов В.* Создание систем поддержки принятия решений на основе хранилища данных // Системы управления базами данных. — 1997. — № 3.
26. *Мандель Н. Д.* Кластерный анализ. — М.: Финансы и статистика, 1988.
27. *Миркин Б. Г.* Анализ качественных признаков и структур. — М.: Статистика, 1960.
28. *Митиг П. П.* Как проводить деловые беседы. — М.: Экономика, 1987.
29. *Михеенкова М. А., Финн В. К.* Правдоподобное рассуждение с информацией о ситуации // Тр. 7-й нац. конф. по искусственному интеллекту с международным участием. — М.: Изд-во физ.-мат. литературы, 2000.
30. *Моргоев В. К.* Метод извлечения и структуризация экспертных знаний: моделирование консультаций // Сб. тр. ВНИИСИ «Человеко-машинные процедуры принятия решений» / Под ред. С. В. Емельянова, О. И. Ларичева. — М.: ВНИИСИ, 1988.
31. *Нечеткие множества в моделях управления и искусственного интеллекта* / Под ред. Д. А. Поспелова. — М.: Наука, 1986.
32. *Обозов И. И.* Психологическая культура взаимных отношений. — М.: Знание, 1986.
33. *Осипов Г. С.* Приобретение знаний интеллектуальными системами. — М.: Наука, 1997.

34. *Поспелов Д. А.* Моделирование рассуждений. Опыт анализа мыслительных актов. — М.: Радио и связь, 1989.
35. *Поспелов Д. А.* Ситуационное управление: Теория и практика. — М.: Наука, 1986.
36. *Поспелов Д. А., Осипов Г. С.* Прикладная семиотика // Новости искусственного интеллекта. — 1999. — № 1.
37. Представление и использование знаний: Пер. с яп. / Под ред. Х.Уэно, М. Исидзука. — М.: Мир, 1989.
38. Приобретение знаний: Пер. с яп. / Под ред. С. Осуги, Ю. Сазэки. — М.: Мир, 1990.
39. *Романов А. Н., Одинцов Б. Е.* Советующие информационные системы в экономике. — М.: ЮНИТИ-ДАНА, 2000.
40. Статические и динамические экспертные системы: Учеб. пособие / Э.В. Попов, И. Б. Фоминых, Е. Б. Кисель, М. Д. Шапот. — М.: Финансы и статистика, 1996.
41. *Терехина А. Ю.* Анализ данных методами многомерного шкалирования. — М.: Наука, 1986.
42. *Терехина А. Ю.* Представление структуры знаний методами многомерного шкалирования. — М.: ВИНТИ, 1988.
43. *Финн В. К.* Правдоподобные рассуждения в интеллектуальных системах типа ДСМ // Итоги науки и техники. Сер. «Информатика», т. 15. Интеллектуальные информационные системы. — М.: ВИНТИ, 1991.
44. *Финн В. К.* Философские проблемы логики интеллектуальных систем // Новости искусственного интеллекта. — 1999. — № 1.
45. *Франселла Ф., Баннистер Д.* Новый метод исследования личности: руководство по репертуарным личностным методикам: Пер. с англ. — М.: Прогресс, 1987.
46. *Фуремс Е. М., Гниденко А. С.* STEPCLASS — система извлечения знаний и проведения экспертизы при решении диагностических задач // Научно-техническая информация. Сер. 2. — 1996. — № 9.
47. *Шапот М.* Интеллектуальный анализ данных в системах поддержки принятия решений // Открытые системы. — 1998 — № 1.
48. *Шапот М., Рошупкина В.* Интеллектуальный анализ данных и управление процессами // Открытые системы. — 1998. — № 4.
49. *Шенк Р., Бирнбаум Л., Мей Дж.* К интерпретации семантики и прагматики // Новое в зарубежной лингвистике. Компьютерная лингвистика. Вып. 14. — М.: Прогресс, 1989.
50. *Элти Дж., Кумбс М.* Экспертные системы: концепции и примеры: Пер. с англ. — М.: Финансы и статистика, 1987.

51. Язык и интеллект. — М.: Прогресс, 1996.
52. *Bennet J. S.* A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system // *Journal of Automated Reasoning*. — 1985. — No. 1.
53. *Boose J. H., Bradshaw J. H., Shema D. B.* Transforming repertory grids to shell-based knowledge bases using AQUINAS, a knowledge acquisition workbench // *Proceedings of the AAAI-88 Integration of Knowledge Acquisition and Performance Systems Workshop*. — St. Paul, 1988.
54. *Codd E. F., Codd S. B., Salley C. T.* Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. — E. F. Codd & Associates, 1993.
55. *Davis R.* Interactive transfer of expertise // In: *Rule-Based Expert Systems* Ed. by Buchanan B.G., Shortliffi E.H. — London: Addison - Wesley, 1984.
56. *De Marco T.* Structured analysis and system specification. — Englewood Cliffs, NJ : Prentice Hall, 1979.
57. *Diderich J., Ruhman I., May M.* KRITON: a knowledge acquisition tool for expert systems // *Int. Journal of Man-Machine Studies*. 1987. — Vol. 26. — No. 1.
58. *Ericson K. A.* Expert and exceptional performance: evidence of maximum adaptation to task constraints // *Annual Review of Psychology*. — 1996. — No. 47.
59. *Eshelman L.* MOLE. Knowledge acquisition tool that buries certainty factors // *Int. Journal of Man-Machine Studies*. — 1987. — Vol. 26. — No. 1.
60. *Fransella F., Dalton P.* Personal construct counseling in action. — London: Sage Publications, 1994.
61. *Gaines B. R., Shaw M.L.G.* Knowledge engineering techniques // *Proceedings of AUTOFACT'86*, Detroit, 1986.
62. *Gane C., Sarson T.* Structured system analysis. — Englewood Cliffs, NJ: Prentice Hall, 1979.
63. *Guarino N., Poli R.* The role of ontology in the information technology // *International Journal of Human- Computer Studies*, Special issue on the ontology. — 1995. — No. 43 (5/6).
64. *Jones A.* The Object model: a conceptual toll for structuring software. Operating Systems. — New York: Springer-Verlag, 1992.
65. *Kaplan R. N., Berry-Rogghe G.* Knowledge-based acquisition of causal relationships in text // *Knowledge Acquisition*. — 1991. — No. 3.

66. *Kelly G. A.* Psychology of personal constructs. — New York: Norton, 1955.
67. *Kihlstrom J.* The Cognitive unconscious // Science. — 1987. — Vol. 237.
68. *Kintsch W.* The representation of meaning in memory. — New York, 1974.
69. *Lewicki P., Hill., Czyzewska M.* Non conscious acquisition of information // American Psychologist. — 1992. — June.
70. *Markus S.* Taking backtracking with a grain of SALT // Int. Journal of Man-Machine Studies. — 1987. — Vol. 26. — No. 4.
71. *Muzen M. A., Fagan L. M., Combs P. M., Shortliffe E. N.* Use of a domain model to interactive knowledge-editing tool // Int. Journal of Man-Machine Studies. — 1987. — Vol. 26. — No. 1.
72. Repertory Grid Technique and Personal Constructs / Edited by Nigel Beail. — Groom Helm London & Sydney, 1993.
73. *Shaw M.* Abstraction techniques in modern programming languages // IEEE Software. — 1984. — Vol. 1(4).
74. *Shaw M. L., Woodward J. B.* Validation of knowledge support system // Proceedings of the 2nd Knowledge Acquisition for Knowledge-Based Workshop, Banff, Canada, 1988.
75. *Yourdon E.* Modern Structured Analysis. — Prentice-Hall Int. Ed., 1989.

Контрольные вопросы и задания

1. Охарактеризуйте основные аспекты процесса извлечения знаний (психологический, лингвистический, гносеологический).
2. В чем заключаются основные особенности структурирования знаний на основе структурного и объектно-ориентированного подходов?
3. Опишите семиотический подход к моделированию человеческих знаний. Приведите пример треугольника Фреге. Охарактеризуйте синтаксис, семантику и прагматику знаков.
4. Дайте сравнительную характеристику методов извлечения знаний.
5. Какие способы можно применять для извлечения «скрытых» знаний? Опишите принципы многомерного шкалирования, достоинства и недостатки этого подхода.
6. Опишите метод репертуарных решеток Дж. Келли. Приведите пример. Расскажите о возможных применениях этого метода.

7. Сформулируйте постановку задачи экспертной классификации с явно заданными признаками при построении баз знаний для экспертных систем диагностики. Что такое «конусы доминирования по характерности»?
8. Расскажите о методах машинного обучения. Приведите классификацию и дайте характеристики методов.
9. Охарактеризуйте индуктивные выводы в логике. В чем их основное отличие от дедуктивных выводов? Приведите пример индуктивного вывода.
10. Сформулируйте основные принципы индукции Милля. Опишите обобщенный алгоритм ДСМ-метода.
11. Чем отличаются хранилища данных от баз данных? Что такое интеллектуальный анализ данных?
12. Дайте характеристику основных методов и алгоритмов, которые используются в технологиях *DM&KDD*.
13. Сформулируйте примеры задач в области экономики и бизнеса, для решения которых актуально применение технологий *DM&KDD*.
14. Подготовьте доклад или реферат о возможностях конкретного программного средства для извлечения знаний.
15. Решите небольшую задачу извлечения знаний из имеющейся БД, используя доступные программные средства (например, Excell Statistica for Windows и др.).
16. Подготовьте набор знаний для индуктивного вывода, используя логику предикатов первого порядка.
17. Решите задачу классификации из хорошо знакомой вам предметной области, используя методику дерева решений.
18. Подготовьте набор знаний для решения задачи экспертной классификации в хорошо знакомой вам области.
19. Сформируйте набор знаний из знакомой вам предметной области, используя семиотическую модель.
20. Подготовьте набор содержательной информации для выполнения абдуктивного вывода. Разработайте алгоритм его реализации.
21. Разработайте программное обеспечение для реализации ДСМ-метода.

Изучая нейрофизиологию, можно многое узнать о машинах. Изучая машины, ничего нельзя узнать о нейрофизиологии.

Джон фон Нейман

Особенностью интеллектуальных систем является способность решать слабоструктурированные и плохо формализованные задачи. Эта способность основана на применении различных методов моделирования рассуждений для обработки символьной информации. Традиционным подходом к построению механизмов рассуждения является использование дедуктивного логического вывода на правилах (*rule-based reasoning*), который применяется в экспертных системах продукционного и логического типа (см. главу 2). При таком подходе необходимо заранее сформулировать весь набор закономерностей, описывающих предметную область. Альтернативный подход основан на концепции обучения по примерам (*case-based reasoning*). В этом случае при построении интеллектуальной системы не требуется заранее знать обо всех закономерностях исследуемой области, но необходимо располагать достаточным количеством примеров для настройки разрабатываемой адаптивной системы, которая после обучения будет способна получать требуемые результаты с определенной степенью достоверности. В качестве таких адаптивных систем применяются искусственные нейронные сети.

5.1. МОДЕЛЬ ИСКУССТВЕННОГО НЕЙРОНА

Искусственная нейронная сеть (ИНС) — это упрощенная модель биологического мозга, точнее нервной ткани [2, 5, 9, 12]. Естественная нервная клетка (нейрон) состоит из тела (*сомы*), содержащего ядро, и отростков — *дендритов*, по которым в нейрон поступают входные сигналы. Один из отростков, ветвящийся на конце, служит для передачи выходных сигналов данного нейрона

другим нервным клеткам. Он называется *аксоном*. Соединение аксона с дендритом другого нейрона называется *синапсом*. Нейрон возбуждается и передает сигнал через аксон, если число пришедших по дендритам возбуждающих сигналов больше, чем число тормозящих.

Сеть ИНС представляет собой совокупность простых вычислительных элементов — искусственных нейронов, каждый из которых обладает определенным количеством входов (дендритов) и единственным выходом (аксоном), разветвления которого подходят к синапсам, связывающим его с другими нейронами. На входы нейрона поступает информация извне или от других нейронов. Каждый нейрон характеризуется функцией преобразования входных сигналов в выходной (функция возбуждения нейрона). Нейроны в сети могут иметь одинаковые или разные функции возбуждения. Сигналы, поступающие на вход нейрона, неравнозначны в том смысле, что информация из одного источника может быть более важной, чем из другого. Приоритеты входов задаются с помощью вектора весовых коэффициентов, моделирующих синаптическую силу биологических нейронов.

Модель искусственного нейрона (рис. 5.1) представляет собой дискретно-непрерывный преобразователь информации. Информация, поступающая на вход нейрона, суммируется с учетом весовых коэффициентов w_i сигналов x_i , $i = 1, \dots, n$, где n — размерность пространства входных сигналов. Потенциал нейрона определяется по формуле

$$P = \sum_{i=1}^n w_i x_i.$$

Взвешенная сумма поступивших сигналов (потенциал) преобразуется с помощью передаточной функции $f(P)$ в выходной сигнал нейрона Y , который передается другим нейронам сети, т. е. $Y = f(P)$. Вид передаточной (активационной) функции является важнейшей характеристикой нейрона. В общем случае эта функция может быть ступенчатой (пороговой), линейной или нелинейной (рис. 5.2). Пороговая функция пропускает информацию только в том случае, если алгебраическая сумма входных сигналов превышает некоторую постоянную величину P^* , например:

$$Y = \begin{cases} 1, & \text{если } P \geq P^*; \\ -1, & \text{если } P < P^*. \end{cases}$$

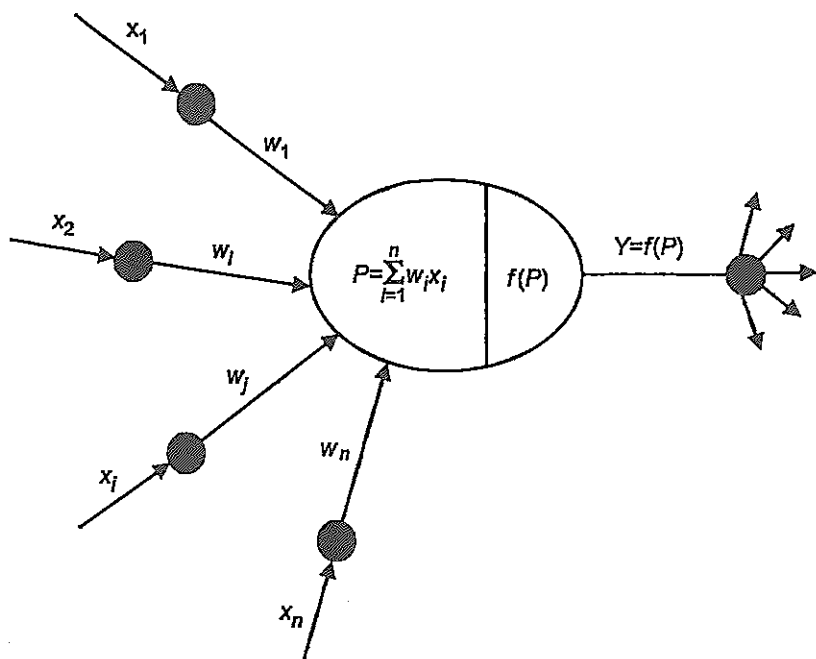


Рис. 5.1. Схема кибернетической модели нейрона

Пороговая функция не обеспечивает достаточной гибкости ИНС при обучении. Если значение вычисленного потенциала не достигает заданного порога, то выходной сигнал не формируется и нейрон «не срабатывает». Это приводит к снижению интенсивности выходного сигнала нейрона и, как следствие, к формированию невысокого значения потенциала взвешенных входов в следующем слое нейронов.

Линейная функция $Y=kP$ дифференцируема и легко вычисляется, что в ряде случаев позволяет уменьшить ошибки выходных сигналов в сети, так как передаточная функция сети также является линейной. Однако она не универсальна и не обеспечивает решения многих задач.

Определенным компромиссом между линейной и ступенчатой функциями является сигмоидальная функция переноса $Y = 1/(1+e^{-kP})$, которая удачно моделирует передаточную характеристику биологического нейрона (рис. 5.2, в). Коэффициент k

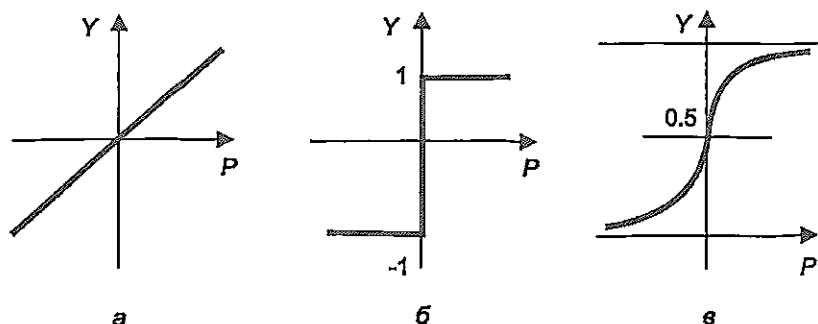


Рис. 5.2. Функции переноса искусственных нейронов:
 а — линейная; б — ступенчатая; в — сигмоидальная

определяет крутизну нелинейной функции: чем больше k , тем ближе сигмоидальная функция к пороговой; чем меньше k , тем она ближе к линейной. Подобно ступенчатой функции она позволяет выделять в пространстве признаков множества сложной формы, в том числе невыпуклые и несвязные. При этом сигмоидальная функция, в отличие от ступенчатой, не имеет разрывов. Она дифференцируема, как и линейная функция, и это качество можно использовать при поиске экстремума в пространстве параметров ИНС.

Тип функции переноса выбирается с учетом конкретной задачи, решаемой с применением нейронных сетей. Например, в задачах аппроксимации и классификации предпочтение отдают логистической (сигмоидальной) кривой. Нейронная сеть представляет собой совокупность искусственных нейронов, организованных слоями. При этом выходы нейронов одного слоя соединяются с входами нейронов другого. В зависимости от топологии соединений нейронов ИНС подразделяются на одноуровневые и многоуровневые, с обратными связями и без них. Связи между слоями могут иметь различную структуру. В однолинейных сетях каждый нейрон (узел) нижнего слоя связан с одним нейроном верхнего слоя. Если каждый нейрон нижнего слоя соединен с несколькими нейронами следующего слоя, то получается пирамидальная сеть. Воронкообразная схема соединений предполагает связь каждого узла верхнего слоя со всеми узлами нижнего уровня. Существуют также древовидные и рекуррентные сети, содержащие обратные связи с произвольной структурой межнейронных соединений. Чтобы построить ИНС для решения конкрет-

ной задачи, нужно выбрать тип соединения нейронов, определить вид передаточных функций элементов и подобрать весовые коэффициенты межнейронных связей [1, 2, 5 – 7, 12].

При всем многообразии возможных конфигураций ИНС на практике получили распространение лишь некоторые из них. Классические модели нейронных сетей рассмотрены ниже.

5.2. МОДЕЛИ НЕЙРОННЫХ СЕТЕЙ

Теоретические основы нейроматематики были заложены в начале 1940-х гг. Попытки построить машины, способные к разумному поведению, были в значительной мере вдохновлены идеями «отца кибернетики» Норберта Винера, который писал в своей знаменитой работе «Кибернетика или управление и связь в животном и машине», что все машины, претендующие на «разумность», должны обладать способностью преследовать определенные цели и приспосабливаться, т.е. обучаться. Идеи Винера были применены Дж. Маккалохом и У. Питтсом, которые разработали собственную теорию деятельности головного мозга [3], основанную на предположении, что функционирование компьютера и мозга сходно. К главным результатам их работы относятся следующие:

- модель нейрона в виде простейшего процессорного элемента, который вычисляет значение переходной функции от скалярного произведения вектора входных сигналов и вектора весовых коэффициентов;
- конструкция нейронной сети для выполнения логических и арифметических операций;
- предположение о том, что нейронная сеть способна обучаться, распознавать образы, обобщать полученную информацию.

В формализме Дж. Маккалоха и У. Питтса нейроны имеют пороговую функцию перехода из состояния в состояние. Каждый нейрон в сети определяет взвешенную сумму состояний всех других нейронов и сравнивает ее с порогом, чтобы определить свое собственное состояние.

Аппаратная реализация ИНС на основе пороговых элементов, оперирующих двоичными числами, оказалась чрезвычайно

трудной из-за высокой стоимости электронных элементов в то время. Самые совершенные системы тогда содержали лишь сотни нейронов, в то время как нервная система муравья содержит более 20 тыс.

Серьезное развитие нейрокибернетики получила в трудах американского нейрофизиолога Ф. Розенблата, который предложил свою модель нейронной сети в 1958 г. и продемонстрировал созданное на ее основе электронное устройство, названное *перцептроном* [8]. Ф. Розенблат ввел возможность модификации межнейронных связей, что сделало ИНС обучаемой. Первые перцептроны были способны распознавать некоторые буквы латинского алфавита. Впоследствии модель перцептрона была значительно усовершенствована, а наиболее удачным ее применением стали задачи автоматической классификации.

Алгоритм обучения перцептрона включает следующие шаги.

1. Системе предъявляется эталонный образ.
2. Если результат распознавания совпадает с заданным, весовые коэффициенты связей не изменяются.
3. Если ИНС неправильно распознает результат, то весовым коэффициентам дается приращение в сторону повышения качества распознавания.

Теоретический анализ перцептрона, проведенный М. Минским и С. Пейпертом [4], показал его ограниченные возможности, поскольку не всегда существует такая комбинация весовых коэффициентов, при которой заданное множество образов будет распознаваться правильно. Причина этого недостатка состоит в том, что однослойный перцептрон реализует линейную поверхность, разделяющую пространство эталонов, вследствие чего происходит неверное распознавание образов в случаях, когда задача не является линейно сепарабельной. Для решения таких проблем предложены модели многослойных перцептронов, способные строить ломаную границу между распознаваемыми образами. Несмотря на то что перцептрон Розенблата имел невысокие возможности обучения, разработка этой концепции привлекла внимание исследователей к проблеме ИНС и привела к созданию более «разумных» интеллектуальных систем.

Многослойные сети. В многослойных сетях устанавливаются связи только между нейронами соседних слоев, как показано на рис. 5.3. Каждый элемент может быть соединен модифицируемой

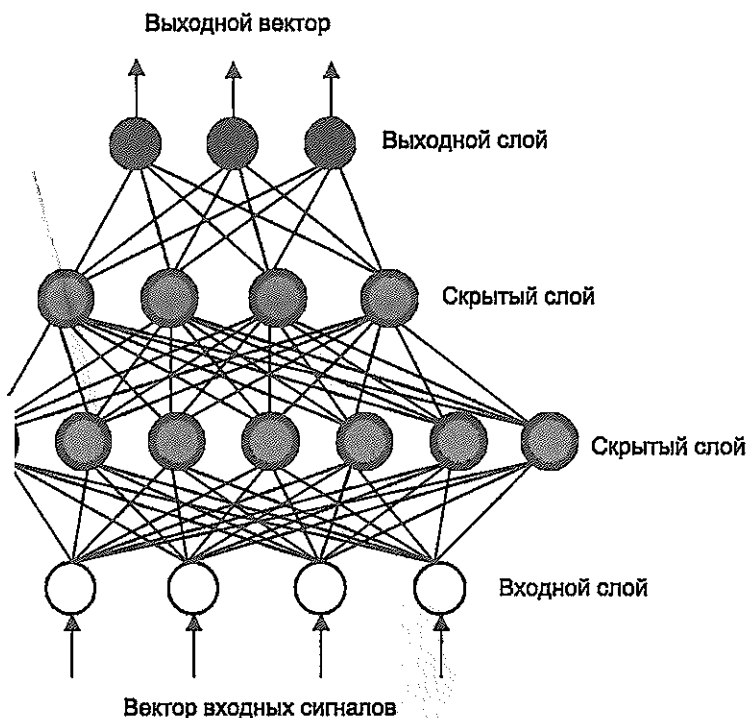


Рис. 5.3. Схема многослойного перцептрона

связью с любым нейроном соседних слоев, но между элементами одного слоя связей нет. Каждый нейрон может посылать выходной сигнал только в вышележащий слой и принимать входные сигналы только с нижерасположенного слоя. Входные сигналы подаются на нижний слой, а выходной вектор сигналов определяется путем последовательного вычисления уровней активности элементов каждого слоя (снизу вверх) с использованием уже известных значений активности элементов предшествующих слоев. При распознавании образов входной вектор соответствует набору признаков, а выходной — распознаваемым образам. Скрытый слой (один или несколько) предназначен для отражения специфики знаний. В таких сетях обычно используются передаточные сигмоидальные функции.

Структура нейронной сети определяется типом, например 25–10–5, т.е. двадцать пять узлов находится в первом слое, де-

сять — в скрытом и пять — в выходном. Определение числа скрытых слоев и числа нейронов в каждом слое для конкретной задачи является неформальной проблемой, при решении которой можно использовать эвристическое правило: *число нейронов в следующем слое в два раза меньше, чем в предыдущем* [10, 14].

Выше отмечалось, что простой перцептрон с одним слоем обучаемых связей формирует границы областей решений в виде гиперплоскостей. Двухслойный перцептрон может выполнять операцию логического И над полупространствами, образованными гиперплоскостями первого слоя весов. Это позволяет формировать любые выпуклые области в пространстве входных сигналов. С помощью трехслойного перцептрона, используя логическое ИЛИ для комбинирования выпуклых областей, можно получить области решений произвольной формы и сложности, в том числе невыпуклые и несвязные. То, что многослойные перцептроны с достаточным множеством внутренних нейроподобных элементов и соответствующей матрицей связей в принципе способны осуществлять любое отображение вход-выход, отмечали еще М. Минский и С. Пейперт, однако они сомневались, что для таких процедур можно открыть мощный аналог процедуры обучения простого перцептрона. В настоящее время в результате возрождения интереса к многослойным сетям предложено несколько таких процедур. Одной из них является алгоритм обратного распространения ошибки, который будет рассмотрен ниже.

Рекуррентные сети. Они содержат обратные связи, благодаря которым становится возможным получение отличающихся значений выходов при одних и тех же входных данных. Наличие рекуррентных нейронов позволяет ИНС накапливать знания в процессе обучения.

Рекуррентные сети (рис. 5.4) являются развитием модели Хопфилда на основе применения новых алгоритмов обучения, исключающих попадание системы в локальные минимумы на поверхности энергетических состояний. Важной особенностью рекуррентных сетей является их способность предсказывать существование новых классов объектов.

Модель Хопфилда. Работы американского биофизика Дж. Хопфилда положили начало современному математическому моделированию нейронных вычислений [11]. Ему удалось привлечь к анализу нейросетевых моделей мощный математический аппарат статистической физики. В результате была сформулиро-

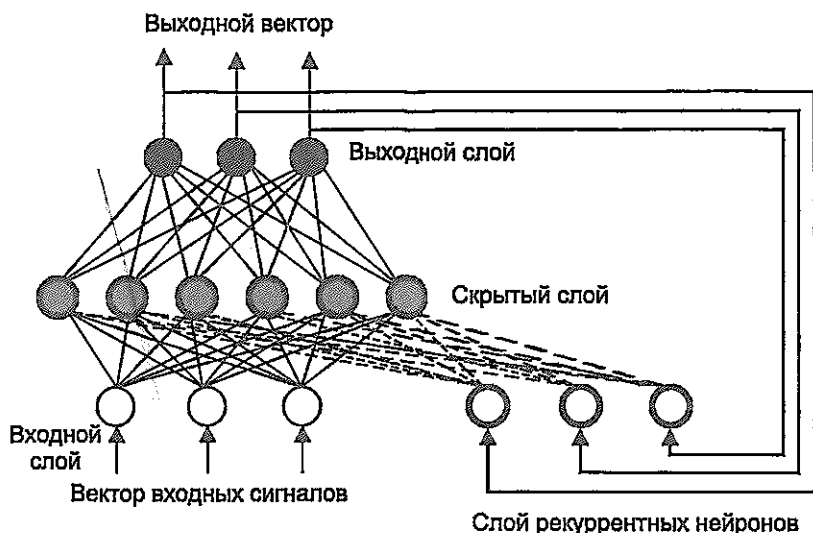


Рис. 5.4. Схема рекуррентной нейронной сети

вана математическая модель ассоциативной памяти на нейронной сети с использованием правила Д. Хейбба для модификации весовых коэффициентов. Это правило основано на простом предположении: если два нейрона возбуждаются вместе, то сила связи между ними возрастает; если они возбуждаются порознь, то сила связи между ними уменьшается.

Сеть Хопфилда строится с учетом следующих условий:

- все элементы связаны со всеми;
- $w_{ji} = w_{ij}$ — прямые и обратные связи симметричны;
- $w_{ii} = 0$ — диагональные элементы матрицы связей равны нулю, т. е. исключаются обратные связи с выхода на вход одного нейрона.

Для однослойной нейронной сети со связями типа «все ко всем» характерна сходимость к одной из конечного множества равновесных точек, которые являются локальными минимумами функции энергии, отражающей структуру всех связей в сети. Введенная Хопфилдом функция вычислительной энергии нейронной сети описывает поведение сети через стремление к минимуму энергии, который соответствует заданному набору образов. В связи с этим сети Хопфилда могут выполнять функции ассоциативной памяти, обеспечивая сходимость к тому образу, в область

притяжения которого попадает начальный паттерн (образец) активности нейронов сети.

Этот подход привлекателен тем, что нейронная сеть для конкретной задачи может быть запрограммирована без обучающих итераций. Веса связей вычисляются на основе вида функции энергии, сконструированной для решаемой задачи.

Развитием модели Хопфилда является машина Больцмана, предложенная и исследованная Дж. Е. Хинтоном и Р. Земелом [5, 7, 12] для решения комбинаторных оптимизационных задач и задач искусственного интеллекта. В ней, как и в других моделях, нейрон имеет состояния $(1, 0)$, межнейронные связи представлены весовыми коэффициентами, а каждое состояние сети характеризуется определенным значением функции консенсуса (аналог функции энергии). Максимум функции консенсуса соответствует оптимальному решению задачи.

Сети Хопфилда получили применение на практике в основном как реализации подсистем более сложных систем. Они имеют определенные недостатки, ограничивающие возможности их применения:

- предположение о симметрии связей между элементами, без которой нельзя ввести понятие энергии;
- нейронная сеть — это устройство для запоминания и обработки информации, а не устройство минимизации энергии. Экономия энергии играет в этих процессах вспомогательную роль;
- сети Хопфилда поддерживают множество лишних, неэффективных, иногда дублирующих друг друга связей. В реальных нервных системах такие связи не поддерживаются, так как их реализация требует определенных затрат. В биологических нервных системах происходит освобождение от лишних связей за счет их структуризации. При этом вместо организации связей «всех ко всем» используется многослойная иерархическая система связей.

Самоорганизующиеся сети Т. Кохонена [15]. Идея сетей с самоорганизацией на основе конкуренции между нейронами базируется на применении специальных алгоритмов самообучения ИНС. Сети Кохонена обычно содержат один (выходной) слой обрабатывающих элементов с пороговой передаточной функцией. Число нейронов в выходном слое соответствует количеству распознаваемых классов. Настройка параметров межнейронных соединений проводится автоматически на основе меры близости

вектора весовых коэффициентов настраиваемых связей к вектору входных сигналов в евклидовом пространстве. В конкурентной борьбе побеждает нейрон, имеющий значения весов, наиболее близкие к нормализованному вектору входных сигналов. Кроме того, в самоорганизующихся сетях возможна классификация входных образцов (паттернов). На практике идея Кохонена обычно используется в комбинации с другими нейросетевыми парадигмами.

5.3. ПОСТРОЕНИЕ НЕЙРОННОЙ СЕТИ

При построении модели ИНС прежде всего необходимо точно определить задачи, которые будут решаться с ее помощью. В настоящее время нейросетевые технологии успешно применяются для прогнозирования, распознавания и обобщения.

Первым этапом построения нейросетевой модели является тщательный отбор входных данных, влияющих на ожидаемый результат. Из исходной информации необходимо исключить все сведения, не относящиеся к исследуемой проблеме. В то же время следует располагать достаточным количеством примеров для обучения ИНС. Существует эмпирическое правило, которое устанавливает рекомендуемое соотношение λ между количеством обучающих примеров, содержащих входные данные и правильные ответы, и числом соединений в нейронной сети: $\lambda \leq 10$.

Для факторов, которые включаются в обучающую выборку, целесообразно предварительно оценить их значимость, проведя корреляционный и регрессионный анализ, и проанализировать диапазоны их возможных изменений.

На *втором* этапе осуществляется преобразование исходных данных с учетом характера и типа проблемы, отображаемой нейросетевой моделью, и выбираются способы представления информации. Эффективность нейросетевой модели повышается, если диапазоны изменения входных и выходных величин приведены к некоторому стандарту, например $[0,1]$ или $[-1,1]$.

Третий этап заключается в конструировании ИНС, т.е. в проектировании ее архитектуры (число слоев и число нейронов в каждом слое). Структура ИНС формируется до начала обучения, поэтому успешное решение этой проблемы во многом оп-

ределяется опытом и искусством аналитика, проводящего исследования.

Четвертый этап связан с обучением сети, которое может проводиться на основе конструктивного или деструктивного подхода. В соответствии с первым подходом обучение ИНС начинается на сети небольшого размера, который постепенно увеличивается до достижения требуемой точности по результатам тестирования. Деструктивный подход базируется на принципе «прореживания дерева», в соответствии с которым из сети с заведомо избыточным объемом постепенно удаляют «лишние» нейроны и примыкающие к ним связи. Этот подход дает возможность исследовать влияние удаленных связей на точность сети. Процесс обучения нейронной сети представляет собой уточнение значений весовых коэффициентов w_{ij} для отдельных узлов на основе постепенного увеличения объема входной и выходной информации. Началу обучения должна предшествовать процедура выбора функции активации нейронов, учитывающая характер решаемой задачи. В частности, в трехслойных перцептронах на нейронах скрытого слоя применяется в большинстве случаев логистическая функция, а тип передаточной функции нейронов выходного слоя определяется на основе анализа результатов вычислительных экспериментов на сети. Индикатором обучаемости ИНС может служить гистограмма значений межнейронных связей [13].

На *пятом* этапе проводится тестирование полученной модели ИНС на независимой выборке примеров.

5.4. ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ

Важнейшим свойством нейронных сетей является их способность к обучению, что делает нейросетевые модели незаменимыми при решении задач, для которых алгоритмизация является невозможной проблематичной или слишком трудоемкой. Обучение нейронной сети заключается в изменении внутренних параметров модели таким образом, чтобы на выходе ИНС генерировался вектор значений, совпадающий с результатами примеров обучающей выборки. Изменение параметров нейросетевой модели может выполняться разными способами в соответствии с различными ал-

горитмами обучения. Парадигма обучения определяется доступностью необходимой информации. Выделяют три парадигмы:

- обучение с учителем (контролируемое);
- обучение без учителя (неконтролируемое);
- смешанное обучение.

При обучении с учителем все примеры обучающей выборки содержат правильные ответы (выходы), соответствующие исходным данным (входам). В процессе контролируемого обучения синаптические веса настраиваются так, чтобы сеть порождала ответы, наиболее близкие к правильным.

Обучение без учителя используется, когда не для всех примеров обучающей выборки известны правильные ответы. В этом случае предпринимаются попытки определения внутренней структуры поступающих в сеть данных с целью распределить образцы по категориям (модели Кохонена).

При смешанном обучении часть весов определяется посредством обучения с учителем, а другая часть получается с помощью алгоритмов самообучения.

Обучение по примерам характеризуется тремя основными свойствами: емкостью, сложностью образов и вычислительной сложностью. Емкость соответствует количеству образов, которые может запомнить сеть. Сложность образов определяет способности нейронной сети к обучению. В частности, при обучении ИНС могут возникать состояния «перетренировки», в которых сеть хорошо функционирует на примерах обучающей выборки, но не справляется с новыми примерами, утрачивая способность обучаться.

Рассмотрим известные правила обучения ИНС.

Правило коррекции по ошибке. Процесс обучения ИНС состоит в коррекции исходных значений весовых коэффициентов межнейронных связей, которые обычно задаются случайным образом. При вводе входных данных запоминаемого примера (стимула) появляется реакция, которая передается от одного слоя нейронов к другому, достигая последнего слоя, где вычисляется результат. Разность между известным значением результата и реакцией сети соответствует величине ошибки, которая может использоваться для корректировки весов межнейронных связей. Корректировка заключается в небольшом (обычно менее 1%) увеличении синаптического веса тех связей, которые усиливают правильные реакции, и уменьшении тех, которые способствуют

ошибочным. Это простейшее правило контролируемого обучения (дельта-правило) используется в однослойных сетях с одним уровнем настраиваемых связей между множеством входов и множеством выходов. При этом на каждом k -м шаге для j -го нейрона вес i -й связи вычисляется по формуле $w_{jik} = w_{ji(k-1)} + \Delta w_{jik}$, где $\Delta w_{jik} = \eta \delta_{jk} x_{jik}$, $\delta_{jk} = T_{jk} - R_{jk}$, T_{jk} — известное (правильное) значение выхода j -го нейрона; R_{jk} — рассчитанное значение выхода j -го нейрона; x_{jik} — величина сигнала на i -м входе, η — коэффициент скорости обучения.

Оптимальные значения весов межнейронных соединений можно определить путем минимизации среднеквадратичной ошибки с использованием детерминированных или псевдослучайных алгоритмов поиска экстремума в пространстве весовых коэффициентов. При этом возникает традиционная проблема оптимизации, связанная с попаданием в локальный минимум.

Правило Хебба [7]. Оно базируется на следующем нейрофизиологическом наблюдении: если нейроны по обе стороны синапса активизируются одновременно и регулярно, то сила их синаптической связи возрастает. При этом изменение веса каждой межнейронной связи зависит только от активности нейронов, образующих синапс. Это существенно упрощает реализацию алгоритмов обучения.

Обучение методом соревнования. В отличие от правила Хебба, где множество выходных нейронов может возбуждаться одновременно, в данном случае выходные нейроны соревнуются (конкурируют) между собой за активизацию. В процессе соревновательного обучения осуществляется модификация весов связей выигравшего нейрона и нейронов, расположенных в его окрестности («победитель забирает все»).

Метод обратного распространения ошибки. Он является обобщением процедуры обучения простого перцептрона с использованием дельта-правила на многослойные сети [2, 6, 10]. В данном методе необходимо располагать обучающей выборкой, содержащей «правильные ответы», т.е. выборка должна включать множество пар образцов входных и выходных данных, между которыми нужно установить соответствие. Перед началом обучения межнейронным связям присваиваются небольшие случайные значения. Каждый шаг обучающей процедуры состоит из двух фаз. Во время первой фазы входные элементы сети устанавливаются в за-

данное состояние. Входные сигналы распространяются по сети, порождая некоторый выходной вектор. Для работы алгоритма требуется, чтобы характеристика вход-выход нейроподобных элементов была неубывающей и имела ограниченную производную. Обычно для этого используют сигмоидальные функции. Полученный выходной вектор сравнивается с требуемым (правильным). Если они совпадают, то весовые коэффициенты связей не изменяются. В противном случае вычисляется разница между фактическими и требуемыми выходными значениями, которая передается последовательно от выходного слоя к входному. На основе этой информации проводится модификация связей в соответствии с обобщенным дельта-правилом, которое имеет вид: $\Delta_p w_{ji} = \eta \delta_{jp} u_{ip}$, где изменение в силе связи w_{ji} для p -й обучающей пары $\Delta_p w_{ji}$ пропорционально произведению сигнала ошибки j -го нейрона δ_{jp} , получающего входной сигнал по этой связи, и выходного сигнала i -го нейрона u_{ip} , посылающего сигнал по этой связи. Определение сигнала ошибки является рекурсивным процессом, который начинается с выходных блоков. Для выходного блока сигнал ошибки $\delta_{jp} = y'_j (T_{jp} - R_{jp})$, где T_{jp} и R_{jp} — соответственно желаемое и действительное значения выходного сигнала j -го блока; y'_j — производная от выходного сигнала j -го блока. Сигнал ошибки для скрытого блока определяется рекурсивно через сигнал ошибки блоков, с которым соединен его выход, и веса этих связей равны $\delta_{jp} = y'_j \sum_k \delta_{kp} w_{kj}$. Для сигмоидальной функции

$y'_j = y_j(1-y_j)$, поэтому на интервале $0 \leq y_j \leq 1$ производная имеет максимальное значение в точке $y_j = 0.5$, а в точках $y_j = 0$ и $y_j = 1$ обращается в ноль. Максимальные изменения весов соответствуют блокам (нейронам), которые еще не выбрали свое состояние. Кроме того, при конечных значениях весовых коэффициентов выходные сигналы блоков не могут достигать значений 0 или 1. Поэтому за 0 обычно принимают значения $y_j < 0.1$, а за 1 — значения $y_j > 0.9$.

Модификация весов производится после предъявления каждой пары вход-выход. Однако если коэффициент η , определяющий скорость обучения, мал, то можно показать, что обобщенное дельта-правило достаточно хорошо аппроксимирует минимизацию общей ошибки функционирования сети D методом гради-

ентного спуска в пространстве весов. Общая ошибка функционирования сети определяется по формуле

$$D = \frac{1}{2} \sum_p \sum_j (T_{jp} - R_{jp})^2.$$

Обучение продолжается до тех пор, пока ошибка не уменьшится до заданной величины. Эмпирические результаты свидетельствуют о том, что при малых значениях η система находит достаточно хороший минимум D . Один из основных недостатков алгоритма обратного распространения ошибки заключается в том, что во многих случаях для сходимости может потребоваться многократное (сотни раз) предъявление всей обучающей выборки. Повышения скорости обучения можно добиться, например, используя информацию о второй производной D или путем увеличения η .

Алгоритм обратного распространения ошибки используется также для обучения сетей с обратными связями. При этом используется эквивалентность многослойной сети с прямыми связями и синхронной сети с обратными связями на ограниченном интервале времени (слой соответствует такту времени).

В настоящее время предложены алгоритмы обучения, более привлекательные в смысле биологической аналогии. Примером является алгоритм рециркуляции для сетей, в которых скрытые блоки соединены с входными. При обучении веса связей перестраиваются таким образом, чтобы минимизировать частоту смены активности каждого блока. Таким образом, обученная сеть имеет стабильные состояния и может функционировать в режиме ассоциативной памяти.

5.5. СПОСОБЫ РЕАЛИЗАЦИИ НЕЙРОННЫХ СЕТЕЙ

Нейронные сети могут быть реализованы программным или аппаратным способом.

Вариантами *аппаратной* реализации являются нейрокомпьютеры, нейроплаты и нейроБИС (большие интегральные схемы). Одна из самых простых и дешевых нейроБИС — модель MD 1220 фирмы Micro Devices, которая реализует сеть с 8 нейронами и 120

синапсами. Среди перспективных разработок можно выделить модели фирмы Adaptive Solutions (США) и Hitachi (Япония). Разрабатываемая фирмой Adaptive Solutions нейроБИС является одной из самых быстродействующих: объявленная скорость обработки составляет 1,2 млрд межнейронных соединений в секунду (мнс/с). Схемы, производимые фирмой Hitachi, позволяют реализовывать ИНС, содержащие до 576 нейронов.

Большинство современных нейрокомпьютеров представляют собой персональный компьютер или рабочую станцию, в состав которых входит дополнительная нейроплата. К их числу относятся, например, компьютеры серии FMR фирмы Fujitsu. Возможностей таких систем вполне хватает для решения большого числа прикладных задач методами нейроматематики, а также для разработки новых алгоритмов. Наибольший интерес представляют специализированные нейрокомпьютеры, в которых реализованы принципы архитектуры нейросетей. Типичными представителями таких систем являются компьютеры семейства Mark фирмы TRW (первая реализация перцептрона, разработанная Ф. Розенблатом, называлась Mark I). Модель Mark III фирмы TRW представляет собой рабочую станцию, содержащую до 15 процессоров семейства Motorola 68000 с математическими сопроцессорами. Все процессоры объединены шиной VME. Архитектура системы, поддерживающая до 65 000 виртуальных процессорных элементов с более чем 1 млн настраиваемых соединений, позволяет обрабатывать до 450 тыс. мнс/с.

Другим примером является нейрокомпьютер NETSIM, созданный фирмой Texas Instruments на базе разработок Кембриджского университета. Его топология представляет собой трехмерную решетку стандартных вычислительных узлов на базе процессоров 80188. Компьютер NETSIM используется для моделирования сетей Хопфилда—Кохонена. Его производительность достигает 450 млн мнс/с.

В тех случаях, когда разработка или внедрение аппаратных реализаций нейронных сетей обходятся слишком дорого, применяют более дешевые *программные реализации*. Одним из самых распространенных программных продуктов является семейство программ *BrainMaker* фирмы CSS (California Scientific Software). Первоначально разработанный фирмой Loral Space Systems по заказу NASA и Johnson Space Center пакет *BrainMaker* был вскоре адаптирован для коммерческих приложений и сегодня использу-

ется несколькими тысячами финансовых и промышленных компаний, а также оборонными ведомствами США для решения задач прогнозирования, оптимизации и моделирования ситуаций.

Назначение пакета *BrainMaker* — решение задач, для которых пока не найдены формальные методы и алгоритмы, а входные данные неполны, зашумлены и противоречивы. К таким задачам относятся прогнозирование курсов валют и акций на биржах, моделирование кризисных ситуаций, распознавание образов и многие другие. *BrainMaker* решает поставленную задачу, используя математический аппарат теории нейронных сетей (более конкретно — сеть Хопфилда с обучением по методу обратного распространения ошибки). В оперативной памяти строится модель многослойной нейронной сети, которая обладает свойством обучаться на множестве примеров, оптимизируя свою внутреннюю структуру. При правильном выборе структуры сети после ее обучения на достаточно большом количестве примеров можно добиться высокой достоверности результатов (97% и выше). Существуют версии *BrainMaker* для MS DOS и MS Windows, а также для Apple Macintosh. Кроме базовой версии пакета в семейство *BrainMaker* входят следующие дополнения:

- *BrainMaker Student* — версия пакета для университетов. Она особенно популярна у небольших фирм, специализирующихся на создании приложений для не очень сложных задач.

- *Toolkit Option* — набор из трех дополнительных программ, увеличивающих возможности *BrainMaker*. *Binary*, которая переводит обучающую информацию в двоичный формат для ускорения обучения; *Hypersonic Training*, где используется высокоскоростной алгоритм обучения; *Plotting*, которая отображает факты, статистику и другие данные в графическом виде.

- *BrainMaker Professional* — профессиональная версия пакета *BrainMaker* с расширенными функциональными возможностями. Включает в себя все опции *Toolkit*.

- *Genetic Training Option* (для *BrainMaker Pro*) — программа автоматической оптимизации нейронной сети для решения заданного класса задач, использующая генетические алгоритмы для селекции наилучших решений.

- *DataMaker Editor* — специализированный редактор для автоматизации подготовки данных при настройке и использовании нейронной сети.

- *Training Financial Data* — специализированные наборы данных для настройки нейронной сети на различные виды аналити-

ческих, коммерческих и финансовых операций, которые включают реальные значения макроэкономических показателей NYSE, NADDAW, ASE, OEX, DOW и др., индексы инфляции, статистические данные биржевых сводок по различным видам продукции, а также информацию по фьючерсным контрактам и многое другое.

- *BrainMaker Accelerator* — специализированная нейроплата-акселератор на базе сигнальных процессоров TMS320C25 фирмы Texas Instruments. Вставленная в персональный компьютер, она в несколько раз ускоряет работу пакета *BrainMaker*.

- *BrainMaker Accelerator Pro* — профессиональная многопроцессорная нейронная плата. Она содержит пять сигнальных процессоров TMS320C30 и 32 Мбайт оперативной памяти.

В настоящее время на рынке программных средств имеется большое количество разнообразных пакетов для конструирования нейронных сетей и решения различных задач. Пакет *BrainMaker* можно назвать ветераном рынка. Кроме представителей этого семейства, к хорошо известным и распространенным программным средствам можно отнести *NeuroShell* (Ward System's Group), *Neural Works* (Neural Ware Inc.) и *NeuroSolutions* (NeuroDimension Inc.). Объектно-ориентированные программные среды семейства *NeuroSolutions* предназначены для моделирования ИНС произвольной структуры. Пользователю систем *NeuroSolutions* предоставлены возможности исследования и диалогового управления. Все данные в сети доступны для просмотра в процессе обучения посредством разнообразных инструментов визуализации. Проектирование ИНС в системе *NeuroSolutions* основано на модульном принципе, который позволяет моделировать стандартные и новые топологии. Важным преимуществом системы является наличие специальных инструментов, позволяющих моделировать динамические процессы в ИНС.

Рассмотрим один из тестовых примеров, в котором решается достаточно трудная задача прогнозирования значений хаотических временных последовательностей. Для предсказания значений трех членов хаотического ряда Mackey-Glass, который является эталонным тестом, выбрана топология TLRN (Time Lagged Recurrent Network) — рекуррентная сеть с запаздыванием во времени, в которой обратные связи локально ограничены. Подобные сети являются расширением многослойного перцептрона, который снабжен структурами памяти. Они содержат обратные

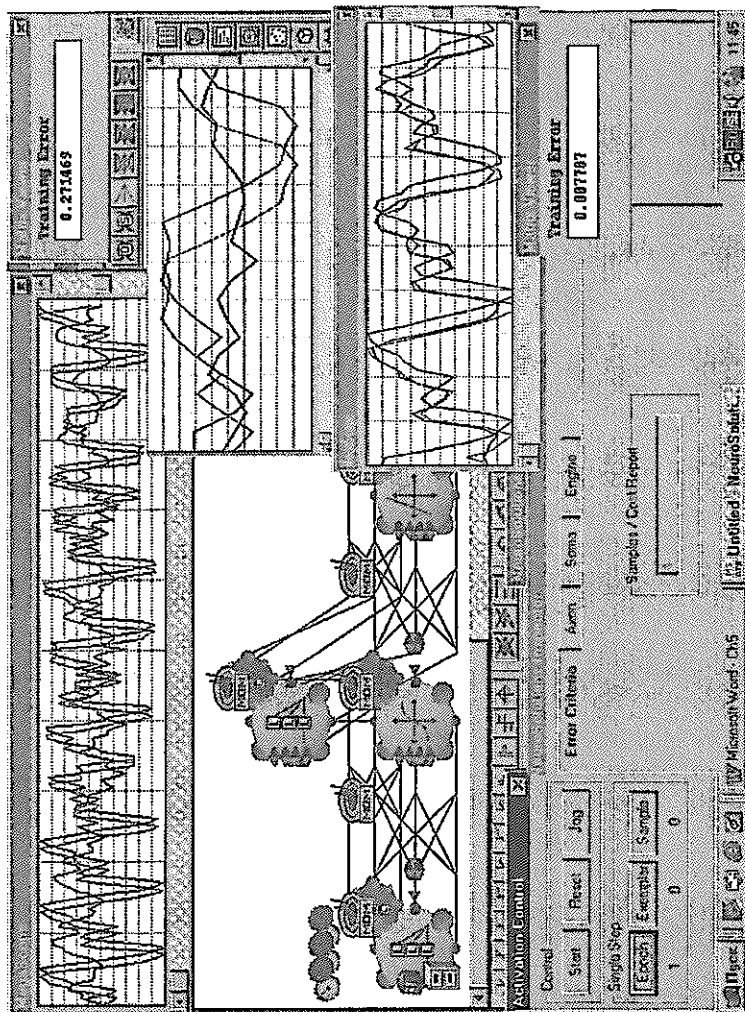


Рис. 5.5. Обучение нейронной сети в системе NeuroSolutions

связи, однако проблема устойчивости здесь легко решается, а поведение таких сетей хорошо объяснимо. Сети с подобной топологией подходят для решения проблем прогнозирования, идентификации и классификации образцов, изменяющейся во времени. Выходной сигнал динамической системы можно предсказать на короткий срок, но любая малая ошибка в таких сетях увеличивается со временем из-за присущего им внутреннего фактора рассогласования, называемого положительной экспонентой Ляпунова. В таких сетях реализована структура памяти Laguette, которая обладает удобным свойством, позволяющим контролировать глубину запоминания с помощью единственного адаптивного параметра. В памяти такого типа может храниться K/u обработанных в прошлом образцов, где K — число выходных сигналов; u — параметр памяти.

Рекуррентная сеть не может быть обучена стандартным методом обратного распространения ошибки. *NeuroSolutions* содержит модификацию этого алгоритма, позволяющую проводить обучение динамической сети. Нейроны выходного слоя обычно имеют линейную передаточную функцию. Такой выбор обусловлен стремлением воспроизвести волновую форму выходного сигнала. Нелинейности могут вызвать получение недопустимых значений в экстремальных точках и привести к ошибкам. Для нейронов входного слоя и скрытых слоев, как правило, подбираются нелинейные активационные функции. В процессе решения задачи определяется объем обучающей и тестирующей выборки, в удобном режиме конструируется архитектура сети (количество скрытых слоев, число нейронов), выбирается вид активационных функций, способ и параметры обучения. Экранные формы, демонстрирующие процесс обучения сети, показаны на рис. 5.5.

5.6. ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ

Применение нейросетевых технологий целесообразно при решении задач, имеющих следующие признаки:

- отсутствие алгоритмов решения задач при наличии достаточно большого числа примеров;

• наличие большого объема входной информации, характеризующей исследуемую проблему;

• зашумленность, частичная противоречивость, неполнота или избыточность исходных данных.

Нейросетевые технологии нашли широкое применение в таких направлениях, как распознавание печатного текста, контроль качества продукции на производстве, идентификация событий в ускорителях частиц, разведка нефти, борьба с наркотиками, медицинские и военные приложения, управление и оптимизация, финансовый анализ, прогнозирование и др.

В сфере экономики нейросетевые технологии могут использоваться для классификации и анализа временных рядов путем аппроксимации сложных нелинейных функций. Экспериментально установлено, что модели нейронных сетей обеспечивают большую точность при выявлении нелинейных закономерностей на фондовом рынке по сравнению с регрессионными моделями [13].

Рассмотрим решение задачи прогнозирования *цены закрытия на завтра* по акциям некоторого предприятия *X*. Для моделирования воспользуемся данными наблюдений за месяц. В качестве исходных данных можно использовать индикаторы Dow Jones, NIKKEI, FTSE100, индексы и акции российских компаний, «сезонные» переменные и др.

Относительный показатель однодневной доходности предприятия можно определить из соотношений:

$$R_i = \begin{cases} \Delta P_i; \\ -\Delta P_i; \\ \Delta P_i = P_i - P_{i-1}, \end{cases}$$

где ΔP_i — оценка операции «вчера купил, сегодня продал»;

— ΔP_i — оценка операции «вчера продал, сегодня купил»;

P_i — значение выбранного показателя доходности в i -й день;

P_{i-1} — значение показателя в $(i-1)$ -й день.

Итоговая доходность за установленный интервал времени (n дней) рассчитывается по формуле

$$R = \sum_{i=1}^n (1 + \Delta P_i) - 1.$$

Результаты оценки доходности предприятия с использованием различных моделей ИНС, а также доходов «идеального» трейдера приведены ниже.

Индикаторы	Доходность за 30 дней
Стандартная трехслойная сеть	0,1919
Стандартная четырехслойная сеть	-0,1182
Рекуррентная сеть с обратной отрицательной связью от скрытого слоя.....	0,1378
Рекуррентная сеть с отрицательной обратной связью	0,4545
Сеть Ворда: с тремя скрытыми блоками, с разными передаточными функциями.....	0,2656
Трехслойная сеть с обходным соединением	-0,1889
Четырехслойная сеть с обходными соединениями	0,0003
Сеть с общей регрессией	0,3835
Сеть метода группового учета аргументов	0,1065
Сеть Ворда: с тремя скрытыми блоками, с разными передаточными функциями, с обходным соединением.....	-0,1166
«Идеальный» трейдер	1,1448

«Идеальный трейдер» знает цену закрытия на следующий день и поэтому получает максимально возможную прибыль. Трейдер пользуется значением нейросетевого индикатора следующим образом: на основе прогнозируемого в $(i - 1)$ -й день значения ΔP_i (величина относительно изменения цены закрытия по акциям рассматриваемого предприятия X на завтрашний i -й день) трейдер принимает решение о покупке ($\Delta P_i > 0$) или продаже ($\Delta P_i < 0$) акций.

Анализ результатов моделирования показывает, что лучшую доходность обеспечила рекуррентная сеть с отрицательной обратной связью (45% за 30 дней). Динамика изменения однодневных показателей доходности, полученных с помощью этой ИНС, приведена на рис 5.6.

Нейросетевые технологии активно используются в маркетинге для моделирования поведения клиентов и распределения долей рынка. Нейросетевые технологии позволяют отыскивать в маркетинговых базах данных скрытые закономерности (см. разд. 4.5).

Моделирование поведения клиентов позволяет определить характеристики людей, которые будут нужным образом реагиро-

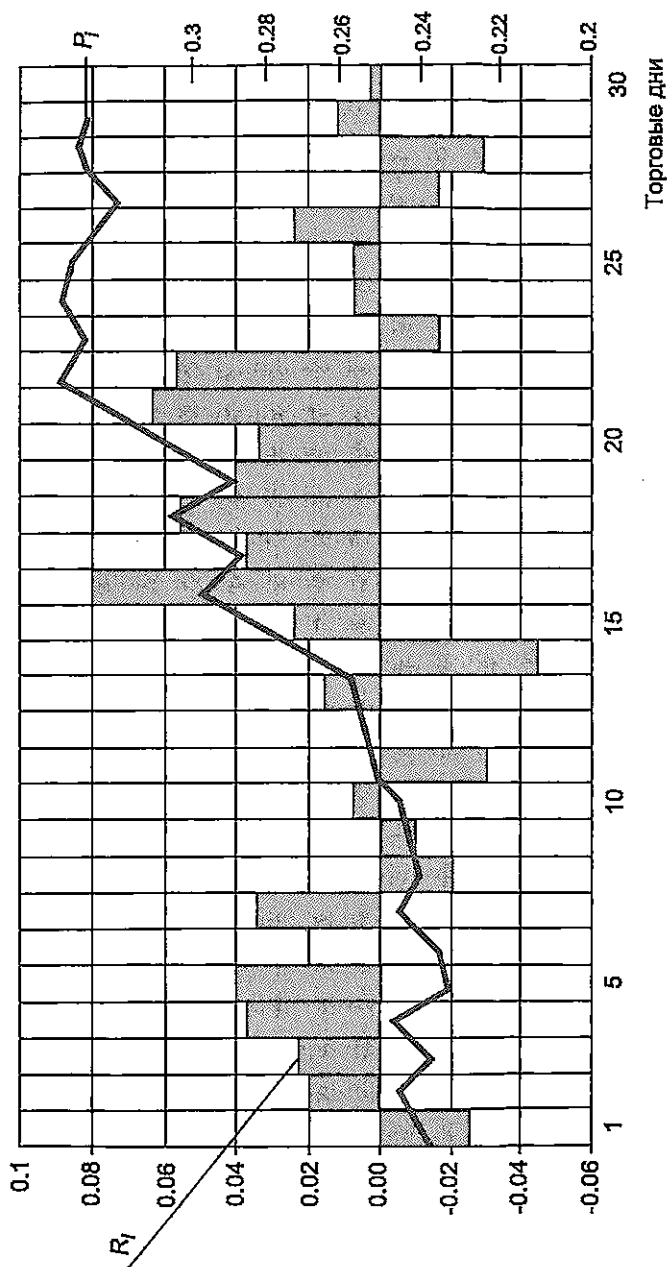


Рис. 5.6. Динамика изменения однодневных доходности (R_i) и цен закрытия (P_i) за 30 торговых дней, полученная на рекуррентной сети с отрицательной обратной связью

вать на рекламу и совершать покупки определенного товара или услуги.

Сегментирование и моделирование рынков на основе нейросетевых технологий дает возможность построения гибких классификационных систем, способных осуществлять сегментирование рынков с учетом многообразия факторов и особенностей каждого клиента.

Технологии ИНС имеют хорошие перспективы при решении задач имитации и предсказания поведенческих характеристик менеджеров и задач прогнозирования рисков при выдаче кредитов. Не менее актуально применение ИНС при выборе клиентов для ипотечного кредитования, предсказания банкротства клиентов банка, определения мошеннических сделок при использовании кредитных карточек, составления рейтингов клиентов при займах с фиксированными платежами и т.п.

Следует помнить о том, что применение нейросетевых технологий не всегда возможно и сопряжено с определенными проблемами и недостатками.

1. Необходимо как минимум 50, а лучше 100 наблюдений для создания приемлемой модели. Это достаточно большое число данных, и они далеко не всегда доступны. Например, при производстве сезонного товара истории предыдущих сезонов недостаточно для прогноза на текущий сезон из-за изменения стиля продукта, политики продаж и т.д. Даже при прогнозировании спроса на достаточно стабильный продукт на основе информации о ежемесячных продажах трудно накопить исторические данные за период от 50 до 100 месяцев. Для сезонных товаров проблема еще более сложна, так как каждый сезон фактически представляет собой одно наблюдение. При дефиците информации модели ИНС строят в условиях неполных данных, а затем проводят их последовательное уточнение.

2. Построение нейронных сетей требует значительных затрат труда и времени для получения удовлетворительной модели. Необходимо учитывать, что излишне высокая точность, полученная на обучающей выборке, может обернуться неустойчивостью результатов на тестовой выборке — в этом случае происходит «переобучение» сети. Чем лучше система адаптирована к конкретным условиям, тем меньше она способна к обобщению и экстраполяции и тем скорее может оказаться неработоспособной при изменении этих

условий. Расширение объема обучающей выборки позволяет добиться большей устойчивости, но за счет увеличения времени обучения.

3. При обучении нейронных сетей могут возникать «ловушки», связанные с попаданием в локальные минимумы. Детерминированный алгоритм обучения не в силах обнаружить глобальный экстремум или покинуть локальный минимум. Одним из приемов, который позволяет обходить ловушки, является расширение размерности пространства весов за счет увеличения числа нейронов скрытых слоев. Некоторые возможности для решения этой проблемы открывают стохастические методы обучения. При модификации весов сети только на основе информации о направлении вектора градиента целевой функции в пространстве весов можно достичь локального минимума, но невозможно выйти из него, поскольку в точке экстремума «движущая сила» (градиент) обращается в нуль и причина движения исчезает. Чтобы покинуть локальный экстремум и перейти к поиску глобального, нужно создать дополнительную силу, которая будет зависеть не от градиента целевой функции, а от каких-то других факторов. Один из простейших методов состоит в том, чтобы просто создать случайную силу и добавить ее к детерминистической.

4. Сигмоидальный характер передаточной функции нейрона является причиной того, что если в процессе обучения несколько весовых коэффициентов стали слишком большими, то нейрон попадает на горизонтальный участок функции в область насыщения. При этом изменения других весов, даже достаточно большие, практически не сказываются на величине выходного сигнала такого нейрона, а значит, и на величине целевой функции.

5. Неудачный выбор диапазона входных переменных — достаточно элементарная, но часто совершаемая ошибка. Если x_i — двоичная переменная со значениями 0 и 1, то примерно в половине случаев она будет иметь нулевое значение: $x_i = 0$. Поскольку x_i входит в выражение для модификации веса в виде множителя, то эффект будет тот же, что и при насыщении: модификация соответствующих весов будет заблокирована. Правильный диапазон для входных переменных должен быть симметричным, например от +1 до -1 [2, 12].

6. Процесс решения задач нейронной сетью является «непрозрачным» для пользователя, что может вызывать с его стороны недоверие к прогнозирующим способностям сети.

7. Предсказывающая способность сети существенно снижается, если поступающие на вход факты (данные) имеют значительные отличия от примеров, на которых обучалась сеть. Этот недостаток ярко проявляется при решении задач экономического прогнозирования, в частности при определении тенденций котировок ценных бумаг и стоимости валют на фондовых и финансовых рынках.

8. Отсутствуют теоретически обоснованные правила конструирования и эффективного обучения нейронных сетей. Этот недостаток приводит, в частности, к потере нейронными сетями способности обобщать данные предметной области в состояниях переобучения (перетренировки).

Литература

1. Барцев С. И., Охонин В. А. Адаптивные сети обработки информации. — Красноярск: Институт физики СО АН СССР, 1986.
2. Компьютер обретает разум. — М.: Мир, 1990.
3. Маккалох Дж., Питтс У. Логические исчисления идей, относящихся к нервной деятельности // Автоматы. — М.: ИЛ, 1956.
4. Минский М., Пейперт С. Перцептроны. — М.: Мир, 1971.
5. Микртчан С. О. Нейроны и нейронные сети (Введение в теорию формальных нейронов и нейронных сетей). — М.: Энергия, 1971.
6. Осовский С. Нейронные сети для обработки информации / Пер. с польск. И. Д. Рудинского. — М.: Финансы и статистика, 2002.
7. Позин Н. В. Моделирование нейронных структур. — М.: Наука, 1970.
8. Розенблат Ф. Принципы нейродинамики. — М.: Мир, 1965.
9. Соколов Е. Н., Вайтнявичус Г. Г. Нейроинтеллект: от нейрона к нейрокомпьютеру. — М.: Наука, 1989.
10. Трикоз Д. В. Нейронные сети: как это делается? // Компьютеры + программы. — 1993. — № 4(5).
11. Тэнк Д. У., Хопфилд Д. Д. Коллективные вычисления в нейронно-подобных электронных схемах // В мире науки. — 1988. — № 2.
12. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. — М.: Мир, 1992.

13. *Фролов Ю. В.* Интеллектуальные системы и управленческие решения. — М.: Изд-во МГПУ, 2000.
14. *Хинтон Дж. Е.* Как обучаются нейронные сети // В мире науки. — 1992. — № 11 — 12.
15. *Kohonen T.* Self-organization and associative memory. — New York: Springer, 1984.

Контрольные вопросы и задания

1. Опишите модель искусственного нейрона. Приведите примеры передаточных функций.
2. Что такое перцептрон? Какие модели нейронных сетей вам известны?
3. Проведите сравнение однослойных и многослойных ИНС.
4. Какими особенностями обладают рекуррентные и самоорганизующиеся сети? Расскажите о моделях ИНС Хопфилда и Кохонена.
5. Дайте характеристику основных этапов построения нейронной сети.
6. Расскажите о методах обучения ИНС (коррекция по ошибке, обучение Хебба, соревновательное обучение, метод обратного распространения ошибки).
7. Опишите алгоритм обратного распространения ошибки. Сформулируйте его достоинства и недостатки.
8. Расскажите об известных вам способах реализации ИНС.
9. Для каких задач целесообразно применять ИНС? Каковы условия применения моделей этого типа? Сформулируйте основные проблемы, возникающие при применении нейронных сетей.
10. Сформулируйте постановку прикладной задачи экономического характера, для решения которой возможно и целесообразно применить нейронную сеть. Опишите, как это можно сделать.
11. Подготовьте доклад или реферат о конкретной нейросетевой инструментальной системе. Опишите методику решения задач определенного класса при поддержке выбранного программного обеспечения.
12. Подготовьте набор содержательных примеров для обучения нейронной сети с заданной целью.
13. Сформулируйте постановку содержательной задачи для решения методами нейронных сетей. Подготовьте обучающую и тестирующую выборки примеров. Выберите топологию сети, выполни-

те проектирование и реализацию в системе *NeuroSolutions* (другой системе).

14. Сформулируйте постановку задачи извлечения знаний для решения с помощью технологии нейронных сетей. Подготовьте необходимые данные. Выберите топологию сети, выполните проектирование и реализацию в системе *NeuroSolutions* (или другой системе).
15. Составьте задачу классификации (диагностики) для решения с помощью технологии нейронных сетей. Подготовьте необходимые данные. Выберите топологию сети, выполните проектирование и реализацию в системе *NeuroSolutions* (или другой системе).
16. Сформулируйте задачу прогнозирования для решения с помощью технологии нейронных сетей. Подготовьте необходимые данные. Выберите топологию сети, выполните проектирование и реализацию в системе *NeuroSolutions* (или другой системе).

ЭВОЛЮЦИОННЫЕ АНАЛОГИИ В ИСКУССТВЕННЫХ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМАХ

Как великий художник, природа умеет и с небольшими средствами достигать великих эффектов.

Генрих Гейне

Эволюционное моделирование можно определить как воспроизведение процесса естественной эволюции с помощью специальных компьютерных программ. Термин *эволюция* в ограниченном смысле, касающемся только смены поколений организмов, начал широко использоваться в XVII в. С появлением в 1859 г. учения Дарвина этот термин приобрел современное толкование: «Биологическая эволюция — историческое развитие организмов». Необходимые и достаточные условия, определяющие главные факторы эволюции, были сформулированы в XX в. на основе созданной популяционно-генетической теории. К факторам, определяющим неизбежность эволюции, относятся:

- наследственная изменчивость как предпосылка эволюции, ее материал;
- борьба за существование как контролирующий и направляющий фактор;
- естественный отбор как преобразующий фактор.

На рис. 6.1 приведена конкретизация факторов эволюции, учитывающая многообразие форм их проявления, взаимосвязей и взаимовлияния [11]. Главные факторы выделены пунктиром.

Современная теория эволюции базируется на теории общей и популяционной генетики. Элементарным объектом эволюции является популяция — сообщество свободно скрещивающихся особей. В популяциях происходят микроэволюционные процессы, приводящие к изменению их генофонда. Преобразования генетического состава популяции происходят под действием элементарных эволюционных факторов (см. рис. 6.1). Случайные структурные или функциональные изменения в генах, хромосо-

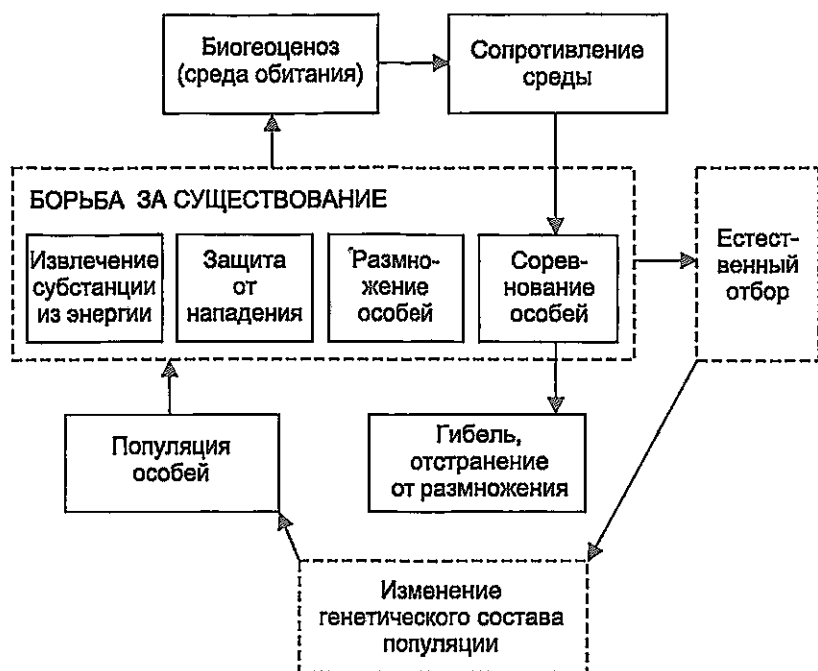


Рис.6.1. Схема взаимодействия факторов эволюции

мах и других воспроизводимых единицах называют *мутациями*, если они приводят к наследственному изменению какого-либо фенотипического признака особи. *Хромосомы* — это специфические структуры клеточного ядра, которые играют важнейшую роль в процессах деления клеток. Хромосомы состоят из генов. *Геном* называется реально существующая, независимая, комбинирующаяся и расщепляющаяся при скрещиваниях единица наследственности.

Преобразования генофонда популяции происходят под управлением естественного отбора.

Эволюция — это многоэтапный процесс возникновения органических форм с более высокой степенью организации, который характеризуется изменчивостью самих эволюционных механизмов.

История эволюционных вычислений началась с разработки ряда независимых моделей, среди которых были генетические алгоритмы и классификационные системы, созданные амери-

канским исследователем Дж. Холландом. Он предложил использовать методы и модели развития органического мира на Земле в качестве механизма комбинаторного перебора вариантов при решении оптимизационных задач [14, 26]. Компьютерные реализации этого механизма получили название «генетические алгоритмы». В 1970-х гг. в рамках теории случайного поиска Л. А. Растрининым был предложен ряд алгоритмов, использующих идеи биологического поведения особей [10]. Развитие этих идей нашло отражение в цикле работ И. Л. Букатовой по эволюционному моделированию [2, 3]. Идеи М. Л. Цетлина, развитые в исследованиях поведения сообществ конечных автоматов, легли в основу алгоритмов поиска глобального экстремума, основанных на моделировании процессов развития и элиминации особей [15]. Большой вклад в развитие эволюционного программирования внесли работы Л. Фогеля, А. Оуэнса и М. Уолша [12, 20, 21].

К основным направлениям развития эволюционного моделирования на современном этапе относятся следующие:

- *генетические алгоритмы* (ГА), предназначенные для оптимизации функций дискретных переменных и использующие аналогии естественных процессов рекомбинации и селекции;

- *классифицирующие системы* (КС), созданные на основе генетических алгоритмов, которые используются как обучаемые системы управления;

- *генетическое программирование* (ГП), основанное на использовании эволюционных методов для оптимизации создаваемых компьютерных программ;

- *эволюционное программирование* (ЭП), ориентированное на оптимизацию непрерывных функций без использования рекомбинаций;

- *эволюционные стратегии* (ЭВС), ориентированные на оптимизацию непрерывных функций с использованием рекомбинаций.

Эволюционные методы целесообразно использовать в тех случаях, когда прикладную задачу сложно сформулировать в виде, позволяющем найти аналитическое решение, или тогда, когда требуется быстро найти приближенный результат, например, при управлении системами в реальном времени.

В России развитием эволюционных методов занимаются научные школы профессоров И. Л. Букатовой [2, 3], Д. И. Батищева [1], В. М. Курейчика [7–9] и И. П. Норенкова [6].

6.1. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

В основе генетических алгоритмов лежат генетика и хромосомная теория эволюции организмов. *Хромосомы* — это нитевидные структуры, находящиеся в клеточном ядре, которые являются носителями наследственности. Каждая хромосома уникальна морфологически и генетически и не может быть заменена другой либо восстановлена при утере (при потере хромосомы клетка, как правило, погибает). Каждый биологический вид имеет определенное, постоянное количество хромосом. Каждая клетка содержит удвоенный набор морфологически и генетически сходных хромосом. Например, в клетках человека содержится 23 пары хромосом, в клетках комара — 3.

На процесс наследования признаков существенно влияет поведение хромосом при делении клеток. Существует митозное и мейозное деление клеток. *Митозное деление* обеспечивает распределение исходных хромосом между двумя образующимися дочерними клетками, которые будут иметь равноценные наборы хромосом и будут очень похожи друг на друга. При этом происходит редупликация исходных хромосом, вследствие чего к моменту деления клетки каждая хромосома состоит из двух копий исходной материнской хромосомы — сестринских *хроматид* (рис. 6.2).

Во время *мейоза* происходит два последовательных деления: редукционное и эквационное. Мейоз приводит к образованию клеток, у которых число хромосом вдвое меньше по сравнению с исходной клеткой.

В фазе *редукции* хроматиды обмениваются генами, т. е. участками дезоксирибонуклеиновой кислоты (ДНК). После этого клетка разделяется на две новые, причем каждая из них содержит удвоенный набор хромосом, структуры которых отличаются от исходных. Механизм обмена генами называется *кроссинговером*.

В результате *эквационного* деления из двух получившихся клеток образуются четыре клетки, каждая из которых содержит одинаковый набор хромосом (см. рис. 6.2).

Таким образом, митоз обеспечивает возобновление клеток, а мейоз отвечает за передачу наследственной информации и способствует генетическому разнообразию организмов данного вида.

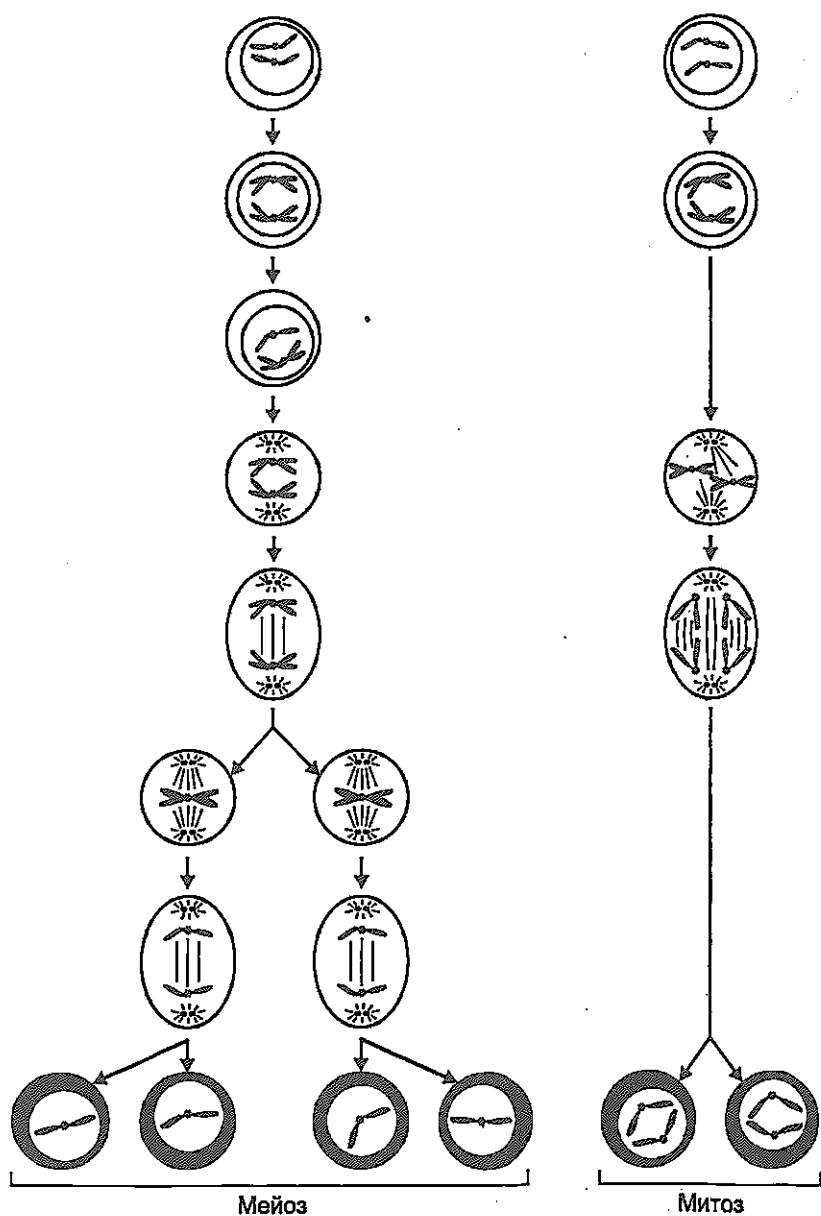


Рис. 6.2. Механизмы деления клеток

Классическая генетика обосновала наследственность и изменчивость благодаря созданию фундаментальной теории гена, основные положения которой формулируются следующим образом:

- все признаки организма определяются наборами генов;
- гены — это элементарные единицы наследственной информации, которые находятся в хромосомах;
- гены могут изменяться — мутировать;
- мутации отдельных генов приводят к изменению отдельных элементарных признаков организма, или фенотипов.

Ген определяется как структурная единица наследственной информации, далее неделимая в функциональном отношении. Он представляет собой участок молекулы ДНК, на котором сохраняется постоянный порядок следования пар нуклеотидов. Комплекс генов, содержащихся в наборе хромосом одного организма, образует *геном*. Роль молекул ДНК, обладающих уникальной способностью к самовоспроизведению, заключается в хранении и передаче генетической информации последующим поколениям.

В задачах поиска оптимальных решений каждое решение из множества возможных можно представить набором информации, который может быть изменен путем введения в него элементов другого решения. Другими словами, возможные решения соответствуют хромосомам, состоящим из генов, причем в ходе оптимизации происходит обмен генами между хромосомами (рекомбинация). При построении генетических алгоритмов важен выбор принципа генетической рекомбинации. Существует несколько типов перераспределения наследственных факторов:

- 1) рекомбинация хромосомных и нехромосомных генов;
- 2) рекомбинация целевых негомологических хромосом;
- 3) рекомбинация участков хромосом, представленных непрерывными молекулами ДНК.

Для построения генетических алгоритмов наибольший интерес представляет третий тип рекомбинации, который используется для накопления в конечном решении лучших функциональных признаков, какие имелись в наборе исходных решений. Существует несколько типов рекомбинации участков хромосом: кроссинговер, сайт, иллегальная рекомбинация.

Кроссинговер соответствует регулярной рекомбинации, при которой происходит обмен определенными участками между го-

мологичными хромосомами. Он приводит к появлению нового сочетания сцепленных генов.

Сайт — это вид рекомбинации, при которой на коротких специализированных участках хромосом происходит обмен *генофоров* (генных носителей), часто различных по объему и составу генетической информации.

Иллегальная рекомбинация допускает негомолгичные обмены, к которым относятся транслокации, инверсии и случаи неравного кроссинговера. Такие способы могут оказаться полезными при генерации новых решений.

В генетических алгоритмах наибольшее распространение получила операция кроссинговера, заключающаяся в разрыве гомологичных хроматид с последующим соединением их в новом сочетании. Схема кроссинговера, демонстрирующая образование двух новых хромосом после обмена генетическим материалом, приведена на рис. 6.3.

Основная цель кроссинговера заключается в создании из имеющегося генетического материала желаемой комбинации признаков в одном решении.

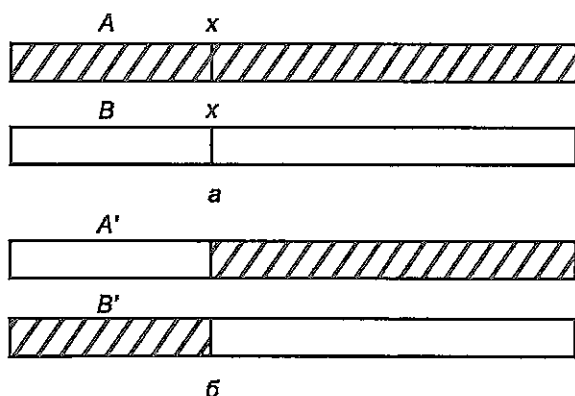


Рис. 6.3. Схема кроссинговера:

а — родительские хромосомы *А*, *В* до кроссинговера;
б — хромосомы-потомки *А'*, *В'* после кроссинговера

Кроссинговер может происходить в нескольких точках. Пример двойного кроссинговера между хромосомами *f* и *w* приведен на рис. 6.4.

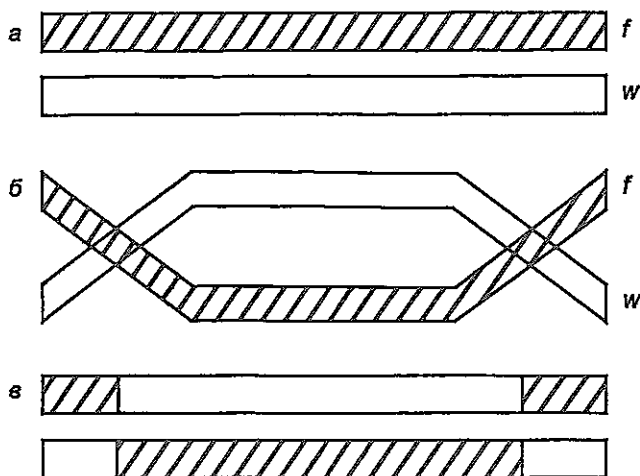


Рис. 6.4. Схема двойного кроссинговера:
 а — до кроссинговера; б — во время кроссинговера;
 в — после кроссинговера

Помимо кроссинговера для решения различных прикладных задач полезными являются такие генетические операции, как мутация, инверсия, транслокация, селекция (инбридинг и гибридизация), генная инженерия.

Под *мутацией* понимается генетическое изменение, приводящее к качественно новому проявлению основных свойств генетического материала: дискретности, непрерывности или линейности. Свойство дискретности позволяет выделить в исходном генетическом материале отдельные фрагменты, контролирующие те или иные функции. Непрерывность означает, что определенные комбинации генов совместно контролируют некоторую функцию. Линейность проявляется в определенной последовательности генов в пределах группы сцепления.

Процессы мутации ведут к получению более разнообразного генетического материала. В связи с этим применение операции мутации в генетических алгоритмах направлено на получение решений, которые не могут быть улучшены качественно посредством кроссинговера.

Инверсия, транслокация, транспозиция, делеция и дупликация относятся к разновидностям хромосомной мутации. При ин-

версии участок хромосомы поворачивается на 180° . *Транслокацией* называют перенос части одной хромосомы в другую. При перемещении небольших участков генетического материала в пределах одной хромосомы используют термин *транспозиция*. *Делеция* — это выпадение отдельных участков хромосом, *дупликация* — повторение участка генетического материала. Кроме перечисленных, существуют другие разновидности хромосомных мутаций [7].

Селекция представляет собой форму искусственного отбора, который может быть массовым или индивидуальным. Установлено, что массовый отбор по фенотипу (совокупности всех внешних и внутренних признаков) менее эффективен, чем индивидуальный, когда популяцию делят на отдельные линии, а для размножения выбирают носителей желаемых свойств. Применение процедуры селекции в генетических алгоритмах оптимизации способствует ускорению процесса синтеза искомого решения.

Генная инженерия представляет собой совокупность методов для получения рекомбинантной ДНК и операции над нею. Рекомбинантная ДНК получается путем объединения фрагментов ДНК различных организмов. Использование подходов генной инженерии позволяет в ряде задач значительно быстрее находить желаемое решение.

Механизм эволюции основан на трех повторяющихся процессах: отборе, амплификации (процесс производства потомков) и мутации. Он используется в качестве механизма случайно направленного комбинаторного перебора при решении задач оптимизации и слабоструктурированных проблем принятия решений.

Генетический алгоритм — это поисковый алгоритм, основанный на природных механизмах селекции и генетики. Эти алгоритмы обеспечивают выживание сильнейших решений из множества сгенерированных, формируя и изменяя процесс поиска на основе моделирования эволюции исходной популяции решений. Генетические алгоритмы сконструированы таким образом, что при генерации каждой новой популяции используются фрагменты исходных решений, к которым добавляются новые элементы, обеспечивающие улучшение решений относительно сформулированного критерия отбора. Другими словами, генетические алгоритмы используют информацию, накопленную в процессе эволюции [2, 3, 7, 8, 16, 22 — 26].

В генетических алгоритмах используются специфические термины, взятые из генетики, которые трактуются следующим образом.

Генетика	Генетические алгоритмы
Хромосома	Решение, стринг, строка, последовательность, родитель, потомок
Популяция	Набор решений (хромосом)
Локус	Местоположение гена в хромосоме
Поклоение	Цикл работы генетического алгоритма, в процессе которого сгенерировано множество решений
Ген	Элемент, характеристика, особенная черта, свойство, детектор
Аллель	Значение элемента, характеристики
Фенотип	Структура
Эпистасис	Множество параметров, альтернативные решения
Скрещивание, рекомбинация, кроссинговер	Оператор рекомбинации
Мутация	Оператор модификации

При разработке генетических алгоритмов преследуются две главные цели:

- абстрактное и формальное объяснение процессов адаптации в естественных системах;
- проектирование искусственных программных систем, воспроизводящих механизмы функционирования естественных систем.

Основные отличия ГА от других алгоритмов оптимизации:

- используются не параметры, а закодированные множества параметров;
- поиск осуществляется не из единственной точки, а из популяции точек;
- в процессе поиска используются значения целевой функции, а не ее приращения;
- применяются вероятностные, а не детерминированные правила поиска и генерации решений;
- выполняется одновременный анализ различных областей пространства решений, в связи с чем возможно нахождение новых областей с лучшими значениями целевой функции за счет объединения квазиоптимальных решений из разных популяций.

6.1.1. ПРОСТОЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ

Согласно репродуктивному плану Холланда [14, 26] генетические схемы поиска оптимальных решений включают следующие этапы процесса эволюции:

1. Конструируется начальная популяция. Вводится начальная точка отсчета поколений $t = 0$. Вычисляются приспособленность хромосом популяции (целевая функция) и средняя приспособленность всей популяции.

2. Устанавливается значение $t = t + 1$. Выбираются два родителя (хромосомы) для кроссинговера. Выбор осуществляется случайным образом пропорционально жизнеспособности хромосом, которая характеризуется значениями целевой функции.

3. Формируется генотип потомка. Для этого с заданной вероятностью над генотипами выбранных хромосом производится операция кроссинговера. Случайным образом выбирается один из потомков $A(t)$, который сохраняется как член новой популяции. Далее к потомку $A(t)$ последовательно с заданными вероятностями применяются операторы инверсии и мутации. Полученный в результате генотип потомка сохраняется как $A'(t)$.

4. Обновление текущей популяции путем замены случайно выбранной хромосомы на $A'(t)$.

5. Определение приспособленности $A'(t)$ и пересчет средней приспособленности популяции.

6. Если $t = t^*$, где t^* — заданное число шагов, то переход к этапу 7, в противном случае — переход к этапу 2.

7. Конец работы.

Основная идея эволюции, заложенная в различные конструкции генетических алгоритмов, проявляется в способности «лучших» хромосом оказывать большее влияние на состав новой популяции за счет длительного выживания и более многочисленного потомства.

Простой генетический алгоритм [23, 26] включает операцию случайной генерации начальной популяции хромосом и ряд операторов, обеспечивающих генерацию новых популяций на основе начальной. Этими операторами являются репродукция, кроссинговер и мутация.

Репродукцией называется процесс копирования хромосом с учетом значений целевой функции, т.е. хромосомы с «лучшими»

значениями целевой функции имеют большую вероятность попадания в следующую популяцию. Этот процесс является аналогией митозного деления клеток. Выбор клеток (хромосом) для репродукции проводится в соответствии принципом «выживания сильнейшего». Простейшим способом представления операции репродукции в алгоритмической форме является колесо рулетки, в котором каждая хромосома имеет поле, пропорциональное значению целевой функции.

Рассмотрим пример применения простого генетического алгоритма для максимизации функции $f(x)=x^2$ на целочисленном интервале $[0, 31]^*$.

Значения аргумента функции $f(x)=x^2$, изменяющегося в интервале от 0 до 31, можно представить пятиразрядными двоичными числами. Первоначальная популяция, состоящая из четырех строк пятиразрядных чисел, полученная с помощью процедуры генерации случайных чисел, приведена во втором столбце табл. 6.1. Значение целевой функции для каждой хромосомы опреде-

Таблица 6.1

Анализ начальной популяции
на первом шаге простого генетического алгоритма

Номер хромосомы	Двоичный код хромосомы	Значение x (десятичный код)	Значение целевой функции $f(x)$	Нормированное значение $f(x)/\text{sum}(f(x))$	Ожидаемое количество копий хромосомы в следующем поколении	Реальное количество копий хромосомы в следующем поколении
1	01101	13	169	0.14	0.56	1
2	11000	24	576	0.49	1.96	2
3	01000	8	64	0.06	0.24	0
4	10011	19	361	0.31	1.24	1
Суммарная целевая функция			1170	1.00	4.00	4
Среднее значение целевой функции			293	0.25	1.00	1
Максимальное значение целевой функции			576	0.49	1.97	2

* Пример взят из монографии В. М. Курейчика «Генетические алгоритмы» [7].

ляется путем возведения в квадрат значения двоичного числа, кодирующего решение x . Претенденты для скрещивания (кроссинговера) могут выбираться из начальной популяции или после выполнения оператора репродукции.

Репродукция начального множества заключается в четырехкратном вращении колеса рулетки (4 — мощность популяции), в результате чего состав исходной популяции может измениться (рис. 6.5). Вероятность выбора i -й хромосомы вычисляется по формуле

$$P_i = \frac{f_i(x)}{\text{sum}f(x)},$$

где $f_i(x)$ — значение целевой функции i -й хромосомы в популяции;
 $\text{sum}f(x)$ — суммарное значение целевой функции всех хромосом в популяции.

Ожидаемое число копий i -й хромосомы после оператора репродукции равно

$$N = nP_i,$$

где n — число анализируемых хромосом.

Число копий хромосомы, переходящих в следующее поколение, определяют по формуле

$$A_i = \frac{f_i(x)}{f_{\text{cp}}(x)},$$

где $f_{\text{cp}}(x)$ — среднее значение целевой функции.

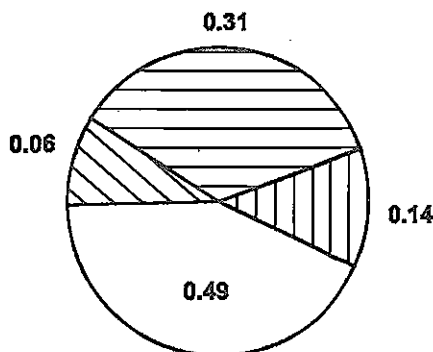


Рис. 6.5. Колесо рулетки

Значение N для первой хромосомы будет равно $0.14 \times 4 = 0.56$ копий, для второй — $0.49 \times 4 = 1.96$ копий, для третьей — $0.06 \times 4 = 0.24$ и для четвертой — $0.31 \times 4 = 1.24$. В результате репродукции в новой популяции (второй столбец в табл. 6.2) будут присутствовать по одной копии первой и четвертой хромосомы и две копии второй, а третья хромосома будет исключена. Таким способом оператор репродукции отбирает *лучших* представителей популяции.

Таблица 6.2

**Результаты операций репродукции
и кроссинговера в простом генетическом алгоритме**

Номер хромосомы	Популяция после репродукции	Случайно выбранные пары	Точка разрыва кроссинговера	Популяция после кроссинговера	Значение x (десятичный код)	Значение $f(x)$
1	0110 1	1 – 2	4	01100	12	144
2	1100 0	1 – 2	4	11001	25	625
3	11 000	3 – 4	2	11011	27	729
4	10 011	3 – 4	2	10000	16	256
Суммарное значение целевой функции $\text{sum} f(x) = 1754$						
Среднее значение целевой функции $f_{cp}(x) = 439$						
Максимальное значение целевой функции $\max f(x) = 729$						

На шаге 2 с помощью колеса рулетки осуществляется выбор хромосом для кроссинговера. Поля колеса рулетки соответствуют нормированным значениям целевой функции. Указатель рулетки после останова колеса определяет выбранную хромосому. Следует заметить, что случайный механизм не гарантирует выбора лучших хромосом, т.е. иногда результатом выбора могут оказаться хромосомы с низкими значениями целевой функции.

После репродукции выполняется оператор кроссинговера, который может повторяться несколько раз. При этом каждый раз будет осуществляться выбор двух кандидатур из множества хромосом. Затем каждая пара хромосом (стрингов) пересекается. Место пересечения K выбирается случайным образом на интер-

вале (1, $L-1$), где L — длина хромосомы, определяемая количеством значащих цифр в ее двоичном коде. В нашем случае $L = 5$. Две новые хромосомы создаются путем взаимного обмена всех значений после точки пересечения, т.е. между позициями $(K + 1)$ и L . При выборе двух первых хромосом из популяции (см. табл. 6.1) и значения $K = 4$ до применения оператора кроссинговера имеем описание:

$$\text{родители} \quad \begin{cases} \text{хромосома 1: } 0110|1 \\ \text{хромосома 2: } 1100|0 \end{cases},$$

а после применения оператора кроссинговера получаем описание

$$\text{потомки} \quad \begin{cases} \text{хромосома 1: } 0110|0 \\ \text{хромосома 2: } 1100|1 \end{cases}.$$

Аналогично были получены потомки от третьей и четвертой хромосом.

Анализ полученных результатов (см. табл. 6.2) показывает, что после проведения одной генерации улучшились и среднее, и максимальное значение целевой функции по сравнению с начальной популяцией (см. табл. 6.1).

Согласно схеме простого генетического алгоритма на шаге 3 выполняется оператор мутации, который играет существенную роль в естественной генетике и эволюции, но менее значим в генетических алгоритмах. Обычно выбирают одну мутацию на 1000 бит. Оператор мутации относится к унарным операциям и реализуется в два этапа.

Этап 1. В хромосоме $A = \{a_1, a_2, a_3, \dots, a_{L-2}, a_{L-1}, a_L\}$ случайным образом определяют две позиции, например, 2 и $L-1$.

Этап 2. Гены, соответствующие выбранным позициям, меняют местами и формируют новую хромосому $A = \{a_1, a_{L-1}, a_3, \dots, a_{L-2}, a_2, a_L\}$.

Если длина обрабатываемых последовательностей невелика, то в процессе мутации можно осуществить полный перебор возможных перестановок генов и найти комбинацию с максимальным значением целевой функции. При длине хромосомы $L=50 - 200$ полный перебор вариантов становится затруднительным, поэтому здесь производится случайно-направленный поиск, который может быть реализован на основе простого генети-

ческого алгоритма. Рассмотрим этот механизм на исследуемой задаче.

Выберем третью хромосому из пятого столбца табл. 6.2 со значением целевой функции $f(x)=729$ и применим операцию мутации к позициям 3 и 4:

хромосома 3: 11011 $\rightarrow$ хромосома 3': 11101.

У новой хромосомы 3' значение целевой функции равно $(29)^2=841$. Сделаем еще одну перестановку 4 и 5 генов в хромосоме 3':

хромосома 3': 11101 $\rightarrow$ хромосома 3'': 11110.

Значение целевой функции для хромосомы 3'' равно 900, что соответствует квазиоптимальному решению задачи нахождения максимального значения функции $f(x)=x^2$ на интервале $[0,31]$.

В генетических алгоритмах и эволюционном программировании используют два основных механизма воспроизводства хромосом:

- воспроизводство без мутаций, соответствующее митозу, результатом которого являются потомки — копии родителей;
- воспроизводство потомков, имеющих большие отличия от родителей. Этот механизм соответствует половому размножению.

В генетических алгоритмах в основном используется механизм родительского воспроизводства [4] с рекомбинацией и мутацией, а в эволюционном программировании — механизм на основе мутации без рекомбинации.

В алгоритмических реализациях механизма воспроизводства хромосом следует придерживаться следующих правил.

1. Выбор начальной популяции можно выполнять произвольным образом, например подбрасыванием монеты.
2. Репродукция осуществляется на основе моделирования движения колеса рулетки.
3. Оператор кроссинговера реализуется как взаимный обмен короткими фрагментами двоичных строк гомологичных хромосом.
4. Вероятность оператора кроссинговера принимается равной $P(CO) \leq 1.0$.
5. Вероятность оператора мутации принимается равной $P(MO) \geq 0.001$.

РАЗНОВИДНОСТИ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Генетический алгоритм Девиса [25] включает следующие шаги:

1. Инициализация популяции хромосом.
2. Оценка каждой хромосомы в популяции.
3. Создание новых хромосом посредством изменения и скрещивания текущих хромосом (применение операторов мутации и кроссинговера).
4. Устранение хромосом из популяции для замены их новыми.
5. Оценка новых хромосом и включение их в популяцию.
6. Проверка условия исчерпания ресурса времени, отведенного на поиск оптимального решения (если время исчерпано, то работа алгоритма завершается и производится возврат к наилучшей хромосоме, в противном случае — переход к шагу 3).

Холланд [14, 26] предложил для генетического алгоритма *оператор инверсии*, который реализуется по схеме:

1. Строинг (хромосома) $B = \{b_1, b_2, \dots, b_L\}$ выбирается случайным образом из текущей популяции.

2. Из множества $Y = \{0, 1, 2, \dots, L + 1\}$ случайным образом выбираются два числа y_1 и y_2 и определяются значения $x_1 = \min\{y_1, y_2\}$ и $x_2 = \max\{y_1, y_2\}$.

3. Из хромосомы B формируется новая хромосома путем инверсии (обратного порядка) сегмента, лежащего справа от позиции x_1 и слева от позиции x_2 в хромосоме B . После применения оператора инверсии строка B примет вид $B' = \{b_1, \dots, b_{x_1}, b_{x_2-1}, b_{x_2-2}, \dots, b_{x_1+1}, b_{x_2}, \dots, b_L\}$.

Например, для строки $B = \{1, 2, 3, 4, 5, 6\}$ при выборе $y_1 = 6$ и $y_2 = 2$ и соответственно $x_1 = 2$, $x_2 = 6$ результатом инверсии будет $B' = \{1, 2, 5, 4, 3, 6\}$.

Операции кроссинговера и мутации, используемые в простом ГА, изменяют структуру хромосом, в том числе разрушают удачные фрагменты найденных решений, что уменьшает вероятность нахождения глобального оптимума. Для устранения этого недостатка в генетических алгоритмах используют *схемы* (схематы или шаблоны), представляющие собой фрагменты решений или хромосом, которые желательно сохранить в процессе эволюции. При использовании схем в генетическом алгоритме вводится новый алфавит $\{0, 1, *\}$, где $*$ интерпретируется как «имеет значение 1 или 0». Например:

схема (*0000) соответствует двум стрингам {10000 и 00000};

схема (*111*) соответствует четырем строкам {01110, 11110, 01111, 11111};

схема (0*1**) может соответствовать восьми пятизначным стрингам.

В общем случае хромосома длиной L максимально может иметь 3^L схем (шаблонов), но только 2^L различных альтернативных стрингов. Это следует из факта, что схеме (**) в общем случае могут соответствовать $3^2=9$ стрингов, а именно {**, *1, *0, 1*, 0*, 00, 01, 10 11}, и только $2^2=4$ альтернативные строки {00, 01, 10, 11}, т. е. одной и той же строке может соответствовать несколько схем.

Если в результате работы генетического алгоритма удалось найти схемы типа (11***) и (**111), то, применив к ним оператор кроссинговера, можно получить хромосому (11111), обладающую наилучшим значением целевой функции.

Схемы небольшой длины называются строительными блоками. Размер строительных блоков заметно влияет на качество и скорость нахождения результата. Вид строительного блока выбирается с учетом специфики решаемой задачи, а их разрыв в генетических алгоритмах допускается только в исключительных случаях, определяемых пользователем. Например, в схеме (****1) строительным блоком является элемент 1, а в схеме (10***) — составной элемент 10.

При использовании большого числа строительных блоков генетические алгоритмы, основанные на случайной генерации популяций и хромосом, переходят в разряд беспорядочных.

Стационарные генетические алгоритмы отличаются от *поколенческих* тем, что у первых размер популяции является заданным постоянным параметром, который определяется пользователем, а у вторых размер популяции в последующих генерациях может увеличиваться или уменьшаться.

Процедура удаления лишних хромосом в стационарных и поколенческих генетических алгоритмах основана на эвристических правилах, примерами которых являются следующие:

- случайное равновероятное удаление хромосом;
- удаление хромосом, имеющих худшие значения целевой функции;
- удаление хромосом на основе обратного значения целевой функции;
- удаление хромосом на основе турнирной стратегии.

Следует иметь в виду, что использование в генетических алгоритмах тех или иных эвристик удаления хромосом может повлечь за собой негативные последствия. Например, удаление худших хромосом приводит к преждевременной утрате разнообразия и, как следствие, к попаданию целевой функции в локальный оптимум, а при наличии большого числа хромосом с плохими значениями целевой функции утрачивается направленность поиска, и он превращается в «слепой» поиск.

Фундаментальная теорема генетического алгоритма

Пусть в момент времени t в популяции $S(t)$ содержится множество хромосом $S_j, j = 1, 2, \dots, n$, а схема H строится на основе алфавита $V = \{0, 1, *\}$. Тогда схема может быть определена на двоичной хромосоме длины L . Очевидно, что для алфавита мощности M существует $(M + 1)^L$ схем и $n \cdot 2^L$ схем, содержащихся в популяции размера n , поскольку стринг представляется двумя схемами.

Для количественной оценки схем введем две характеристики: порядок схемы $O(H)$ и определенная длина схемы $L(H)$. Порядок схемы определяет число закрепленных позиций (в двоичном алфавите — число единиц и нулей), представленных в шаблоне. Определенная длина схемы — это расстояние между первой и последней числовой позицией стринга.

Предположим, что заданы шаг по времени t и m примеров схем H , содержащихся в популяции $S(t)$, которые определяют возможное число различных схем H при заданном t , т.е. $m = m(H, t)$.

В процессе репродукции вероятность попадания хромосомы S_i в репродуцированное множество равна $P_i = \frac{f_i(x)}{\sum f(x)}$, т.е. зависит от значения целевой функции. За время $t + 1$ в популяции $S(t)$ ожидается получить $m(H, t + 1)$ представителей схемы H , которое вычисляется по формуле

$$m(H, t + 1) = m(H, t) n \frac{f(H)}{\sum f(x)},$$

где $f(H)$ — среднее значение целевой функции хромосом, представленных схемой H за время t .

Так как среднее значение целевой функции для всей популяции равно

$$f_{cp}(S) = \text{sum} f(x) / n, \text{ то } m(H, t+1) = m(H, t) \frac{f(H)}{f_{cp}(S)}.$$

Из этой формулы можно сделать вывод о том, что увеличение количества частных схем определяется отношением среднего значения целевой функции схемы к среднему значению целевой функции популяции. Поэтому схема, для которой значение целевой функции $f(H)$ выше $f_{cp}(S)$, имеет большую вероятность копирования.

Правило Холланда: Схема со значением целевой функции выше среднего живет и копируется, а схема со значением ниже среднего умирает.

Если предположить, что схема H является жизнеспособной, то $f(H) \geq f_{cp}(S)$. Тогда значение целевой функции для схемы H можно выразить через среднее значение для всей популяции, например, следующим образом: $(1+c)f_{cp}(S)$, где c — константа. Число представителей схемы в следующем поколении будет

$$m(H, t+1) = m(H, t) \frac{(1+c)f_{cp}(S)}{f_{cp}(S)} = (1+c)m(H, t).$$

Если принять значение c постоянным во времени, то за период $0 \leq t \leq t^*$ можно вычислить количество представителей схемы H по формуле $m(H, t) = (1+c)^t m(H, 0)$, из которой следует, что репродукция может приводить к экспоненциальному увеличению ($c > 0$) или уменьшению ($c < 0$) числа схем.

Лемма. Если на некотором шаге генетического алгоритма P_1 есть вероятность того, что хромосома A порождает потомка, и P_2 есть вероятность, что A уничтожается, то ожидаемое число потомков хромосомы A равно P_1/P_2 [26].

Вероятность выживания хромосомы A на шаге t после операции репродукции определяется по формуле $P_S(t) = (1 - P_2)^{t-1} P_2$, где $t = 1, 2, \dots, g$; g — число шагов (генераций) генетического алгоритма. Значение вероятности выживания хромосомы изменяется после операций кроссинговера и мутации. Использование оператора кроссинговера может вызывать увеличение или уменьшение числа схем в популяции. Если кроссинговер не применяется, то обмен между хромосомами отсутствует, поэтому поисковое пространство не увеличивается, и процесс затухает.

Вероятность выживания схемы после применения оператора кроссинговера определяется по формуле

$$P_S(H) = 1 - \frac{O(H)}{L-1},$$

где $O(H)$ — порядок схемы;
 L — длина стринга.

Если оператор кроссинговера выполняется на основе случайного выбора с вероятностью $P(CO)$, то вероятность выживания схемы определяется по формуле

$$P_S(H) \geq 1 - \frac{P(CO)L(H)}{L-1},$$

где $L(H)$ — определенная длина схемы.

Приведенное выражение свидетельствует о том, что вероятность выживания схемы уменьшается при возрастании $P(CO)$.

Вычислим число схем H в новой генерации после операций репродукции и кроссинговера, допуская их взаимную независимость:

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{f_{cp}(S)} \left[1 - P(CO) \frac{L(H)}{L-1} \right].$$

Из этого выражения следует, что число схем $m(H, t+1)$ зависит от значений целевой функции для схемы и для всей популяции, а также от длины схемы $L(H)$.

Рассмотрим влияние мутации на выживание схем. Известно, что единственная хромосома выживает с вероятностью $1 - P(MO)$, где $P(MO)$ — вероятность оператора мутации. Если учесть тот факт, что частная схема выживает в случаях, когда выживает каждая из $L(H)$ закрепленных позиций схемы, то для малых величин $P(MO) \ll 1$ вероятность выживания схемы при мутации может быть представлена выражением [23]: $P_S(MO) = 1 - L(H)P(MO)$.

С учетом вышесказанного и согласно [26] для частной схемы H можно определить ожидаемое число копий в следующей генерации после реализации операторов репродукции, кроссинговера и мутации по следующей формуле:

$$m(H, t+1) \geq m(H, t) \frac{f(H)}{f_{cp}(S)} \left[1 - P(CO) \frac{L(H)}{L-1} - L(H)P(MO) \right].$$

Данная формула отражает фундаментальную теорему генетического алгоритма, которая определяет асимптотическое число схем, выживающих при его реализации на каждой итерации. Наиболее существенное влияние на число выживающих схем оказывают значения целевых функций отдельной схемы и всей популяции, а эффективность реализации генетических алгоритмов зависит от размера строительных блоков.

6.1.3. ПРИМЕРЫ ПРАКТИЧЕСКОГО ПРИМЕНЕНИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Генетические алгоритмы нашли широкое практическое применение в менеджменте и управлении [13] для решения задач поиска оптимальных решений, формирования моделей и прогнозирования значений различных показателей. Они осуществляют поиск лучших решений на основе заданной целевой функции. Значение целевой функции для многих задач весьма непросто вычислить, поэтому в ряде случаев при исследовании плохо обусловленных проблем с этой целью применяются нейронные сети, позволяющие найти решение при отсутствии явной модели. Кроме того, для вычисления целевых функций в условиях неопределенности применяются статистические методы и методы логического вывода в четкой или нечеткой среде.

Формирование системы прогнозирующих правил. Генетические алгоритмы могут использоваться для нахождения оптимального набора правил, позволяющих прогнозировать страховые риски с учетом ряда определяющих его факторов [13]. Для решения этой задачи необходимо иметь базу данных, содержащую фактические значения переменных, влияющих на страховой риск.

Рассмотрим пример использования генетического алгоритма для оптимизации экспертных правил в сфере страхования.

Допустим, что компания, занимающаяся страхованием автомобилей, использует базу данных, которая помимо прочих включает следующие факторы: максимальную скорость автомобиля (км/час), возраст автомобиля (лет), возраст водителя (лет) и риск, определенный экспертно по некоторой шкале на основе анализа обращений клиентов о выплате компенсации по страховым слу-

чаям. Правила, задающие оценку страхового риска, сконструированы в виде:

ЕСЛИ максимальная скорость автомобиля лежит в диапазоне $[A_i]$ И возрастной диапазон автомобиля $[B_i]$ И возраст водителя находится в диапазоне $[C_i]$, ТО страховой риск имеет значение $[D_i]$.

Для конкретной выборки из БД это правило может иметь следующий вид:

ЕСЛИ максимальная скорость $[91 - 100 \text{ км/час}]$ И возраст автомобиля $[11 - 15 \text{ лет}]$ И возраст водителя $[31 - 40 \text{ лет}]$, ТО риск $[3]$. Здесь уровень риска отображается на интервал $[1, 5]$, при этом высокие значения соответствуют большим страховым рискам.

Подобные правила, основанные на фактических значениях переменных, случайным образом выбранных из БД, составляют исходную популяцию. Для каждой из переменных, входящих в популяцию, предварительно задается диапазон состояний. Например, переменная «возраст автомобиля», может иметь пять возможных состояний: $1 - 5, 6 - 10, 11 - 15, 16 - 20, 21 - 25 \text{ лет}$. Далее сформированная популяция обрабатывается генетическими операторами с учетом специфики рассматриваемой задачи. Целевая функция должна показывать, насколько точно сгенерированные правила описывают реальные страховые случаи, хранящиеся в БД. Например, если какое-то правило описывает 4 случая из 5, то значение целевой функции будет $4/5$, или 80%.

Новые члены популяции образуются в результате скрещивания и мутации начального набора правил. В данном случае при скрещивании двух правил происходит обмен парами «атрибут — значение» на участке строки после точки кроссинговера. В результате образуются два новых правила, жизнеспособность которых оценивается по тому, насколько удачно они описывают страховые случаи, которые имели место в прошлом. Мутация правил обеспечивает необходимое разнообразие признаков и заключается в изменении значений атрибутов с заданной вероятностью. Таким образом, первоначально сформированный набор правил преобразуется случайно направленным способом в другой набор, который лучше остальных описывает накопленную статистику страховых случаев. Результирующая система правил в дальнейшем используется для прогнозирования страховых рисков.

Следует отметить, что подобный подход к формированию системы правил может приводить к некорректным правилам продукций. В то же время он освобождает разработчиков и экспертов

от трудоемкой работы по формулированию и оценке правил, так как некорректные результаты отбрасываются при сопоставлении сгенерированных продуктов с реальными страховыми ситуациями. Привлечение прошлого опыта для оценки пригодности прогнозирующих правил не позволяет предвидеть новые ситуации, которые не имели места в прошлом. Поэтому при решении задач описанным способом очень важно следить за своевременным пополнением и модификацией информации в БД, которая отражает появление новых фактов, атрибутов и тенденций.

Классифицирующие системы. На основе генетических алгоритмов Дж. Холланд предложил классифицирующие системы, которые можно использовать для целей управления [11, 14, 17, 30]. Классифицирующая система состоит из трех вложенных друг в друга подсистем (рис. 6.6): классификатора, системы обучения и генетического алгоритма. В классификатор поступают внешние

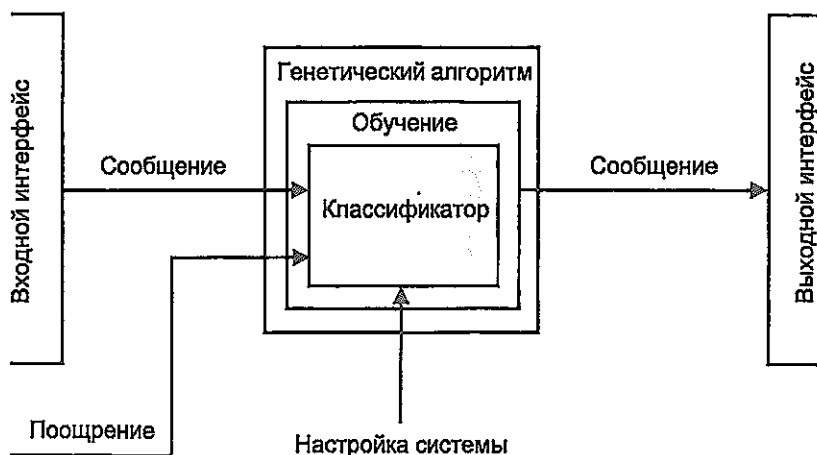


Рис. 6.6. Схема классифицирующей системы

сообщения и положительные оценки (поощрения) его действий. Классификатор содержит правила вида ЕСЛИ<условие>, ТО<сообщение>, с помощью которых формируются выходные сообщения. Обучающая система выполняет оценку используемых правил. Генетический алгоритм предназначен для случайно направленной модификации правил. Схема обработки правил представлена на рис. 6.7.

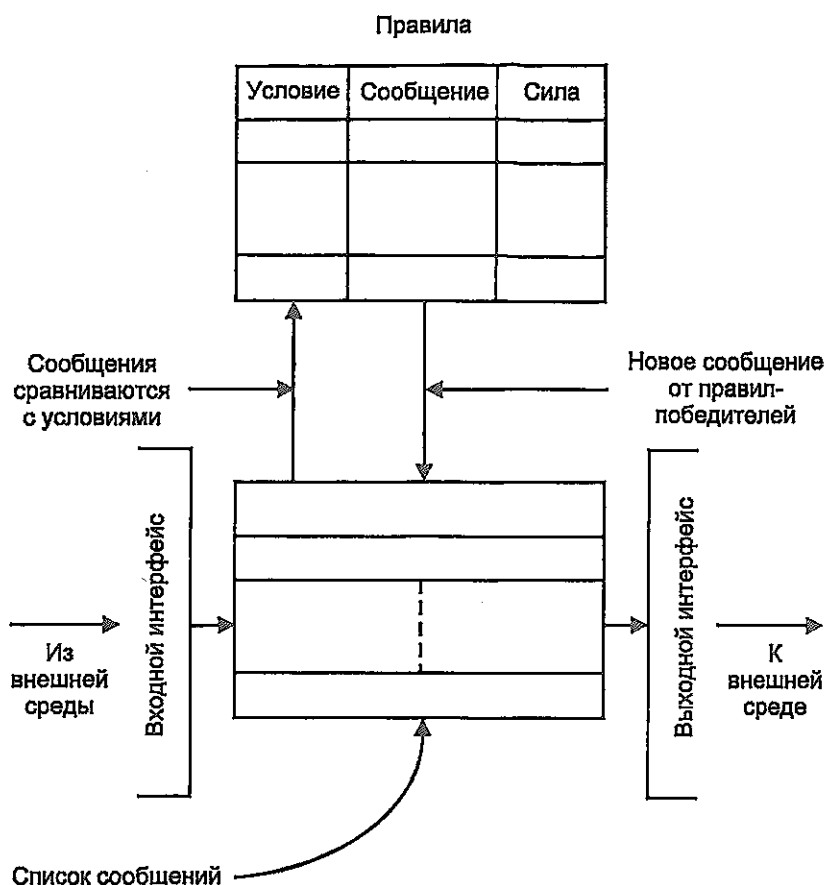


Рис. 6.7. Схема обработки правил в классифицирующей системе

Каждому правилу приписывается численная оценка *силы правила*. Сообщения и условные части правил (антецеденты) формулируются в одних и тех же терминах. Список сообщений содержит все текущие сообщения — поступающие из внешней среды и те, что формируются внутри системы. В процессе работы КС все сообщения из списка сравниваются с условиями всех правил. Классификатор выполняет следующие действия.

Шаг 1. В список сообщений (рабочую память) добавляются все сообщения, поступившие извне.

Шаг 2. Проводится сравнение всех сообщений из списка с антецедентами всех правил. Все правила, антецеденты которых совпадают с присутствующими в рабочей памяти сообщениями, записываются в список правил M .

Шаг 3. Выполняются правила из списка M , при этом сообщения каждого правила посылаются в список новых сообщений.

Шаг 4. Обновление списка сообщений.

Шаг 5. Сообщения из списка посылаются в выходной интерфейс. Вероятность выдачи сообщения зависит от силы правила: не каждое сообщение выдается на управляемый объект, часть их может быть связана с изменением внутренней структуры системы (правил).

Шаг 6. Возврат к шагу 1.

В процессе обучения каждому правилу присваивается численное значение силы, а алгоритм обучения регулирует это значение с учетом полезности правила для системы. На шаге 3 описанного алгоритма для каждого отобранного правила C вычисляется цена по формуле $B(C, t) = bR(C)s(C, t)$, где $s(C, t)$ — сила правила C в момент t ; $R(C)$ — специфичность условия в правиле, равная числу символов, отличающихся от символа * в условии, деленному на длину условия; b — коэффициент, который обычно принимают равным $1/8$ или $1/6$.

Цена B определяет вероятность того, что правило пошлет сообщение в список новых сообщений. Вероятностный подход позволяет аутсайдерам тоже изредка посылать сообщения, что при благоприятных условиях может сделать их лидерами.

Послать сообщение могут все правила с допустимым значением B , т.е. такие, у которых B превышает определенный порог. Правило, пославшее сообщение в новый список, расплачивается за это уменьшением своей силы:

$$s(C, t + 1) = s(C, t) - B(C, t).$$

Для правил C' , пославших сообщения, которые на следующем шаге работы оказались полезными (совпали с условиями правила-победителя, имеющего высокую цену), оценка силы возрастает на долю B :

$$s(C', t + 1) = s(C', t) + aB(C, t),$$

где $a = 1/K$, K — число правил C' , т.е. каждый поставщик получает равную долю B .

Правило полезно только тогда, когда его потребители в своих локальных действиях тоже получают выигрыш. В противном случае правило обесценивается, так как его цена s уменьшается при отсылке сообщения. В свою очередь, полезность потребителей зависит от их потребителей и т.д. Цепочка приводит к конечным потребителям, достигающим цели и получающим поощрения от внешней среды.

Классификатор и обучающая система не порождают новых правил. Эту функцию выполняет генетический алгоритм, который работает с учетом силы правил, определенной в системе обучения. Работа генетического алгоритма рассмотрена в предшествующем примере.

Комбинированные методы и интеллектуальные системы. В настоящее время активно развиваются методы, основанные на объединении технологий инженерии знаний и генетических алгоритмов. В области ГА разрабатываются операторы, ориентированные на обработку знаний.

Генетические алгоритмы используют в теории нечетких систем для настройки параметров функций принадлежности. Интеграция четких и нечетких нейронных сетей и генетических алгоритмов обеспечивает реализацию оптимизационной задачи. Средства *fuzzy-neuro-genetic* используются в интеллектуальных системах и содержат следующие процедуры:

- преобразование входных примеров в нечеткое представление;
- извлечение знаний, представленных в виде продукций ЕС-ЛИ-ТО из нечеткой обучающей выборки с помощью нейронной сети;
- оптимизацию структуры продукционных правил с помощью генетического алгоритма.

Активно развивается направление, ориентированное на использование генетических алгоритмов для обучения нейронных сетей [5] и корректировки структуры уже обученной сети [18]. В отличие от метода обратного распространения ошибки генетические алгоритмы мало чувствительны к архитектуре сети. Напомним, что основными характеристиками нейронной сети являются следующие:

- HLN — количество скрытых слоев;
- N_k — число нейронов в каждом слое;
- w_{ij} — весовые коэффициенты межнейронных связей;

• $F_j(X, W)$ — передаточные функции нейронов скрытых слоев, а также нейронов выходного слоя.

Сформулируем общую задачу оптимизации сети: при заданных количествах входных и выходных нейронов на основе заданного множества обучающих примеров определить оптимальные значения HLN , N_k , $k = 1, \dots, HLN$, значения всех весовых коэффициентов межнейронных связей w_{ij} , где j — индекс нейрона; i — индекс межнейронной связи (синапса), $F_j(X, W)$ — передаточные функции всех нейронов за исключением нейронов входного слоя. Критерием оптимизации является максимальное отклонение выходного вектора сети Y' от эталонного значения выхода Y , полученное в результате обработки всех примеров, т.е. необходимо найти

$$\delta^* = \min_{\substack{HLN, N_k, \\ W, F_j(X, W)}} \max_Q \delta,$$

где $\delta = Y' - Y$; Q — множество обучающих примеров, содержащих значения X, Y ; $Y' = F(HLN, N_k, X, W)$;
 $F(HLN, N_k, X, W)$ — передаточная функция ИНС, которая строится на основе частных функций отдельных нейронов $F_j(X, W)$.

Даже для простых сетей эта задача является очень сложной, поэтому для ее решения применяется декомпозиция, т.е. сеть оптимизируется в процессе последовательного решения частных задач оптимизации. Например, на первом шаге подбираются оптимальные значения HLN и N_k , затем определяется оптимальный вид передаточных функций нейронов, а на конечной стадии подбираются веса межнейронных связей.

Генетические алгоритмы чаще всего применяются для улучшения характеристик ИНС, уже созданных и обученных с применением других методов (см. главу 5).

6.1.4.

КРАТКИЙ ОБЗОР ПРОГРАММНЫХ СРЕДСТВ

Коммерческое программное обеспечение, реализующее генетические алгоритмы, можно разделить на программные средства общего назначения, прикладные и алгоритмические программные продукты.

Программное обеспечение общего назначения включает разнообразные наборы инструментальных средств для построения конкретных программ, которые содержат библиотеки алгоритмов, программы моделирования, средства визуализации и другие инструменты. Пакеты подобного типа рассчитаны на опытных программистов, требуют знания основ теории эволюционных вычислений и характеризуются высокой трудоемкостью освоения, которая в значительной мере зависит от квалификации пользователя.

Прикладные программные продукты ориентированы на решение проблем определенного класса в конкретных предметных областях (реинжиниринг, маркетинг, стратегическое планирование и др.). Такие средства не требуют от пользователя теоретических знаний в области методологии создания интеллектуальных систем. Достаточно, чтобы он был специалистом в своей предметной области.

Алгоритмическое программное обеспечение поддерживает один (или несколько) генетический алгоритм. Преимущества таких программных продуктов — их гибкость и простота использования. При этом пользователям необходимо иметь представление об основах теории ГА.

В табл. 6.3 перечислены некоторые популярные программные средства, реализующие технологии оптимизации с применением генетических алгоритмов [13].

Система *PC/Beagle* представляет собой программу поиска решающих правил, классифицирующих примеры из базы данных. Она превращает данные в знания за счет использования машинного обучения. Один из модулей системы путем репродукции и селекции порождает правила, представленные в виде логических выражений.

Система *Evolver* реализует шесть методов генетической оптимизации и выполнена в виде расширения MS Excel (рис. 6.8). Основные области применения пакета — оптимизация доходности с учетом уровня риска и максимизация прибыли с учетом возможных издержек.

Genesis — известный алгоритмический программный продукт, который используется в качестве инструмента тестирования генетических алгоритмов. Он позволяет создать модифицированную программную среду и обеспечивает пользователя статистической информацией на выходе.

**Программные средства, реализующие методологию
генетических алгоритмов**

Программные продукты	Фирма	Краткая характеристика
Прикладные		
<i>Evolver</i>	Axcelis	Генетический бизнес-инструмент, ориентированный на использование совместно с электронными таблицами MS Excel
<i>Omega</i>	KiO and CAP Volmac	Прикладное программное обеспечение для финансовых приложений
<i>PC/Beagle</i>	Pathway Research	Извлечение решающих правил из баз данных
<i>Xpert Rule Gen Asy</i>	Attar Software	Оболочка экспертной системы со встроенным генетическим алгоритмом
Алгоритмические		
<i>EM</i>	Technical University Berlin	Библиотека с набором эволюционных алгоритмов
<i>Escapade</i>	University of Portmund	Программное обеспечение, поддерживающее эволюционные алгоритмы
<i>GAGA</i>	University College London	Генетические алгоритмы, предназначенные для минимизации сложных функций производственных издержек
<i>Gauesd</i>	US San Diego	Программное обеспечение, представляющее большинство С-функций в качестве оценочных функций
<i>Genesis</i>	The Software Partnership	Набор программ генетических алгоритмов, написанных на языке С
<i>Genitor</i>	Colorado State University	Пакет программ, использующих алгоритм Whitley's Genitor
<i>OOGA</i>	The Software Partnership	Программное обеспечение на языке LISP (методология генетического программирования)

Программные продукты	Фирма	Краткая характеристика
<i>Алгоритмические программные продукты общего назначения</i> <i>EnGENEer</i>	Logica	Программы общего назначения на языке C
<i>Game</i>	University College London	Программная среда, содержащая библиотеку генетических алгоритмов
<i>GA Workbench</i>	Cambridge Consultants	Образовательная система с мощным графическим интерфейсом
<i>MicroGA</i>	Emergent Behavior	Программная подсистема генетических алгоритмов, написанная на языке C++
<i>Pegasus</i>	German National Research Center	Набор инструментов, написанных в ANSI-C
<i>Splicer</i>	NASA Johnson & Metre Corp.	Универсальный программный продукт: ядро + библиотека + целевые модели
<i>GeneHunter</i>	Ward Systems Group	Программная среда, позволяющая писать программы на Visual Basic и с помощью специальных встроенных функций

Программный продукт общего назначения *EnGENEer* помогает адаптировать генетические алгоритмы к новым проблемным областям за счет использования следующих инструментов:

- специального языка, предназначенного для описания структурных понятий генетики (генов, хромосом и т.д.);
- эволюционного модельного языка, используемого для отображения таких атрибутов, как размер популяции, типы скрещивания и мутации;
- графических инструментов мониторинга;
- библиотеки инструкций.

Объектно-ориентированная среда *Game* содержит пять основных частей:

- виртуальную машину;
- высокоуровневый генетический язык;

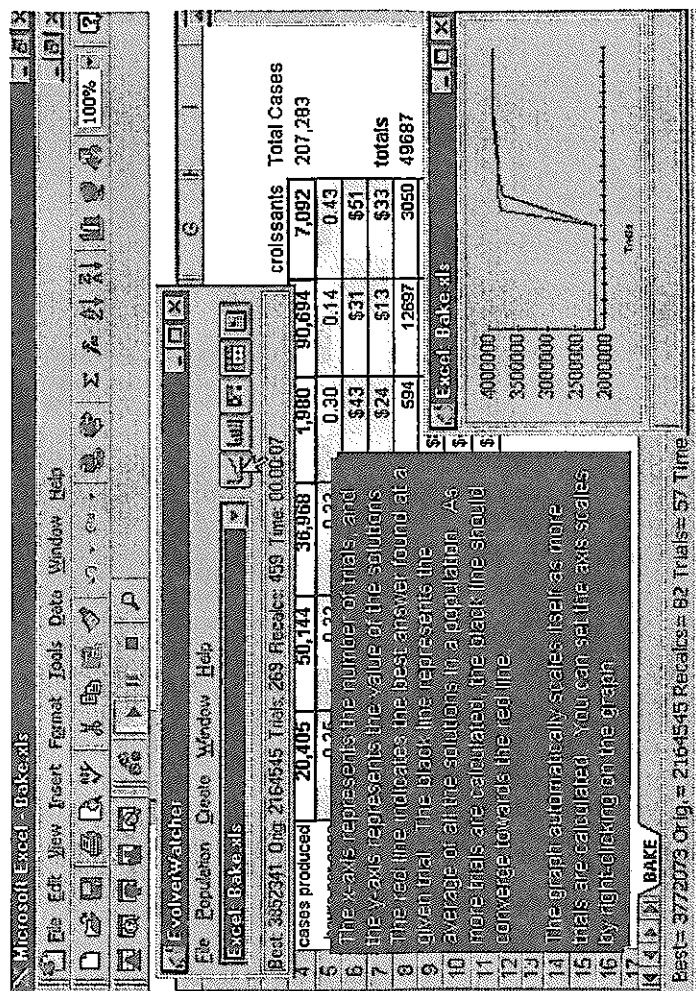


Рис. 6.8. Демонстрация работы генетического алгоритма в системе Evolver

- библиотеку генетических алгоритмов;
- графический монитор;
- компилятор.

Система спроектирована так, что допускает параллельное использование нескольких алгоритмов. Для создания конкретного приложения используются библиотечные модули, из которых строится *макропрограмма* с помощью специального высокоуровневого языка.

Известный дистрибьютер программного обеспечения фирма «Тора-Инфо-Центр» распространяет пакет *GeneHunter*, который может использоваться как приложение MS Excel и допускает составление собственных программ на языках C и Visual Basic.

6.2. МЕТОДЫ ЭВОЛЮЦИОННОГО ПРОГРАММИРОВАНИЯ

6.2.1. ГЕНЕТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Методы генетического программирования были разработаны в начале 1990-х гг. Дж. Козой [27 — 29]. Генетическое программирование — это способ создания компьютерных программ для задач с неизвестным алгоритмом решения. Объектом эволюции является программа, а популяция содержит множество различных программ. Совершенствование объекта осуществляется на основе отбора в соответствии с определенной функцией ценности (*fitness function*). Программы строятся из блоков, которые представляют собой примитивные функции и терминалы. В качестве примитивных функций обычно рассматриваются арифметические и логические операции, математические функции и функции специального вида, характерные для класса решаемых задач. Множество терминалов содержит разнообразные данные, не создаваемые программой. Цель состоит в построении наилучшей программы в пространстве программ, которые могут быть составлены из заданных функций и терминалов с учетом определенных правил синтаксиса.

Технология генетического программирования включает следующие этапы.

Этап 1. Формирование множества терминалов, множества примитивных функций, синтаксических правил и критериев оценки создаваемых программ.

Этап 2. На основе закона случайности создается начальная популяция компьютерных программ, ориентированных на решение поставленной задачи.

Этап 3. Каждая программа выполняется, а результаты ее работы оцениваются с помощью *fitness function* (целевой функции).

Этап 4. Формируется новая популяция программ, в которую сгенерированные программы могут попасть с вероятностью, пропорциональной значению целевой функции.

Этап 5. Реализуются генетические операторы репродукции, скрещивания и мутации. В результате репродукции осуществляется копирование уже созданных программ с *хорошими* значениями целевой функции. Оператор скрещивания создает новые программы путем комбинирования фрагментов существующих программ. Мутация заключается в замене некоторого фрагмента программы случайно порожденным символьным выражением.

Этап 6. Производится тестирование программ—членов новой популяции и принимается решение о продолжении процесса эволюции. Продолжать генерацию новых популяций имеет смысл тогда, когда максимальные и средние значения целевой функции улучшаются.

Рассмотрим пример [27] модификации программы на языке LISP, где в качестве терминалов используются переменные логического типа D_i , $i = 0, 1$; а для их обработки применяются логические операции NOT, OR, AND. Пусть на некотором шаге имеется следующее множество допустимых выражений:

```
NOT( $D_1$ );  
NOT( $D_0$ );  
AND( $D_0$ ,  $D_1$ );  
OR(AND( $D_0$ ,  $D_1$ ), NOT( $D_1$ ));  
AND(NOT ( $D_0$ ), NOT( $D_1$ ));  
OR( $D_1$ , NOT( $D_0$ ));  
OR(OR( $D_1$ , NOT( $D_0$ )), AND(NOT ( $D_0$ ), NOT( $D_1$ ))).
```

Эти выражения можно представить в виде деревьев (рис. 6.9). В процессе эволюции на уровне поддеревьев осуществляется рекомбинация и получаются потомки (рис. 6.10). Первый из этих

потомков представляет собой реализацию операции исключающего ИЛИ:

$OR(AND(NOT(D_0), NOT(D_1)), AND(D_0, D_1)).$

Результатом применения оператора мутации является замена части дерева другим выражением, сгенерированным случайным образом. Точка мутации также выбирается случайно.

Идеи генетического программирования положены в основу программ, которые называются симуляторами «искусственной жизни». В работе Дж. Козы [27] приводится следующий пример подобной программы. На тороидальной сетке размером 32×32 , в 89 ячейках помещается «пища». Существуют некие препятствия, мешающие «насекомым» добраться до «пищи». «Насекомые» попадают на сетку из одной точки, и каждое движется согласно командам своей программы. В начальной популяции эти программы формируются случайным образом из операторов, которые проверяют наличие препятствий и предписывают движение прямо, влево или вправо. Задается время жизни популяции (400 ша-

Объекты-родители

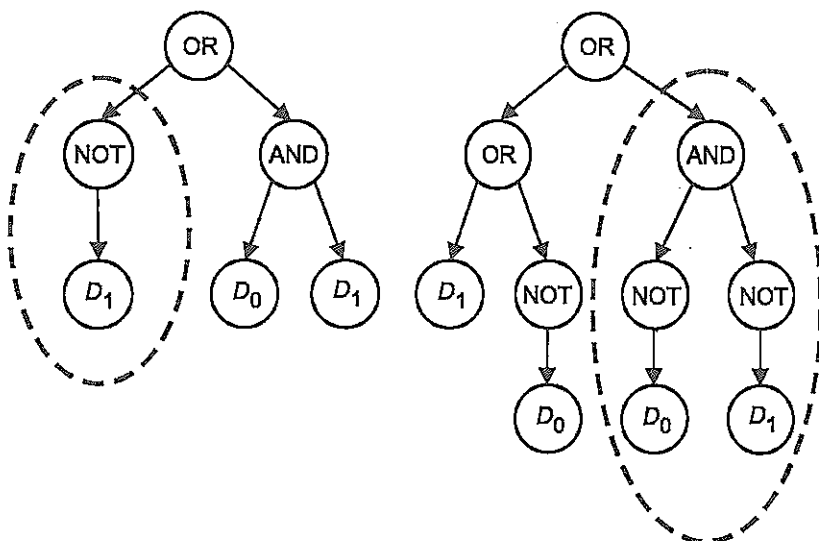


Рис. 6.9. Представление символьных выражений языка LISP в виде деревьев

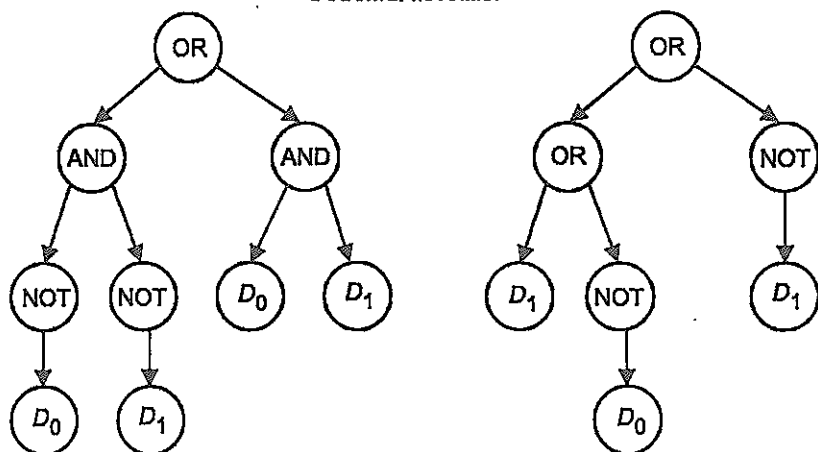


Рис. 6.10. Потомки от скрещивания родителей на уровне поддеревьев

гов). Цена каждой программы определяется числом шагов, которые необходимо совершить, чтобы обойти все ячейки с «пищей». Каждая следующая популяция формируется из предыдущей с помощью генетических операторов репродукции, скрещивания и мутации с учетом ценности программ предыдущей популяции. Решение для популяции из 4000 «насекомых» было найдено за 20 итераций.

Последователи Дж. Козы исследовали в своих работах возможность использования ГП для синтеза сложных автоматов, а также для структурной идентификации динамических систем [19].

В примере построения экономической балансовой модели [13] поставлена цель уточнения эконометрического уравнения обмена, связывающего уровень цен, валовой национальный продукт (ВНП), запас денег и скорость оборота денег в экономике.

В качестве терминалов здесь используются следующие переменные: *ВНП82* – уровень ВНП за 1982 г.; *GD* – дефлятор ВНП (выходная переменная модели), нормализованный к единице для 1982 г.; *FM* – ежемесячная величина запаса денег. Приведенные переменные являются функциями времени. Их значения определяются на основе статистических данных в виде временных рядов. Кроме того, используется множество обобщенных констант действительного типа *R*.

Для обработки переменных предусмотрены следующие операции: сложение (+); вычитание (-); умножение (*); защищенное деление (%), результатом которого является единица при попытке разделить на 0; защищенное логарифмирование (RLOG), дающее 1 при нулевом значении аргумента; вычисление экспоненты (EXP).

Грубая оценка пригодности сгенерированных уравнений вычисляется как сумма квадратов отклонений расчетных значений от фактических в заданных экспериментальных точках:

$$R_h(t) = \sum_{j=1}^N [V_j^h - S_j]^2,$$

где S_j — фактическое значение выходного параметра модели;
 $j = 1, \dots, N$; N — число точек;
 V_j^h — расчетное значение вычисляемого показателя;
 h — индекс сгенерированной модели.

Значение $R_h(t)$ масштабируется в целях получения нормированной оценки пригодности $a_h(t) = \frac{1}{1 + R_h(t)}$, которая изменяется на интервале $[0, 1]$ и позволяет перейти к задаче максимизации.

На основе $a_h(t)$ вычисляется относительная нормированная оценка пригодности $n_h(t) = a_h(t) / \sum_{m=1}^M a_m(t)$, которая имеет более высокие значения для лучших членов популяции и обладает свойством $\sum_{m=1}^M n_m(t) = 1$, допускающим вероятностную интерпретацию.

Критерием окончания процесса эволюции является достижение заданного числа генераций (50) или достижение наилучшего значения целевой функции. Численность популяции была принята равной 500. В процессе генерации новых поколений скрещивание проводилось на 90% численности популяции, т.е. из каждого поколения выбиралось 225 пар родителей с вероятностью, равной относительной оценке их пригодности. Кроме того, для каждой новой популяции осуществлялась репродукция 10% лучших представителей поколения.

Генерируемые модели (программы) удобно представить в виде древовидных структур (рис. 6.11).

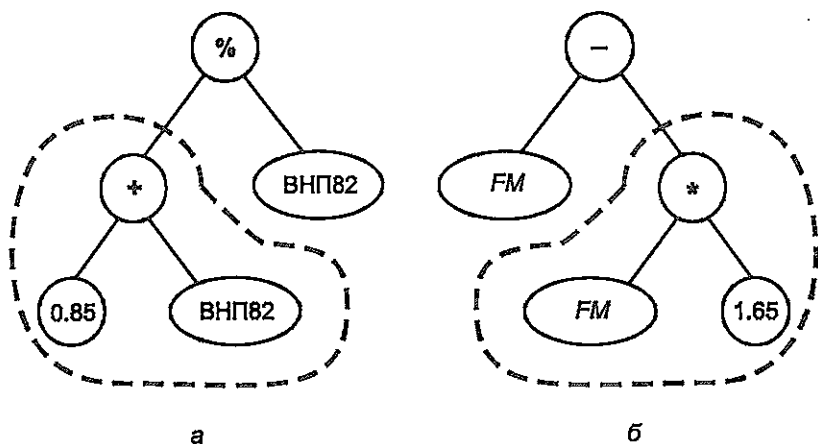


Рис. 6.11. Древоидное представление компьютерных моделей, отобранных для скрещивания:
а – родитель 1; б – родитель 2

Представленные на рис. 6.11 модели соответствуют выражениям $V_i = (0.85 + \text{ВНП82}) / \text{ВНП82}$ и $V_j = 1.65 \text{ FM} - \text{FM}$. Операция скрещивания начинается со случайного и независимого выбора точки кроссинговера в каждой из двух моделей-родителей. Отсеченные фрагменты программ-родителей, обозначенные пунктиром, меняются местами, и в результате образуются две модели-потомка (рис. 6.12). Потомкам соответствуют уравнения $V_k = 1.65 \text{ FM} / \text{ВНП82}$ и $V_l = 0.85 + \text{ВНП82} - \text{FM}$.

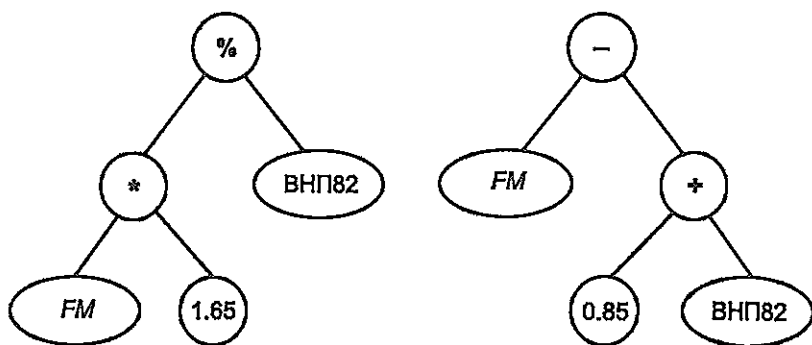


Рис. 6.12. Модели-потомки, полученные в результате скрещивания

Оператор мутации в данном примере выполнялся путем замены функций в узлах деревьев либо путем случайного изменения значений констант.

Используя статистические данные за 30 лет (заметим, что в примере исследовалась экономика США в период с 1959 по 1988 г.), генетический алгоритм за 50 последовательных генераций выдал наилучшее решение $GD = 1.656 FM / \text{ВНП82}$, которое хорошо согласуется с известным эконометрическим уравнением обмена $P = MV/Q$, где P — уровень цен; M — запасы денежной массы; V — скорость обращения денег в экономике; Q — валовой национальный продукт.

6.2.2. ЭВОЛЮЦИОННОЕ ПРОГРАММИРОВАНИЕ

В 1960-х гг. Л. Фогель, А. Оуэнс и М. Уолш предложили схему эволюции логических автоматов, решающих задачи предсказания, диагностики, распознавания и классификации образцов, а также задачи управления объектом с неизвестным характером [12]. Исследования, идейно очень близкие к работам Л. Фогеля с сотрудниками, были разносторонне развиты и описаны в работах И.Л. Букатовой [2, 3]. В более поздних работах Л. Фогеля [20, 21] эволюционное программирование используется для решения систем линейных алгебраических уравнений.

Логические (конечные) автоматы — это модели, описывающие средствами формальной логики возможные переходы исследуемой системы из некоторого начального состояния в заключительное. Удобной формой представления конечных автоматов являются ориентированные графы (рис. 6.13), где вершина q_0 — начальное состояние, q_f — заключительное состояние, q_1, q_2 — промежуточные состояния; $\{0, 1\}$ — символы входного словаря.

Конечные автоматы используются в задачах распознавания, управления и многих других приложениях. Знаменитая машина Тьюринга является разновидностью конечного автомата.

Эволюционная программа реализует моделирование процессов естественной эволюции моделей-автоматов, причем в каждый момент времени сохраняется тот «организм», который наилучшим образом может справиться с данной задачей. «Родительский» организм оценивается в зависимости от способности принимать требуемое решение на основе имеющихся данных. Этот

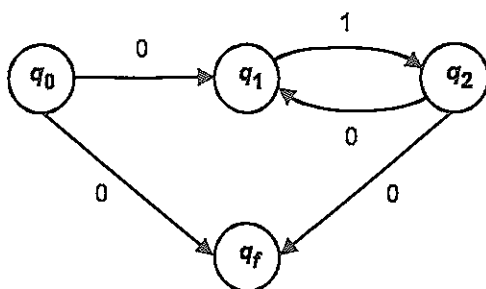


Рис. 6.13. Ориентированный граф, соответствующий конечному автомату

организм подвергается мутации и производит на свет «потомка», которому ставится та же задача и который оценивается таким же образом. Автомат, который демонстрирует наилучшую способность выполнять требуемые функции, сохраняется и поставляет «потомков» в следующее поколение. Таким образом производятся все лучшие и лучшие модели (программы) для решения поставленной задачи. Процесс завершается, когда получена достаточно хорошая программа или исчерпаны ресурсы времени. Всякий раз, когда поступает новая информация, происходит эволюционный поиск логической структуры, обеспечивающей получение наиболее приемлемого решения.

В эволюционном программировании объектами эволюции являются конечные автоматы, способные реагировать на стимулы, поступающие из внешней среды. Каждый автомат на основе текущей информации предсказывает состояние, соответствующее определенному значению функции ценности. Решение ищется постепенным отбором автоматов-родителей, к которым применяется мутация на следующем шаге эволюции.

В эволюционном программировании используются следующие способы реализации оператора мутации:

- изменение заключительного состояния;
- изменение условия перехода из одного состояния в другое;
- добавление нового состояния;
- удаление состояния;
- изменение начального состояния.

Обобщенный алгоритм эволюционного программирования включает следующие шаги.

1. Формулируется постановка задачи. Формируются входной словарь, множество входных и выходных состояний, набор воз-

можных состояний, условия переходов из состояния в состояние, функция ценности для характеристики генерируемых моделей.

2. Случайным образом генерируется начальная популяция конечных автоматов-родителей.

3. Выполняется тестирование автоматов-родителей путем решения поставленной задачи (на вход модели подается заданный образец) и оценка полученных результатов на основе выбранной функции ценности.

4. Отсев неперспективных моделей.

5. На основе случайного применения оператора мутации к автоматам-родителям производятся потомки—члены новой популяции.

6. Тестирование моделей-потомков путем решения поставленной задачи и оценка полученных результатов.

7. Отбор наиболее перспективных потомков.

8. Проверка условий окончания процесса эволюции, в качестве которых могут быть: достижение оптимального значения функции ценности и/или достижение предельных значений, ограничивающих длительность процесса. Если условия завершения эволюции удовлетворены, то переход на шаг 9, в противном случае — возврат на шаг 5, где объекты последней сгенерированной популяции выступают в качестве родителей.

9. Конец алгоритма.

Дальнейшая эволюция автоматов возможна на основе предъявления автоматам более сложных задач.

6.2.3.

ЭВОЛЮЦИОННЫЕ СТРАТЕГИИ

Эволюционные стратегии были предложены в 1970-х гг. [31, 32] в качестве стохастического метода нахождения глобального минимума функций многих переменных $F(X)$, суть которого состоит в следующем. Из случайных векторов решения задачи многокритериальной оптимизации $x = \{x_{ij}\}$, $j = 1, \dots, P$, P — размерность пространства параметров оптимизации, формируется начальная популяция объектов эволюции, над которыми выполняются следующие действия.

1. Из решений x формируются новые объекты — потомки x'_i путем сложения каждой компоненты $x_{ij} = x_{ij} + \xi_{ij}$ со случайной переменной ξ_{ij} , имеющей нормальный закон распределения с нулевым математическим ожиданием.

2. Вычисляются значения целевой функции $F(x_i)$ и $F(x_i')$ и осуществляется выбор наилучшего (минимального) решения, которое отбирается в новую популяцию.

3. Процесс продолжается до тех пор, пока не будет достигнуто приемлемое решение.

Каждый объект в популяции характеризуется двумя векторами — вектором решения и случайным вектором, модифицирующим это решение. Случайный вектор характеризуется вектором дисперсии, который хранится в процессе поиска, и может быть дополнен корректирующим вектором, ускоряющим сходимость алгоритма. Значение ξ_{ij} моделирует величину шага изменения параметров, выбираемую случайным образом. В общем случае ξ_{ij} может принимать любые значения, однако в схеме моделирования эволюционных механизмов величина ξ_{ij} отражает интенсивность мутаций «родителя» и поэтому не слишком велика. Совокупность полученных точек составляет очередное поколение решений, которые оцениваются по значениям минимизируемой функции $F(X)$. В результате отбора одни особи гибнут, а другие живут и размножаются. Эту простую схему легко усовершенствовать, вводя по аналогии с естественными закономерностями зависимость числа порожденных потомков от значений функций ценности «родителей». Соответствующие эволюционные стратегии поиска известны и широко используются на практике. Популяции можно формировать следующими способами:

1) μ родителей порождают λ потомков, все решения борются за выживание и лучшие μ объектов отбираются в следующую популяцию;

2) время жизни объекта ограничено одной генерацией, т.е. μ родителей, произведя λ потомков, погибают. За место в следующей популяции соревнуются только λ потомков, причем в данном способе должно выполняться условие $\lambda > \mu$ (рекомендуемое соотношение $\lambda/\mu = 7$). Такой подход применим к задачам с изменяющимся оптимумом и с зашумленными данными.

В эволюционных стратегиях используется оператор рекомбинации (в эволюционном программировании, в отличие от эволюционных стратегий, рекомбинация не применяется), который аналогичен скрещиванию в генетических алгоритмах. При этом компоненты вектора «потомка» создаются из компонент векторов решений двух «родителей». Это можно сделать разными способами, например:

• компоненты вектора потомка выбираются случайным образом из векторов родителей;

• компоненты вектора потомка получаются как средние арифметические значения компонент обоих родителей, а затем к полученному потомку применяется оператор мутации.

В эволюционных стратегиях иногда применяется *глобальная рекомбинация*, при которой компоненты вектора каждого потомка случайным образом выбираются из векторов всей популяции родителей.

Следует отметить, что моделирование естественных процессов развития, в том числе и эволюции, было и остается одним из самых перспективных научных направлений. Кроме описанных методов эволюционных вычислений, на основе естественных аналогий придуманы нейронные сети, предложены методы эволюционного синтеза систем (см. главу 8) и методы эволюционного проектирования технических объектов. Особенностью подходов, базирующихся на эволюционных аналогиях, является контраст между достаточно простым математическим аппаратом (по сравнению с другими методами) и впечатляющими результатами в области решения слабоструктурированных и плохо обусловленных проблем.

Великий Гёте назвал природу «творцом всех творцов», поэтому разработчикам ИИС еще предстоит очень многому у нее научиться.

Литература

1. *Батищев Д. И.* Генетические алгоритмы решения экстремальных задач: Учеб. пособие. — Воронеж: Изд-во ВГТУ, 1995.
2. *Букатова И. Л.* Эволюционное моделирование и его приложения. — М.: Наука, 1979.
3. *Букатова И. Л.* и др. Эвоинформатика. Теория и практика эволюционного моделирования. — М.: Наука, 1991.
4. *Гудман Э. Д., Коваленко А. П.* Эволюционные вычисления и генетические алгоритмы // Обзорение прикладной и промышленной математики. — М.: ТВП, 1996. Т. 3. — Вып. 5.
5. *Корнеев В. В.* и др. Базы данных. Интеллектуальная обработка информации. — М.: Нолидж, 2000.
6. *Корячко В. П., Курейчик В. М., Норенков И. П.* Теоретические основы САПР. — М.: Энергоатомиздат, 1987.

7. Курейчик В. М. Генетические алгоритмы: Монография. — Таганрог: Изд-во ТРТУ, 1998.
8. Курейчик В. М. Генетические алгоритмы в проектировании СБИС: Учеб. пособие. — Таганрог: Изд-во ТРТУ, 1997.
9. Курейчик В. М. Методы генетического поиска: Учеб. пособие. — Таганрог: Изд-во ТРТУ, 1998.
10. Растринин Л. А. Статистические методы поиска. — М.: Наука, 1968.
11. Стецюра Г. Г. Эволюционные методы в задачах управления, выбора и оптимизации // Приборы и системы управления. — 1998. — № 3.
12. Фогель Л., Оуэнс А., Уолш М. Искусственный интеллект и эволюционное моделирование. — М.: Мир, 1969.
13. Фролов Ю. В. Интеллектуальные системы и управленческие решения. — М.: Изд-во МГПУ, 2000.
14. Холланд Дж. Генетические алгоритмы // В мире науки. — 1992. — № 9-10.
15. Цетлин М. Л. Исследование по теории автоматов и моделирование биологических систем. — М.: Наука, 1969.
16. Ackley D. H. A Connection machine for genetic hillclimbing. — Boston: Kluwer Academic Publishers, USA, 1987.
17. Booker L. B., Goldberg D. E., Holland J. H. Classifier systems and genetic algorithms // Ibid. — 1989. — Vol. 40. — No. 1-3.
18. Branke J., Kohlmorgen U., Sshmeck H. A distributed genetic algorithm improving the generalization behavior of neural networks // LNAI. — 1995. — Vol. 912.
19. Dzerovski S., Petrovski I. Discovering dynamics with genetic programming // LNAI. — Vol. 784.
20. Fogel D. Evolutionary computation. — IEEE press, 1995.
21. Fogel D. B., Atmar J. W. Comparing genetic operators with gaussian mutations in simulated evolutionary process using linear systems // Biol. Cybernetics. — 1990. — Vol. 63.
22. Foundation of Genetic Algorithms / Ed. by rawens gregory. — San Mateo: Morgan Kaufman Publishers, California, USA, 1991.
23. Goldberg David E. Genetic algorithms in search, optimization and machine learning. — Addison-Wesley Publishing Company, Inc. 1989.
24. Gruckles B. P., Party F. E. Genetic algorithms. — Los Alamos: IEEE Computer Society Press, LA, USA, 1992.
25. Handbook of Genetic Algorithms // Ed. by Lawrence Davis, Van Nostrand Reinholds. — New York, 1991.

26. *Holland J. H.* Adaptation in natural and artificial systems. An introductory analysis with application to biology, control and artificial intelligence. — University of Michigan, 1975.
27. *Koza J.* Genetic evolution and co-evolution of computer programs // Artificial life 2. Proceedings of the Workshop on Artificial life, 1990.
28. *Koza J.* Genetic programming // MIT press, 1992.
29. *Koza J.* GP2: Automatic discovery of reusable programs // MIT press, 1994.
30. *Ono N., Rahmani A.* Self-organization of communication in distributed learning classifier systems. Artificial neural nets and genetic algorithms // Proceedings of the International Conference, 1993, p. 361 — 367.
31. *Schwefel H.-P.* Numerical optimization of computer models. — New York: John Wiley, 1981.
32. *Schwefel H.-P.* Evolution and optimum searching. — New York: John Wiley, 1995.

Контрольные вопросы и задания

1. Перечислите основные направления эволюционного моделирования и приведите основные факторы, определяющие неизбежность эволюции.
2. Какие алгоритмы называют генетическими? Сформулируйте основные особенности генетических алгоритмов.
3. Охарактеризуйте простой генетический алгоритм. Приведите пример.
4. Опишите операторы репродукции и кроссинговера в простом генетическом алгоритме. Приведите примеры.
5. Приведите примеры использования простого генетического алгоритма для вычисления функции $f(x) = x^4$ на интервале $[0, 1, 2, 3, 4]$.
6. Составьте примеры, иллюстрирующие работу операторов репродукции, кроссинговера, мутации и инверсии.
7. Дайте характеристику понятию «схема» в простом генетическом алгоритме. Расскажите о назначении и способах использования схем. Приведите примеры.
8. Расскажите о фундаментальной теореме генетического алгоритма.
9. Приведите пример применения фундаментальной теоремы генетического алгоритма.

10. Сформулируйте прикладную экономическую или управленческую оптимизационную задачу и опишите ее решение с применением генетического алгоритма.
11. Расскажите о классифицирующих системах Холланда. Приведите пример.
12. Перечислите основные этапы технологии генетического программирования.
13. В чем особенности эволюционного программирования? Приведите основные шаги обобщенного алгоритма эволюционного программирования.
14. Охарактеризуйте метод эволюционных стратегий. В чем его отличие от эволюционного программирования и от генетических алгоритмов?
15. Расскажите о применении эволюционных вычислений в ИИС. Каким образом применяют ГА для обучения нейронных сетей? Приведите небольшой содержательный пример, демонстрирующий применение ГА для формирования продукционных правил интеллектуальной системы.

ИНТЕЛЛЕКТУАЛЬНЫЕ МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ

... Будущее оставляет мало надежд для тех, кто ожидает, что наши новые механические рабы создадут для нас мир, в котором мы будем освобождены от необходимости мыслить. Помочь они нам могут, но при условии, что наши честь и разум будут удовлетворять требованиям самой высокой морали.

Норберт Винер

Интеллектуальные мультиагентные системы — одно из новых перспективных направлений искусственного интеллекта, которое сформировалось на основе результатов исследований в области распределенных компьютерных систем, сетевых технологий решения проблем и параллельных вычислений. В мультиагентных технологиях заложен принцип автономности отдельных частей программы (агентов), совместно функционирующих в распределенной системе, где одновременно протекает множество взаимосвязанных процессов. Под *агентом* подразумевают автономный искусственный объект (компьютерную программу), обладающий активным мотивированным поведением и способный к взаимодействию с другими объектами в динамических виртуальных средах. Каждый агент может принимать сообщения, интерпретировать их содержание и формировать новые сообщения, которые либо передаются на «доску объявлений», либо направляются другим агентам.

Агентно-ориентированный подход уже нашел применение в таких областях, как распределенное решение сложных задач, реинжиниринг предприятий, электронный бизнес и т.п. Важной областью применения мультиагентных технологий является моделирование. В этой области Д.А. Поспелов [9] выделяет два класса задач. К первому классу он относит задачи распределенного управления и задачи планирования достижения целей, где

усилия разных агентов направлены на решение общей проблемы и необходимо обеспечение эффективного способа кооперации их деятельности. В задачах второго класса агенты самостоятельно решают свои локальные задачи, используя общие, как правило, ограниченные ресурсы.

7.1. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ АГЕНТОВ

Понятие *агент* соответствует аппаратно или программно реализованной сущности, которая способна действовать в интересах достижения целей, поставленных перед ней владельцем и/или пользователем [3, 12, 23].

В мультиагентных системах (МАС) множество автономных агентов действуют в интересах различных пользователей и взаимодействуют между собой в процессе решения определенных задач. Примерами таких задач являются: управление информационными потоками и сетями, управление воздушным движением, поиск информации в сети Интернет, электронная коммерция, обучение, электронные библиотеки, коллективное принятие многокритериальных управленческих решений и другие.

Идея мультиагентных систем появилась в конце 1950-х гг. в научной школе М.Л. Цетлина, которая занималась исследованием коллективного поведения автоматов [14]. Агентами (*маленькими животными*) были названы искусственные существа, обладающие свойством *реактивности*, т. е. способные воспринимать и интерпретировать сигналы, поступающие из внешней среды, и формировать ответные сигналы. В роли маленьких животных выступали конечные автоматы, которые не имели априорных знаний о свойствах окружающей среды и о наличии в ней других существ. Единственным знанием, которым они обладали, была цель их деятельности и способность оценивать поступающие сигналы относительно достижения этой цели. Оказалось, что даже такие простые структуры, как конечные автоматы (см. разд. 6.2.2), демонстрируют хорошие способности к адаптации в стационарных вероятностных средах. Одной из главных характеристик агентов-автоматов была *рациональность*, которая определялась как сумма положительных откликов среды, накопленных

агентом за некоторый период его существования. В дальнейших исследованиях структура *маленьких животных* усложнялась. Сначала появились вероятностные автоматы с переменной структурой, адаптирующейся к характеристикам среды, затем появились агенты, способные изменять свои реакции на основании предыстории и анализа состояния окружения. Серьезным шагом в развитии мультиагентных технологий стала реализация способности агентов к рассуждениям [7, 12]. Простейшие модели взаимодействия агентов предусматривали их общение через среду. При этом на каждом шаге функционирования агенты совершают выбор возможных для них действий. Множество действий всех агентов обуславливает распределение откликов среды для всех участников, которые могут его использовать либо не использовать при формировании своих ответных реакций.

Новый шаг к современному пониманию агентов был сделан при переходе к коллективной работе в распределенных компьютерных системах. Этот шаг стал началом бурного развития мультиагентных технологий. К настоящему времени в данном направлении накоплен определенный опыт. Предложены разнообразные модели агентов и способы их реализации, решены практические задачи и созданы инструментальные средства для разработки мультиагентных систем, сформулированы различные принципы взаимодействия агентов и т. п. В этой главе мы остановимся на вопросах, связанных с построением и применением интеллектуальных МАС.

Одна из возможных классификаций агентов [3, 19] приведена в табл. 7.1, из которой следует, что для *интеллектуальных агентов характерно целесообразное поведение*, которое предполагает наличие у агента целей функционирования и способностей использовать знания об окружающей среде, партнерах и о своих возможностях.

7.1.1.

ХАРАКТЕРИСТИКИ ИНТЕЛЛЕКТУАЛЬНЫХ АГЕНТОВ

Интеллектуальным агентам присущи следующие основные свойства:

- автономность — способность функционировать без вмешательства со стороны своего владельца и осуществлять контроль собственных действий и внутреннего состояния. Автономность

Классификация агентов

Признак	Тип агента			
	простой	смысловый	интеллектуальный	действительно интеллектуальный
Автономность	+		+	+
Взаимодействие с другими агентами и/или пользователями	+	+	+	+
Реактивность	+	+	+	+
Способность использования абстракции		+	+	+
Адаптивное поведение		+	+	+
Обучение на основе взаимодействия с окружением			+	+
Толерантность к ошибкам и/или неверным входным сигналам			+	
Функционирование в режиме реального времени			+	
Взаимодействие на естественном языке			+	

предполагает относительную независимость агента от окружающей среды, т.е. наличие «свободы воли», обуславливающей собственное поведение, которое должно быть обеспечено необходимыми ресурсами;

- активность — способность к организации и реализации действий;

- общительность — взаимодействие и коммуникация с другими агентами;

- реактивность — адекватное восприятие состояния среды и реакция на его изменение;

- целенаправленность, предполагающая наличие собственных источников мотивации;

- наличие базовых знаний о себе, о других агентах и об окружающей среде;

- убеждения — переменная часть базовых знаний, меняющихся во времени;

- желания — стремление к определенным состояниям;

- намерения — действия, которые планируются агентом для выполнения своих обязательств и/или желаний;

- обязательства — задачи, которые выполняет один агент по просьбе и/или поручению других агентов.

Иногда к этому списку добавляются другие качества, в том числе:

- правдивость — неспособность к подмене истинной информации заведомо ложной;

- благожелательность — готовность к сотрудничеству с другими агентами в процессе решения собственных задач, что обычно предполагает отсутствие конфликтующих целей, поставленных перед агентами;

- альтруизм — приоритетность общих целей по сравнению с личными;

- мобильность — способность агента мигрировать по сети в поисках необходимой информации.

В работе [12] для классификации агентных программ используются два основных признака: 1) степень развития внутреннего представления о внешнем мире; 2) способ поведения.

По первому признаку выделяются интеллектуальные (когнитивные, рассуждающие) и реактивные агенты. Интеллектуальные агенты обладают хорошо развитой и пополняемой символической моделью внешнего мира благодаря наличию у них БЗ, механизмов рассуждения и анализа действий. Реактивные агенты не имеют развитого представления о внешней среде. Они не используют рассуждений и могут не иметь собственных ресурсов. Их поведение определяется целью, в соответствии с которой формируются реакции на предъявляемые ситуации. В связи с этим реактивные агенты не имеют внутренних источников мотивации и не способны планировать свои действия (реактивность в чистом виде — это обратная связь без прогноза).

7.1.2. АРХИТЕКТУРЫ МУЛЬТИАГЕНТНЫХ СИСТЕМ

Интеллектуальная мультиагентная система представляет собой множество интеллектуальных агентов, распределенных в сети, которые мигрируют по ней в поисках релевантных данных, знаний, процедур и кооперируются для достижения поставленных перед ними целей.

В зависимости от концепции, принятой при разработке МАС, возможны различные варианты ее архитектуры, среди которых выделяют три базовых типа:

- 1) архитектуры, основанные на методах работы со знаниями;
- 2) архитектуры, в которых используются поведенческие модели «стимул-реакция»;
- 3) гибридные архитектуры.

В архитектурах *первого* типа для представления и обработки знаний используются традиционные модели, методы и средства искусственного интеллекта, а принятие решений осуществляется на основе механизмов формальных рассуждений. В самых первых системах такого типа для представления и обработки знаний использовалась логика предикатов первого порядка. Развитие исследований в этой области привело к появлению специальных расширений логических исчислений, ориентированных на учет таких свойств агентов, как убеждения, желания, намерения и обязательства [9, 12]. Основной недостаток архитектур первого типа — сложность или принципиальная невозможность построения достаточно полных баз знаний, которые являются необходимой частью создаваемых систем. В частности, интеллектуальный агент может иметь архитектуру типичной продукционной системы, которая способна воспринимать информацию из внешней среды и осуществлять те или иные действия в результате обработки этой информации. Главные отличия агентной программы от обычной продукционной ЭС связаны с наличием механизма формирования целей и модуля коммуникации, который обеспечивает взаимодействие с другими агентами. Агент с такой архитектурой способен к рассуждениям, но не способен к обучению. Адаптивное поведение агента позволяет реализовать архитектура на основе классифицирующих систем Дж. Холланда (см. рис. 6.6, 6.7). Важнейшими отличиями классифицирующих систем от продукционных являются: 1) возможность формирования новых правил с применением генетического алгоритма; 2) наличие механизма поощрений.

В архитектурах *второго* типа, которые называют *реактивными*, не используются традиционные для ИИ символичные модели представления знаний [16]. Модели поведения агентов представлены либо наборами правил, которые позволяют выбрать *действие*, соответствующее *ситуации*, либо конечными автоматами, либо другими средствами, обеспечивающими формирование адек-

ватных реакций агента на возникающие в системе стимулы. Системы этого типа, как правило, имеют высокую степень специализации и строгие ограничения на сложность решаемых задач.

Наиболее перспективными считаются *гибридные* интеллектуальные мультиагентные системы, которые позволяют использовать возможности интеллектуальных и реактивных архитектур. Примером может служить архитектура с иерархической базой знаний, которая содержит структурированную БЗ, рабочую память, модуль управления коммуникацией и человеко-машинный интерфейс. Агент с подобной архитектурой обладает способностью к рассуждениям и к реактивному поведению. Его БЗ содержит три уровня: 1) знания предметной области; 2) знания о взаимодействии, которые позволяют принимать решения в условиях неопределенности; 3) управляющие знания. Интеллектуальное поведение агента обеспечивается способностью принимать решения, а реактивное — системой контроля за содержимым рабочей памяти, которая функционирует по принципу глобальной доски объявлений. Агент взаимодействует с пользователем, используя человеко-машинный интерфейс. В общем случае гибридные архитектуры являются многоуровневыми и отличаются друг от друга структурой и содержанием уровней, которые могут соответствовать различным уровням управления, абстракции либо отдельным функциональным свойствам агента.

Одно из новых направлений — применение нейронных сетей для реализации МАС. Коннекционистские архитектуры (на основе ИНС) позволяют создавать самообучающихся агентов, знания которых формируются в процессе решения практических задач. Хорошие перспективы для реализации самообучающихся агентов имеют сети с обратными связями и нечеткие ИНС [12].

7.2. КОЛЛЕКТИВНОЕ ПОВЕДЕНИЕ АГЕНТОВ

Главная черта МАС, отличающая их от других интеллектуальных систем, — взаимодействие между агентами. Взаимодействие означает установление двусторонних и многосторонних динамических отношений между субъектами. Оно является не только следствием деятельности агентов, но и необходимым условием формирования виртуальных сообществ. Взаимодействие — не

просто связь между сосуществующими агентами, но и предпосылка для взаимных превращений самих агентов и отношений между ними.

Главными характеристиками любого взаимодействия являются направленность, избирательность, интенсивность и динамичность. В контексте МАС эти понятия можно интерпретировать следующим образом:

- *направленность* — положительная или отрицательная; кооперация или конкуренция; сотрудничество или конфронтация; координация или субординация и т. п.;

- *избирательность* — взаимодействие происходит между агентами, которые каким-либо образом соответствуют друг другу и поставленной задаче. При этом агенты могут быть связаны в одном отношении и независимы — в другом;

- *интенсивность* — взаимодействие между агентами не сводится к наличию или отсутствию, а характеризуется определенной силой;

- *динамичность* — наличие, сила и направленность взаимодействий могут изменяться с течением времени.

Общая проблема анализа взаимодействия между агентами включает следующие задачи [12]:

- 1) идентификацию ситуации взаимодействия агентов;
- 2) выделение основных ролей и их распределение между агентами;
- 3) определение числа и типов взаимодействующих агентов;
- 4) построение формальной модели взаимодействия;
- 5) определение набора возможных стратегий поведения агентов;
- 6) формирование множества коммуникативных действий.

7.2.1. СПОСОБЫ И ПРИЧИНЫ ВЗАИМОДЕЙСТВИЯ МЕЖДУ АГЕНТАМИ

К базовым видам взаимодействия между агентами относятся:

- *кооперация* (сотрудничество);
- *конкуренция* (конфронтация, конфликт);
- *компромисс* (учет интересов других агентов);
- *конформизм* (отказ от своих интересов в пользу других);
- *уклонение от взаимодействия*.

Интеллектуальные агенты сотрудничают с другими агентами «сознательно», преследуя при этом определенные цели. Кооперацию в сообществе реактивных агентов можно назвать непреднамеренной, поскольку она базируется на естественных реакциях отдельных агентов, направленных на выживание вида. Показатели выживания отражают способность особи или группы сохранять свою целостность при воздействиях факторов, которые могут ее разрушить. Кооперация между агентами может возникать на принудительных началах (*директивная* кооперация) или на основе добровольных отношений (*ситуативная* кооперация). Эти два вида сотрудничества часто представлены так называемой *контрактной* формой кооперации, когда взаимодействие агентов регламентируется набором формальных или неформальных соглашений между ними.

Взаимодействие агентов обусловлено целым рядом причин, важнейшими среди которых являются следующие.

Совместимость целей (общая цель). Эта причина обычно порождает взаимодействие по типу кооперации или сотрудничества. При этом следует выяснить, не ведет ли взаимодействие к снижению *жизнеспособности* отдельных агентов. Несовместимость целей или убеждений обычно порождает конфликты, позитивная роль которых заключается в стимулировании процессов развития. Известная модель *хищник-жертва* представляет собой пример одновременного взаимодействия по двум типам *кооперация-конфронтация*.

Отношение к ресурсам. Ресурсами будем называть любые средства, используемые для достижения агентами своих целей. Задачи распределения долей рынка, затрат и прибылей совместных предприятий можно рассматривать как примеры взаимодействия, обусловленного общими ресурсами. Ограниченность ресурсов, которые используются многими агентами, обычно порождает конфликты. Одним их самых простых и эффективных способов разрешения подобных конфликтов является *право сильного* — сильный агент отбирает ресурсы у слабых. Более тонкие способы разрешения конфликтов обеспечивают переговоры [13], направленные на достижение компромиссов, в которых учитываются интересы всех агентов.

Необходимость привлечения недостающего опыта. Каждый агент обладает ограниченным набором знаний, необходимых ему для реализации собственных и общих целей. В связи с этим ему приходится взаимодействовать с другими агентами. При этом

возможны различные ситуации: а) агент способен выполнить задачу самостоятельно; б) агент может обойтись без посторонней помощи, но кооперация позволит решить задачу более эффективным способом; в) агент не способен решить задачу в одиночку. В зависимости от ситуации агенты выбирают тип взаимодействия и могут проявлять разную степень заинтересованности в сотрудничестве.

Взаимные обязательства. Обязательства являются одним из инструментов, позволяющих упорядочить хаотические взаимодействия агентов. Они позволяют предвидеть поведение других агентов, прогнозировать будущее и планировать собственные действия. Можно выделить следующие группы обязательств: а) обязательства перед другими агентами; б) обязательства агента перед группой; в) обязательства группы перед агентом; г) обязательства агента перед самим собой. Формальное представление целей, обязательств, желаний и намерений, а также всех остальных характеристик составляет основу ментальной модели интеллектуального агента, которая обеспечивает его мотивированное поведение в автономном режиме.

Перечисленные причины в различных сочетаниях могут приводить к разным формам взаимодействия между агентами, например:

- простое сотрудничество, которое предполагает интеграцию опыта отдельных агентов (распределение задач, обмен знаниями и т. п.) без специальных мер по координации их действий;
- координируемое сотрудничество, когда агенты вынуждены согласовывать свои действия (иногда привлекая специального агента-координатора) для того, чтобы эффективно использовать ресурсы и собственный опыт;
- непродуктивное сотрудничество, когда агенты совместно используют ресурсы или решают общую проблему, не обмениваясь опытом и мешая друг другу (как лебедь, рак и щука в басне И.А. Крылова).

7.2.2.

МОДЕЛИРОВАНИЕ ВЗАИМОДЕЙСТВИЯ В МУЛЬТИАГЕНТНЫХ СИСТЕМАХ

Рассматривая проблему моделирования взаимодействия агентов друг с другом и с окружающей средой, Д. А. Поспелов [2] выделил следующие основные признаки естественных систем,

которые необходимо учитывать при моделировании виртуальных сред.

1. Конечность времени существования любого агента. Длительность *жизни* агента зависит от различных обстоятельств, в частности от поставленной перед ним задачи, от величины доступных ресурсов и т. п.

2. Использование механизма биологического отбора в моделях *искусственной жизни*. Естественный отбор эффективных агентов может осуществляться в адаптивных системах с использованием различных эволюционных механизмов (обучаемых нейронных сетей, генетических алгоритмов, автоматов с перестраиваемой структурой и т. д.).

3. Учет уровня организации сообщества агентов. Если модель описывает взаимодействие сложных организмов, имеющих социальную организацию, то помимо реактивности, активности и когнитивности (способность к рассуждениям) агенты приобретают еще одно свойство — *социальность*. В таких моделях возникает необходимость учета социального статуса и социальных отношений. Распределение труда в обществе служит основой для выделения классов агентов, выполняющих специализированные функции, в том числе функции управления искусственной средой. Задача распределения функций приводит к необходимости реализации механизма *социального отбора*, который принципиально отличается от биологического принципа.

Вопросы организации сообщества искусственных организмов по образу и подобию человеческого общества связывают теорию МАС с системным анализом, теорией организаций, теорией административного управления и т. п. Серьезной и пока не решенной проблемой является морально-этическая основа организации мультиагентных систем, связанная с формированием понятий об основных ценностях и нормах, принятых в обществе. Ориентация на модели *нормативного поведения* агентов вызывает дискуссии, так как наряду с нормативным в реальном обществе имеет место и ненормативное поведение [9].

Коллективное поведение агентов в МАС предполагает кооперацию агентов при коллективном решении задач. В процессе работы мультиагентной системы агент может обращаться за помощью к другим агентам, если не в состоянии решить поставленную перед ним задачу самостоятельно. При этом агенты могут строить планы совместных действий, не только полагаясь на свои

возможности, но анализируя планы и намерения других членов коллектива. Моделирование коллективного поведения необходимо также в случаях, когда агенты для решения своих задач используют общий ограниченный ресурс. Каждый агент вынужден учитывать наличие других агентов, а выбор стратегии действий одного агента обычно зависит от поведения остальных.

Проблемы коллективного поведения рассматриваются в теории систем, в теории управления и в теории игр. Основной идеей системного анализа является применение декомпозиции исходной задачи на более простые, из решения которых может быть найдено решение задачи в целом. В мультиагентных системах идея декомпозиции воплотилась в принцип распределенного решения подзадач с их координацией для получения стратегии коллективного поведения.

В процессе моделирования коллективной работы агентов возникает множество проблем [4]:

- распознавание необходимости кооперации;
- выбор подходящих партнеров;
- возможность учета интересов партнеров;
- организация переговоров о совместных действиях;
- формирование планов совместных действий;
- синхронизация совместных действий;
- декомпозиция задач и разделение обязанностей;
- выявление конфликтующих целей;
- конкуренция за совместные ресурсы;
- формирование правил поведения в коллективе;
- обучение поведению в коллективе и т. д.

Особенностью коллективного поведения агентов является то, что их взаимодействие в процессе решения частных задач (или одной общей) порождает новое качество решения этих задач. При этом в моделях координации поведения агентов используются следующие основные идеи [4].

1. Отказ от поиска наилучшего решения в пользу «хорошего», что приводит к переходу от процедуры строгой оптимизации к поиску приемлемого компромисса, реализующего тот или иной принцип координации.

2. Использование самоорганизации в качестве устойчивого механизма формирования коллективного поведения.

3. Применение рандомизации (случайно-вероятностного способа выбора решений) в механизмах координации для разрешения конфликтов.

4. Реализация рефлексивного управления [6], сущность которого заключается в том, чтобы заставить субъекта *осознанно* подчиняться влиянию извне, т. е. сформировать у него такие желания и намерения (интенции), которые совпадают с требованиями окружения.

7.2.3. КООРДИНАЦИЯ ПОВЕДЕНИЯ АГЕНТОВ В МУЛЬТИАГЕНТНОЙ СИСТЕМЕ

Наиболее известными моделями координации поведения агентов являются: теоретико-игровые модели, модели коллективного поведения автоматов, модели планирования коллективного поведения, модели на основе BDI-архитектур (Belief — Desire — Intention), модели координации поведения на основе конкуренции.

Теоретико-игровые модели. Предметом теории игр являются задачи выбора решений в условиях неопределенности и конфликта. Наличие конфликта предполагает существование как минимум двух участников, которых называют игроками. Множество решений, возможных для выбора каждым игроком, называется стратегией. *Равновесными точками* игры (оптимальными решениями) называют такие состояния, когда ни одному из игроков невыгодно менять свою позицию. Понятие равновесия оказалось весьма полезным в теории МАС, поскольку механизм поиска равновесных ситуаций может использоваться как средство самоорганизации коллективного поведения агентов. Следствием подобной интерпретации является подход, в котором необходимые атрибуты коллективного поведения агентов обеспечиваются путем конструирования правил игры. Кроме того, на основе развития теории игр в области МАС предпринимаются попытки построения эффективных, устойчивых, полностью распределенных протоколов переговоров, направленных на координацию коллективного поведения агентов.

В работе [24] множество возможных ситуаций выбора поведения пары агентов классифицируется следующим образом.

1. *Симметричная кооперация*, когда существует непустое множество стратегий (переговорное множество); при использовании которых оба агента достигают своих целей и получают больший эффект, чем в ситуациях, когда они действуют поодиночке.

2. *Симметричный компромисс*, когда достижение цели в одиночку более выгодно для каждого агента, однако невозможно в присутствии другого агента.

3. *Несимметричная кооперация* или *несимметричный компромисс* — один из агентов может самостоятельно достичь своей цели в присутствии другого агента, а другой — только за счет кооперации с первым.

4. *Конфликт* — переговорное множество пусто, т. е. не существует стратегий, обеспечивающих достижение целей обоих агентов.

В этой же работе показано, что теоретико-игровые модели позволяют для всех перечисленных случаев сконструировать наборы правил переговоров, следуя которым агенты придут к некоторому соглашению, отвечающему состоянию равновесия. Это достигается за счет использования множества дополнительных предположений и специальных приемов. Например, кроме *стоимости* цели в рассмотрение вводится понятие *ценности* цели, а в качестве одной из возможных стратегий может выступать стратегия манипулирования информацией о ценности целей (т. е. агенты могут сообщать друг другу заведомо ложные значения). При этом «нечестные» агенты могут либо увеличить свой доход, либо освободиться от части своей работы.

Модели коллективного поведения автоматов. Они основаны на идеях рандомизации, самоорганизации и полной распределенности [1, 14]. Модели этого типа подходят для построения протоколов переговоров в задачах, которые характеризуются большим количеством очень простых взаимодействий с неизвестными характеристиками.

Модели планирования коллективного поведения. Планирование может быть централизованным, частично централизованным или распределенным (децентрализованным). В последнем случае агенты сами принимают решения о выборе своих действий в процессе координации частных планов, в связи с чем возникают вопросы о рациональной децентрализации, о возможности изменения целей при возникновении конфликтов, а также проблемы вычислительной сложности.

Модели на основе BDI-архитектур [12, 17]. В моделях этого класса применяются аксиоматические методы теории игр и логической парадигмы искусственного интеллекта. Для описания агентов используются логические средства, в том числе темпоральные и модальные логики. Акцент делается на описании ин-

тенсиональных понятий, таких, как убеждения (belief), желания (desire) и намерения (intention). Задача координации поведения агентов решается путем согласования результатов логического вывода в базах знаний отдельных агентов, полученных для текущего состояния внешней среды, в которой действуют агенты. Логический вывод осуществляется непосредственно в процессе функционирования агентов, что приводит к высокой сложности моделей, вычислительным трудностям и к проблемам, связанным с аксиоматическим описанием нетривиальных ситуаций, например, когда перед агентом возникает выбор между решением собственной задачи и выполнением обязательств по отношению к партнерам.

Модели на основе конкуренции. В моделях данного класса используется понятие *аукцион* в качестве механизма координации поведения агентов. Использование механизма аукциона основано на предположении о возможности явной передачи «полезности» от одного агента к другому или к агенту-аукционеру, причем эта полезность обычно имеет смысл денег [4].

Аукционы принято разделять на открытые и закрытые. В первом случае предлагаемые цены объявляются всем участникам. В закрытом аукционе о предлагаемых ценах знает только аукционер. Открытые аукционы различаются по способу проведения. В так называемых английских аукционах обычно задается стартовая цена, которая может увеличиваться участниками в ходе торгов. Побеждает тот, кто даст максимальную цену. Голландский аукцион начинается с верхней цены, которая постепенно снижается. Победителем считается тот, кто дал наибольшую текущую цену. Закрытые аукционы разделяют на аукционы первой и второй цены. В аукционах первой цены побеждает тот, кто предложил самую высокую цену, известную только аукционеру. В аукционах второй цены победитель определяется таким же способом, но платит за товар не свою цену, а вторую по величине. Теоретически доказано, что все разновидности аукционов эквивалентны для аукционера, однако практика показывает иное. Например, если участники аукциона не склонны к риску, то аукционер стимулирует повышение цены продажи при проведении голландского аукциона первой цены. Существуют варианты «групповых» аукционов, когда один или несколько участников представляют интересы группы, и в случае выигрыша проводится аукцион внутри группы. При этом на внутреннем аукционе товар продается

по более высокой цене по сравнению с ценой внешнего аукциона. Полученная разница делится между участниками группы.

Сам по себе механизм аукциона не затрагивает способов принятия решений участниками. Решения могут приниматься на основе некоторой модели рассуждений, которая может использовать различные типы знаний, доступных агентам, и разнообразные способы их обработки.

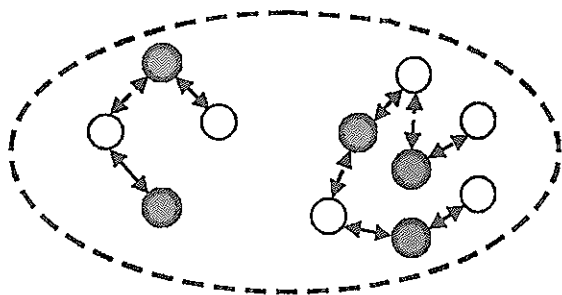
Аукцион всегда должен заканчиваться. Для этого в стратегии его проведения должны быть заложены средства для разрешения возможных конфликтов (например, при наличии нескольких победителей). Одним из самых простых способов разрешения конфликтов является рандомизация, когда применяется случайный механизм выбора.

7.3. ПРИМЕРЫ МУЛЬТИАГЕНТНЫХ СИСТЕМ

Рассмотрим практические примеры организации взаимодействия в мультиагентных системах с использованием различных механизмов координации поведения.

7.3.1. КООРДИНАЦИЯ ПОВЕДЕНИЯ НА ОСНОВЕ МОДЕЛИ АУКЦИОНА

Электронный магазин. Рассмотрим типичную задачу электронной коммерции [4], в которой участвуют агенты-продавцы и агенты-покупатели (рис. 7.1). Торговля осуществляется в электронном магазине, который представляет собой программу, размещенную на сервере. Ее основным назначением является организация взаимодействия агентов, интересы которых совпадают. Агенты действуют по поручению своих персональных пользователей. При этом агенты-продавцы стремятся продать свой товар по максимально возможной цене, а агенты-покупатели стремятся купить нужный товар по минимальной цене. Оба вида агентов действуют автономно и не имеют целей кооперации. Электронный магазин регистрирует появление и исчезновение агентов и организует контакты между ними, делая их «видимыми» друг для друга.



• агент-продавец ○ — агент-покупатель $\leftrightarrow$ — процесс переговоров

Рис. 7.1. Схема электронного магазина

Поведение агента-продавца характеризуется следующими параметрами:

- желаемая дата, до наступления которой необходимо продать товар;

- желаемая цена, по которой пользователь хочет продать товар;

- самая низкая допустимая цена, ниже которой товар не продается;

- функция снижения цены во времени (линейная, квадратичная и др.);

- описание продаваемого товара.

Агент-покупатель имеет «симметричные» параметры:

- крайний срок покупки товара;

- желаемая цена покупки;

- самая высокая приемлемая цена;

- функция роста цены во времени;

- описание покупаемого товара.

Торги ведутся по схеме закрытого аукциона первой цены. Поведение агентов описывается простой моделью, в которой не используются знания и рассуждения. Агент-продавец, получив от электронного магазина информацию о потенциальных покупателях своего товара, последовательно опрашивает их всех с целью принять решение о возможности совершения сделки. Сделка заключается с первым агентом-покупателем, который готов дать за товар запрашиваемую цену. Продавец не может вторично вступить в контакт с любым покупателем до тех пор, пока не опросит всех потенциальных покупателей. При каждом контакте агент-прода-

вещ ведет переговоры, предлагая начальную цену либо снижая ее. Агент-покупатель действует аналогичным образом, отыскивая продавцов нужного товара и предлагая им свою цену покупки, которую он может увеличить в процессе переговоров. Любая сделка завершается только в случае ее одобрения пользователем агента.

Данная схема переговоров представляет собой простейший случай взаимодействия автономных агентов, действующих реактивно. Тем не менее итоговое поведение системы вполне адекватно реальности.

Виртуальное предприятие. Создание виртуальных предприятий является одним из современных направлений бизнеса, которое в значительной мере стимулируется быстрым ростом информационных ресурсов и услуг, предоставляемых в сети Интернет. Кроме того, появлению виртуальных предприятий способствует сокращение времени жизненного цикла создаваемых изделий и повышение уровня их сложности, так как при этом возникает необходимость оперативного объединения производственных, технологических и интеллектуальных ресурсов. Еще одна немаловажная причина — ужесточение конкуренции на товарных рынках, стимулирующее объединение предприятий в целях выживания.

Виртуальное предприятие можно определить как кооперацию юридически независимых предприятий, организаций и индивидуумов, которые производят продукцию или услуги в общем бизнес-процессе. Во внешнем мире виртуальное предприятие выступает как единая организация, в которой используются методы управления и администрирования, основанные на применении информационных и телекоммуникационных технологий. Целью создания виртуального предприятия является объединение производственных, технологических, интеллектуальных и инвестиционных ресурсов для продвижения на рынок новых товаров и услуг.

Поскольку каждое реальное предприятие в рамках виртуального выполняет только часть работ из общей технологической цепочки, то при его создании решаются две главные задачи. Первая — это декомпозиция общего бизнес-процесса на компоненты (подпроцессы). Вторая задача заключается в выборе рационального состава реальных предприятий-партнеров, которые будут осуществлять технологический процесс. Первая задача решается с применением методов системного анализа, а для решения второй могут применяться средства мультиагентных технологий.

Задача оптимального распределения множества работ (подпроцессов) среди множества работников (реальных предприятий) в исследовании операций формулируется как задача о назначениях [5]. Ее решение начинается с формирования множеств подпроцессов и потенциальных предприятий-участников. Затем строятся возможные отображения из множества участников на множество подпроцессов и делается выбор наиболее приемлемого отображения, которое соответствует конкретным назначениям предприятий на бизнес-процессы. Для этого можно использовать механизм аукциона. На рис. 7.2 приведена схема аукциона по созданию виртуального предприятия, в котором выделены бизнес-процессы A, B, C, D, E и участвуют четыре предприятия: P_1, P_2, P_3, P_4 , претендующие на их реализацию. Каждое из предприятий представлено интеллектуальным агентом, при этом одно из них (P_1) выступает в роли инициатора (аукционера).

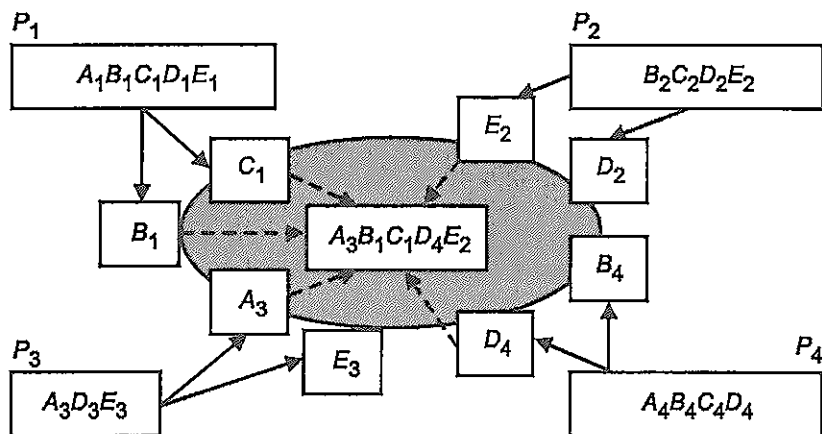


Рис. 7.2. Схема создания виртуального предприятия

Перед началом аукциона аукционер (менеджер) формирует базу данных и базу знаний об участниках аукциона. Затем он выставляет на продажу отдельные бизнес-процессы, информация о которых представлена стартовой ценой и требованиями по заданному набору показателей. Каждый претендент выдвигает свои предложения по параметрам, которые он в состоянии обеспечить, и свою цену. Собрав и обработав эти предложения, аукцио-

нер с помощью некоторой модели рассуждения упорядочивает потенциальных претендентов с учетом собственной информации о них. После этого он принимает решение о выборе назначений или отвергает их и выдвигает новые предложения.

Следует отметить, что задачу создания виртуального предприятия можно отнести к задачам структурного синтеза сложных систем, удовлетворяющих заданным требованиям. Подходы к решению таких задач рассмотрены в главе 8.

7.3.2.

МУЛЬТИАГЕНТНАЯ СИСТЕМА ДЛЯ ПОДДЕРЖКИ ПРОЦЕССОВ ПРИНЯТИЯ РЕШЕНИЙ НА ПРЕДПРИЯТИИ

Интеллектуальные мультиагентные системы принятия решений предназначены для оценки качества организационно-технических и экономических решений в процессе деятельности предприятия.

В настоящее время происходит переход от концепции стабильного бизнеса к *мобильному*, в котором главную роль играют понятия *конкурентоспособность* и *гибкость*. Для работы в новых быстроизменяющихся условиях предприятиям необходимо постоянно трансформировать свои производственные структуры и структуры бизнес-процессов. При этом становится неизбежным привлечение сторонних специалистов из области технологий, маркетинга, реинжиниринга и т. д. Оценка предлагаемых решений является сложным и постоянным видом деятельности, требующим участия высококвалифицированных экспертов из разных областей знаний, которые, как правило, территориально удалены друг от друга. Этим обусловлена актуальность распределенной компьютерной поддержки процессов принятия решений на предприятиях, которая может быть реализована с применением мультиагентных систем.

Рассмотрим пример мультиагентной системы принятия решений для многокритериальной оценки инновационной деятельности предприятия [11].

Общая схема принятия решений включает следующие этапы:

- 1) спецификация требований;
- 2) генерация решений;
- 3) оценка альтернатив;
- 4) выбор эффективного решения.

Оценку решений проводит рабочая группа, которая состоит из руководителя, аналитика и экспертов. Функции между участниками рабочей группы распределяются следующим образом. Руководитель формирует набор показателей (критериев), которые будут использоваться для оценки проектов (решений); подбирает состав группы экспертов; составляет персональный календарь, в соответствии с которым эксперты выполняют свои задания. Каждый эксперт работает по индивидуальному сценарию, предложенному руководителем. Аналитик, функции которого может выполнять руководитель, высказывает свое мнение о результатах проведенной экспертами работы.

Для поддержки группового процесса принятия решений используется программная реализация метода анализа иерархий [10], где реализованы следующие основные процедуры:

- формирование и согласование иерархической структуры показателей;
- оценка и согласование качественных показателей проекта;
- оценка и согласование важности показателей;
- ранжирование альтернативных решений и согласование результатов.

В решении перечисленных задач участвует множество экспертов, поэтому на каждом этапе предусмотрены процедуры согласования их мнений.

Ядром мультиагентной системы «Multi Expert» (рис. 7.3) является *менеджер знаний*, использующий три внешних компонента:

- информационную модель проблемной области в виде упорядоченного набора показателей качества решений;
- средства технической и программной поддержки;
- множество типов пользователей (руководитель, координатор, эксперт, аналитик).

Для координации работы коллектива экспертов используется двухуровневый механизм согласования. Каждый из экспертов представлен агентом, в задачу которого входит оценка предлагаемых руководителем альтернатив по заданному набору показателей качества. С помощью редактора знаний руководитель формирует задания экспертам и проводит анализ полученной от них информации. Задача координации поведения агентов возложена на агента-координатора. Результатом работы системы являются согласованные экспертные оценки, на основании которых проводится многокритериальное ранжирование альтернатив.



Рис. 7.3. Обобщенная структура мультиагентной системы принятия решений «Multi Expert»

Рассмотрим основные функции агентов в системе «Multi Expert».

Агент-руководитель:

- предоставляет набор процедур для облегчения работы руководителя в распределенной системе;
- вычисляет конечный результат на основании данных, полученных от других агентов;
- отслеживает согласованность решения, вырабатываемого группой;
- предоставляет средства визуализации результатов работы;
- подготавливает сообщения агенту-координатору;
- выполняет почтовые функции в распределенной среде.

Агент-координатор:

- обеспечивает выполнение пошагового алгоритма принятия решения;
- поддерживает целостность баз данных системы на групповом уровне и вносит в них необходимые изменения;

• подготавливает диалоговые формы для информационного обмена через Интернет.

Агент-эксперт:

- поддерживает выполнение текущего шага задания;
- готовит сообщения агенту-координатору;
- поддерживает целостность локальных баз данных;
- выполняет почтовые функции в распределенной среде.

Работа агентов осуществляется следующим образом. Руководитель формирует задания, оперируя справочниками, содержащими знания об экспертах, показателях качества и решениях, требующих рассмотрения. Далее задание в виде входного сообщения M_{in} поступает агенту-координатору, определяющему состав изменений, которые необходимо сделать в базах данных на локальном уровне. Координатор с помощью предоставленного ему набора функций готовит информацию для всех агентов-экспертов рабочей группы. Агенты-эксперты выполняют задания, предназначенные для своих пользователей, анализируя поступившие от координатора сообщения M_{ij} (j – номер эксперта), и отправляют ему ответные сообщения M_{oj} .

Агент-координатор собирает сообщения о готовности выполненных заданий от всех членов группы. При выполнении всего пакета заданий его состояние изменяется, и посылается сообщение агенту руководителя M_{out} .

Руководитель может выполнять проверку согласованности экспертных суждений либо на основе вычислений, либо с помощью логического анализа предоставленной ему информации. Решение руководителя о степени согласованности суждений посылается агенту-координатору, который продвигает задание на следующий шаг или возвращает экспертов на предыдущий этап в целях достижения лучшей согласованности.

7.4. ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ МУЛЬТИАГЕНТНЫХ СИСТЕМ

Программно реализованные агенты, в том числе и интеллектуальные, относятся к классу программного обеспечения, которое способно действовать самостоятельно от лица пользователя. Созданию программных агентов предшествовал опыт разработки так называемых открытых систем [8], результатом внедрения ко-

торых в практику явилось создание архитектуры «клиент-сервер». В настоящее время наибольшее распространение получили две модели такого взаимодействия: «толстый клиент — тонкий сервер» и «тонкий клиент — толстый сервер». В первой модели серверная часть реализует доступ к ресурсам, а приложения находятся на компьютерах клиентов. Во второй модели клиентское приложение обеспечивает только реализацию интерфейса, а сервер объединяет все остальные части программного обеспечения. При создании МАС используются обе модели. При этом может применяться либо статический подход, при котором осуществляется передача только данных, либо динамический подход, обеспечивающий также передачу программного кода.

Динамический подход опирается на парадигму мобильных агентов, которые в отличие от статических могут перемещаться по сети. Они могут покидать клиентский компьютер и перемещаться на удаленный сервер для выполнения своих действий, после чего могут возвращаться обратно. Использование мобильных агентов имеет положительные и отрицательные последствия, поэтому их применение оправдано в тех случаях, когда они обеспечивают следующие возможности [3]:

- уменьшение времени и стоимости передачи данных;
- расширение ограниченных локальных ресурсов;
- облегчение координации;
- выполнение асинхронных вычислений.

При использовании мобильных агентов возникает ряд серьезных проблем, в том числе: легальность способов перемещения агентов по сети; верификация агентов (например, защита от вирусов); соблюдение прав частной собственности; сохранение конфиденциальности информации; перенаселение сети агентами; совместимость кода агента и программно-аппаратных средств сетевой машины.

Для реализации мультиагентных систем, основанных как на статических, так и на динамических распределенных приложениях, наиболее перспективными на сегодняшний день являются следующие технологии [3, 18]: *DCOM* (Microsoft Distributed Component Object Model), *Jawa RMI* (Jawa Remote Method Invocation) и *CORBA* (Common Object Request Broker Architecture).

Главной особенностью объектно-ориентированной технологии *DCOM* является возможность интеграции приложений, реализованных в разных системах программирования.

В приложениях *Java RMI* на сервере создаются объекты и методы их обработки, доступные для вызова удаленными приложениями, которые размещаются на компьютерах-клиентах.

Технология *CORBA* — одно из наиболее гибких средств реализации распределенных приложений. Ее преимуществом по сравнению с *Java RMI* является наличие специального языка описания интерфейсов IDL, унифицирующего средства коммуникации между приложениями и способы взаимодействия с другими приложениями.

Подробную информацию о программных продуктах, предназначенных для разработки мультиагентных систем, можно найти в Интернете по адресу: [http:// www.Reticular.com](http://www.Reticular.com).

7.4.1. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДЛЯ ПОСТРОЕНИЯ МУЛЬТИАГЕНТНЫХ СИСТЕМ

Для поддержки процессов проектирования агентов и мультиагентных систем разработаны специальные инструментальные средства. Чтобы получить представление об их возможностях и о технологии создания MAC, рассмотрим в качестве примера систему *Agent Builder*.

Инструментарий *Agent Builder* (Reticular Systems, Inc.) предназначен для разработки мультиагентных систем на основе Java-программ, что позволяет исполнять их на любом компьютере, где установлена виртуальная Java-машина (Java Virtual Machine). Общая схема процесса проектирования и реализации приложений на основе *Agent Builder ToolKit* представлена на рис.7.4.

Модель «жизненного цикла» создаваемых агентов включает следующие этапы:

- обработку новых сообщений;
- определение правил поведения;
- выполнение действий;
- обновление ментальной модели в соответствии с заданными правилами;
- планирование действий.

Ментальная модель включает описание намерений, желаний, обязательств и возможностей, а также правил поведения агентов. На основе этой модели осуществляется выбор тех или иных действий интеллектуального агента.

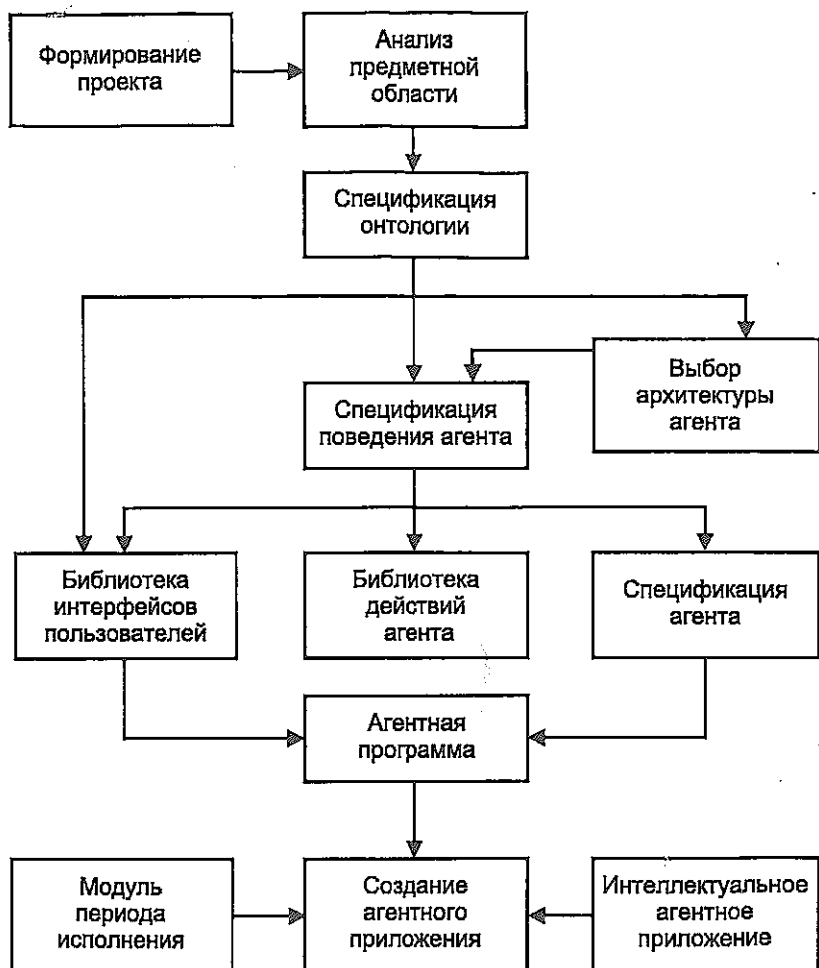


Рис. 7.4. Схема процесса проектирования приложений в системе *Agent Builder ToolKit*

Правила поведения в системе *Agent Builder* реализуются на специальном объектно-ориентированном языке *RADL* (Reticular Agent Definition Language) в виде конструкции *When-If-Then*. Составные части этого правила выполняют следующие функции:

When <...> содержит новые сообщения, полученные от других агентов;

- If <...> сравнивает текущую ментальную модель с условиями применимости правила;
Then <...> определяет действия, соответствующие текущим событиям, состоянию ментальной модели и внешнего окружения.

Правила поведения агентов записываются в формате:

- Name <Имя правила>
When <Message Conditions>
If <Mental Conditions>
Then <Private Actions; Mental Changes; Message Actions>.

В языке *RADL* используются структуры данных, подобные фреймам, а правила представляют собой продукции специального вида. При проектировании приложений необходимо составить спецификации моделей поведения агентов, которые будут применяться совместно с классами и методами из библиотеки действий агентов и библиотеки интерфейсов.

Являясь достаточно мощным средством для представления и обработки знаний, *Agent Builder* не предусматривает применения средств явного управления логическим выводом, которые могли бы существенно расширить возможности используемого языка.

7.4.2. МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ ДЛЯ ПОИСКА ИНФОРМАЦИИ

В связи с быстрым развитием интернет-технологий возникла необходимость применения средств искусственного интеллекта для поиска и обработки интернет-ресурсов. Применение интеллектуальных МАС для решения задач сбора, поиска и анализа информации в глобальных сетях дает следующие существенные преимущества перед традиционными средствами обработки информации [3]:

- обеспечение доступа пользователя к сетевым протоколам в сети Интернет;
- параллельное решение нескольких задач;
- выполнение поиска информации после отключения пользователя от сети;

• увеличение скорости и точности поиска, а также уменьшение загрузки сети за счет поиска информации непосредственно на сервере;

• создание собственных баз информационных ресурсов, постоянно обновляемых и расширяемых;

• реализация возможности сотрудничества между агентами, которая позволяет использовать накопленный опыт;

• возможность автоматически корректировать и уточнять запросы, используя контекст и применяя модели пользователей.

В табл. 7.2 приведены отличительные особенности известных в России коммерческих мультиагентных систем *Autonomy* [15] и *WebCompass* [22], предназначенных для интеллектуального поиска и обработки информации в сети Интернет.

Недостатком современных систем интеллектуального поиска и обработки информации является их слабая способность к обу-

Таблица 7.2

Анализ систем интеллектуального поиска и обработки информации

Характеристика	Autonomy	WebCompass
Категория пользователей, на которую ориентирована система	Конечные пользователи	«Продвинутые» пользователи
Подход к описанию предметной области	Технология нейронных сетей и специальные методы распознавания образов и обработки сигналов	Иерархии понятий, связанных отношениями типа IS-A, PART-OF, HAS-PART, IS-A KIND OF и т. д.
Средства спецификации запросов	Естественный язык	«Прямое» использование сформированного пользователем описания предметной области
Методы поиска релевантной информации	Нечеткая логика	Поиск по списку ключевых слов одновременно на 35 машинах поиска
Режим обучения поисковых агентов	Есть	Нет

чению. Поэтому основные усилия по совершенствованию интеллектуальных систем информационного поиска в сети Интернет направлены на развитие моделей представления знаний, механизмов вывода новых знаний, моделей рассуждения и способов обучения агентов [20].

Одним из успешных исследовательских проектов, выполненных в этом направлении, стал проект системы *MARRI* [21], которая была разработана для поиска Web-страниц, релевантных запросам в определенной предметной области. Для решения поставленной задачи система использует знания, представленные в виде онтологии, под которой в данном случае понимается упорядоченное множество понятий предметной области. Архитектура системы *MARRI* показана на рис. 7.5.

Система *MARRI* включает следующие типы агентов:

- *интерфейсный агент* (агент пользователя) обеспечивает интеллектуальное взаимодействие с пользователем. Он поддерживает процесс формулирования запросов и представляет результаты поиска в виде списка URL или Web-страниц;

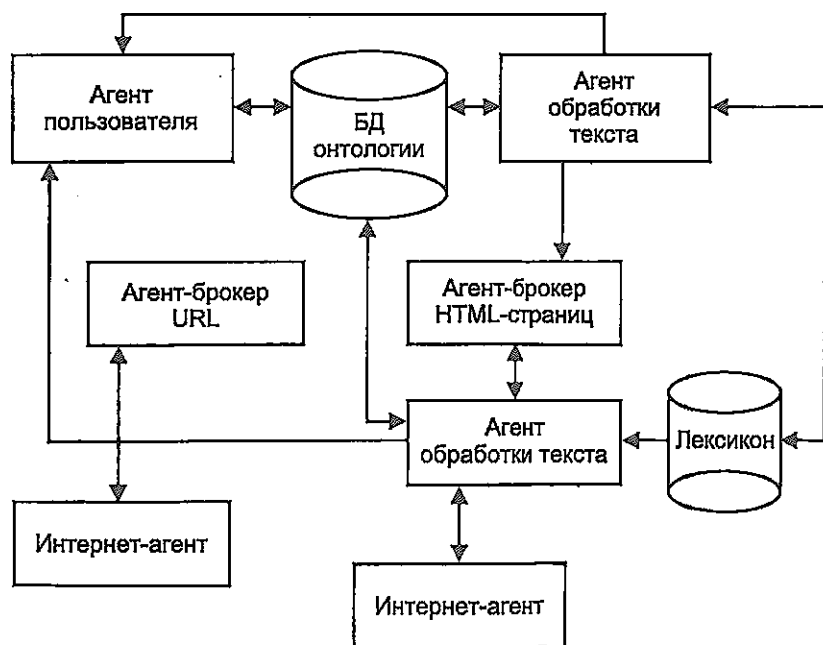


Рис. 7.5. Архитектура системы *MARRI*

• *агенты-брокеры* двух типов: 1) брокер типа URL предназначен для формирования списков интернет-адресов, поставляемых браузером; 2) брокер типа HTML выполняет функции запоминания полученных Web-страниц и их распределения между агентами обработки текста;

• *агент сети* (интернет-агент) обеспечивает считывание и анализ заданной страницы URL или Web-страницы (URL — автономная Java-программа с собственным сетевым адресом). Он должен уметь выполнять обработку исключительных ситуаций (например, когда страница недоступна), а также производить анализ текста на считанной странице;

• *агент обработки текста* сначала преобразует HTML-текст к представлению, с которым работают морфологический и синтаксический анализаторы, а затем проводит семантический анализ Web-страниц для проверки их релевантности запросу на основе соответствующей онтологии. Результат обработки текста представляется в виде синтаксического дерева, которое должно соответствовать какому-нибудь фрагменту используемой онтологии.

Каждый из перечисленных типов агентов наделен специальными знаниями, которые используются для повышения эффективности поиска информации. Агенты способны взаимодействовать друг с другом; обмениваться информацией, контактировать с Web-браузерами, анализаторами естественного языка и онтологическими базами данных.

Отличительной чертой системы *MARRI* является представление агентов автономными Java-программами, каждая из которых имеет собственный сетевой адрес (URL). Это обеспечивает мобильность агентов, но противоречит политике безопасности, не допускающей запуск подобных программ, если они не сертифицированы на данном сервере.

7.5. ПЕРСПЕКТИВЫ МУЛЬТИАГЕНТНЫХ ТЕХНОЛОГИЙ

В работе [9] сформулированы проблемы, решение которых может существенно продвинуть вперед технологию мультиагентных систем и исследования в области «искусственной жизни».

1. Применение принципов гомеостатического управления. Предположение о том, что наилучшее взаимодействие агентов в

МАС достигается при бесконфликтной кооперации, не всегда справедливо. Это утверждение можно аргументировать примерами биологических систем, в которых эффективной оказывается кооперация противоборствующих сторон (*хищник-жертва*, взаимодействие симпатической и парасимпатической нервных систем). Противодействующие структуры позволяют поддерживать системы с многокритериальным управлением в границах области «неулучшаемых» решений (область Парето). Поэтому одним из актуальных направлений развития теории МАС является применение принципов гомеостатического управления (гомеостаз — равновесие), основы которого были заложены в работе отечественной научной школы Ю. М. Горского.

2. Создание адекватных механизмов активизации знаний, требующихся при решении конкретных проблем. Опыт создания интеллектуальных систем показывает, что увеличение количества знаний приводит к эффекту «государственной публичной библиотеки». Обладая огромным запасом знаний, библиотека не имеет каких-либо умений и навыков. Поэтому одной из существенных проблем интеллектуальных агентов является повышение их активности, которая связана не с накоплением знаний, а с умением активизировать нужные знания в процессе решения задач. Разработка процедур активизации знаний будет способствовать созданию действительно интеллектуальных агентов.

3. Перспективным направлением является использование идей рефлексивного управления в МАС. Эксперименты с агентами, наделенными способностью к рефлексивным рассуждениям, показали эффективность данного подхода.

4. Создание конструктивных моделей этических систем и моделей поступков в среде обитания агентов.

5. Исследование влияния внешних факторов на поведение коллектива искусственных агентов и личностных характеристик агентов (психологические типы, оптимизм в оценках достижимости целей, азартность, упорство, конфликтность и т.п.).

Литература

1. *Варшавский В. И., Поспелов Д. А.* Оркестр играет без дирижера. — М.: Наука, 1984.
2. *Гаазе-Раппопорт М. Г., Поспелов Д. А.* От амебы до робота. Модели поведения. — М.: Наука, 1987.

3. *Гаврилова Т.А., Хорошевский В.Ф.* Базы знаний интеллектуальных систем. — СПб.: Питер, 2000.
4. *Городецкий В. И.* Многоагентные системы: основные свойства и модели координации поведения // Информационные технологии и вычислительные системы. — 1998. — № 1.
5. *Кудрявцев Е.М.* Исследование операций в задачах, алгоритмах и программах. — М.: Радио и связь, 1984.
6. *Лефевр В.* Конфликтующие структуры. — М.: Советское радио, 1973.
7. Моделирование обучения и поведения. — М.: Наука, 1975.
8. *Орлик С.* Многоуровневые модели в архитектуре клиент-сервер. — 1997. — [http:// www.citforum.ru/database/osbd/glava_95.html](http://www.citforum.ru/database/osbd/glava_95.html).
9. *Поспелов Д. А.* Многоагентные системы — настоящее и будущее// Информационные технологии и вычислительные системы. — 1998. — № 1.
10. *Саати Т.Л.* Принятие решений. Метод анализа иерархий. — М.: Радио и связь, 1989.
11. *Смирнов А. В., Пашкин М. П., Рахманова И. О.* Многоагентные системы поддержки принятия решений для предприятий малого и среднего бизнеса // Информационные технологии и вычислительные системы. — 1988. — № 1.
12. *Тарасов В.Б.* От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. — М.: Эдиториал УРСС, 2002.
13. *Трахтенгерц Э.А.* Взаимодействие агентов в многоагентных системах // Автоматика и телемеханика. — 1998. — № 9.
14. *Цетлин М. Л.* Исследования по теории автоматов и моделированию биологических систем. — М.: Наука, 1969.
15. *Autonomy.* Autonomy Technology Whitepaper. — 1998. — [http:// www.autonomy.com/tech/wp.html](http://www.autonomy.com/tech/wp.html).
16. *Brooks R.A.* Intelligence without Representation // Artificial Intelligence. — 1991. — No. 47.
17. *Georgeff M.P., Rao S.* Social plans: Preliminary report // Proceedings of 3rd European Workshop on Modeling Autonomous Agents and Multi-Agent Worlds, 1992, North Holland.
18. *Gopalan R.S.* A Detailed Comparison of CORBA, DCOM and Java/RNI (with specific code examples). — [http:// www.execpc.com/~gopalan/index.html](http://www.execpc.com/~gopalan/index.html).

19. *Nwana H.S.* Software Agents: An Overview // Knowledge Engineering Review. — 1996. — Vol. 11. — No. 3, Cambridge University Press.
20. *Pagina H.* Intelligent Software Agent on the Internet. — 1996/ - <http://www.hermans.org/agents/index.html>.
21. *Villemain F.-Y.* Ontologies — based relevant information retrieval. — 1999. — <http://www.cnam.fr/f-gv>.
22. WebCompass. WebCompass Page. — 1999. — http://www.syman-tec.com/techsupp/webcompass/kbase_webcompass.html.
23. *Wooldridge M., Jennings N.* Intelligent Agents: Theory and Practice // Knowledge Engineering Review. — 1995. — № 10(2).
24. *Zlotkin G., Rosenschein J. S.* Mechanisms for Automated Negotiation in State Oriented Domain // Journal of Artificial Intelligence Research. — 1996. — № 5.

Контрольные вопросы и задания

1. Расскажите о сущности мультиагентных технологий. Что подразумевается под *агентом* и как он может быть реализован?
2. Какими свойствами обладают «интеллектуальные агенты»?
3. Дайте характеристику архитектурам мультиагентных систем.
4. Сформулируйте основные проблемы, возникающие при моделировании коллективного поведения интеллектуальных агентов.
5. Охарактеризуйте основные модели координации поведения агентов в мультиагентных системах: теоретико-игровые, модели коллективного поведения автоматов, модели планирования коллективного поведения, модели на основе BDI-архитектур, модели координации поведения на основе конкуренции.
6. Сформулируйте постановки задач координации поведения агентов на основе модели аукциона.
7. Проведите сравнительный анализ свойств мобильных и статических агентов.
8. Опишите технологию построения мультиагентных систем. Приведите примеры инструментальных средств, предназначенных для этой цели.
9. Перечислите основные преимущества интеллектуальных поисковых мультиагентных систем перед традиционными средствами поиска информации.
10. Для каких задач актуально применение мультиагентных технологий? Приведите примеры.

11. Сформулируйте содержательный пример задачи кооперации и покажите возможный способ ее решения средствами мульти-агентных технологий.
12. Приведите пример задачи координации коллективного поведения, для решения которой актуально применение мультиагентных технологий. Сформулируйте принцип координации и правила нормативного поведения агентов.
13. Спроектируйте виртуальный магазин. Опишите виды агентов, их функции и способы возможной реализации.
14. Спроектируйте структуру мультиагентной системы для реализации конкретного виртуального предприятия. Опишите виды агентов, их функции и способы возможной реализации. Охарактеризуйте механизм координации поведения агентов.
15. Спроектируйте интеллектуальную мультиагентную систему для решения прикладной задачи в области экономики и управления. Реализуйте спроектированную систему на ЭВМ.

ИНТЕЛЛЕКТУАЛЬНЫЕ МЕТОДЫ ПРОЕКТИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

Метод науки — анализ...
Метод искусства — синтез.

В. Я. Брюсов

Профессиональный подход к задачам проектирования и совершенствования производственных, промышленно-торговых и других социально-экономических систем является одной из важнейших предпосылок их успешного функционирования. Экономические системы являются открытыми, поскольку не могут существовать в изоляции от окружающего мира, и целенаправленными, так как в их состав входят люди. Свойства открытости и целенаправленности присущи развивающимся системам, состав и структура которых изменяются с течением времени. В процессе развития происходит изменение свойств, выполняемых функций и механизмов функционирования систем, причем специфической чертой антропогенных систем (систем с участием человека), отличающей их от природных (например, биологических), является наличие множества целей внутри системы, в то время как для естественных систем характерны внешние цели. Формируя множественные и часто противоречивые цели, люди далеко не всегда могут правильно предвидеть пути и последствия их достижения. Наверное, это невозможно в принципе, поэтому нам ничего не остается, кроме хаотического, эвристического или систематического поиска вариантов достижения целей, попыток прогнозирования возможных последствий принимаемых решений и субъективных оценок этих последствий на основе доступных знаний. Существенную поддержку при решении столь сложных проблем могут оказать информационные технологии. Проблемы выработки творческих решений, их оценки и прогнозирования последствий в условиях неопределенности требуют для своего решения применения интеллектуальных методов и компьютерных систем, обладающих знаниями и наделенных способностью к рассуждениям.

8.1. ПРОБЛЕМЫ ПРОЕКТИРОВАНИЯ И РЕИНЖИНИРИНГА ЭКОНОМИЧЕСКИХ СИСТЕМ

Традиционное проектирование содержит задачи анализа систем, связанные с накоплением и исследованием необходимой информации, и задачи синтеза конкретных вариантов систем, удовлетворяющих заданным требованиям. Исследования проблем проектирования получили наибольшее развитие в области техники, поэтому, рассматривая вопросы проектирования экономических систем, будем пользоваться накопленным там опытом. Проектирование технического объекта может проводиться на основе прототипа или без такового.

Цель проектирования на основе прототипа обычно заключается в его совершенствовании, которое ориентировано либо на улучшение качества выполнения уже существующих функций, либо на обеспечение новых функций. Кроме того, задача может ставиться как изобретательская, когда одной из важнейших целей является патентоспособность создаваемого устройства, т. е. оно должно иметь существенные отличия по ряду признаков от всех известных.

Задача совершенствования прототипа может включать дополнительные условия, связанные с технологическими возможностями конкретного предприятия для перехода от производства прототипа к выпуску нового образца. Задача проектирования без прототипа обычно не содержит подобных требований. Отсутствие прототипа вовсе не означает, что проектировщики не используют информацию о существующих устройствах рассматриваемого класса и не проводят анализ на новизну проектируемого объекта. Если этого не делать, то можно «изобрести велосипед», т.е. прийти к известным решениям, впустую потратив время и средства.

При проектировании экономических систем обычно не преследуются цели новизны. Проектированию без прототипа здесь соответствует задача создания новой экономической системы (предприятия, корпорации и т.п.), т.е. системы, которая не существовала прежде. При этом цели, функции, а иногда структура и многие свойства создаваемой системы могут быть давно и хоро-

шо известными, но в процессе проектирования не используется конкретный аналог в качестве отправной точки для совершенствования или как ориентира. С одной стороны, проектирование без прототипа осложняется в связи с отсутствием начального или «идеального» варианта, но, с другой стороны, расширяет пространство поиска проектных решений, допуская произвольный выбор траектории поиска.

Задачи проектирования на основе прототипа в экономике могут иметь различные постановки.

1. Проектирование не существовавшей прежде системы по образу и подобию уже известной. В качестве прототипа здесь принимается некоторая модель успешно работающей фирмы.

2. Кардинальное перепроектирование существующей системы, которая при этом уничтожается, а на ее месте с использованием ряда прежних элементов предусматривается создание новой. Отечественная история богата примерами подобного рода, большую часть которых трудно отнести к рациональным решениям, однако среди них есть весьма успешные и хорошо обоснованные. Если руководствоваться положениями теории развития [2, 29], то данная постановка задачи перепроектирования возникает тогда, когда существующая система исчерпала свои ресурсы совершенствования, стала неустойчивой и неэффективной (достигла точки бифуркации), поэтому даже небольшие флуктуации могут привести ее к гибели или к такой структурной перестройке, которая обеспечит возможности дальнейшего развития, но уже в соответствии с новым законом.

3. Кардинальное перепроектирование существующей системы, ориентированное на ее совершенствование и не допускающее ее уничтожения (реинжиниринг). Такая постановка задачи возможна тогда, когда система «осознает» характер процессов развития, «следит» за изменением своего окружения и заранее «принимает меры», направленные на изменение своей самоорганизации в целях выживания в приближающейся точке бифуркации.

4. Постоянное и постепенное совершенствование существующей системы на основе мониторинга ее окружения и тенденций развития организационных систем (концепция постоянно развивающегося предприятия).

Следует заметить, что в зависимости от конкретных условий возможна любая из приведенных постановок задачи проектиро-

вания. С начала 1990-х гг. концепция реинжиниринга бизнес-процессов (BPR) приобрела особую популярность, которая, очевидно, обусловлена согласованием в ее рамках противоречивых целей, а именно: существенное повышение эффективности и качества функционирования предприятия (компании) без снижения производительности. Основные особенности BPR заключаются в следующем [34]:

- смена концепции управления компанией: от управления по целям к управлению по результатам [32];

- переход от ориентации на функции к ориентации на процессы;

- переход от пирамидальной (иерархической) структуры управления компанией к плоской (горизонтальной);

- повышение уровня квалификации кадров и изменение роли исполнителей конкретных работ, обусловленное необходимостью роста производительности труда, увеличением разнообразия решаемых задач и расширением прав принятия самостоятельных решений;

- изменение роли потребителя в процессах управления деятельностью компании (она становится более важной);

- ориентация на многовариантное исполнение процессов;

- рациональный подход к размещению и распределению ресурсов, а также к организации бизнес-процессов;

- усиление роли информационных технологий в процессе деятельности компании.

Реинжиниринг позволяет достичь многократного улучшения основных показателей качества деятельности компании за счет изменения принципов ее организации. Цикл BPR включает следующие этапы.

1. Постановка задачи и создание команды для разработки проекта реинжиниринга предприятия.

2. Разработка модели существующего предприятия и оценка эффективности его функционирования.

3. Перепроектирование бизнес-процессов, суть которого состоит в разработке более эффективных рабочих процедур и определении изменений, необходимых для перехода к ним от существующих.

4. Разработка бизнес-процессов на уровне трудовых ресурсов (определение видов работ, разработка системы мотивации, создание информационных систем, обучение и подбор кадров).

5. Разработка поддерживающих информационных систем.

6. Внедрение перепроектированных процессов.

Реинжиниринг становится одним из видов постоянной деятельности предприятия, так как наряду с мерами постоянного совершенствования текущих процессов периодически возникает потребность в реорганизации общего бизнес-процесса. На успех реинжиниринга влияют следующие факторы:

- точное понимание задачи;
- мотивация;
- заинтересованность в кардинальном улучшении результатов деятельности и готовность руководства компании пойти на риск, вкладывая средства в реинжиниринг;
- уровень управления и квалификация кадров;
- хорошая методологическая основа.

Научную и методологическую базу проектирования и реинжиниринга бизнес-процессов на предприятии составляют методы теории систем, математического моделирования, исследования операций, информатики, искусственного интеллекта, а также знания из соответствующих областей бизнеса.

8.2.

СИСТЕМНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ СЛОЖНЫХ СИСТЕМ

Основным методом исследования сложных систем, в том числе социально-экономических и социально-производственных, является системный анализ. Общие этапы для любой методики системного анализа — это постановка задачи, формирование описания системы и выбор наилучших решений.

На этапе постановки задачи определяются цели исследования, производится выделение системы из среды, рассматриваются способы взаимодействия системы со средой, формулируются основные допущения.

Этап формирования описания системы включает следующие действия:

- расчленение системы на элементы;
- выделение подсистем;
- определение общей структуры системы;
- определение связей системы со средой и выявление внешних факторов, подлежащих учету;

- выбор подхода к представлению системы;
- формирование вариантов представления системы.

Особенностью методик системного анализа является то, что они опираются на понятие системы и используют общие закономерности строения, функционирования и развития систем [1, 3, 4, 8, 11, 12, 17, 21, 25, 42]. Специфическими для конкретного исследования этапами могут быть синтез возможных структур изучаемой системы, прогнозирование поведения системы при заданных условиях, выявление способов достижения определенных состояний и др.

Стандартного определения понятия «система» не существует. Основоположник теории систем Л. Берталанфи [8] определил систему «как совокупность элементов, находящихся в определенных отношениях друг с другом и со средой». Позднее в определении системы появилось понятие цели [3, 4]. У.Р. Эшби [42] указал на необходимость учета взаимодействия исследователя с изучаемой системой, что привело к включению исследователя в систему наряду с элементами, связями, свойствами и целями.

С философской точки зрения природа систем двойственна, т.е. это понятие одновременно отражает объективное существование и субъективное восприятие некоторой реальности [16]. Значительным вкладом в теорию систем является определение системы, данное академиком П.К. Анохиным [7]: «Системой можно назвать только такой комплекс избирательно вовлеченных компонентов, у которых взаимодействие и взаимоотношение приобретают характер взаимосодействия компонентов, направленного на получение фокусированного полезного результата», т.е. системообразующим фактором является *полезный результат*.

Дж. Клир в своей книге «Системология» [17] пишет о том, что общее определение системы можно сделать более полезным для практики, введя классы элементов и отношений между ними. При этом классификация систем по типам элементов имеет экспериментальную, а по типам отношений — эпистемологическую основу, позволяющую выделять изоморфные классы систем. Дж. Клир выделяет два класса системных задач — задачи исследования и задачи проектирования. *Задача исследования* систем состоит в накоплении знаний о свойствах и отношениях существующих объектов в соответствии с конкретными целями. *Задача проектирования* систем заключается в создании новых объектов с заданными свойствами.

Сложным системам присущи следующие основные закономерности [25]:

- целостность — свойства системы не являются суммой свойств ее элементов, хотя и зависят от них;

- коммуникативность — любая система является подсистемой в системе более высокого уровня;

- иерархичность;

- эквифинальность — способность системы достигать не зависящего от времени состояния, полностью детерминированного начальными условиями;

- историчность;

- закон необходимого разнообразия;

- закономерности целеобразования — зависимость целей от уровня познания объекта, а также от внешних и внутренних факторов.

Академик И. В. Прангишвили [30, 31] к главным общесистемным атрибутам относит первичные элементы системы, отношения между элементами, законы композиции отношений, фон системы. Он формулирует восемь законов композиции, которым подчиняется функционирование любых сложных систем:

- 1) перевод системы из одного качественного состояния в другое путем минимального воздействия в критическую точку фазового перехода системы;

- 2) закон эволюции, который утверждает, что любая система в процессе развития проходит в сокращенной форме собственный эволюционный путь, включая все его этапы (этот закон отрицает возможность «большого скачка»);

- 3) закон пирамиды, который гласит, что коэффициент полезного действия любой реальной системы не может достигать 100%, в связи с чем энергия, почерпнутая системой извне, постепенно уменьшается по мере приближения к конечной цели;

- 4) закон «островного эффекта», позволяющий определить возможную степень автономности системы в зависимости от ее параметров и от свойств окружения;

- 5) закон единства и борьбы противоположностей, определяющий возможность и условия объединения противоборствующих сторон;

- 6) закон причинно-следственных связей;

- 7) закон проявления нестабильностей системы, которые выражены нарушением когерентного взаимодействия с фоном этой системы;

8) закон существенной зависимости потенциала системы от изменения характера взаимодействия между ее элементами.

Свойства сложных систем можно разделить на три основные группы.

1. Свойства, определяющие взаимодействие системы с внешней средой. Важнейшие среди них — это устойчивость и характеристики состояний системы.

2. Свойства, характеризующие внутреннее строение системы (структура). Структура систем любой природы может изменяться как в результате взаимодействия с внешней средой, так и в результате протекания внутренних процессов. Параметром, характеризующим изменение структуры во времени, является энтропия. В открытых системах энтропия может не только увеличиваться, но и уменьшаться за счет ее увеличения во внешней среде.

3. Интегральные свойства, описывающие поведение системы. К ним относятся полезность, эффективность, надежность, управляемость, безопасность, живучесть и др. Одни свойства системы наследуются от свойств входящих в нее элементов, другие определяются как некоторая комбинация свойств элементов, ряд свойств системы — это новые свойства, отсутствующие у ее элементов. В общем случае интегральные свойства сложной системы не являются суммой свойств ее частей (нарушение принципа суперпозиции).

Системный подход к проектированию заключается в рассмотрении всего комплекса проблем, возникающих в течение жизненного цикла исследуемой системы. Основные аксиомы системного проектирования сформулированы в работе [28]:

1. Из неразрешимости общей задачи проектирования вытекает необходимость ее декомпозиции на совокупность локальных задач, упорядоченных многоуровневой параллельно-последовательной логической схемой проектирования.

2. Из неопределенности исходных данных и ограничений в общей задаче проектирования вытекает необходимость их прогнозирования и обмена проектными решениями между функциональными ячейками системы проектирования в соответствии с определенной логической схемой.

3. Из логической противоречивости общей задачи проектирования вытекает необходимость организации итерационных циклов, которые определяют сходимость системных решающих процедур.

4. Из невозможности сконструировать априори «сквозное» правило предпочтения следует необходимость «индивидуального» построения многоуровневого критерия оценки проектных решений, который может быть получен эвристически только в конце итерационного цикла.

Построение формального описания сложной системы является необходимым этапом исследования. Формальные модели нужны для изучения внутреннего строения систем, для прогнозирования, а также для определения оптимальных режимов функционирования. Наиболее высокий уровень абстрагирования имеют лингвистический и теоретико-множественный способы описания систем.

Лингвистический подход к описанию систем [17] исходит из «характерных черт» системы. Этот подход весьма привлекателен, так как знания в виде высказываний являются наиболее доступным видом информации. Лингвистический подход уступает теоретико-множественному способу описания в плане компактности. Алгебраический подход к описанию систем, развитый Месаровичем и Тахакарой [21], во многих случаях позволяет получить полезные результаты, но существуют задачи, для которых более естественными являются описания других типов.

На сегодняшний день самый распространенный способ описания — теоретико-множественный. При этом наиболее развитым направлением теории систем является феноменологическое, которое основано на представлении системы S как некоторого преобразования входных воздействий X в выходные величины Y , т.е. $S \subset X \times Y$.

Открытые системы невозможно представить в виде функции, позволяющей получить выход по входу из-за наличия неопределенностей. Входные воздействия в открытой системе можно разделить на две составляющие: определенные U и неопределенные M , тогда $X = U \times M$, $S \subset U \times M \times Y$.

В условиях неопределенности традиционным приемом является усиление степени размытости языка описания, поэтому часто в таких случаях переходят от рассмотрения входных и выходных величин к рассмотрению подмножеств их элементов. Это приводит к вероятностным и нечетким описаниям систем.

Характерная черта систем, в которых участвует человек (антропогенные системы), — это стремление к определенной цели

(целенаправленность). Для описания целенаправленных систем применяется подход, отличный от феноменологического. Он заключается в том, что система описывается с помощью некоторой задачи принятия решений (ЗПР).

Система $S \subset X \times Y$ называется системой принятия решений, если найдется такое семейство ЗПР D_x , $x \in X$, решения которых принадлежат множеству D_x , и такое отображение $T: Z \rightarrow Y$, что для любого $x \in X$ и $y \in Y$ пара (x, y) принадлежит системе S тогда и только тогда, когда найдется такое $z \in Z$, что z является решением задачи D_x , а $T(z) = y$.

Любую систему преобразования входов в выходы можно формально представить как систему принятия решений и наоборот [25], поэтому для конкретной системы можно получить как феноменологическое, так и целенаправленное описание, если определена цель.

Существуют разные подходы к формированию целей проектирования. Сложившийся на практике стереотип — ясно поставленная цель есть главнейшее условие успеха — противоречит опыту специалистов по системному анализу, утверждающих, что цель создания системы и ее составляющие уточняются в ходе работы. Цель деятельности формируется на начальных стадиях проектирования, а достигается в момент времени $t = T$, поэтому в процессах проектирования необходимо осуществлять исследование и разработку образов будущего на интервале $0 < t < T$. В связи с этим одним из условий, обеспечивающих успех любого проекта, является построение следующих прогнозов:

- прогноз состояния или поведения внешней среды (надсистемы), взаимодействующей с исследуемой системой (объектом);
- прогноз изменения целей функционирования и структуры исследуемой системы;
- прогноз развития конкурирующих или противоборствующих подсистем, их характеристик и стратегий поведения.

Взаимодействие системы со средой, а также элементов системы друг с другом может быть представлено моделями структуры и моделями функционирования. Модель структуры в зависимости от цели исследования может иметь следующие разновидности:

- внешняя модель — система представляется в каноническом виде, а ее связи с внешней средой выражаются посредством входов и выходов;

- иерархическая модель — система расчленяется по уровням согласно принципу подчинения низших уровней высшим;
- внутренняя модель — отражает состав и взаимосвязь между элементами системы.

Функционирование системы может быть представлено:

- моделью жизненного цикла системы, характеризующей процесс существования системы от ее замысла до гибели;
- операциональной моделью системы, представляющей совокупность процессов ее функционирования по основному назначению.

Существуют различные подходы к моделированию сложных систем [33].

Статистический подход основан на построении макромоделей больших систем, которые могут использовать различные типы описаний: теоретико-множественный, лексикографический, топологический и др. Макромодели дают представление о зависимости состояния системы от поведения человека и среды, но не учитывают состязательный характер развития и не содержат представлений об эффективности [14, 20].

Структурно-функциональный подход [23, 26, 40] связан с построением модели структуры, элементами которой являются функции. Для построения механизма, порождающего функции, используют теоретико-множественный аппарат и математико-лингвистические средства.

Ситуационное моделирование разработано для задач динамического управления сложными системами в условиях неопределенности [27]. Система описывается конечным набором возможных ситуаций и соответствующих им управленческих решений. Главным условием применимости этого подхода является возможность классификации ситуаций.

Имитационное моделирование основано на использовании субъективных предположений исследователей о динамике рассматриваемых процессов [13, 19]. При моделировании динамики сложных систем дифференциальные уравнения можно составить только для идеализированной структуры при усредненных значениях параметров. Примерами могут служить модели системной динамики Дж. Форрестера [37, 38]. Разработанные им точечные модели дают хорошие результаты в тех случаях, когда они достаточно просты и когда оперируют с большими пространствами ус-

реднения переменных. Усложнение обычно сопровождается переходом к моделям, распределенным в пространстве. При этом возникают серьезные затруднения с заданием граничных условий, определением неизвестных коэффициентов уравнений, а также сложности вычислительного характера.

Имитационное моделирование дискретных систем и процессов до недавнего времени было связано в основном с разработкой специализированных языков (SIMULA, GPSS и др.). Основная сфера их приложения — моделирование технологических процессов производства, а также моделирование функционирования организационных систем. Развитие технологий программирования и искусственного интеллекта значительно облегчило процесс разработки имитационных моделей. Появились языки и системы интеллектуального имитационного моделирования [13], доступные широкому кругу пользователей. На рынке программного обеспечения популярны визуальные пакеты системной динамики *Ithink*, *ReThink*, *PowerSim* и т. д. [43].

Синергетический подход к моделированию сложных систем [2, 29, 41] учитывает нелинейность сложных систем, одним из проявлений которой является нарушение принципа суперпозиции. В данном подходе используется динамическая имитационная модель системы, описывающая процесс ее развития [2]. Основопологающие понятия в теории развития — информация и энтропия. Наиболее полное выражение идея о связи энтропии и информации получила в виде негэнтропийного принципа Бриллюэна [9]: «Энтропия есть мера недостатка информации о действительной структуре системы». Для эволюции существенно не количество информации, а ее ценность, которая непосредственно связана со степенью использования информации в системе. Ценность информации оказывается тем больше, чем меньше существует способов выполнить нужную функцию без данной информации. Другими словами, ценность характеризует незаменимость информации. В процессе использования информации происходит отбор тех элементов-признаков, которые дают преимущества в ходе развития [10, 36].

Согласно концепции И. Пригожина [29] абсолютно устойчивая система не способна к развитию, так как она подавляет любые отклонения от устойчивого состояния [39].

Задачи проектирования систем всегда предполагают решение задач синтеза, которые в зависимости от исходных данных разделяют на три класса [25]:

- 1) синтез структуры системы при заданных алгоритмах функционирования;
- 2) синтез оптимального поведения и алгоритмов функционирования системы при известной структуре;
- 3) синтез структуры и алгоритмов функционирования системы, распределение функций по элементам и определение их оптимального состава.

Требования, предъявляемые к синтезируемым системам, зависят от конкретных условий и определяющим образом влияют на качество получаемых решений.

Проблема выбора оптимальных решений из множества синтезированных объектов осложняется по следующим причинам:

- объекты выбора (альтернативы) реально не существуют;
- отсутствует возможность количественного или объективного измерения свойств проектируемой системы;
- представление о «наилучшем решении» может отсутствовать или быть неоднозначным и противоречивым;
- необходим учет многих критериев в процессах принятия решений;
- требования к проектируемой системе изменяются с течением времени.

Задача структурного синтеза считается наиболее трудной для формализации проектной процедурой [23]. Для синтеза необходима информация о базовых элементах системы, макроэлементах (типовая совокупность взаимосвязанных элементов) и обобщенных структурах проектируемого объекта. Автоматизация структурного синтеза, как правило, сопряжена с разработкой интеллектуальных систем. Для представления знаний об объектах синтеза обычно используются объектно-ориентированные модели. Распространенным средством представления обобщенных структур являются И-ИЛИ-деревья. Известны следующие подходы к алгоритмизации структурного синтеза:

- перебор законченных структур;
- наращивание структуры;
- выделение варианта из обобщенной структуры;
- трансформация готовых описаний [5].

Отсутствие строгого общего решения задачи синтеза сложной неоднородной системы обусловлено следующими причинами:

- начальная информация о составе элементов и отношений между ними ограничена, различным уровням описания системы соответствуют свои составы элементов;

- возможны альтернативные способы членения сложной системы на элементы;

- не все свойства системы можно получить суммированием свойств ее частей;

- в процессе формализации связи между элементами обычно выражаются через бинарные отношения или отношения, сводимые к бинарным, но известно, что система, будучи целостной организацией, не сводима к бинарным отношениям;

- проектирование новых систем часто является поиском принципиально новых решений, следовательно, эффективный метод решения задачи должен включать процессы, ведущие к образованию новых структур с новым составом элементов и отношениями между ними.

Для моделирования и проектирования сложных систем широко применяются средства компьютерной поддержки автоматизированного проектирования, имеющие статус методик и стандартов [23], в частности, наиболее известны методики функционального, инфологического и поведенческого проектирования, закрепленные стандартами серии IDEF (Icam DEFinition). Методика Structured Analysis Design Technology (стандарт IDEF0) ориентирована на построение *функциональных* моделей систем управления производством, бизнесом и т. п., которые затем анализируются различными специалистами и используются при разработке структурных и имитационных моделей. *Поведенческое* моделирование (стандарты IDEF2, IDEF3) основано на построении имитационных моделей сложных систем с использованием теории массового обслуживания, сетей Петри и конечных автоматов. Методики *инфологического* проектирования (стандарты IDEF1, IDEF1X) ориентированы на поддержку создания баз данных на основе ER-моделей (сущность — связь). Главным результатом применения этих средств является экономия времени, которое затрачивается на преобразование описаний и форматов данных, используемых на разных этапах и в различных задачах проектирования.

8.3. ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ ПОДДЕРЖКИ ПРОЦЕССОВ РЕИНЖИНИРИНГА

Проведение реинжиниринга связано с решением сложных исследовательских задач и разработкой средств информационной поддержки деятельности предприятия, поэтому невозможно без применения программных средств. Большинство современных фирм при решении таких задач ориентируются на использование CASE-технологий и средств быстрой разработки приложений. Применяемые программные средства можно разделить на пять категорий [34].

1. Средства создания диаграмм и инструментальные пакеты, например: *ABC Flowcharter* (Micrografx), *Process Charter* (Scriptor), *Ithink* (High Performance Systems). Эти средства имеют невысокую стоимость и могут использоваться при разработке моделей бизнес-процессов, представляемых в виде диаграмм. Они, как правило, не связаны со средствами быстрой разработки приложений.

2. Средства построения потоковых моделей (*Action-Workflow Analyzer*, *Process Architect*), которые применяются для описания потоков работ. Они позволяют строить планы работы над проектами, но обычно не обладают мощными инструментами анализа получаемых планов.

3. Системы имитационного моделирования (*Modsim*, *Arena*, *ProModel*, *ReThink*, РДО) позволяют строить и анализировать модели поведения исследуемых систем с помощью графических средств, специализированных библиотек и языков. Эта категория средств имеет высокую стоимость и сложна в освоении.

4. CASE-системы и средства быстрой разработки приложений (*Framework*, *Designer 2000*, *System Architect*) ориентированы на поддержку процессов создания компьютерных баз данных и приложений.

5. Многофункциональные средства автоматизации реинжиниринга (*WorkFlow Analyzer*, *ARIS Toolset*, *ReThink+G2*) ориентированы на профессиональных программистов и требуют специальной подготовки. Содержат программное обеспечение для поддержки различных стадий реинжиниринга и средства стыковки

приложений, закрепленные стандартами серии IDEF. Большинство этих средств позволяют осуществлять коллективную разработку моделей.

Ни одна из перечисленных категорий программных средств не охватывает всего многообразия задач проектирования и реинжиниринга, поэтому развитие методической и программной базы в этом направлении активно продолжается. Проектирование и совершенствование экономических систем сопряжено с решением задач прогнозирования, структурного синтеза, коллективного выбора рациональных решений. Поддержку в решении этих проблем могут оказать разработки в области нейросетевых и мульти-агентных технологий, *DM&KDD*, экспертные системы и эволюционные алгоритмы.

Кратко остановимся на возможностях одной из самых перспективных многофункциональных систем *ReThink+G2*, разработанной фирмой Gensym. В этой системе соединены графический объектно-ориентированный язык для описания моделей, средства имитационного моделирования с анимацией исследуемых процессов, интеллектуальные способы представления и обработки экспертных знаний об исследуемых процессах.

Система *ReThink* является проблемно-ориентированным приложением, построенным на базе инструментального комплекса *G2*, который предназначен для создания экспертных систем реального времени [34]. Для представления моделей бизнес-процессов используются диаграммы, состоящие из блоков и соединений. Блоками обозначают задачи, а соединения соответствуют потокам информации и материальных сущностей (например, товаров). Свойства блоков могут быть представлены точными или случайными величинами. Система *ReThink* позволяет описывать модели с различной степенью детализации на основе иерархического представления. Процесс функционирования моделей демонстрируется средствами анимации. Для объективного сравнения проектов в системе предусмотрено формирование их стоимостных и временных характеристик. Кроме того, существует возможность проверки гипотез типа «Что — Если» на основе реализации различных сценариев.

Система является открытой для добавления приложений и баз знаний. Она ориентирована на коллективную работу с приложениями на основе архитектуры «клиент-сервер».

Следует отметить, что успех применения BPR (Business Process Reengineering) зависит от множества внешних условий, в

которых работают предприятия, а также от ряда внутренних факторов. Проанализируем возможности применения BPR на отечественных предприятиях в современных условиях.

1. BPR ориентирован на увеличение эффективности работы компаний в десятки раз. За счет чего это можно обеспечить? Здесь есть два основных источника: внедрение информационных технологий и повышение производительности труда. Существенный эффект от внедрения информационных технологий можно ожидать там, где управление было практически неавтоматизированным. При этом часто возникают проблемы, связанные с ошибками, допущенными в моделях и в программном обеспечении, отсутствием подготовленных кадров, с необязательностью производственных и коммерческих партнеров, быстро меняющейся правовой и налоговой базой и т.п. Что касается повышения производительности и качества труда, то этот ресурс не бесконечен, поэтому трудно ожидать очень большого эффекта.

2. На каких предприятиях возможно проведение BPR? Попытаемся ответить на этот вопрос, помня о том, что, во-первых, на проведение BPR нужны немалые средства, а во-вторых, внедрение результатов BPR предполагает наличие кадров, которые способны к выполнению новых задач и труд которых будет должным образом оплачиваться. Очевидно, к таким предприятиям нельзя отнести простаивающие заводы, бюджетные организации со скудным финансированием, предприятия ЖКХ и большую часть малого бизнеса. Следовательно, BPR по силам компаниям с хорошими финансовыми показателями, которые имеют возможность привлекать квалифицированные кадры.

3. Статистические данные о результатах проведения реинжиниринга для развитых стран свидетельствует о том, что далеко не во всех случаях они были успешными и что существенного повышения эффективности добиваются, как правило, компании с высокими показателями своей управленческой деятельности. Получается, что там, где действительно есть серьезные проблемы с управлением, таким способом их решить нельзя.

Все вышесказанное следует учитывать при выборе стратегии проектирования или совершенствования структуры предприятия. Многие из российских предприятий просто не готовы к проведению BPR, пока не сформированы стратегии их устойчивого развития, направления и виды деятельности, не обследованы рынок и состояние материальных и трудовых ресурсов. Если руко-

водство предприятия понимает BPR как компьютеризацию управленческой деятельности (весьма распространенная точка зрения), то оно должно разобраться прежде всего с тем, подходит ли предлагаемая система автоматизации для работы в конкретных условиях данного предприятия и каковы гарантии того, что она будет работать. Быть может, такой подход позволит избежать неоправданных затрат огромных средств на технику, программное обеспечение и услуги консалтинговых фирм и исключит возможность применения этой перспективной технологии в качестве инструмента для «отмывания денег».

Таким образом, ясное понимание целей и возможностей применяемых технологий проектирования, в частности BPR, является необходимым условием для правильного выбора стратегии и методов совершенствования экономических систем.

8.4.

ПОДХОД К КОЛЛЕКТИВНОМУ ВЫБОРУ РЕШЕНИЙ ПРИ ПРОЕКТИРОВАНИИ ЭКОНОМИЧЕСКИХ СИСТЕМ

Одной из ключевых проблем в процессах проектирования экономических систем является принятие решений. Следует заметить, что большая часть рассмотренных ранее интеллектуальных систем ориентирована именно на поддержку процессов принятия решений, которые отличаются большим разнообразием. Интеллектуальные информационные системы, используя заложенные в них знания, способны помочь специалисту поставить диагноз, построить прогноз, выбрать рациональный вариант действий и т. п. При проектировании социально-экономических систем возникают задачи коллективного многокритериального выбора, осложненные наличием взаимных требований его участников. Например, предприниматель, который собирается открыть собственное дело, должен выбрать вид и форму деятельности, подобрать кадры, найти инвесторов и партнеров, т. е. он должен совершить одновременный выбор из нескольких множеств взаимосвязанных альтернатив. Кроме того, выбираемые альтернативы могут предъявлять определенные требования к предпринимателю. Подобные задачи не вписываются в традиционные постановки задач принятия решений, где обычно рассматривается одно

конечное множество альтернатив, элементы которого считаются не зависящими друг от друга. Коллективный выбор с взаимными требованиями участников включает задачу синтеза вариантов возможных решений, каждое из которых представляет собой комбинацию альтернатив из различных множеств, и последующую многокритериальную оценку полученных вариантов.

Рассмотрим подход к решению проблемы коллективного выбора с учетом взаимных требований сторон, которая типична для задач проектирования и совершенствования экономических систем.

Пусть предприятие-производитель транспортных средств (X_1) выпускает для них пружинные рессоры. В условиях затянувшегося кризиса спрос на готовую продукцию невелик, в связи с чем возникает проблема модернизации производства и освоения выпуска продукции, требующейся на рынке. Маркетинговые исследования выявили множество предприятий-потребителей (X_2), готовых приобретать виброзащитные устройства для транспортных средств, которые они производят или собирают. При этом определилось множество виброзащитных систем (ВЗС), интересующих потребителей. Для удовлетворения запросов потребителей необходимо привлечение инвесторов (X_3) и приобретение соответствующих технологий (X_4), которые позволяют выпускать требуемые виброзащитные системы (X_5). При решении вопроса о модернизации производства следует учитывать характеристики инфраструктуры предприятия (X_6) и позиции местных властей (X_7). Супермножество участников процесса принятия решений $X = \{X_1, X_2, \dots, X_7\}$ включает семь объектов, каждый из которых является множеством:

$X_1 = \{X_1\}$ — производители ВЗС;

$X_2 = \{X_{21}, X_{22}, X_{23}\}$ — потребители ВЗС;

$X_3 = \{X_{31}, X_{32}, X_{33}\}$ — инвесторы;

$X_4 = \{X_{41}, X_{42}, X_{43}, X_{44}\}$ — технологии производства ВЗС;

$X_5 = \{X_{51}, X_{52}, X_{53}\}$ — типы ВЗС;

$X_6 = \{X_6\}$ — инфраструктура предприятия;

$X_7 = \{X_7\}$ — местные власти.

Каждый элемент из множеств $X_i, i = 1, \dots, 7$, описывается набором свойств $P_{Xi} = \{P_{Xij}\}$ и набором требований $R_{Xi} = \{R_{Xik}\}$, которые он может предъявлять к остальным участникам процесса принятия решений. Главная цель — найти решение, наиболее приемлемое для всех участников, т.е. найти оптимальную в определенном смысле комбинацию из элементов множеств X_i .

Граф задачи коллективного выбора приведен на рис. 8.1, где дуги показывают направления требований участников. В задаче участвует одно предприятие-производитель, которое является организатором и главным действующим лицом процесса принятия решений. Множества X_6 и X_7 также являются одноэлементными. Множество потребителей продукции включает три элемента, при этом X_{21} обозначает самого производителя (так как он использует часть выпускаемых ВЗС для дальнейшего производства транспортных средств), а X_{22} и X_{23} соответствуют внешним потребителям. Элементы множества X_3 – возможные инвесторы, X_{31} и X_{32} – внешние инвесторы, X_{33} – сам производитель, осуществляющий частичное инвестирование.

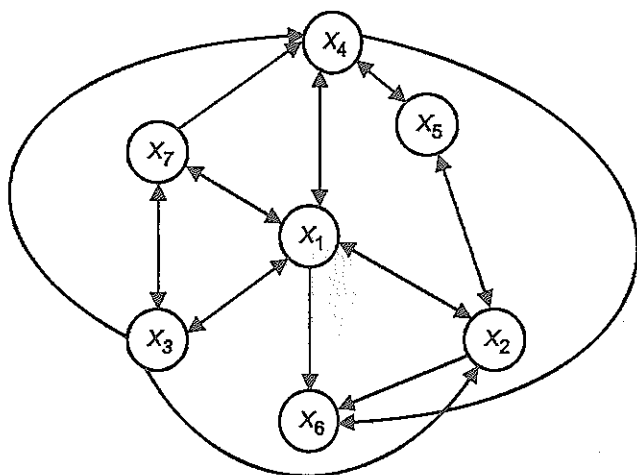


Рис. 8.1. Граф задачи коллективного принятия решений

В задаче рассматриваются четыре доступных для приобретения технологии производства ВЗС (элементы множества X_4), при этом технология X_{41} уже есть у производителя и используется для выпуска пружинных рессор. Элементами множества X_5 являются типы ВЗС, которые можно было бы производить. Среди них X_{51} – пневматическая рессора; X_{52} – пружинная рессора; X_{53} – управляемая (активная) ВЗС.

Информация о свойствах и требованиях рассматриваемых объектов, каждый из которых является участником процесса вы-

бора, может быть представлена лингвистическими или количественными оценками. Лингвистическим оценкам могут соответствовать либо дескрипторы (наименования), либо нечеткие множества. Требования могут формулироваться в виде равенств или неравенств, например

$$P_{Xilk} = R_{Xjmk}; P_{Xilk} \leq R_{Xjmk}; P_{Xilk} \geq R_{Xjmk}; R_{Xjmk}^{\min} \leq P_{Xilk} \leq R_{Xjmk}^{\max},$$

где k — индекс соответствующего критерия, по которому производится сравнение свойства с требованием;

l — индекс выбираемого объекта из множества X_j ;

m — индекс выбирающего объекта из множества X_j .

Для каждого из субъектов множества X_j объекты (субъекты) множества X_i можно упорядочить на основе обобщенной меры сходства множества требований R_{Xj} со свойствами конкретных объектов P_{Xj} . В свою очередь, каждый из объектов множества X_j может предъявлять определенные требования к объектам из множества X_i .

Структура графа взаимных требований описывается матрицей B , в каждой строке которой ненулевые элементы показывают наличие требований, предъявляемых соответствующим участником к остальным субъектам выбора. Число предъявляемых требований может различаться у конкретных представителей каждого множества, но не может превышать количества свойств субъекта, от которого требуется удовлетворение этим требованиям.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_1	0	1	1	1	0	1	1
X_2	1	0	0	0	1	1	0
X_3	1	1	0	1	0	0	1
X_4	1	0	0	0	1	1	0
X_5	0	1	0	1	0	0	0
X_6	0	0	0	0	0	0	0
X_7	1	0	1	1	0	0	0

Степень удовлетворения требования субъекта A к k -му свойству объекта (субъекта) B отражает мера сходства $S_k(R_{Ak}, P_{Bk})$, способ вычисления которой зависит от представления информации о свойствах и требованиях.

Для требований в виде равенств и свойств, выраженных количественными оценками, меру сходства можно вычислить по формулам:

$$S_k(R_{Ak}, P_{Bk}) = \frac{2\min(R_{Ak}, P_{Bk})}{R_{Ak} + P_{Bk}}; \quad S_k(R_{Ak}, P_{Bk}) = \frac{\min(R_{Ak}, P_{Bk})}{\max(R_{Ak}, P_{Bk})}.$$

Мера сходства равна единице, если $R_{Ak} = P_{Bk}$, и уменьшается с увеличением разности значений R_{Ak} и P_{Bk} . При лингвистическом описании свойств символьные строки либо совпадают, либо нет, при этом сходство по k -му критерию оценивается по двух-балльной шкале [1, 0]:

$$S_k(R_{Ak}, P_{Bk}) = \begin{cases} 1, & \text{если } R_{Ak} \cap P_{Bk} \neq \emptyset; \\ 0, & \text{если } R_{Ak} \cap P_{Bk} = \emptyset. \end{cases}$$

Если R_{Ak} и P_{Bk} представляют собой нечеткие множества, соответствующие лингвистическим оценкам, скалярный индекс сходства можно вычислить, например, по формуле Жаккара [22]:

$$S_k(R_{Ak}, P_{Bk}) = \frac{|R_{Ak} \cap P_{Bk}|}{|R_{Ak} \cup P_{Bk}|},$$

где операция $|X|$ обозначает скалярную мощность нечеткого множества X , заданного на базовом множестве U : $|X| = \sum_{u \in U} \mu_X(u)$.

Индекс сходства $S(A, B)$ нечетких множеств A и B обладает следующими свойствами:

1) $S(A, B) = 1$ тогда и только тогда, когда симметрическая разность нечетких множеств $A \Delta B = \emptyset$;

2) если носители нечетких множеств A и B не пересекаются, то $S(A, B) = 0$;

3) $S(A, B) = S(B, A)$.

Степень удовлетворения требований, сформулированных в виде неравенств вида $P_{Bk} \leq R_{Ak}$ или $P_{Bk} \geq R_{Ak}$, отражается с помощью мер включения, которые для количественных оценок можно вычислить соответственно по формулам:

$$I_k(P_{Bk}, R_{Ak}) = \frac{\min(P_{Bk}, R_{Ak})}{P_{Bk}}$$

или

$$I_k(R_{Ak}, P_{Bk}) = \frac{\min(P_{Bk}, R_{Ak})}{R_{Ak}}.$$

Если требование удовлетворяется, значение меры включения равно 1, в противном случае $0 < I_k(P_{Bk}, R_{Ak}) < 1$ (предполагается, что значения P_{Bk}, R_{Ak} положительны). Недостатком такой меры является ее неспособность отразить степень удовлетворения требованиям при соблюдении условий $P_{Bk} \leq R_{Ak}$ или $P_{Bk} \geq R_{Ak}$. При дескриптивном описании оценка $S_k(R_{Ak}, P_{Bk})$ не отличается от случая равенства. Если требования и свойства представлены нечеткими множествами, можно использовать скалярные индексы включения нечеткого множества P_{Bk} в R_{Ak} :

$$I_k(P_{Bk}, R_{Ak}) = \frac{|R_{Ak} \cap P_{Bk}|}{|P_{Bk}|}$$

или R_{Ak} в P_{Bk} :

$$I_k(R_{Ak}, P_{Bk}) = \frac{|R_{Ak} \cap P_{Bk}|}{|R_{Ak}|}.$$

Этот индекс обладает следующими свойствами:

- 1) индекс включения нечеткого множества A в B $I(A, B) = 1$ тогда и только тогда, когда $A \cap \bar{B} = \emptyset$;
- 2) если носители нечетких множеств A и B не пересекаются, то $I(A, B) = 0$.

Если требования заданы в виде интервала $R_{Ak}^{\min} \leq P_{Bk} \leq R_{Ak}^{\max}$, то для количественных оценок можно использовать меру частично-го совпадения

$$PM_k(R_{Ak}, P_{Bk}) = \min \left(\frac{\min(P_{Bk}, R_{Ak}^{\max})}{P_{Bk}}, \frac{\min(P_{Bk}, R_{Ak}^{\min})}{R_{Ak}^{\min}} \right).$$

Если значение $PM_k(R_{Ak}, P_{Bk}) = 1$, то свойства объекта B полностью удовлетворяют требованиям субъекта A . Для нечетких оценок используется скалярный индекс частичного совпадения

$$PM_k(R_{Ak}, P_{Bk}) = \min \left(\frac{|R_{Ak}^{\min} \cap P_{Bk}|}{|R_{Ak}^{\min}|}, \frac{|R_{Ak}^{\max} \cap P_{Bk}|}{|P_{Bk}|} \right),$$

обладающий свойствами:

- 1) $PM(A, B) = 0$ тогда и только тогда, когда $A \cap B = \emptyset$;
- 2) условие $PM(A, B) = 1$ не равносильно требованию строгого включения.

При дескриптивных описаниях свойств и требований применяется стандартная оценка на совпадение.

Формирование интегральной оценки степени удовлетворения взаимных требований для каждой пары участников процесса выбора является двухэтапной процедурой, которая может выполняться различными способами в зависимости от используемых принципов оптимальности. Один из этапов связан с вычислением обобщенной оценки удовлетворения множества требований каждого объекта, другой этап заключается в построении оценки степени взаимного удовлетворения требований в паре. Для формирования обобщенных оценок в работе [6] предложены функции, отражающие разные способы достижения компромисса при многих критериях и участниках процесса выбора. Рассмотрим некоторые из них.

Аддитивная функция выбора:

$$F_1 = \frac{1}{2} \left[\frac{1}{K_1} \sum_{k=1}^{K_1} S_k(R_{Ak}, P_{Bk}) + \frac{1}{K_2} \sum_{k=1}^{K_2} S_k(R_{Bk}, P_{Ak}) \right],$$

где K_1 — количество требований, предъявляемых субъектом A к B ;
 K_2 — число требований B к A ;

Взвешенная аддитивная функция:

$$F_2 = \alpha_A \sum_{k=1}^{K_1} w_k S_k(R_{Ak}, P_{Bk}) + \alpha_B \sum_{k=1}^{K_2} v_k S_k(R_{Bk}, P_{Ak}),$$

где w_k и v_k — нормированные весовые коэффициенты требований субъектов A и B соответственно;

α_A, α_B — весовые коэффициенты влияния субъектов выбора.

Аддитивные функции F_1 и F_2 реализуют принцип равномерной оптимальности, который применяется как для свертки множества требований каждого участника, так и для вычисления оценки в паре. Операция суммирования допускает возможность компенсации низких значений высокими, что может привести к выбору такой пары, в которой требования одной стороны учтены полностью, а требования другой — проигнорированы. Главным преимуществом аддитивных функций является высокая вероятность получения непустого выбора, который соответствует выполнению условий $(\exists d) F_1(d) \neq 0$ или $(\exists d) F_2(d) \neq 0, d \in D, D = A \times B$.

Мультипликативные функции, реализующие принцип справедливого компромисса:

$$F_3 = \prod_{k=1}^{K_I} S_k(R_{Ak}, P_{Bk}) \prod_{k=1}^{K_z} S_k(R_{Bk}, P_{Ak});$$

$$F_4 = \left[\prod_{k=1}^{K_I} S_k^{w_k}(R_{Ak}, P_{Bk}) \right]^{\alpha_A} \left[\prod_{k=1}^{K_z} S_k^{v_k}(R_{Bk}, P_{Ak}) \right]^{\alpha_B}.$$

В отличие от аддитивных эти функции не допускают взаимной компенсации оценок. Недостатком мультипликативной целевой функции является весьма высокая вероятность пустого выбора, так как даже при одном неудовлетворенном требовании значение функции обобщенной оценки обращается в нуль.

Сочетание различных принципов компромисса в одной функции, как правило, дает наиболее приемлемый результат [6].

Аддитивно-мультипликативная функция:

$$F_5 = \frac{1}{K_I K_z} \sum_{k=1}^{K_I} S_k(R_{Ak}, P_{Bk}) \sum_{k=1}^{K_z} S_k(R_{Bk}, P_{Ak}).$$

Эта функция допускает компенсацию оценок степени удовлетворения требований одного субъекта, но не допускает компенсации оценок в паре.

Взвешенная аддитивно-мультипликативная функция:

$$F_6 = \left[\prod_{k=1}^{K_I} w_k S_k(R_{Ak}, P_{Bk}) \right]^{\alpha_A} \left[\prod_{k=1}^{K_z} v_k S_k(R_{Bk}, P_{Ak}) \right]^{\alpha_B}.$$

При использовании таких функций выбор может оказаться пустым только в том случае, если требования одной из сторон полностью не удовлетворены.

Обобщение задачи взаимного выбора на множество участников $X = \{X_1, X_2, \dots, X_N\}$ связано с поиском такой комбинации субъектов из множеств $X_i, i = 1, \dots, N$, требования которых удовлетворяются наилучшим образом. Для этой комбинации значение обобщенной оценки степени удовлетворения взаимных требова-

ний субъектов будет максимальным. Функции для вычисления обобщенной степени удовлетворения требований на основе различных принципов компромисса имеют вид:

Принцип максимальной эффективности:

$$G_1(d) = \max_{d \in D} \sum_{j=1}^N \sum_{\substack{i=1 \\ i \neq j}}^N v_j \sum_{k=1}^{K_i} w_{jk} S_k(R_{X_j^m k}, P_{X_i^l k}).$$

Принцип максимального равенства участников выбора:

$$G_2(d) = \max_{d \in D} \prod_{j=1}^N \prod_{\substack{i=1 \\ i \neq j}}^N \left[\prod_{k=1}^{K_i} S_k^{w_{jk}}(R_{X_j^m k}, P_{X_i^l k}) \right]^{v_j}.$$

Комбинированный принцип «эффективность – равенство»:

$$G_3(d) = \max_{d \in D} \prod_{j=1}^N \prod_{\substack{i=1 \\ i \neq j}}^N \left[\sum_{k=1}^{K_i} w_{jk} S_k(R_{X_j^m k}, P_{X_i^l k}) \right]^{v_j}.$$

Здесь $D = X_1 \times X_2 \times \dots \times X_N$; d – кортеж из N -арного отношения D ; k – индекс требования; w_{jk} – вес k -го требования j -го субъекта; v_j – весовой коэффициент j -го субъекта; K_i – число свойств i -го субъекта; m, l – индексы альтернатив из множеств X_j и X_i соответственно; $S_k(R_{X_j^m k}, P_{X_i^l k})$ – мера сходства k -го требования участника X_j^m с соответствующим свойством субъекта (объекта) X_i^l .

Вернемся к задаче о модернизации производства (см. рис. 8.1). В табл. 8.1 – 8.7 приведены описания свойств и требований субъектов выбора, представленные количественными и лингвистическими оценками.

В рассматриваемой задаче возможно 108 различных комбинаций из элементов множеств $X_1, \dots, X_7$, образующих систему. Степень удовлетворения требований сторон вычислялась на основе принципа «эффективность – равенство» G_3 , при этом важность участников и требований считалась одинаковой.

Полученные значения обобщенных оценок характеризуют степень согласия всех сторон. На рис. 8.2 приведены результаты для 36 комбинаций. Для остальных решений степень согласия

была равна нулю. Так как множества X_1 , X_6 и X_7 содержат по одному элементу, которые входят во все комбинации, они не включены в приведенные обозначения.

На графиках хорошо видно, что в комбинациях с участием инвестора X_{32} степень согласия всех сторон существенно ниже, чем в комбинациях с другими инвесторами. Легко заметить, что невысокая степень согласия имеет место в комбинациях с участием потребителя X_{23} . Следовательно, можно сделать вывод, что

Таблица 8.1

Свойства и требования к производителям ВЗС (X_1)

Субъект	Экономический потенциал $P_1(X_1)$	Квалификация кадров $P_2(X_1)$	Технический потенциал $P_3(X_1)$	Репутация $P_4(X_1)$	Производственная мощность $P_5(X_1)$
X_1	Средний	Высокий	Средний	Средняя	1500
Субъект	Требования к производителям				
	$R_1(X_{ij}, X_1)$	$R_2(X_{ij}, X_1)$	$R_3(X_{ij}, X_1)$	$R_4(X_{ij}, X_1)$	$R_5(X_{ij}, X_1)$
X_{21}	—	—	—	—	300 — 500
X_{22}	—	—	—	Высокая	500
X_{23}	—	—	—	Высокая	1000
X_{31}	Не ниже Среднего	Высокая	Не ниже Среднего	Высокая	Не менее 1000
X_{32}	Высокий	Высокая	Не ниже Среднего	Не ниже Средней	Не менее 500
X_{33}	—	—	—	—	—
X_{41}	Не ниже Среднего	Не ниже Средней	—	—	—
X_{42}	Не ниже Среднего	Высокая	—	—	—
X_{43}	Не ниже Высокого	Высокая	—	—	—
X_{44}	Не ниже Среднего	Высокая	—	—	—
X_7	Не ниже Среднего	—	—	Не ниже Средней	—

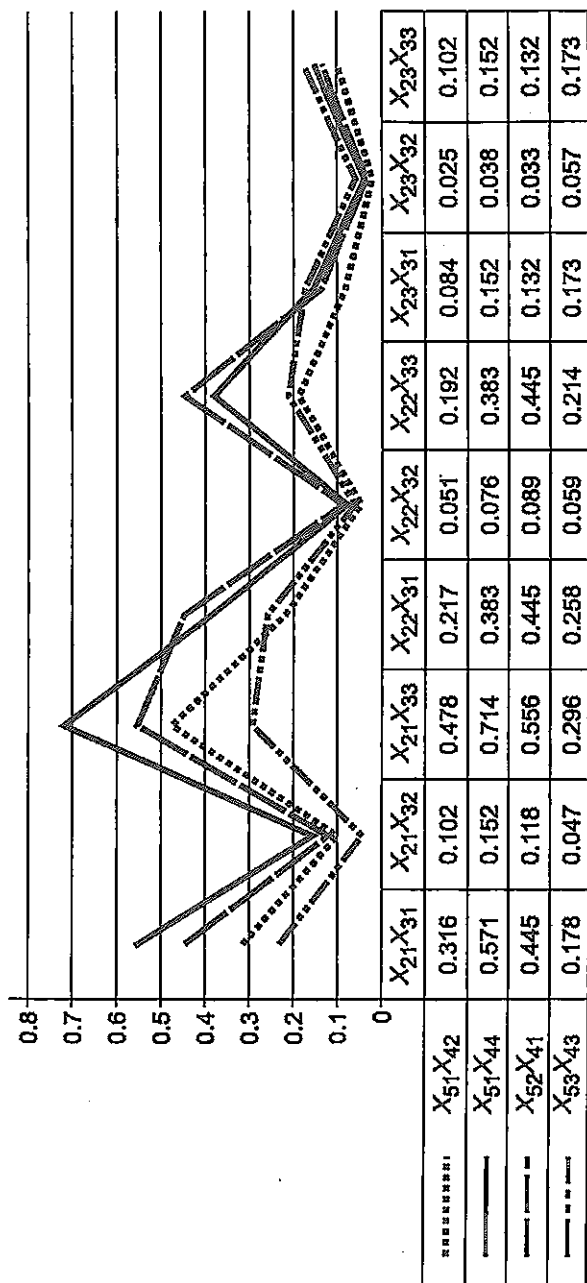


Рис. 3.2. Результаты решения задачи коллективного выбора

Свойства потребителей ВЗС и требования к ним (X_2)

Субъект	Потребность в ВЗС	Возможность оплаты продукции	Репутация	Условия эксплуатации		
				Опасность механических повреждений $P_4(X_{2j})$	Пространство для размещения $P_5(X_{2j})$	Вероятность коррозии $P_6(X_{2j})$
	$P_1(X_{2j})$	$P_2(X_{2j})$	$P_3(X_{2j})$			
X_{21}	300	Средняя	Высокая	Есть	Большое	Высокая
X_{22}	1000	Средняя	Средняя	Нет	Среднее	Высокая
X_{23}	500 — 1500	Высокая	Низкая	Есть	Малое	Низкая
Субъект	Требования к потребителям ВЗС					
	$R_1(X_{ij}, X_2)$	$R_2(X_{ij}, X_2)$	$R_3(X_{ij}, X_2)$	$R_4(X_{ij}, X_2)$	$R_5(X_{ij}, X_2)$	$R_6(X_{ij}, X_2)$
X_1	300 — 1500	Не ниже Средней	Не ниже Средней	—	—	—
X_{31}	Не менее 500	Не ниже Средней	Не ниже Средней	—	—	—
X_{32}	Не менее 500	Не ниже Высокой	Не ниже Средней	—	—	—
X_{33}	Не менее 500	Не ниже Средней	Не ниже Средней	—	—	—
X_{51}	—	—	—	Требуется защиты	Среднее	Не требует защиты
X_{52}	—	—	—	Не требует защиты	Большое	Требуется защиты
X_{53}	—	—	—	Не требует защиты	Малое	Требуется защиты

Таблица 8.3

Свойства и требования к инвесторам (X_3)

Субъект	Объем инвестиций $P_1(X_3)$	Процентная ставка $P_2(X_3)$	Репутация $P_3(X_3)$
X_{31}	$2 \cdot 10^6$	40	Высокая
X_{32}	$5 \cdot 10^6$	60	Низкая
X_{33}	$1 \cdot 10^6$	—	Высокая
Субъект	Требования к инвесторам		
	$R_1(X_{ij}, X_3)$	$R_2(X_{ij}, X_3)$	$R_3(X_{ij}, X_3)$
X_1	Не менее $1 \cdot 10^6$	Не выше 40	Не ниже Средней
X_7	—	Не выше 50	Высокая

Таблица 8.4

Свойства технологий производства ВЗС и требования к ним (X_4)

Субъект	Стоимость $P_1(X_4)$	Экологич- ность $P_2(X_4)$	Срок освое- ния $P_3(X_4)$	Стоимость материалов $P_4(X_4)$	Энергоем- кость $P_5(X_4)$	Тип ВЗС $P_6(X_4)$	Доступность материалов $P_7(X_4)$
X_{41}	0	Высокая	0	Низкая	Средняя	Пружина	Высокая
X_{42}	$3 \cdot 10^6$	Высокая	1 год	Средняя	Высокая	Пневмо	Средняя
X_{43}	$8 \cdot 10^6$	Высокая	1.5 года	Высокая	Низкая	Активная	Низкая
X_{44}	$1.5 \cdot 10^6$	Средняя	0.5 года	Средняя	Средняя	Пневмо	Высокая
Субъект	Требования к технологиям						
	$R_1(X_{ij}, X_4)$	$R_2(X_{ij}, X_4)$	$R_3(X_{ij}, X_4)$	$R_4(X_{ij}, X_4)$	$R_5(X_{ij}, X_4)$	$R_6(X_{ij}, X_4)$	$R_7(X_{ij}, X_4)$
X_1	Не боль- ше $5 \cdot 10^6$	Не хуже Средней	Не боль- ше 1 года	Не выше Средней	Не выше Средней	—	Не ниже Средней
X_{31}	Не боль- ше $2 \cdot 10^6$	—	Не боль- ше 1 года	—	—	—	—
X_{32}	Не боль- ше $5 \cdot 10^6$	—	Не боль- ше 1 года	—	—	—	—
X_{33}	—	—	Не боль- ше 1 года	—	—	—	—
X_{51}	—	—	—	—	—	Пневмо	—
X_{52}	—	—	—	—	—	Пружина	—
X_{53}	—	—	—	—	—	Актив- ная	—
X_7	—	Высокая	—	—	—	—	—

Свойства и требования к ВЗС (X_5)

Субъект	Качество виброзащиты $P_1(X_5)$	Надежность $P_2(X_5)$	Стоимость $P_3(X_5)$	Композитные свойства $P_4(X_5)$	Принцип действия $P_5(X_5)$	Ориентация $P_6(X_5)$	Разные частоты $P_7(X_5)$
X_{51}	Высокое	Средняя	Средняя	Средние	Пневмо	Средняя	Узкий спектр
X_{52}	Среднее	Высокая	Низкая	Низкие	Пружина	Средняя	Одна частота
X_{53}	Высокое	Высокая	Высокая	Высокие	Активная	Высокая	Широкий спектр
Субъект	Требования к ВЗС						
	$R_1(X_{ij}, X_5)$	$R_2(X_{ij}, X_5)$	$R_3(X_{ij}, X_5)$	$R_4(X_{ij}, X_5)$	$R_5(X_{ij}, X_5)$	$R_6(X_{ij}, X_5)$	$R_7(X_{ij}, X_5)$
X_{21}	Не ниже	Не ниже	Не выше	Не ниже	—	Не ниже	Одна и более
X_{22}	Высокого	Средней	Средней	Средних	—	Средней	Узкий спектр
X_{23}	Не ниже	Не ниже	Не выше	Не ниже	—	Не ниже	Средний спектр
X_{23}	Среднего	Высокой	Средней	Высоких	—	Средней	Широкий спектр
X_{23}	Не ниже	Не ниже	Не выше	Не ниже	—	Не ниже	Широкий спектр
X_{41}	Высокого	Высокой	Высокой	Высоких	—	Высокой	—
X_{41}	—	—	—	—	Пружина	—	—
X_{42}	—	—	—	—	Пневмо	—	—
X_{43}	—	—	—	—	Активная	—	—
X_{44}	—	—	—	—	Пневмо	—	—

Таблица 8.6

Свойства и требования к инфраструктуре (X_6)

Субъект	Цена электроэнергии $P_2(X_6)$	Состояние коммуникаций $P_3(X_6)$	Транспортное сообщение $P_1(X_6)$
X_6	Высокая	Хорошее	Хорошее
Субъект	Требования к инфраструктуре		
	$R_1(X_{ij}, X_6)$	$R_2(X_{ij}, X_6)$	$R_3(X_{ij}, X_6)$
X_1	Не выше Высокой	Хорошее	Хорошее
X_{21}	—	—	—
X_{22}	—	—	Не хуже Среднего
X_{23}	—	—	Не хуже Хорошего
X_{41}	Не выше Высокой	Хорошее	Не хуже Среднего
X_{42}	Не выше Средней	Хорошее	Не хуже Среднего
X_{43}	—	Хорошее	Не хуже Хорошего
X_{44}	Не выше Высокой	Хорошее	Не хуже Хорошего

Свойства и требования к властям (X_7)

Субъект	Поддержка в СМИ $P_1(X_7)$	Налоговые льготы $P_2(X_7)$	Лоббирование интересов $P_3(X_7)$
X_7	Если <i>Репутация</i> $X_{ij} = \text{Высокая}$, то Высокая, иначе Низкая	Если <i>Экономический</i> <i>потенциал</i> $\geq$ <i>Средний</i> и <i>Технический потен-</i> <i>циал</i> $\geq$ <i>Средний</i> , то Предоставляются, иначе Не предостав- ляются	Низкое
Субъект	Требования к властям		
	$R_1(X_{ij}, X_7)$	$R_2(X_{ij}, X_7)$	$R_3(X_{ij}, X_7)$
X_1	—	Требуются	—
X_{31}	Высокая	—	—
X_{32}	Средняя	—	Высокое
X_{33}	Низкая	Требуются	—

наиболее приемлемыми партнерами для производителя будут инвестор X_{31} и потребитель X_{22} . Для собственного потребления (X_{21}) желательно производить пневматические рессоры по технологии X_{44} , на втором месте по согласованию требований находится пружинная рессора, на третьем — пневматическая подвеска, производимая по технологии X_{42} , и на последнем месте — активная виброзащитная система. Для потребителя X_{22} лучшей является пружина, с небольшим отставанием от нее следует пневматическая ВЗС по технологии X_{44} , затем активная ВЗС и пневматическая по технологии X_{42} . Этот результат соответствует совокупности требований данного потребителя (см. табл. 8.5) и аддитивному принципу свертки отдельных требований. Значительно уступающая по качеству виброзащиты другим альтернативам пружинная рессора выходит на первое место, поскольку потребитель X_{22} не требует очень высокого качества виброизоляции и располагает достаточным пространством для размещения ВЗС при компоновке подвески транспортного средства. В то же время для него важны надежность и низкая стоимость. Третий потребитель предпочитает активную ВЗС, снабженную средствами управления, так как его требования к качеству виброизоляции очень высоки (см. табл. 8.5) и он располагает достаточными финансовыми ресурсами, чтобы заплатить высокую цену.

Полученные результаты позволяют рекомендовать производителю приобрести технологию X_{44} для производства пневматических рессор и продолжать производить пружинные рессоры. При этом имеет смысл ориентироваться на кооперацию с инвестором X_{31} и потребителем X_{22} . Варианты с участием X_{21} и X_{33} , имеющие очень хорошие оценки согласованности, свидетельствуют о том, что производитель является надежным потребителем своей продукции и инвестором нового производства.

Анализ результатов промежуточных расчетов степени согласия в формируемых комбинациях облегчает процесс интерпретации решений. Например, на рис. 8.3 приведены результаты расчета степени согласия в парах «потребитель (X_2) – Тип производимой ВЗС (X_5)». Самые высокие оценки степени удовлетворения взаимных требований для всех потребителей имеет активная ВЗС, второе место принадлежит пневматической системе, а пружинной – последнее.

Однако учет требований производителя и инвесторов к технологии производства активной ВЗС (X_{43}) приводит в дальнейшем к существенному снижению обобщенных оценок степени удовлетворения требований для комбинаций, содержащих альтернати-

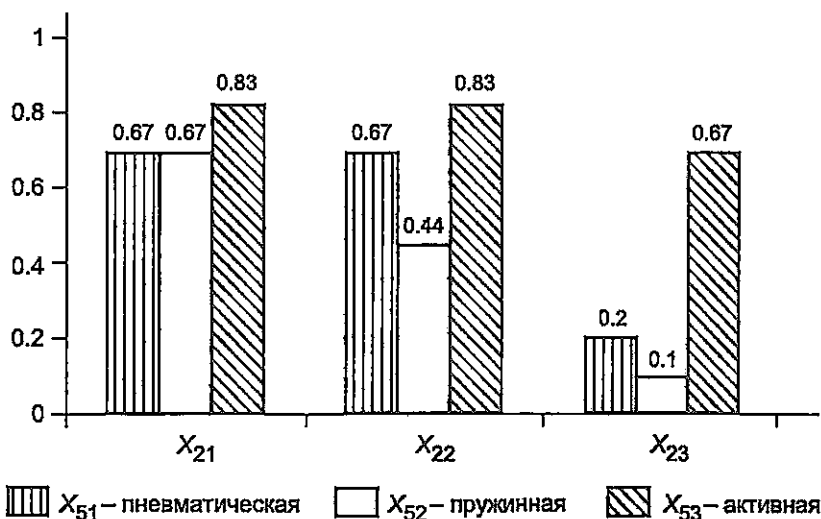


Рис. 8.3. Сходство взаимных требований в комбинациях «Потребитель – Тип ВЗС»

вы $X_{53}X_{43}$. Это связано с высокой стоимостью технологии производства активных ВЗС, малой доступностью материалов и длительным сроком освоения. Наиболее приемлемыми технологиями для участников рассматриваемой задачи коллективного выбора являются X_{41} и X_{44} . В связи с этим предпочтительными для производства становятся пневматическая и пружинная системы виброзащиты. Такие системы мало подходят потребителю X_{23} , поэтому комбинации с его участием имеют невысокую степень согласия взаимных требований.

Низкие оценки степени удовлетворения требований в комбинациях с инвестором X_{32} обусловлены его плохой репутацией и высокой процентной ставкой. Это значит, что партнерство с X_{32} сопряжено с высоким риском.

В рассматриваемой задаче нет ни одной комбинации, где удовлетворяются все без исключения требования сторон. При жестком способе формирования обобщенной оценки мультипликативный критерий будет иметь нулевое значение. Использование весовых коэффициентов требований и их изменение в процессе поиска компромисса может приводить к другим результатам. Так, ослабление требований, связанных со стоимостью технологий и материалов (что возможно при достаточном финансировании), приводит к увеличению степени согласия в комбинациях, включающих активную ВЗС и технологию ее производства ($X_{53}X_{43}$). Производство активной ВЗС становится предпочтительнее, чем производство пружины, однако уступает пневматическим рессорам из-за низкой доступности материалов и длительного срока окупаемости, который важен для производителя и инвесторов.

8.5.

РАЗРЕШЕНИЕ КОНФЛИКТОВ ПРИ КОЛЛЕКТИВНОМ ВЫБОРЕ РЕШЕНИЙ

Описанный подход к задачам коллективного принятия решений с взаимными требованиями участников позволяет выявить конфликты между требованиями субъектов выбора. В практике часто встречаются ситуации с конфликтующими требованиями, поэтому для их разрешения в интеллектуальных системах должны быть предусмотрены средства разрешения конфликтов.

Обычно конфликты разрешаются либо путем взаимных уступок (изменения требований), либо путем изменения множества рассматриваемых решений. Можно привести не один пример из реальной жизни, когда два собственника, боровшиеся за контроль над предприятием и не склонные к компромиссу, были вынуждены изменить структуру управления и/или состав собственников.

Для разрешения конфликтов между участниками коллективного выбора хорошие перспективы имеют мультиагентные системы. Рассмотрим работу мультиагентной системы «Коллективный выбор» на примере задачи модернизации производства, график которой показан на рис. 8.1. В этой задаче выделено семь элементов-множеств $X_1, X_2, \dots, X_7$, среди которых есть одноэлементные множества X_1, X_6, X_7 . Организатором и инициатором процесса является производитель $X_1 = \{X_1\}$, который отвечает за координацию поведения агентов в мультиагентной системе. Элемент $X_6 = \{X_6\}$ содержит описание параметров инфраструктуры предприятия, возможности которой ограничены и поэтому представлены в виде требований (имеющих смысл ограничений) к другим субъектам выбора. Так как эти требования чаще всего не подлежат изменениям, то субъект X_6 не требует представления в виде агента, хотя это возможно в случае необходимости. В рассматриваемой задаче агентами представлены производитель (X_1), множества потребителей (X_2), инвесторов (X_3), владельцев технологий (X_4), полномочных представителей местных властей (X_7). Информация о вариантах продукции, которую можно было бы производить (X_5), как и информация о параметрах инфраструктуры, будет храниться в базе знаний мультиагентной системы, поскольку множества X_5 и X_6 имеют статус объектов, а не активно действующих субъектов.

Производитель X_1 , исполняющий множество «ролей», представлен несколькими агентами: агентом-производителем (X_1), агентом-потребителем (X_{21}), агентом-инвестором (X_{33}) и агентом-координатором процесса коллективного выбора.

На первом этапе решения задачи агент-координатор осуществляет поиск потенциальных участников процесса выбора и рассылает им сообщения-запросы об их свойствах и их требованиях к другим участникам. На основе этой информации, а также сведений о возможностях производства, видах продукции и параметрах инфраструктуры заполняется база знаний агента-

координатора и формируются множества объектов и субъектов выбора.

Второй этап связан с предварительным расчетом степени удовлетворения взаимных требований в соответствии с алгоритмом, описанным в разд. 8.4. Предварительный расчет позволяет определить участников с непротиворечивыми требованиями и проанализировать полученные варианты с использованием специальных программ для оценки эффективности полученных вариантов производства. Следует заметить, что на этапе предварительного расчета целесообразно последовательное применение различных принципов компромисса, начиная с самого «жесткого» G_2 и заканчивая самым «мягким» G_1 . Между ними находится комбинированный принцип G_3 . Если результаты предварительного расчета не выявили наличия конфликтов между участниками и устраивают пользователя агента-координатора, то задачу можно считать решенной. В противном случае осуществляется переход к следующему этапу.

На третьем этапе агент-координатор предпринимает попытки разрешить конфликты между субъектами выбора. Эта работа начинается с выявления конфликтов, разрешение которых наиболее актуально. Множество таких конфликтов формируется на основе вычисленных оценок мер сходства требований со свойствами и упорядочивается в соответствии с весовыми коэффициентами субъектов, а также важностью их требований. В множество конфликтов заносятся оценки мер сходства, значения которых ниже заданного порога Δ . Агент-координатор рассылает владельцам соответствующих требований сообщения, содержащие предложения изменить конфликтующие требования или их весовые коэффициенты в направлении достижения компромисса. Ответные сообщения агентов анализируются, и в случае положительных изменений производится повторный расчет (возврат на второй этап). Если никто из участников не пожелал внести изменения, то агент-координатор сообщает об этом своему пользователю, который может предпринять следующие меры:

- Повторно попросить субъектов об уступках, предоставив им возможность получения компенсаций.
- Изменить весовые коэффициенты конкретных субъектов выбора.
- Дать указание изменить состав множеств субъектов выбора.

- Изменить структуру задачи и/или информацию в базе знаний путем расширения пространства возможных решений.

- Дать указание скорректировать информацию о свойствах субъектов выбора.

В соответствии с решениями своего пользователя агент-координатор обращается с запросами к агентам субъектов выбора и обрабатывает их ответные сообщения. При этом делаются соответствующие изменения информации в базе знаний.

Структура базы знаний показана на рис. 8.4. Кроме информации об объектах, субъектах, их свойствах и требованиях в ней хранятся граф взаимных требований участников процесса выбора, а также процедуры вычисления результата.

В процессе разрешения конфликтов у агентов запрашивается информация о требованиях, выполнение которых является обязательным. Такие требования снабжаются специальными метками и не могут исключаться из дальнейшего рассмотрения. Остальные требования могут быть исключены, если это способствует достижению согласия. Компенсации представляют собой дополнительные требования участников, которые они могут выдвигать, совершая определенные уступки по основным требованиям. Дополнительные требования могут относиться к имеющимся в БЗ свойствам объектов и субъектов или вызвать необходимость формирования новых записей. Кроме того, дополнительные требования могут быть связаны с выполнением условий, которые невозможно представить средствами субъектно-объектного описания проблемы. Например, инвестор может снизить требуемую долю прибыли в обмен на лишнее место в совете директоров предприятия. Такая информация собирается руководителем и учитывается на неформальной основе.

Предложенный подход имеет следующие преимущества при поиске компромиссных решений:

- упорядоченная процедура сбора и обработки информации;
- уменьшение вероятности манипулирования информацией, получаемой от агентов;
- использование различных принципов достижения компромисса;
- поэтапное приближение к консенсусу путем постепенного изменения требований сторон и их приоритетов;



Рис. 8.4. Структура базы знаний мультиагентной системы для поддержки процессов коллективного выбора

- выявление и устранение противоречий, содержащихся в исходной информации;

- развернутое обоснование получаемых решений, которое позволяет понять, почему принятое решение считается лучшим и для кого оно таковым является.

Описанная выше процедура соответствует естественным человеческим возможностям, не требует от субъектов выбора лишней информации и освоения таких трудоемких процедур, как сравнение векторных оценок альтернатив, построение функций принадлежности, парные сравнения по предпочтительности и т.п., которые необходимы в других методах принятия решений.

8.6.

ЭВОЛЮЦИОННЫЙ СИНТЕЗ СИСТЕМ И ОБЪЕКТОВ

Задача проектирования заключается в создании модели некоторой системы, которая будет способна выполнять предписанные функции с заданным уровнем качества. Требования, предъявляемые к проектируемой системе, формируются вне ее, в метасистеме более высокого уровня, которая также является антропогенной развивающейся системой. Следовательно, формируемые в ней цели и требования к системе более низкого уровня (подсистеме) могут изменяться с течением времени. В связи со сказанным задачи проектирования и перепроектирования экономических систем являются динамическими задачами принятия проектных решений в условиях неопределенности, поэтому им органически присущи многовариантность, многокритериальность, открытость (постоянный обмен информацией с внешней средой) и адаптивность (способность изменять свои свойства и структуру при изменении факторов внешнего окружения). Эффективное решение подобных задач основано на применении принципов эволюционного проектирования (рис. 8.5).

Информация об изменениях метасистемы (внешней среды) используется для моделирования ее поведения. Результаты моделирования позволяют сформировать вектор требований к проектируемой системе $R(t)$, который может изменяться во времени. Задача проектирования сложной системы согласно [17] заключается в синтезе вариантов ее структуры (S_1, S_2, S_3) и выборе варианта, который характеризуется совокупностью свойств (P_1, P_2, P_3), наилучшим образом удовлетворяющих внешним требованиям $R(t)$.

Эволюционный подход к синтезу заключается в построении целостной системы из более простых частей с позиций теории развития, а именно: сложная система синтезируется из элементов под контролем факторов внешней среды, при этом структура системы и состав элементов подбираются так, чтобы обеспечить максимальное удовлетворение внешних требований (естественный отбор). В процессе синтеза происходит соединение элементов (скрещивание), результатом которого являются более крупные объекты, обладающие новыми свойствами и наследующие некоторые свойства «родителей». Из множества возможных ком-

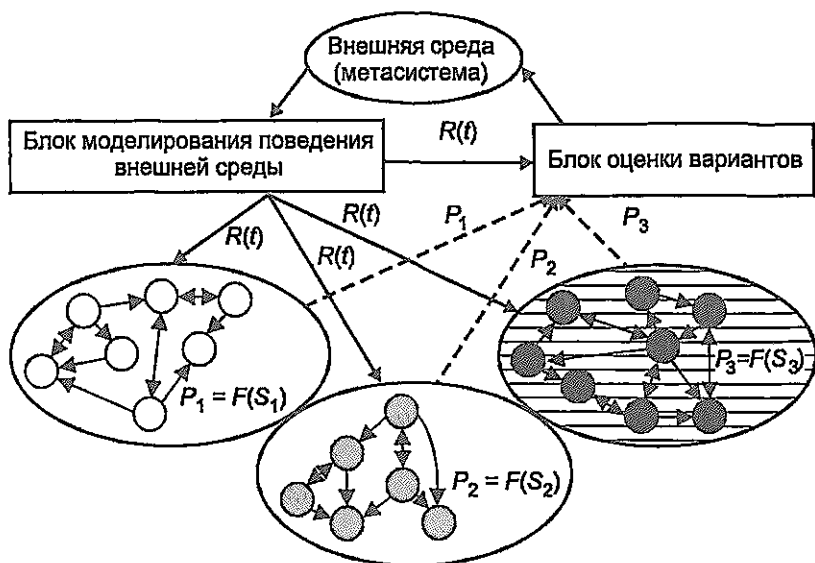


Рис. 8.5. Схема эволюционного проектирования

бинаций элементов на каждом шаге синтеза отбираются только те, которые имеют высокие шансы «выживания» во внешней среде и не имеют внутренних противоречий.

Для реализации эволюционной синтеза можно использовать идеи генетических алгоритмов, широко применяемые для решения задач оптимизации (см. главу 6). Задачи синтеза сложных систем существенно отличаются от оптимизационных, поэтому для их решения необходима модификация известных генетических алгоритмов. Это связано с тем, что в синтезе сложной системы участвуют объекты с различными структурами описаний, в то время как в процессе оптимизации рассматриваются объекты с идентичными описаниями. Другими словами, состав популяции объектов синтеза разнороден, т.е. содержит множество видов, а в процессе синтеза возможно межвидовое скрещивание. Кроме того, описания объектов синтеза представляют собой наборы структурированных данных различных типов, а не двоичные цепочки генов, как в алгоритмах оптимизации. Поэтому в процессе синтеза возникает проблема формирования структур описаний потомков, а также проблема выбора

претендентов для скрещивания, так как кроссинговер возможен только между определенными видами, присутствующими в популяции. В известных генетических алгоритмах проблемы отбора родителей и формирования описаний потомков успешно решаются с использованием механизмов случайного выбора. В задачах синтеза применение случайного отбора и комбинирования ограничено.

Рассмотрим основные этапы эволюционного синтеза систем.

Этап 1. Создается популяция исходных объектов синтеза, которая представляет собой множество альтернативных реализаций функциональных подсистем (ФПС), наделенных определенными свойствами и имеющих требования к своему окружению внутри системы. Формирование множества ФПС осуществляется на основе функционально-структурного или морфологического анализа систем рассматриваемого класса. Результат декомпозиции может представлять собой множество ФПС (рис. 8.6, а) или иерархию таких множеств (рис. 8.6, б).

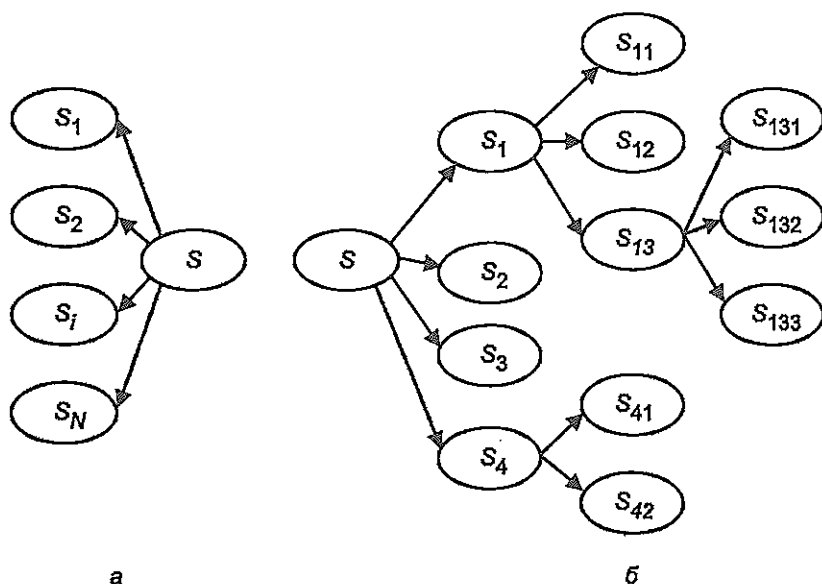


Рис. 8.6. Одноуровневая (а) и иерархическая (б) структуры синтезируемых систем

Этап 2. Формулируется набор обобщенных требований к синтезируемой системе, отражающих ее жизнеспособность или эффективность. Если структура системы задана иерархией, то формулируются также требования к подсистемам, имеющим внутреннее строение (подсистемы S_1, S_4, S_{13}).

Этап 3. Формируется функция ценности вариантов проектируемой системы, позволяющая оценивать степень соответствия сгенерированных объектов заданному набору внешних требований. При иерархическом представлении задается набор таких функций.

Этап 4. С помощью генетического оператора скрещивания из объектов исходной популяции создаются новые объекты — представители следующей популяции.

Этап 5. Вычисляются значения функции ценности полученных вариантов и на их основе производится отбор лучших представителей из новой популяции.

Этапы 4 и 5 могут повторяться неоднократно до выполнения условия завершения процесса синтеза.

Рассмотрим представление знаний, используемых в процессе эволюционного синтеза систем. Допустим, структура синтезируемой системы S задана графом (рис. 8.6, а). Варианты целостного объекта (системы) могут содержать не более чем N элементов $S_i, i = 1, \dots, N$. Набор информации, используемой для синтеза, может включать различные характеристики, представленные совокупностью свойств каждой подсистемы $P_{S_i} = \{Y_1, Y_2, \dots, Y_{K_i}\}$, где K_i — число свойств i -й подсистемы, $i = 1, \dots, N$. В отличие от традиционных бинарных цепочек генов свойства ФПС могут выражаться целыми или вещественными числами, а также символьными строками. Кроме набора свойств каждая подсистема S_i описывается набором требований к ее внутреннему окружению (к другим элементам), $R_{S_i} = \{X_1, X_2, \dots, X_{M_i}\}$; M_i — число требований i -й подсистемы к другим подсистемам. Порядок взаимных требований подсистем можно представить с помощью графа, пример которого приведен на рис. 8.7.

Элементы матрицы B соответствуют числу требований, которые предъявляет элемент, указанный в строке, к элементу, указанному в столбце.

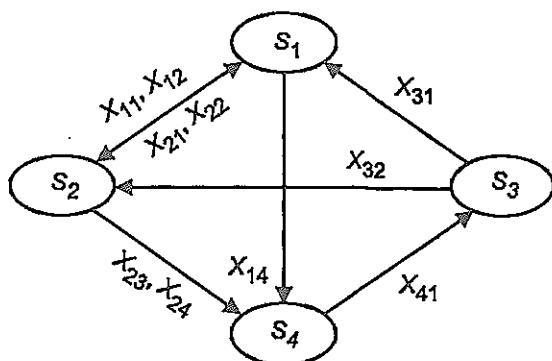


Рис. 8.7. Пример графа взаимных требований подсистем

$$B = \begin{array}{c|cccc} & S_1, & S_2, & S_3, & S_4 \\ \hline S_1 & 0 & 2 & 0 & 1 \\ S_2 & 2 & 0 & 0 & 2 \\ S_3 & 1 & 1 & 0 & 1 \\ S_4 & 0 & 0 & 1 & 0 \end{array}$$

Первый индекс любого требования X_{ij} совпадает с индексом подсистемы, которая выдвигает это требование, второй индекс является порядковым номером требования в списке требований R_{S_i} . Граф требований хранится в матричном представлении (B).

Важным моментом является установление соответствия между свойствами ФПС и требованиями, которые к ней предъявляются со стороны других подсистем. С этой целью заполняются матрицы соответствия для каждой подсистемы S_i . Эти матрицы являются промежуточным представлением объектов синтеза, удобным для дальнейшей обработки. Например, матрица соответствия для ФПС S_1 имеет вид

$$C_1 = \begin{array}{c|ccccc} S_1 & Y_{11} & Y_{12} & \dots & Y_{1K1} \\ S_2 & X_{21} & X_{22} & \dots & - \\ S_3 & - & X_{31} & \dots & - \\ \dots & \dots & \dots & \dots & \dots \\ S_N & - & - & - & - \\ X_1^* & X_{11}^* & X_{12}^* & \dots & X_{1K1}^* \end{array}.$$

В первой строке такой матрицы записываются индексы свойств рассматриваемой подсистемы, в остальных $(N-1)$ строках — индексы требований, которые другие подсистемы выдвигают к соответствующим свойствам S_i . В матрицах соответствия могут присутствовать столбцы, в которых не записано ни одного требования. Это значит, что данное свойство не влияет на совместимость элементов внутри системы. Оно, например, может являться характеристикой качества подсистемы. В процессе синтеза для каждой матрицы C_i создается копия, в которую записываются конкретные значения требований выбранных альтернативных реализаций A_{ij} . Последние строки этих матриц служат для записи значений обобщенных требований к j -му свойству i -й подсистемы (X_{ij}^*) со стороны всех остальных элементов, которые вычисляются после подстановки значений. Описания альтернативных реализаций ФПС удобно представить с помощью таблиц следующего вида:

$$P_i = \begin{array}{c|cccc} S_i & Y_{i1} & Y_{i2} & \dots & Y_{iK_i} \\ \hline A_{i1} & y_{11} & y_{21} & \dots & y_{K11} \\ A_{i2} & y_{12} & y_{22} & \dots & y_{K12} \\ \dots & \dots & \dots & \dots & \dots \\ A_{iL_i} & y_{1L_i} & y_{2L_i} & \dots & y_{KL_i} \end{array}$$

$$R_i = \begin{array}{c|cccc} S_i & X_{i1} & X_{i2} & \dots & X_{iM_i} \\ \hline A_{i1} & x_{11} & x_{21} & \dots & x_{M11} \\ A_{i2} & x_{12} & x_{22} & \dots & x_{M12} \\ \dots & \dots & \dots & \dots & \dots \\ A_{iL_i} & x_{1L_i} & x_{2L_i} & \dots & x_{ML_i} \end{array}$$

Таким образом, альтернативные реализации каждой ФПС S_i описываются парой таблиц, в которых хранятся конкретные значения свойств (y_{jk}) и требований (x_{jk}). Значениями свойств являются числа или текстовые данные, значениями требований — равенства или неравенства. При таком представлении принимается допущение о том, что наборы свойств и требований для всех альтернативных реализаций i -й подсистемы имеют одинаковые размерности (K_i, M_i). Общее число свойств или требований подсистемы можно получить путем объединения соответствующих характеристик альтернативных реализаций A_{ij} . При проверке на удовлетворение требованиям конкретных альтернатив действуют следующие правила. Если альтернатива обладает свойством, к которому не предъявляется требований, то результат проверки считается удовлетворительным. Если у альтернативы отсутствует свойство, к которому предъявляются требования со стороны дру-

гих подсистем, то результат проверки требований будет отрицательным.

Множество векторов свойств $P_S = \{P_{S1}, P_{S2}, \dots, P_{SN}\}$ и требований подсистем $R_S = \{R_{S1}, R_{S2}, \dots, R_{SN}\}$, набор матриц соответствия $C = \{C_1, C_2, \dots, C_N\}$, а также таблицы свойств $P = \{P_1, P_2, \dots, P_N\}$ и требований реализаций ФПС $R = \{R_1, R_2, \dots, R_N\}$ составляют совокупность исходных данных об объектах эволюционного синтеза.

Этапы 2 и 3 эволюционного синтеза связаны с построением функции ценности синтезируемых вариантов, которая должна отражать их жизнеспособность. Для построения такой функции необходимо сформулировать внешние требования к проектируемой системе $R_G = \{R_{G1}, R_{G2}, \dots, R_{GM}\}$, где M — количество внешних требований. Эти требования обычно сопряжены с выполнением определенного набора функций и заданным уровнем качества системы в целом. Требования к синтезируемой системе могут представлять собой выражения, включающие операции $=, \neq, <, \leq, >, \geq$, $\max$, $\min$ и др. Операндами таких выражений являются свойства целостной системы $P_G = \{P_{G1}, P_{G2}, \dots, P_{GK}\}$, константы $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_H\}$, отражающие специфику требований, а также имена и значения свойств ФПС. В процессе синтеза необходимо сформировать вектор свойств целостной системы P_G , который является функцией ее структуры (т.е. свойств элементов и связей между ними), и установить способы вычисления компонентов этого вектора на основе информации о составе конкретных вариантов. Значения свойств целостной системы можно получить различными способами. При *морфологическом* подходе к синтезу [5] характеристики, представленные количественными оценками, получаются с помощью аддитивной или мультипликативной свертки.

Логический подход к синтезу предусматривает применение формул логики предикатов (правил) для формирования характеристик целостного объекта на основе свойств элементов, что расширяет возможности представления знаний, но требует дополнительной информации.

Эволюционный подход к синтезу допускает применение разных способов получения параметров: это могут быть математические и алгоритмические функции, логические формулы, правила, а также комбинированное представление. Выбор конкретного способа зависит от количества и качества доступной информации. При недостатке знаний для вычисления параметров могут

использоваться аддитивный или мультипликативный обобщенный критерий. Процедура вычисления компонентов вектора свойств P_G для варианта системы V_q может выглядеть, например, следующим образом:

$$P_{G1} = Y_{11};$$

$$P_{G2} = \max_{si \in V_q} (Y_{i2});$$

$$P_{G3} = \sum_{i=1}^N Y_{i3};$$

$$(P_{G4} = \lambda_1) \rightarrow (Y_{21} \leq \lambda_2) \wedge (Y_{31} = \lambda_3);$$

$$(P_{GK} = y_{53}) \rightarrow S_4 = A_{43};$$

$$(P_{GK} = \lambda_9) \rightarrow S_4 \neq A_{43}.$$

... ..

Последние три формулы для вычисления интегральных свойств представляют собой записи продукционных правил в виде фраз Хорна. При такой форме записи в левой части правила (до стрелки) записывается заключение, а в правой — условие. Например, последнее правило следует интерпретировать так: установить значение свойства P_{GK} равным λ_9 , если реализацией ФПС S_4 не является A_{43} . Предпоследняя запись предусматривает присваивание другого значения свойству $P_{GK} = y_{53}$ в том случае, если в качестве S_4 выбрана реализация A_{43} .

После того как сформирована процедура получения значений параметров целостной системы, значения функции ценности вариантов можно вычислить как обобщенную меру сходства вектора внешних требований R_G со свойствами конкретного варианта $P_G(V_q)$. Для построения функции ценности применяются меры сходства, описанные в разд. 8.5. Самый простой способ формирования интегрального значения функции ценности — это вычисление среднего арифметического мер удовлетворения всех требований для q -го варианта системы:

$$F_q = \frac{1}{K} \sum_{i=1}^K S(R_{Gi}, P_{Gi}^{(q)}),$$

где $S(R_{Gi}, P_{Gi}^{(q)})$ — мера сходства i -го внешнего требования с соответствующим свойством q -го варианта;

K — количество требований.

Если требования имеют разную значимость, вводятся весовые коэффициенты $w = \{w_1, w_2, \dots, w_K\}$ и функция ценности приобретает вид

$$F_q = \sum_{i=1}^K w_i S(R_{Gi}, P_{Gi}^{(q)}), \quad \sum_{i=1}^K w_i = 1; \quad w_i \geq 0.$$

Если структура проектируемой системы задана иерархией (см. рис. 8.6, б), необходимо сформировать функции ценности для всех подсистем, имеющих сложную внутреннюю структуру.

Рассмотрим процедуру соединения N элементов с разными описаниями, соответствующую оператору скрещивания в генетических методах поиска оптимальных решений. Заметим, что в данном случае потомок происходит от N родителей. Отбор кандидатов для скрещивания основан на вычислении степени удовлетворения взаимных требований соединяемых ФПС. Для каждого i -го элемента синтезируемой системы вычисляется мера удовлетворения требованиям остальных элементов, претендующих войти в комбинацию, в соответствии с формулами

$$G_{qi} = \frac{1}{K_i} \sum_{j=1}^{K_i} S(X_{ij}^*, Y_{ij})$$

или

$$G_{qi} = \sum_{j=1}^{K_i} w_j S(X_{ij}^*, Y_{ij}).$$

Здесь $S(X_{ij}^*, Y_{ij})$ — мера сходства j -го свойства i -й ФПС с объединенным набором требований к ней со стороны остальных элементов; w_j — нормированные весовые коэффициенты требований к j -му свойству i -й ФПС. Обобщенная оценка степени удовлетворения взаимных требований элементов в q -м варианте системы вычисляется либо как среднеарифметическое, либо как среднегеометрическое в зависимости от выбранного принципа компромисса, т.е.

$$G_q = \frac{1}{N} \sum_{i=1}^N G_{qi}$$

или

$$G_q = \left(\prod_{i=1}^N G_{qi} \right)^{1/N}.$$

Аддитивная функция G_q принимает нулевое значение только в том случае, если все ФПС, вошедшие в q -й вариант, не удовлетворяют требованиям друг друга. Мультипликативная функция принимает нулевое значение, когда в комбинацию входит хотя бы один элемент, для которого $G_{q_i} = 0$, поэтому вторая функция более эффективна для поиска вариантов с высокой степенью согласия между участниками (ФПС). При использовании аддитивной функции проверку на наличие внутренних противоречий следует проводить с использованием некоторого порога Δ , $G_q \geq \Delta$. В случае применения мультипликативной функции проверка может выполняться по условию $G_q > 0$.

Таким образом, для любого генерируемого варианта системы $V_q = \{A_{1j}, A_{2j}, A_{3k}, \dots, A_{Nz}\}$ вычисляется значение функции G_q и на его основе принимается решение о возможности существования данной комбинации ФПС. Если решение отрицательное, то не проводится вычисление свойств варианта V_q , т.е. фактически он не порождается и не включается в новую популяцию. В противном случае формируется описание полученного варианта с использованием соответствующей процедуры вывода интегральных свойств, и он участвует в последующем отборе, который осуществляется на основе значений функции F_q , отражающей жизнеспособность целостной системы во внешней среде. Таким образом, выполняется скрещивание N разнородных объектов с учетом их способности к соединению. Следует отметить, что в данном случае не происходит никакого обмена фрагментами хромосом, как в генетических алгоритмах, так как описания объектов являются структурированными, разнородными, и поэтому взаимный обмен фрагментами таких описаний лишен какого-либо смысла.

В зависимости от объема знаний о проектируемой системе и от ее сложности эволюционный синтез можно проводить в различной последовательности. В случае одноуровневого представления проектируемой системы (см. рис. 8.6, а) алгоритм синтеза включает следующие шаги.

1. Подготавливается и вводится исходная информация, содержащая описания ФПС наборами свойств и требований, граф взаимных требований ФПС и матрицы соответствия требований свойствам. В базу знаний системы эволюционного синтеза помещается процедура (грамматика) вычисления параметров синтезируемой системы через параметры составляющих элементов.

Пользователь должен также сформировать набор внешних требований к синтезируемой системе.

2. Генерируется вариант V_q из N реализаций функциональных подсистем, выбранных случайным образом из соответствующих множеств. При небольшой мощности множеств альтернативных реализаций ФПС генерируется множество всех возможных вариантов целостной системы путем полного перебора.

3. Вычисляются оценки меры удовлетворения требований для каждой подсистемы G_{qi} и обобщенная оценка для варианта G_q .

4. Если значение $G_q \geq \Delta$, то формируется описание варианта $V_q(P_{G1}, P_{G2}, \dots, P_{GM})$ с использованием заданной процедуры вычисления свойств целостного объекта. Если $G_q < \Delta$, то вариант исключается из популяции и происходит возврат на шаг 2.

5. Вычисляется значение функции ценности варианта F_q , которое используется для отбора жизнеспособных вариантов. Отбор может производиться по условию $F_q \geq \Delta$ или путем запоминания заданного количества вариантов с наилучшими значениями F_q .

6. Шаги 2 – 5 повторяются до тех пор, пока не будет выполнено условие завершения синтеза. Синтез заканчивается, когда значения F_q перестают улучшаться либо при достижении заданного количества шагов.

В данном алгоритме формируется единственная популяция, членами которой являются варианты системы, полученные от N родителей. При большом количестве альтернативных реализаций ФПС этот алгоритм может включать процедуру предварительного отбора родителей. Следует иметь в виду, что «родители» имеют различные описания, поэтому для их оценки нужно применять либо разные оценочные функции, либо усредненные оценки по множеству критериев качества.

В приведенном алгоритме не учитывалось наличие подсистем, способных выполнять несколько функций. При наличии таких подсистем варианты проектируемой системы могут содержать число элементов $N_1 < N$. Набор выполняемых функций включается в список свойств объектов синтеза. Функции, реализуемые целостной системой, могут отличаться от объединенного набора функций, которые выполняют подсистемы. Поэтому в набор правил формирования свойств целостной системы (грамматику) необходимо добавить правила вывода функций целостной системы.

Если различные ФПС способны выполнять одинаковые функции, то в процессе синтеза следует установить контроль за дублированием функций. В случае если дублирование функций нежелательно, то генерируются только неизбыточные (рациональные) варианты системы, что актуально, например, для задач реинжиниринга. Функции могут иметь различную степень необходимости, поэтому в списке требований к проектируемой системе отмечаются функции, выполнение которых является обязательным. При наличии многофункциональных ФПС в результате скрещивания порождается не один, а несколько вариантов, отличающихся количеством элементов. При этом объекты синтеза заменяются схемами, в которых свободными переменными являются необязательные функции. Синтезированные объекты можно рассматривать как схемы, в которых свободными переменными являются ФПС, реализующие необязательные функции. Путем исключения таких элементов можно получить L реализаций q -й схемы, которые соответствуют вариантам системы с числом элементов, меньшим N . Для всех таких вариантов вычисляется значение G_q , характеризующее совместимость элементов, и если $G_q \geq \Delta$, то порождаются описания новых объектов. Остальные шаги приведенного выше алгоритма остаются прежними.

Приведенный алгоритм синтеза основан на одновременном скрещивании N элементов и одношаговой процедуре порождения потомков.

Рассмотрим механизм эволюционного синтеза, основанный на последовательном соединении элементов. В этом случае в процессе синтеза участвуют исходная популяция базовых объектов и множество популяций потомков. Алгоритм последовательного синтеза включает следующие основные шаги.

1. Подготовка и ввод исходной информации.

2. Формирование начальной популяции объектов синтеза с использованием или без использования процедуры предварительного отбора.

3. Образование новых объектов, включающих по два элемента. Управление процессом соединения элементов осуществляется с использованием графа взаимных внутренних требований. Два объекта соединяются, если между ними есть связь. В случае если граф требований является несвязным, вершины, не имеющие связей, участвуют в синтезе на последнем этапе. Для задач небольшой размерности осуществляется синтез всех комбинаций.

При большом количестве элементов случайным образом генерируется популяция ограниченного размера.

4. Проверка на совпадение функций, выполняемых элементами в полученной комбинации. Если множество функций, выполняемых одним элементом, является подмножеством функций другого, комбинация заменяется этим элементом, и управление передается на шаг 7. В противном случае формируется список функций, реализуемых новым объектом. Если имеется частичное совпадение функций, выполняемых соединяющимися элементами, то дублирующиеся функции отмечаются в списке.

5. Определение степени удовлетворения требований для каждой подсистемы $G_{q_i}^I$, входящей в полученную комбинацию, и обобщенной оценки G_q^I , вычисляемой путем перемножения. Если значение $G_q^I > 0$, формируется описание комбинации V_q^I путем объединения описаний элементов. В описание также включаются индексы вошедших подсистем. Значение G_q^I включается в описание комбинации. Если $G_q^I = 0$, то комбинация исключается из рассмотрения и осуществляется возврат на шаг 3.

6. Формирование свойств нового объекта с использованием заданных грамматических правил и набора его требований к окружению путем объединения требований «родителей».

7. Формирование новой популяции вариантов, прошедших проверку на шаге 5. При этом возможно применение дополнительного условия отбора на основе обобщенной оценки по критерию качества.

8. Проверка на выполнение требований объектов новой популяции к внутреннему окружению. Если не все требования удовлетворены, то объект участвует в дальнейшем синтезе. Осуществляется переход на шаг 3, где происходит поиск кандидатов для скрещивания, управляемый невыполненными требованиями. В противном случае — переход к следующему шагу.

9. Проверка на полноту функций, которые должна выполнять синтезируемая система. Если описание сгенерированного варианта содержит все заданные функции, описание полученного объекта включается в популяцию готовых вариантов и осуществляется переход на шаг 12. В противном случае — переход на следующий шаг.

10. Возврат на шаг 3, где генерируются новые объекты путем присоединения элементов исходной популяции к членам новой популяции (процесс управляется графом требований).

11. Повторение шагов 3 — 10 до выполнения условия окончания синтеза. Таким условием является завершение обхода графа требований, включая отдельные вершины.

12. Отбор вариантов из популяции целостных систем, проводимый на основе значений функции ценности F_q . Если $F_q \geq \Delta$, то описание варианта V_q записывается в файл результатов. В задачах небольшой размерности имеет смысл сделать полный перебор, иначе осуществляется случайный отбор заданного количества вариантов с максимальными значениями функции F_q .

Если структура синтезируемой системы задана иерархией (см. рис. 8.6, б), то формируется множество популяций, соответствующих вершинам графа, имеющим исходящие дуги. Оценочные функции для подсистем, генерируемых в таких узлах, могут строиться или на основе сходства заданных требований со свойствами, или на основе обобщенной оценки по критериям качества. В любом случае необходимо сформировать вычислительные процедуры для получения параметров подсистем, имеющих сложное внутреннее строение. Вопрос о наследовании свойств в процессе эволюционного синтеза решается на этапе представления знаний. При синтезе экономических систем свойства составляющих элементов могут наследоваться, утрачиваться или вычисляться заданным способом. Применение аналогов операций мутации, репродукции, инверсии и т.д. в процессах эволюционного проектирования возможно в принципе, но на сегодняшний день не представляется нам целесообразным, так как трудно найти естественное объяснение произвольному изменению свойств объектов, из которых синтезируется система, в процессе ее проектирования. Помимо этого, реализация упомянутых операторов проблематична в связи со сложностью представления информации об объектах синтеза, а также может привести к трудно предсказуемым последствиям на этапе вывода свойств целостной системы и при интерпретации результатов. Использование названных приемов может иметь определенный смысл при формировании и исследовании множества объектов синтеза.

Метод эволюционного синтеза можно применить, например, для решения задачи, рассмотренной в разд. 8.5. Это целесообразно, если при проектировании структуры системы (см. рис. 8.1) существует большое количество альтернативных реализаций подсистем X_i . Тогда необходимо сформировать функцию ценности генерируемых вариантов, которая будет отражать эффективность

системы в целом. Заметим, что использование в качестве функций ценности мер сходства позволяет оценивать проектируемый объект по множеству критериев качества без усреднения значений показателей. Такие функции на основе выбранного принципа компромисса вычисляют обобщенную меру близости полученного варианта к заданным внешним требованиям. Главной проблемой в этом случае является формирование правил вычисления свойств целостной системы в процессе ее синтеза из элементов. Интегральными показателями качества проектируемой системы могут служить оценки прибыли, рентабельности, спроса и другие, которые можно получить на основе информации, характеризующей конкретные реализации ФПС, например, мощность производства $P_5(X_{1i})$, спрос на конкретный вид продукции $P_1(X_{2i})$, объем возможных инвестиций $P_1(X_{3i})$ и процентная ставка $P_2(X_{3i})$, стоимость технологий $P_1(X_{4i})$, срок освоения технологий $P_3(X_{4i})$, стоимость материалов $P_4(X_{4i})$ и электроэнергии $P_1(X_{6i})$ (см. табл. 8.1 – 8.7).

Применение процедур эволюционного синтеза существенно облегчает анализ полученных вариантов, так как выходные результаты работы программы содержат только совместимые варианты, наиболее близкие к заданным требованиям. Следует отметить, что, в отличие от проектирования технических объектов, в задачах проектирования экономических систем допустимо и часто желательно построение вариантов, содержащих в своем составе не по одной альтернативной реализации ФПС, а по нескольку (диверсификация продукции, привлечение многих партнеров). Подобная постановка задачи приводит к значительному увеличению количества возможных вариантов и вызывает необходимость изменения процедур вывода свойств целостной системы с тем, чтобы учесть наличие многоальтернативных ФПС. В таких случаях целесообразно применять двухэтапную процедуру синтеза, позволяющую снизить размерность исходной задачи. На первом этапе проводится синтез моноальтернативных вариантов проектируемой системы и отбирается подмножество наилучших. Второй этап заключается в построении вариантов многоальтернативных систем на основе множества структур, отобранных на первом этапе. При этом используется модифицированная процедура вывода свойств целостной системы, в которой учитывается наличие множества элементов, принадлежащих к одному классу ФПС.

8.7. ЛОГИЧЕСКИЙ ПОДХОД К СИНТЕЗУ СЦЕНАРИЕВ РАЗВИТИЯ СЛОЖНЫХ СИСТЕМ

Сценарный подход к решению сложных плохо формализованных задач приобретает все большую популярность. Он активно используется в системах динамического и интеллектуального моделирования (см. разд. 8.3), а также как средство представления и структурирования знаний (см. разд. 4.2.2). Рассмотрим один из возможных подходов к построению сценариев с использованием логики предикатов в качестве средства представления знаний.

Решение задач проектирования развивающихся систем требует построения прогнозов возможных изменений в окружающей их среде. Существуют два принципиальных подхода к прогнозированию: прогнозирование будущего на основе прошлого и прогнозирование будущего с учетом появления новых тенденций и событий, которые могли не иметь места в прошлом. Прогнозы, получаемые первым способом, обычно имеют статистическое или теоретическое обоснование, однако они не способны описать новые ситуации. В отличие от них прогнозы, связанные с генерацией гипотез, не имеют строгого обоснования, но позволяют получить представление о новых вариантах возможного будущего, которые не встречались в прошлом.

Первый подход широко используется для прогнозирования крупномасштабных явлений, при описании которых обычно не выделяются активно и непредсказуемо действующие субъекты. Второй подход чаще всего применяется для описания возможных вариантов поведения систем, содержащих активных участников (игроков), которые, не имея информации о стратегиях противодействующих сторон, вынуждены их генерировать на основе доступных им знаний. Синтез сценариев, описанный в настоящем разделе, является реализацией второго подхода к прогнозированию.

Под *сценарием* будем понимать последовательность взаимосвязанных событий, которая может иметь место при определенных условиях. Между событиями существуют причинно-следственные связи, которые можно представить правилами, записанными на языке логики. Синтез сценария осуществляется с использованием

базы знаний, содержащей описание элементов сценария и связей между ними. Результатом синтеза является множество возможных сценариев, качество и достоверность которых зависят от исходной информации.

На первом этапе представления знаний может оказаться полезным построение когнитивной карты — математической модели, представленной в виде графа и позволяющей описывать субъективное восприятие человеком или группой людей какого-либо сложного объекта, проблемы или функционирования системы. Когнитивная карта предназначена для выявления структуры причинных связей между элементами сложного объекта и оценки последствий внешних воздействий на элементы и связи между ними [35]. Элементы изучаемой системы представлены набором вершин графа, связи — множеством направленных дуг, которым могут быть приписаны знаки, определяющие характер влияния. Графическое представление в виде карты наглядно показывает связи между элементами сценария. Перебор всех возможных путей на когнитивной карте дает количества возможных сценариев. Следует заметить, что для большого количества знаний граф может стать необозримым. В этом случае желательно прибегнуть к структуризации знаний, выделяя различные уровни описания информации.

В качестве базовой модели представления знаний для реализации синтеза сценариев на основе причинно-следственных связей может использоваться логика предикатов первого порядка. В простых случаях для представления знаний можно применять логику высказываний. Синтез сценария сводится к построению формальной порождающей грамматики на определенном множестве терминальных и нетерминальных символов. Синтаксису логики высказываний соответствует контекстно-свободная порождающая грамматика, которая описывается четверкой $G = (V, T, P, S)$, где V — конечное множество нетерминальных символов; T — конечное множество терминальных символов, не пересекающееся с V ; P — конечное множество правил Productions, записанных в виде нотаций Бэкуса и представляющих собой расшифровку нетерминальных символов грамматики логическими выражениями, содержащими символы из T и V ; S — начальный символ грамматики [18].

Термин *грамматика*, взятый из лингвистики, означает набор правил некоторого языка, которые позволяют строить и распознавать «правильные» фразы на этом языке.

Фраза — это конечная последовательность слов, которые являются неделимыми элементами (терминальными символами). В любом языке разрешены не любые сочетания слов, поэтому «правильными» считаются фразы, соответствующие определенным правилам синтаксиса и семантики. Формальная грамматика представляет собой набор логических правил (синтаксиса), которые устанавливают способы сочетания слов и словосочетаний (нетерминальных символов) для образования более сложных выражений. Правила применяются рекурсивно и позволяют порождать бесконечное множество фраз (логических формул), составляющих некоторый язык. Продукции формальных грамматик можно интерпретировать как логическую формализацию правил грамматики естественного языка, а фразы — как подмножества множества логических формул. Кроме формализации естественных языков формальные грамматики применяются для описания и распознавания языков программирования, в частности, в программах-компиляторах. Этот механизм можно также использовать для генерации сложных систем и объектов любой природы при достаточной степени изученности исследуемой предметной области.

Рассмотрим применение формализма контекстно-свободных грамматик для построения языка, с помощью которого можно описывать сценарии развития энергетической отрасли.

СЦЕНАРИЙ → *Список_событий*

Список_событий → *Событие*

→ *Событие, Список_событий*

Событие → *Действие_фактора*

→ *Действие_субъекта*

Действие_фактора → *Фактор*

→ *Условие, Фактор*

Фактор → *Экономический_рост*

→ *Спад_производства*

→ *Глобальное_потепление*

→ *Снижение_инфляции*

→ *Увеличение_доходов_населения*

→ *Ухудшение_состояния_атмосферы*

Действие_субъекта → *Субъект, Фактор, Условие, Действие*

→ *Субъект, Условие, Действие*

→ *Условие, Субъект, Действие*

Субъект → РАО_ЕЭС

→ Промышленность

→ ТЭК

→ Население

→ Государство

→ Армия

Действие → Снизить_цену_электроэнергии

→ Повысить_цену_электроэнергии

→ Сократить_производство_электроэнергии

→ Увеличить_потребление_электроэнергии

→ Снизить_потребление_электроэнергии

→ Дотировать_производство_электроэнергии

→ Увеличить_производство

→ Модернизировать_производство

Условие → (Фактор≠Экономический_рост)∨(Действие_фактора
= Ухудшение_состояния_атмосферы)

→ (Фактор≠Экономический_рост)∨(Действие_фактора
= Повышение_уровня_жизни)

→ (Действие≠Снизить_цену_электроэнергии)∨(Фактор
= Снижение_инфляции)

→

→ (Субъект≠РАО_ЕЭС)∨(Действие=Повысить_цену_эле-
ктроэнергии)

→ (Субъект≠Государство)∨(Действие=Дотировать про-
изводство_электроэнергии)

→ (Субъект≠Армия)∨(Действие=Увеличить потребе-
ние_электроэнергии)

→

→ (Действие≠Увеличить_потребление_электроэнергии)∨
(Субъект≠ТЭК)∨(Действие = Модернизация_производ-
ства)

→

→ (Субъект≠РАО_ЕЭС)∨(Фактор≠Экономический_рост)∨
(Действие = Снизить_цену_электроэнергии)

→ (Субъект≠РАО_ЕЭС)∨(Фактор≠Спад_производства)∨
(Действие=Сократить_производство_электроэнергии)

→

→ (Субъект≠Промышленность)∨(Фактор≠Экономичес-
кий_рост)∨(Действие=Увеличить_производство)

→

→ (Субъект≠Население)∨(Фактор≠Глобальное_потепление)∨(Действие=Снизить_потребление_электроэнергии)

→

В приведенном фрагменте грамматики каждое правило описывает возможные варианты подстановки логических формул, записанных в правой части (после стрелки), в качестве значения высказывания, содержащегося в левой части. При этом возможны альтернативные способы подстановки. Запятая, присутствующая в правых частях грамматических правил, соответствует операции конъюнкции.

СЦЕНАРИЙ (начальный символ грамматики) определяется как рекурсивная последовательность событий. События интерпретируются как действия, которые могут происходить произвольно или при выполнении определенных условий. При этом событие может быть обусловлено действием некоторого фактора (например, природного или макроэкономического) или действием активного субъекта. Варианты возможных событий показаны на рис. 8.8.

Ввод нетерминального символа *Условие* потребовался для того, чтобы ограничить произвольные сочетания действий, не соответствующих семантике. Правила, содержащие *Условие*, отражают связи между субъектами и их действиями, а также причинно-следственные отношения между событиями. При отсутствии подобных ограничений порождающая грамматика генерировала бы все комбинации терминальных и нетерминальных символов, т.е. морфологическое множество.

Рассмотрим примеры построения фраз на приведенной грамматике, последовательно применяя правила подстановки:

Сценарий→Событие→Действие_фактора→Фактор→Экономический_рост;

Событие→Действие_фактора→Фактор→Ухудшение_состояния_атмосферы;

Событие→Действие_субъекта→(Субъект, Фактор, Условие, Действие)→(РАО ЕЭС, Экономический_рост, ЕСЛИ Субъект = РАО_ЕЭС И Фактор=Экономический_рост, ТО Действие=Снизить_цену_электроэнергии, Снизить_цену_электроэнергии);

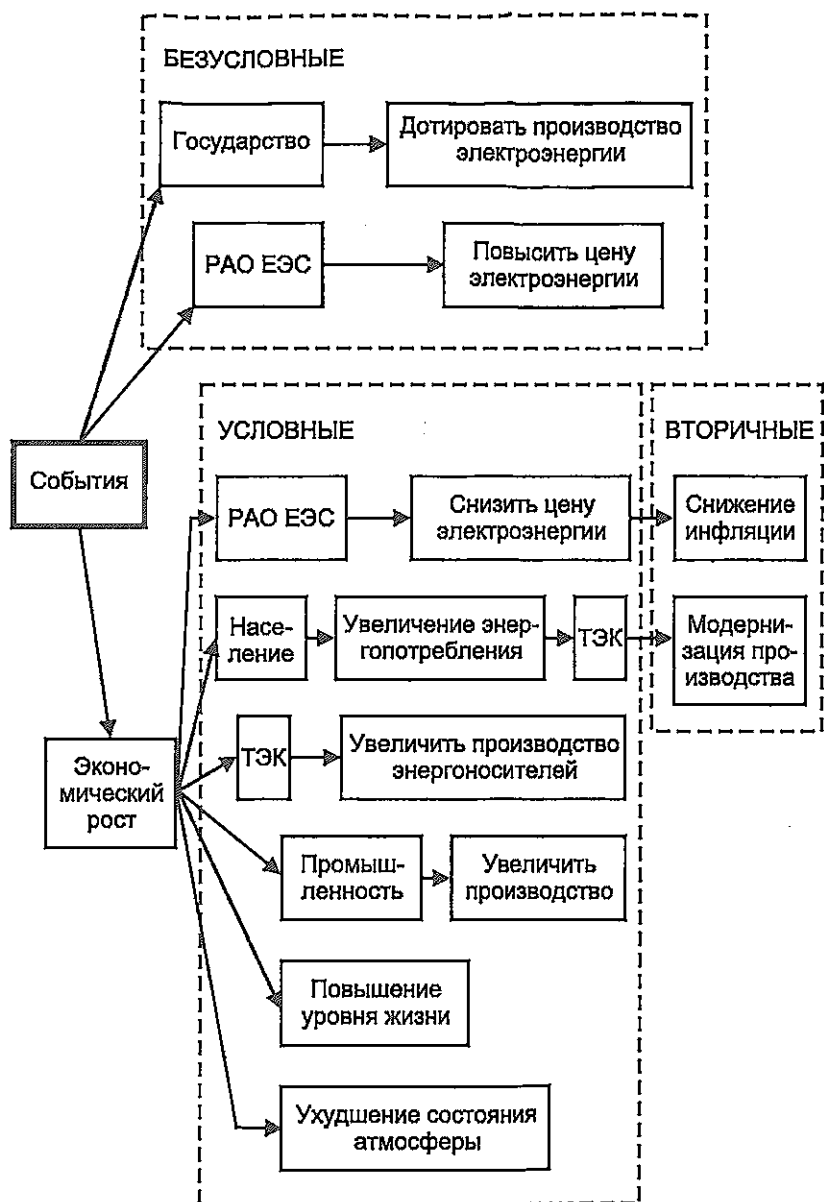


Рис. 8.8. Варианты возможных событий в сценарии развития энергетики

Событие→Действие_субъекта→(Субъект, Условие, Действие)→
(РАО_ЕЭС, ЕСЛИ Субъект = РАО_ЕЭС, ТО Действие = Повы-
сить_цену_электроэнергии, Повысить_цену_электроэнергии);

... ..

В процессе порождения фраз на грамматике возникает проблема выбора последовательности применения правил подстановки, так как при синтезе сценариев допускается наличие циклов событий и больших множеств T и V . В таких случаях необходимым компонентом программного обеспечения является подсистема, управляющая процессом формирования сценариев. При этом становится необходимой классификация событий и символов, например, в соответствии с возможностью использования случайного выбора, со статусом активности субъекта (фактора), отношениями принадлежности и причинности.

Использование многоместных предикатов позволяет лучше отразить специфику знаний предметной области, так как появляется возможность описать внутреннюю структуру используемых категорий, учитывать контекст и сделать грамматику более компактной. Грамматики с многоместными предикатами называются *грамматиками определенных клауз*, или контекстно-зависимыми грамматиками [18]. Применение таких грамматик для синтеза сценариев позволяет представить элементы знаний в виде описаний, содержащих наборы параметров, причем в процессе подстановки существует возможность согласования значений этих параметров. Например, любой Субъект в вышеописанном сценарии может быть представлен описанием следующего вида: Субъект(<имя субъекта>, <статус>, <местоположение>, <условие выхода из игры>, <последнее совершенное действие>, <ожидаемое событие>). С помощью этих атрибутов значительно упрощается процесс организации причинно-следственных связей между событиями, а также установление связей между субъектами и возможными для них действиями.

Важным вопросом при синтезе сценариев является представление *темпоральной* информации, т.е. описание времени и атрибутов, изменяющихся с течением времени. Многие события имеют определенную длительность, а их последствия могут наступать с запаздыванием. В отличие от классических моделей динамических систем в виде систем дифференциальных уравнений, при моделировании сценариев происходит приближенная ими-

тация динамических процессов, при которой важен качественный характер зависимостей и не выдвигаются жестких требований к точности вычислений. Введение в рассмотрение времени необходимо также для указания параллельно протекающих событий. При этом используется понятие параллельно-временной грамматики, которая описывается шестеркой $G = (V, T, P, S, Temp, F)$, где множество $Temp = \{t \in Temp\}$ состоит из элементов, характеризующих время выполнения действий, а множество $F = \{', \square\}$ отражает параллелизм событий. Апостроф в левой части правила означает начало параллельных процессов, а пустой надстрочный символ ($\square$) соответствует правилам, не входящим в циклы [35].

Грамматические правила могут быть представлены в следующем виде:

- $A \rightarrow B(t_B)$, где t_B — время выполнения события B в моделируемом процессе;

- $A' \rightarrow B(t_B)$, где апостроф указывает начало выполнения параллельного процесса. Это значит, что существует по крайней мере еще одно правило вида $A' \rightarrow C(t_C)$, которое выполняется одновременно с правилом $A' \rightarrow B(t_B)$;

- $Z(t_Z) \rightarrow B(t_B)$, где Z — последний символ в циклически повторяемой части строки, B — первый символ в цикле.

Время t может быть представлено вещественным либо целым числом. В первом случае оно соответствует длительности события, а во втором — количеству повторений операций, входящих в цикл.

Более широкие возможности для отражения взаимосвязи между событиями, непрерывно протекающими во времени, предоставляют формальные модели времени, или Т-модели [15], в которых система понятий формируется на основе четырех базовых объектов: ТОЧКА, ИНТЕРВАЛ, ЦЕПЬ и КОЛИЧЕСТВО. Между объектами определены множества бинарных отношений. Например, между ТОЧКАМИ введены следующие отношения: 1) тождества ($x_1 = x_2$); 2) раньше ($x_1 < x_2$); 3) не тождественны ($x_1 \neq x_2$); 4) не раньше ($x_1 \geq x_2$).

Отношения между интервалами имеют более сложную природу. Их можно разделить на две группы. В первую входят отношения, которые определяются путем конъюнкции отношений над концевыми точками интервалов. Примерами таких отношений являются: «Событие A_1 начинается раньше события A_2 »; «Со-

бытие A_1 начинается одновременно с событием A_2 ; «Событие A_1 заканчивается раньше события A_2 »; «Событие A_1 следует за событием A_2 » и т.д.

Вторая группа отношений над интервалами не вписывается в модель конъюнктивной семантической сети, поэтому определяется с помощью первичных понятий, синтаксис которых совместим с синтаксисом этой модели. Например, отношение $\text{МЕЖДУ}(t_1, \{t_2, t_3\}) \leftrightarrow (\text{СЛЕД}(t_2, t_1) \wedge \text{СЛЕД}(t_1, t_3)) \vee (\text{СЛЕД}(t_3, t_1) \wedge \text{СЛЕД}(t_1, t_2))$.

Понятия Т-моделей реализуются наборами фреймов. Разработка и реализация семантических моделей с глубокой детализацией отношений времени и пространства — это отдельное направление исследований в искусственном интеллекте. Результаты этих исследований могут быть использованы при разработке сценариев либо на формальном уровне, либо с применением средств программно-инструментальной поддержки, содержащих темпоральные компоненты.

Весьма плодотворным является подход, предложенный в работе [24], где моделируются динамические системы, обладающие как дискретной, так и континуальной компонентами. Динамика исследуемой системы описывается последовательностью состояний, каждое из которых образуется путем замыкания некоторого порождающего множества фактов и аксиом. Множество используемых знаний описывается с помощью специального языка, который содержит описание событий, процессов, фактов, свойств, законов и т.п. Время является дискретным. Начало и конец любого события маркируются специальными символами. События рекурсивно определяются на базе элементарных событий. Элементарным процессом называется элементарное событие, имеющее определенную длительность. Элементарным фактом называется завершенное во времени элементарное событие. Факты рекурсивно строятся из элементарных фактов. Законы — это отношения на множестве событий. База знаний включает базу правил и базу процедур. Чтобы система была управляемой, достаточно, чтобы в БЗ для каждого события присутствовало его отрицание, а также событие, следствием которого оно является. Такой подход актуален при моделировании динамики сложных открытых систем, не имеющих точного аналитического описания и допускающих неполноту используемого набора знаний.

Мы применяли сценарный подход для построения образов будущего в задачах прогнозирования и стратегического планиро-

вания социально-экономических систем. При этом кроме известных программных средств проводились разработка и исследование специального программного обеспечения для генерации сценариев на основе причинно-следственных связей с представлением знаний средствами логики предикатов. В разработанной системе используется дискретное время, длительность событий задается указанием начала и конца. Количественные характеристики объектов, факторов, событий и других элементов сценария вычисляются с помощью процедур, хранящихся в БЗ.

В системе присутствуют следующие основные категории знаний.

Объекты. Это сущности, для характеристики которых могут использоваться символьные и числовые атрибуты. Набор конкретных значений атрибутов определяет состояние объекта. В процессе синтеза сценария значения атрибутов объекта могут изменяться с течением времени, а также вследствие наступления случайных событий и действий, совершенных субъектами. Некоторые состояния объектов могут вызывать наступление определенных событий, в том числе завершение сценария. Зависимости атрибутов объекта от времени моделируются специальными процедурами, задающими темпы изменения характеристик на основе известных законов или гипотез, а также на основе эмпирических зависимостей. Таким образом, любой объект описывается предикатом вида

Объект(t , <наименование>, <тип>, <список атрибутов>).

Темпы. Характеризуют изменения состояний объектов и субъектов во времени, обусловленные действием объективных законов. Для задания темпов используется следующая структура данных:

Темп(<имя атрибута>, <наименование объекта>, <наименование зависимости>, <список коэффициентов>).

Для моделирования темпов предусмотрен следующий набор зависимостей:

$$\langle linear \rangle \rightarrow a_0 + a_1 t$$

$$\langle square \rangle \rightarrow a_0 + a_1 t + a_2 t^2$$

$$\langle exponent \rangle \rightarrow a_0 + a_1 e^{a_2 t}$$

$$\langle log \rangle \rightarrow a_0 + a_1 \log(t + 1)$$

$$\langle sin \rangle \rightarrow a_0 + a_1 t^n \sin(t + a_2)$$

Субъекты, или действующие лица (ДЛ). Предназначены для описания элементов сценария, способных к совершению активных действий. Субъекты, как и объекты, характеризуются набором атрибутов, описывающих состояние субъекта. Некоторые из них могут изменяться во времени в соответствии с заданными темпами. Тип субъекта показывает, может ли он являться объектом. Если — да, то характеристики субъекта могут изменяться в результате действия факторов и других субъектов. Субъекты могут самостоятельно изменять свои характеристики, например *<местоположение>*.

Применение этого атрибута является попыткой ввести пространственную компоненту в представление знаний. В свою очередь, некоторые атрибуты могут оказывать влияние на значение характеристик объектов и ограничивать набор возможных для субъектов действий. Способность к действиям вызывает необходимость использования ряда специальных атрибутов для характеристики субъектов. Каждый субъект имеет определенный *статус*, характеризующий его способность к действиям. Значение этого атрибута выбирается из набора {активен; ожидает события; вышел}. Активное ДЛ может в течение данного кванта времени выполнять доступные ему действия, выбирая их в соответствии с установленными приоритетами или случайным образом. Действующее лицо может оказаться временно неспособным действовать, если ожидает наступления некоторого события, или выйти из игры навсегда. Для реализации этих случаев в список атрибутов ДЛ добавляются *<ожидаемое событие>* и *<условие выхода>*, *<сообщение о выходе ДЛ>*. Ожидаемое событие может быть связано с достижением определенного значения некоторого атрибута, в том числе значения времени.

События. Это основная информационная единица сценария, предназначенная для активизации действующих лиц (субъектов), запуска, запрещения и разрешения появления других событий, изменения значений характеристик объектов, а также для задания этих характеристик. Событие может иметь определенную длительность во времени. Событие может вызываться одним из следующих способов:

- другим событием (действием факторов или субъектов);
- достижением заданного состояния объекта или субъекта;
- при выполнении некоторого условия;
- случайным образом.

Возможны также произвольные сочетания перечисленных способов. В таких случаях описываемое событие наступает при истинности всех условий. Наступление события может быть запрещено другим событием. В этом случае даже при истинности условий наступления данного события оно не будет активизировано до тех пор, пока не завершится запрещающее событие или не наступит другое разрешающее событие. Для определения порядка наступления событий служит атрибут «приоритет события» — целое число от 0 до 20. Первыми выполняются события с более высоким приоритетом (числом).

Кроме того, события характеризуются следующим набором системных атрибутов:

- кратность появления — двоичный флаг, разрешающий событию участвовать в сценарии после своего первого появления;

- время запаздывания — целое число, задающее в условных единицах интервал времени, по истечении которого с момента активизации наступят последствия события (вызываемые события, изменение характеристик объектов, запрещаемые/разрешаемые события и т.п.);

- группа взаимоисключения — целочисленный атрибут, исключающий одновременное наступление двух или более событий, активизированных в один и тот же момент времени. При этом выбирается одно событие, имеющее наивысший приоритет.

В системе синтеза сценариев элементарное событие определяется либо как действие субъекта, в результате которого могут измениться характеристики объектов (субъектов), либо как случайное (закономерное) изменение этих характеристик, либо как действие внешнего фактора, которое может приводить к изменению определенных атрибутов. Элементарное событие протекает в течение единичного кванта времени. Совокупность элементарных событий в течение одного кванта времени t называется состоянием. Последовательность состояний есть сценарий (см. грамматику ниже).

Факторы. Это разновидность неуправляемых событий, которые обусловлены действием внешней среды. Действие факторов может приводить к изменению характеристик объектов и субъектов, а также к вызову активизации и запрещению определенных событий и действий ДЛ. Фактор описывается предикатом следующего вида:

Фактор(*<имя фактора>*, *<название события>*, *<тип события>*, *<длительность события>*, *<группа исключения>*, *<время запаздывания>*, *<список вызываемых событий>*, *<список активизируемых событий>*, *<список отключаемых событий>*, *<имя объекта>*, *<имя субъекта>*, *<имя атрибута>*, *<способ изменения значения атрибута>*, *<значение>*).

Действия субъектов. Это вид событий, инициируемых действующими лицами. В отличие от факторов этот вид событий причинно обусловлен. Каждое действие связано с определенным субъектом. Действия ДЛ влияют на состояние объектов, могут вызывать, запрещать и разрешать события, могут иметь отложенные последствия, но не могут отключаться событиями, поскольку решения о совершении действий принимают действующие лица. На выбор действий может быть наложен запрет (маска) со стороны действующих факторов, местоположения субъектов и со стороны действий других действующих лиц. Это сделано для своевременного реагирования на действия других субъектов. Подмножество разрешенных действий называется активным набором действий. Если активный набор пуст, субъект не может действовать. Если в активном наборе содержится несколько альтернатив, то действие выбирается в соответствии с приоритетом или случайным образом. Действия ДЛ описываются предикатом

Действие(*<название события>*, *<тип события>*, *<имя субъекта>*, *<длительность события>*, *<условие наступления>*, *<группа исключения>*, *<приоритет действия>*, *<степень случайности>*, *<время запаздывания>*, *<список вызываемых событий>*, *<список активизируемых событий>*, *<список отключаемых событий>*, *<имя объекта>*, *<имя субъекта>*, *<имя атрибута>*, *<способ изменения значения атрибута>*, *<значение>*).

Механизм обработки знаний, хранящихся в БЗ, определяется на основе причинно-следственных связей и выявленных зависимостей между элементами сценария, которые формулируются аналитиками, поставляющими знания в систему. Программно реализованный интерпретатор синтезирует различные последовательности событий на основе информации из БЗ. Синтаксис порождающего языка описывается следующей грамматикой:

Сценарий(<Список атрибутов>)→<Последовательность состояний(Список атрибутов)>

<Последовательность состояний(Список атрибутов)>→Состояние(*t*, Атрибуты состояния), {Список атрибутов = Атрибуты состояния}

→Состояние(*t*), <Последовательность состояний>{Формирование Списка атрибутов из наборов атрибутов состояний}

Состояние(*t*, Атрибуты состояния)→<Совокупность событий(Атрибуты совокупности)>

<Совокупность событий(Атрибуты совокупности)>→Событие(*t*, <Атрибуты события>), {Атрибуты совокупности = Атрибуты события}

→Событие(*t*, <Атрибуты события>), <Совокупность событий(Атрибуты совокупности)>, {Формирование атрибутов совокупности из атрибутов событий}

Событие(*t*, <Атрибуты события>)→Объект(*t*, <Атрибуты объекта>), <Набор темпов(список значений)>, {Вычисление атрибутов события}

→Субъект(*t*, <Атрибуты субъекта>), <Набор темпов(список значений)>, {Вычисление атрибутов события}

→Фактор(<Атрибуты фактора>), {Вычисление атрибутов события}

→Фактор(<Атрибуты фактора>), Объект(*t*, <Атрибуты объекта>), {Вычисление атрибутов события}

→Фактор(<Атрибуты фактора>), Субъект(*t*, <Атрибуты субъекта>), {Вычисление атрибутов события}

→Субъект(*t*, <Атрибуты субъекта>), Действие(<Атрибуты действия>), {Вычисление атрибутов события}

→Субъект(*t*, <Атрибуты субъекта>), Действие(<Атрибуты действия>), Объект(*t*, <Атрибуты объекта>), {Вычисление атрибутов события}

<Атрибуты события>→<название события, тип события, имя инициатора, длительность события, условие наступления, приоритет, время запаздывания, наименование объекта, имя субъекта, характеристика, значение, местоположение субъекта, сообщение>

<имя инициатора>→<имя субъекта>

→<имя фактора>

<условие наступления>→<наименование объекта/субъекта>, <характеристика>, <значение>

→<сравнение характеристик>

→<название события>

→<степень случайности>

<сравнение характеристик>→<наименование объекта/субъекта>,
<характеристика>, <значение>, <наименование объекта/субъекта>, <характеристика>, <значение>, <код операции>

Объект(t, <Атрибуты объекта>)→

→Объект(t, <Список значений атрибутов>), {Присваивание значений атрибутам объекта}

<Атрибуты объекта>→<наименование объекта>, <список характеристик>

<список характеристик>→<характеристика>

→<характеристика>, <список характеристик>

<список значений атрибутов>→<значение>

→<список значений атрибутов>

<значение>→<число>

→<символьная строка>

<Набор темпов(список значений)>→Темп(<характеристика>, <наименование объекта>, <наименование зависимости>, <список коэффициентов>), {Вычисление значения характеристики}

→Темп(<характеристика>, <наименование объекта>, <наименование зависимости>, <список коэффициентов>), {Вычисление значения характеристики}, <Набор темпов(список значений)>

Субъект(t, <Атрибуты субъекта>)→Субъект(t, <Список значений атрибутов>), {Присваивание значений атрибутам субъекта}

<Атрибуты субъекта>→<имя субъекта>, <местоположение> <список характеристик>, <статус>, <ожидаемое событие>, <условие выхода из игры>, <сообщение о выходе ДЛ>

<местоположение>→<характеристика>

<статус>→активен

→ожидает

→вышел

- <ожидаемое событие> → <название события>
 → <наименование объекта/субъекта>, <характеристика>, <значение>
 → <сравнение характеристик>
- <условие выхода из игры> → <название события>
 → <наименование объекта/субъекта>, <характеристика>, <значение>
 → <сравнение характеристик>
- Фактор(<Атрибуты фактора>) → Фактор(<Список значений атрибутов>), {Присваивание значений атрибутам фактора}
- <Атрибуты фактора> → <название события, тип события, имя фактора, длительность события, группа исключения, время запаздывания, список вызываемых событий, список активизируемых событий, список отключаемых событий, имя объекта, имя субъекта, характеристика, способ изменения значения атрибута, значение>
- <группа исключения> → <название события>
 → <название события>, <группа исключения>
- <список вызываемых событий> → <название события>
 → <название события>, <список вызываемых событий>
- <список активизируемых событий> → <название события>
 → <название события>, <список активизируемых событий>
- <список отключаемых событий> → <название события>
 → <название события>, <список отключаемых событий>
- <способ изменения значения атрибута> → <имя процедуры>
- Действие(<Атрибуты действия>) → Действие(<Список значений атрибутов>), {Присваивание значений атрибутам действия}
- <Атрибуты действия> → <название события, тип события, имя субъекта, длительность события, условие наступления, группа исключения, приоритет действия, степень случайности, время запаздывания, список вызываемых событий, список активизируемых событий, список отключаемых событий, имя объекта, имя субъекта, имя атрибута, способ изменения значения атрибута, значение>

Приведенная грамматика не содержит расшифровок очевидных символов и детального описания вызываемых процедур. В фигурных скобках прокомментированы процедуры вычисления аргументов предикатов, находящихся в левых частях грамматических правил, на основе значений аргументов предикатов, записанных в правых частях. Состояние рассматриваемой системы в момент времени t включает описание совокупности элементарных событий, сгенерированных для заданного момента. Элементарным событием является изменение состояния объекта или субъекта в соответствии с заданным темпом, действием фактора или ДЛ, а также действие фактора или субъекта, не приводящее к изменению состояния объекта или субъекта. Если действие фактора или ДЛ происходит мгновенно (время запаздывания равно нулю), то вызываемые или активизируемые события включаются в описание данного состояния. Если имеет место запаздывание, то порождаемые элементарные события включаются в постепенно формируемые описания последующих состояний. События, растянутые во времени, влияют на наборы возможных действий на протяжении всего срока их длительности. При переходе к каждому следующему кванту времени проверяются условия завершения протекающих событий и обновляются списки активизируемых, ожидаемых и запрещаемых событий. На рис.8.9 приведена форма для ввода и редактирования информации о событиях в системе синтеза сценариев.

Сценарный подход к прогнозированию требует больших затрат труда и времени для сбора и представления знаний, а также для экспертной оценки элементов информации. Целесообразность затрат зависит от того, как будут использоваться полученные результаты. Наиболее известны приложения подобного подхода в политике. Генерация сценариев актуальна для мультигентных систем. Весьма полезным, на наш взгляд, может оказаться применение сценариев в задачах стратегического планирования для синтеза вариантов желаемого будущего и политик его достижения. Синтез сценариев в автоматическом режиме с использованием БЗ большого объема порождает огромное количество вариантов, многие из которых имеют незначительные отличия и не представляют интереса для последующего анализа. Поэтому актуальной задачей является выделение подмножества ценных сценариев. Ее решение возможно только в том случае, если сформулированы принципы и критерии оценки порождаемых сценари-

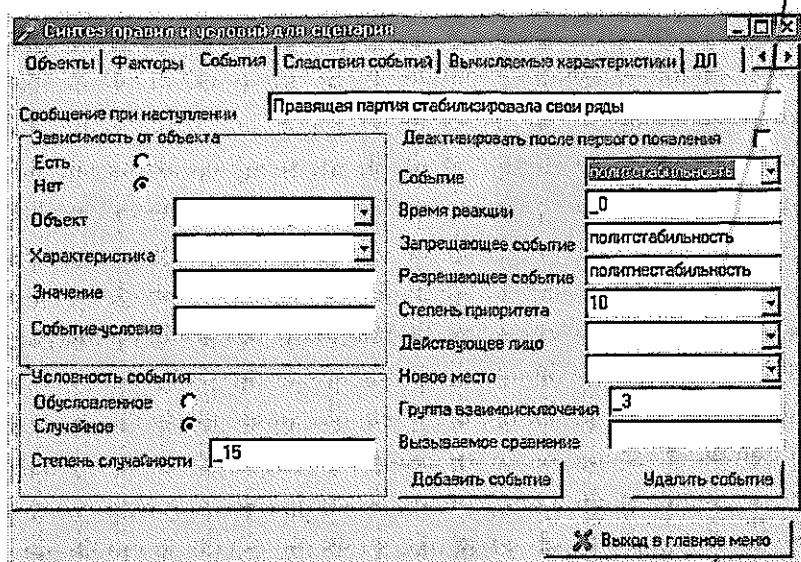


Рис. 8.9. Диалоговое окно просмотра изменяющихся характеристик объектов в системе синтеза сценариев

ев. Рассмотрим некоторые подходы, которые здесь можно использовать.

Сценарий можно оценивать по достигнутому исходу или с учетом всего пройденного пути. В общем случае компьютерная система должна предоставлять разнообразные возможности для оценки полученных сценариев.

Для оценки сценариев в БЗ необходимо добавить следующую информацию:

- о целях (конечных и промежуточных), если таковые имеются. В качестве целей могут выступать желаемые события или состояния объектов (субъектов);
- о критериях качества, по которым оцениваются состояния объектов, события и исходы;
- о предпочтениях экспертов, проводящих оценку сценариев по критериям качества;
- о дополнительных условиях, сформулированных на основе представления об эффективном целостном сценарии (например, минимальное число шагов до заданной цели, наибольшее сред-

нее значение заданной характеристики объекта, присутствие в сценарии множества заданных событий и т.п.).

Элементы сценария, являющиеся целями, снабжаются соответствующими метками. Кроме того, формируются новые структуры данных (таблицы) для представления имеющихся и новых целей. Желаемые цели могут иметь разную важность и могут оказаться противоречивыми. Достижимость целей можно оценивать с использованием различных принципов, например с использованием мер близости, описанных выше. Иногда достижение желаемой цели определить достаточно легко. Такие ситуации характерны для компьютерных игр. При наличии множества целей в качестве оценки достижимости можно выбрать средневзвешенное арифметическое оценок достижения отдельных целей или среднегеометрическое.

Альтернативный принцип оценки достижимости целей ориентирован на обязательное достижение всех целей. Когда цели имеют нечеткие формулировки, то имеет смысл перейти от двузначной шкалы (0, 1) к более информативным шкалам. Экспертные оценки значимости целей используются в процессе формирования обобщенного показателя качества сценария. При этом берется не абсолютная, а относительная оценка достижения целей, вычисляемая как отношение обобщенной оценки сценария к максимально возможному значению, соответствующему достижению всех целей.

Наряду с желательными событиями и состояниями объектов/субъектов в сценариях могут присутствовать нежелательные события и состояния, которые оцениваются с помощью отрицательных значений. В сценариях могут встречаться неоднозначные события или состояния — такие, которые могут наступать многократно и/или в разные моменты времени, причем наступление событий (состояний) при одних обстоятельствах является благом, а при других — злом. Такие события или состояния оцениваются с помощью продукционных правил.

В разработанной версии программного обеспечения для синтеза сценариев предусмотрено вычисление аддитивных и мультипликативных оценок целостного сценария на основе экспертных оценок желательных и нежелательных событий и состояний объектов. В процессе синтеза сценариев используются фильтры для отсеивания вариантов, не удовлетворяющих заданным дополнительным условиям (списки желательных и нежелательных собы-

тий, длительность сценария, время достижения главной цели и т. п.). Обработку знаний выполняет блок синтеза, реализованный на основе допущения о том, что синтезируемые сценарии разбиты на равные отрезки (кванты) времени, в течение которых могут происходить (или не происходить) события (действия факторов и ДЛ). Данный блок обеспечивает функционирование спроектированной грамматики, а также осуществляет разделение событий во времени. Функционирование системы иллюстрируется схемой на рис.8.10.

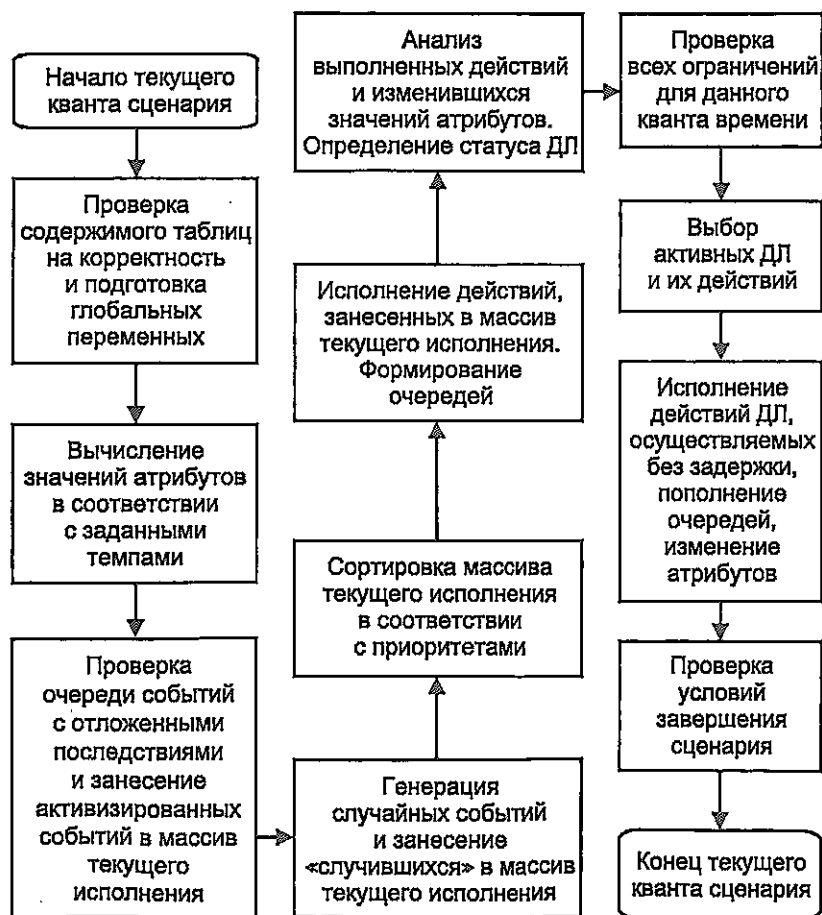


Рис. 8.10. Упрощенная схема функционирования генератора сценариев

Рассмотрим результаты анализа сценариев, сгенерированных для задачи прогнозирования развития политической ситуации в абстрактной стране, где четыре главные партии стремятся к победе на выборах. Партия, победившая на выборах, осуществляет исполнительную власть в течение четырех лет, после чего наступают следующие выборы. В периодах между выборами действующие лица совершают различные действия. Кроме того, могут происходить неуправляемые события.

На рис. 8.11 приведено частотное распределение аддитивных оценок на множестве, включающем 100 сценариев. Оценки формировались на базе двух показателей: состояние экономики и состояние социальной сферы. По оси абсцисс отмечены оценки сценариев (по 100-балльной шкале), а по оси ординат — число сценариев, получивших соответствующую оценку. Нулевое значение свидетельствует о том, что в данном сценарии оценки экономического и социального состояния не попали в заданные диапазоны.

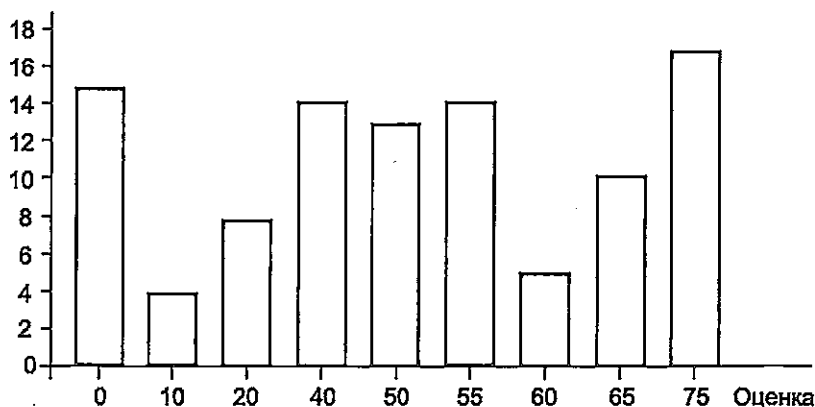


Рис. 8.11. Распределение интегральных оценок синтезированных сценариев

Полученное распределение в этом эксперименте и в ряде других не дает оснований считать какую-либо группу оценок более вероятной, за исключением низкой вероятности максимально возможных оценок, что вполне естественно при аддитивном принципе обобщения.

В процессе синтеза с дополнительными условиями вид распределения свидетельствует о наличии множества сценариев, имеющих оценки, которые встречаются чаще других (рис. 8.12). Это позволяет судить о влиянии сформулированных дополнительных условий на качество сценария в целом. Результаты синтеза множества сценариев такой же мощности (100), но с дополнительными условиями, которые требовали появления событий «война» и «падение цен на нефть», показали, что значения оценок большинства сценариев находятся в диапазоне от 10 до 50, т.е. вероятность синтеза сценариев с высокими оценками экономического и социального состояния снижается при появлении названных событий.

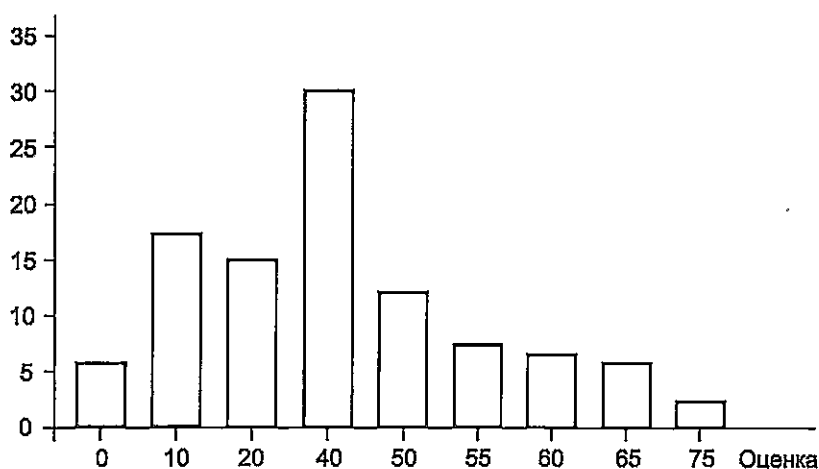


Рис. 8.12. Распределение оценок сценариев, синтезированных при дополнительных условиях

Вид распределения (см. рис. 8.12) свидетельствует о существовании определенной стратегии отбора (или генерации) сценариев, которая позволяет формировать последовательности событий с желательными свойствами.

Перспективным направлением является создание мульти-агентных систем для генерации сценариев развития различных ситуаций, в которых автономные агенты, имитирующие поведение субъектов, принимают решения о выборе действий по согла-

сованию со своими пользователями. Координатор процесса информирует агентов о том, что можно и что нельзя, а также проводит настройку состояний объектов в соответствии с действиями агентов и неуправляемых факторов внешней среды. Такой подход можно воплотить в виде распределенных версий деловых игр или мозгового штурма, ориентированных на выработку, с одной стороны, творческих, а с другой стороны, рациональных вариантов желаемого будущего.

Литература

1. *Абоевский Н. П.* Творчество: системный подход, законы развития, принятие решений. — М.: СИНТЕГ, 1998.
2. *Айламазян А. К., Стась Е. В.* Информатика и теория развития. — М.: Наука, 1989.
3. *Акофф Р.* Искусство решения проблем. — М.: Мир, 1982.
4. *Акофф Р., Эмери Ф.* О целеустремленных системах. — М.: Советское радио, 1974.
5. *Андрейчиков А. В., Андрейчикова О. Н.* Компьютерная поддержка изобретательства (методы, системы, примеры применения). — М.: Машиностроение, 1998.
6. *Андрейчикова О. Н.* Новый подход к проблеме коллективного выбора на базе удовлетворения взаимных требований сторон // Программные продукты и системы. — 2001. — № 3.
7. *Анохин П. К.* Теория функциональной системы // Успехи физиологических наук. — 1970. — № 1.
8. *Берталанфи Л.* Общая теория систем: Критический обзор // Исследования по общей теории систем. — М.: Прогресс, 1969.
9. *Бриллюэн Л.* Научная неопределенность и информация. — М.: Мир, 1966.
10. *Букатова И. Л., Михасев Ю. И., Шаров А. М.* Эвоинформатика: теория и практика эволюционного моделирования. — М.: Наука, 1991.
11. *Гейн К., Сарсон Т.* Системный структурный анализ: средства и методы. — М.: Эйтекс, 1992.
12. *Дружинин В. В., Канторов Д. С.* Системотехника. — М.: Радио и связь, 1985.
13. *Емельянов В. В., Ясиновский С. И.* Введение в интеллектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО. — М.: АНВИК, 1998.

14. *Ивахненко А. Г.* Непрерывность и дискретность. — Киев: Наукова думка, 1990.
15. *Кандрашина Е. Ю.* Средства представления темпоральной информации в базах знаний // Изв. АН СССР. Техническая кибернетика. — 1984. — № 5.
16. *Карташев В. А.* Система систем. Очерки общей теории и методологии. — М.: Прогресс-Академия, 1995.
17. *Клир Дж.* Системология. Автоматизация решения системных задач. — М.: Радио и связь, 1990.
18. Логический подход к искусственному интеллекту. От классической логики к логическому программированию. — М.: Мир, 1990.
19. *Максимей И. В.* Имитационное моделирование на ЭВМ. — М.: Радио и связь, 1988.
20. *Матвеевский С. Ф.* Основы системного проектирования комплексов летательных аппаратов. — М.: Машиностроение, 1987.
21. *Месарович М., Тахакара Я.* Общая теория систем: математические основы. — М.: Мир, 1978.
22. Нечеткие множества и теория возможностей. Последние достижения.: Пер. с англ. / Под ред. Р. Р. Ягера. — М.: Радио и связь, 1986.
23. *Норенков И. П.* Основы автоматизированного проектирования. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2000.
24. *Осинов Г. С.* Динамика в системах, основанных на знаниях // Изв. РАН. Сер. Теория и системы управления. — 1998. — № 5.
25. *Острейковский В. А.* Теория систем. — М.: Высшая школа, 1997.
26. *Половинкин А. И.* Основы инженерного творчества. — М.: Машиностроение, 1988.
27. *Поспелов Д. А.* Ситуационное управление: теория и практика. — М.: Наука, 1986.
28. Построение современных систем автоматизированного проектирования / Под ред. К.Д. Жука. — Киев: Наукова думка, 1983.
29. *Пригожин И., Стенгерс И.* Порядок из хаоса. Новый диалог человека с природой. — М.: Прогресс, 1986.
30. *Прангишвили И. В.* Основные системные законы управления сложными системами различной природы в кризисной ситуации // Приборы и системы управления. — 1997. — № 2.
31. *Прангишвили И. В.* Системный подход и общесистемные закономерности. — М.: СИНТЕГ, 2000.
32. *Санталайнен Т., Воутилайнен Э., Поренне П. и др.* Управление по результатам. — М.: Прогресс, 1993.

33. *Советов Б. Я., Яковлев С. А.* Моделирование систем. — М.: Высшая школа, 1985.
34. Статические и динамические экспертные системы: Учеб. пособие / Э. В. Попов, И. Б. Фоминых, Е. Б. Кисель, М. Д. Шапот. — М.: Финансы и статистика, 1996.
35. *Трахтенгерц Э. А.* Компьютерная поддержка принятия решений. — М.: СИНТЕГ, 1998.
36. *Фогель Л., Оуэнс А., Уолли М.* Искусственный интеллект и эволюционное моделирование. — М.: Мир, 1969.
37. *Форрестер Дж.* Мировая динамика. — М.: Наука, 1978.
38. *Форрестер Дж.* Основы кибернетики предприятия. — М.: Прогресс, 1971.
39. *Хакен Т.* Информация и самоорганизация: Макроскопический подход к сложным системам. — М.: Наука, 1991.
40. *Хубка В.* Теория технических систем. — М.: Мир, 1987.
41. *Цымбал Л. А.* Синергетика информационных процессов. — М.: Наука, 1995.
42. *Эшби У. Р.* Введение в кибернетику. — М.: Изд-во иностр. лит., 1959.
43. *Analytical Power Tools.* — 1998. — Vol. 5. — № 2.

Контрольные вопросы и задания

1. Приведите примеры постановок задач проектирования социально-экономических систем.
2. Сделайте постановку задачи реинжиниринга известных вам предприятия или компании. Перечислите основные этапы и принципы реинжиниринга бизнес-процессов.
3. Сформулируйте основные принципы системного подхода. Дайте определение системы и расскажите о свойствах сложных систем. Приведите примеры сложных систем.
4. Охарактеризуйте методы формального представления и методы исследования сложных систем.
5. Приведите пример задачи коллективного выбора с взаимными требованиями сторон. Представьте проблему в виде графа и дайте характеристику ее элементов. Сформулируйте примеры свойств и взаимных требований.
6. Охарактеризуйте формальные принципы достижения компромисса в задачах коллективного выбора с взаимными требованиями.

ями. Приведите пример задачи и попытайтесь построить для нее обобщенный критерий удовлетворения взаимных требований.

7. Какие существуют способы разрешения конфликтов между участниками коллективного выбора с взаимными требованиями? Какова роль координатора коллективного поведения агентов в мультиагентной системе принятия решений?
8. Сформулируйте основные идеи эволюционного проектирования систем. Чем отличается эволюционный синтез от эволюционных методов оптимизации? Перечислите основные этапы эволюционного синтеза.
9. Проанализируйте различия в интерпретации биологических понятий «родитель», «потомок», «хромосома», «ген», «кроссинговер», «мутация», «популяция» в генетических алгоритмах и в методах эволюционного проектирования.
10. Как формируется функция ценности вариантов в эволюционном проектировании? Расскажите о различных модификациях алгоритма эволюционного синтеза.
11. Дайте характеристику сценарного подхода к прогнозированию. Из каких элементов строятся сценарии? Что такое формальная грамматика? Расскажите об известных вам способах учета темпоральной информации.
12. Для каких задач могут применяться контекстно-свободные грамматики? Приведите примеры грамматических правил, описывающих фрагмент определенной проблемы. Чем отличаются от них контекстно-зависимые грамматики?
13. Расскажите о способах оценки генерируемых сценариев. Придумайте небольшой сценарий и приведите примеры применения сценариев в реальной жизни.
14. Подготовьте набор информации для реализации синтеза сценария. Выделите объекты, субъекты, темпы, факторы и т. д. Сформулируйте правила, описывающие причинно-следственные связи между событиями.
15. Постройте контекстно-свободную грамматику для синтеза сценариев известной вам игры.

Предметный указатель

А

- Абдукция 124
- Агент 306
 - интеллектуальный 308
 - когнитивный 310
 - реактивный 310
- Аналитик 158, 178
- Анкетирование 179, 180
- Антецедент 66
- Архитектура ЭС 35
- Аукцион 320

Б

- База
 - данных 33
 - знаний 33
 - правил 89

В

- Вербализация знаний 158
- Виртуальное предприятие 323

Г

- Ген 261
- Генерация речи 19
- Генетика 265
- Генетический алгоритм 263, 268
 - Девиса 276
 - поколенческий 277
 - простой 270
 - стационарный 277
- Генетическое программирование 292
- Геном 265
- Гештальт 166
- Гештальт-психология 166
- Гипертекстовая система 23

- Гносеологический аспект 164
- Гомеостатическое управление 336
- Грамматика 394
 - контекстно-зависимая 399
 - контекстно-свободная 394

Д

- Демоны 71
- Денотат 170
- Деревья решений 218
- ДСМ-метод 208

Е

- Евклидово расстояние 192
- Естественно-языковой интерфейс 18
- Естественный отбор 260

З

- Задача
 - исследования системы 345
 - модернизации производства 358
 - принятия решений 349
 - проектирования системы 345
 - синтеза систем 341
- Знания
 - декларативные 58
 - имплицитные 189
 - недостоверные 112
 - нечеткие 126
 - поверхностные 188
 - процедурные 58
 - скрытые 159

И

Игра

— карточная со многими участниками 106

— компьютерная позиционная 135

Извлечение знаний

— из данных 214

— Date Mining 214

— Knowledge Discovery 214

И-ИЛИ-граф 95

Илlegalная рекомбинация 266

Инверсия 276

Индуктивные выводы 205

Инженер по знаниям 158

Инженерия знаний 55

Интеллектуальный анализ данных 214

Искусственные нейронные сети 231

Искусственный интеллект 17

Информационные хранилища 214

К

Квантор 63

Классификация с явно заданными признаками 198

Кластерный анализ 216

Когнитивный слой 161

Когнитивный стиль 161

Когнитолог 158

Коллективный выбор решений 357

Композиция нечетких отношений 131

Компонент

— вывода 89

— приобретения знаний 33

— ЭС 33

Компьютерный игрок 107

Консеквент 66

Конструкт 194

Кроссинговер 265

Л

Лингвистическая переменная 140

Логика

— высказываний 59

— предикатов 62

— — первого порядка 62

Логическая модель представления знаний 59

Логическая регрессия 219

Логическая формула 65

— правильно построенная 65

Логические операции 60

Логический вывод

— обратный 89

— прямой 89

М

Масштаб ЭС 36

Машинный перевод 18

Мейоз 263

Метазнания 58

Метасистема 378

Метод

— метафорический 193

— многомерного шкалирования 191

— обратного распространения ошибки 244

— релаксации 117

— соревнования 244

Методы извлечения знаний

— коммуникативные 176

— — активные 176

— — — групповые 183

- — — — дискуссии 185
- — — — мозговой штурм 183
- — — — ролевые игры 185
- — — — индивидуальные 179
- — — — интервьюирование 181
- — — — свободный диалог 182
- — — — экспертные игры 185
- — пассивные 176
- — — — текстологические 185

Методы проектирования 341

Механизм

- вывода 44
- наследования 98

Митоз 263

Многозначность 116

Модели

- координации поведения 318
- — коллективного поведения автоматов 307, 319
- — на основе конкуренции 320
- — на основе BDI-архитектур 319
- — теоретико-игровые 318
- представления знаний 59
- — логическая 59
- — продукционная 66
- — семантическая сеть 74
- — фреймовая 68

Моделирование сложных систем 350

Модель

- закрытого мира 123
- нейрона 231
- Хопфилда 238

Монотонность 123

Мутация 267

Н

Недетерминированность выводов 113

Нейронная сеть 231

Немонотонная логика 122
 Немонотонные выводы 123
 Ненадежность 118
 Неполные знания 122
 Нечеткая логика 138
 Нечеткие знания 126
 Нечеткие множества 127
 Нечеткие отношения 130
 Нечеткие оценки 147
 Нечеткий вывод 133

О

Оболочка ЭС 41

Обучение

- интеллектуальных систем 202
- нейронных сетей 242
- по аналогии 204

Общий код 163

Объектно-ориентированный подход 167

Онтология 164

Операционализация знаний 168

П

Парадигмы программирования 43

Параметрическое обучение 204

Перепроектирование 343

Перцептрон 236

Поле знаний 175

Популяция 260

Правило

- коррекции по ошибке 243
- подстановки 80
- продукционное 66
- резолюций 84
- Хебба 244
- Холланда 279
- *Modus Ponendo Ponens* 78
- *Modus Ponendo Tollens* 78
- *Modus Tollendo Ponens* 78
- *Modus Tollendo Tollens* 78

- Предваренная нормальная форма 65
 Предикат 61
 Приведение к противоречию 83
 Принцип
 — единственного различия 208
 — единственного сходства 208
 — компромисса 365
 Присоединенная процедура 71
 Прототип ЭС 46
 Протофрейм 173
 Процедурный слой 161
 Психологический аспект 159
 Психосемантика 190
- Р**
 Рабочая память 33, 89
 Распознавание образов 19
 Реализация ИНС 246
 Реверси-игра 135
 Реинжиниринг бизнес-процессов 354
 Рекомбинация участков хромосом 265
 Рекуррентные сети 238
 Репродукция 272
 Решетка репертуарная 194
- С**
 Сайт 266
 Свойства нечетких отношений 131
 Семантическая репрезентативность 158
 Семиотика 170
 Силлогизм 77
 Синергетический подход 351
 Синтаксис порождающего языка 405
 Системный подход к проектированию 344
- Системы
 — диагностики 197
 — индуктивные 205
 — контекстной помощи 23
 — с нечеткими выводами 138
 — основанные на знаниях 18
 — основанные на прецедентах 21
 — принятия решений 349
 — самообучающиеся 28
 Ситуационное моделирование 350
 Сколемизация 86
 Сколемовская константа 86
 Слот 68
 Стратегия доказательства 83
 Стратегия поиска 96
 Стратификация знаний 168
 Структурирование знаний 167
 Структурно-функциональный подход 168
 Сценарий 393
- Т**
 Т-модели 400
 Тавтология 79
 Темпоральная информация 400
 Теоретико-игровые модели 318
 Технология
 — DM&KDD 214
 — разработки ЭС 47
 Транслокация 268
- У**
 Указатель
 — наследования 71
 — типа данных 71
 Управляющая компонента 90
- Ф**
 Факт 58
 Формула Байеса 122

Фреге треугольник 170

Фрейм 68

Фундаментальная теорема генетического алгоритма 278

Функция достижения компромисса

— аддитивная 363

— аддитивно-мультипликативная 364

— мультипликативная 364

Х

Хранилище данных 214

Хромосома 261

Ц

Цикл реинжиниринга 343

Ш

Шкалирование 192

Шпация 68

Э

Эволюционное моделирование 260

Эволюционное программирование 298

Эволюционное проектирование 378

Эволюционные аналогии 260

Эволюционные стратегии 300

Эволюционный синтез 378

Эволюция 260

Эвристика 58

Экспертная классификация 198

Экспертная система

— динамическая 26

— доопределяющая 26

— классифицирующая 26

— мультиагентная 27, 374

— статическая 26

— трансформирующая 26

Электронный магазин 321

Учебное издание

**Андрейчиков Александр Валентинович
Андрейчикова Ольга Николаевна**

**ИНТЕЛЛЕКТУАЛЬНЫЕ
ИНФОРМАЦИОННЫЕ
СИСТЕМЫ**

Заведующая редакцией *Л. А. Табакова*

Ведущий редактор *Л. Д. Григорьева*

Редактор *А. М. Маторина*

Младший редактор *Н. А. Федорова*

Художественный редактор *Ю. И. Артюхов*

Технический редактор *Т. С. Маринина*

Корректоры *Т. М. Васильева, Н. П. Сперанская*

Компьютерная верстка *Н. А. Пиминовой*

Оформление художника *А. В. Алексеева*

ИБ № 4465

Сдано в набор 15.04.2003. Подписано в печать 17.05.2004

Формат 60×88/16. Печать офсетная. Гарнитура «Таймс»

Усл. п. л. 25,97. Уч.-изд. л. 24,86

Тираж 4000 экз. Заказ 944. «С» 038

Издательство «Финансы и статистика»

101000, Москва, ул. Покровка, 7

Телефон (095) 925-35-02, факс (095) 925-09-57

E-mail: mail@finstat.ru <http://www.finstat.ru>

ГП Псковской области «Великолукская городская типография»

Комитета по средствам массовой информации

182100, Великие Луки, ул. Полиграфистов, 78/12

Тел./факс: (811-53) 3-62-95

E-mail: VTL@MART.RU