СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
SIBERIAN FEDERAL UNIVERSITY

Т. Г. Пенькова, Ю. В. Вайнштейн

МОДЕЛИ И МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебное пособие

ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Министерство науки и высшего образования Российской Федерации
Сибирский федеральный университет
Институт вычислительного моделирования СО РАН

Т. Г. Пенькова, Ю. В. Вайнштейн

МОДЕЛИ И МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебное пособие

Красноярск
СФУ
2019

УДК 004.89(07)
ББК 32.813я73
П256

Р е ц е н з е н т ы:
Л. Ф. Ноженкова, доктор технических наук, профессор, главный научный сотрудник Института вычислительного моделирования Сибирского отделения Российской академии наук;
Г. М. Рудакова, кандидат физико-математических наук, профессор кафедры информационно-управляющих систем Сибирского государственного университета науки и технологии им. академика М. Ф. Решетнёва

Пенькова, Т. Г.
П256 Модели и методы искусственного интеллекта : учеб. пособие / Т. Г. Пенькова, Ю. В. Вайнштейн. – Красноярск : Сиб. федер. ун-т, 2019. – 116 с.
ISBN 978-5-7638-4043-8

Рассмотрены основные этапы и направления развития искусственного интеллекта, особенности построения систем, основанных на знаниях, принципы функционирования и технология разработки экспертных систем, основные стратегии поиска решений в задачах искусственного интеллекта, а также методы представления и использования знаний: продукционный, фреймовый подходы, семантические сети, формальные логические модели, методы обработки нечётких знаний.

Предназначено студентам направлений 09.03.04 «Программная инженерия», 09.03.02 «Информационные системы и технологии», 27.03.03 «Системный анализ и управление».

Электронный вариант издания см.:
http://catalog.sfu-kras.ru

УДК 004.89(07)
ББК 32.813я73

ISBN 978-5-7638-4043-8

ОГЛАВЛЕНИЕ

СПИСОК СОКРАЩЕНИЙ

БЗ	–	База знаний
ИИ	–	Искусственный интеллект
ИПС РАН	–	Институт программных систем Российской академии наук
ИСА РАН	–	Институт системного анализа Российской академии наук
КНФ	–	Конъюнктивная нормальная форма
КУ	–	Коэффициент уверенности
МВ	–	Машина вывода
МИФИ	–	Национальный исследовательский ядерный университет «МИФИ»
ОР	–	Блок объяснения решений
ПЗ	–	Блок приобретения знаний
РГГУ	–	Российский государственный гуманитарный университет
РП	–	Рабочая память
СПб ГПУ	–	Санкт-Петербургский политехнический университет Петра Великого
СУБД	–	Системы управления базами данных
ЭВМ	–	Электронно-вычислительная машина
ЭС	–	Экспертная система
ЯПЗ	–	Язык представления знаний

ВВЕДЕНИЕ

Искусственный интеллект – одна из современных и стремительно развивающихся областей информатики, занимающаяся разработкой интеллектуальных компьютерных систем. Модели и методы искусственного интеллекта – дисциплина, изучающая основы теории представления знаний в системах искусственного интеллекта. Данная учебная дисциплина обеспечивает повышение профессиональной компетентности в сфере интеллектуальных технологий, способствует расширению профессионального кругозора и умению ориентироваться в тенденциях и направлениях развития современных информационных технологий, позволяет сформировать систему знаний, умений и практических навыков, необходимых для научно-исследовательской, аналитической, проектной и технологической деятельности.

Данное учебное пособие состоит из пяти глав. В первой главе даётся понятие искусственного интеллекта; рассматривается история развития и основные направления искусственного интеллекта; описаны основные гипотезы существования искусственного интеллекта: тест А.Тьюринга, мысленный эксперимент Дж. Сёрля, гипотеза Ньюэлла-Саймона; вводится понятие знаний, определяются их основные типы и свойства, принципы решения задач с применением знаний.

Во второй главе внимание уделяется технологическим аспектам проектирования интеллектуальных систем, основанных на знаниях; рассматривается понятие экспертных систем, их структура и особенности функционирования; даётся классификация и описание области применения экспертных систем; представлена технология их разработки, стадии и этапы создания, включая разработку прототипа.

В третьей главе рассмотрены стратегии поиска решений задач искусственного интеллекта в пространстве состояний; описаны методы полного перебора: стратегия поиска в ширину и глубину, методы эвристического поиска: алгоритм A^*, стратегия первого уровня, метод минимакса, процедура альфа-бета отсечения. Принципы управления поиском рассматриваются на примере решения практических задач.

В четвертой главе даётся описание основных классических моделей представления знаний в системах искусственного интеллекта; рассматриваются продукционная модель, фреймовый подход и семантические сети, используемые для моделирования человеческого образа мышления при

решении многих задач, построения экспертных систем и других приложений искусственного интеллекта; логические методы представления знаний; естественные дедуктивные системы и системы, основанные на методе резолюции; даётся понятие формальной системы, определяются базовые элементы логики предикатов первого порядка как наиболее часто используемой в инженерии знаний формальной системы. Для каждой модели представления знаний приводятся основные теоретические положения, примеры построения баз знаний и организации логического вывода для решения интеллектуальных задач.

Пятая глава посвящена методам представления и использования нечётких знаний; даётся понятие ненадёжных знаний, рассматриваются наиболее известные методы их обработки: метод К. Нейлора, коэффициенты уверенности Шортлиффа; механизм организации нечёткого вывода; вводится понятие размытых знаний, определяются понятия нечёткого множества и нечёткого отношения. Возможности и принципы методов демонстрируются на практических примерах.

1. ОСНОВНЫЕ ПОНЯТИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

1.1. Понятие искусственного интеллекта

Понятие *«искусственный интеллект»* (ИИ) можно определить как *область компьютерной науки, занимающейся автоматизацией разумного (интеллектуального) поведения*. В основе данного определения лежит понятие интеллектуального поведения, или интеллекта. Термин «*интеллект*» происходит от латинского *intellectus*, что означает ум, рассудок, разум, мыслительные способности человека.

На первых этапах развития данной области понятие ИИ определяли как *модель естественного интеллекта*. Однако проблема реальности создания ИИ в данном толковании послужила предметом многих и долгих дискуссий, что в итоге замедлило развитие всего направления в целом.

Постепенно «идеализированное» определение уступило место более прагматичному: ИИ нужен для того, чтобы решать полезные человеку задачи или заменить его в трудных условиях. В 1969 г. Р. Бенерджи сформулировал определение, которое на сегодняшний день остаётся основополагающим для специалистов в области искусственного интеллекта: ИИ – *это направление исследований, целью которого является создание машин, способных решать такие задачи, с которыми до сих пор мог справиться только человек*.

В настоящее время ИИ всё чаще интерпретируется как свойство автоматических систем брать на себя отдельные функции интеллекта человека. При этом для создания даже самой простой модели, реализующей ИИ, применяются знания многих наук. Поэтому в более широком смысле ИИ представляет собой *совокупность моделей, методов и технологий для решения плохо формализуемых (интеллектуальных) задач*. Другими словами, ИИ – *это способность к мышлению, которая воспроизведена с помощью искусственных носителей*.

Обобщая понятия ИИ, можно представить структурную формулу интеллекта следующим образом: *Интеллект = носитель + знания + способы мышления + обучение → решение интеллектуальных задач*. Моделирование отдельных структурных элементов приведённой формулы на разных этапах исторического развития порождало различные парадигмы ИИ.

1.2. История развития и основные направления искусственного интеллекта

1.2.1. Гипотезы существования искусственного интеллекта

История ИИ как нового научного направления начинается в середине XX в. К этому времени уже было сформировано множество предпосылок для его зарождения. Среди философов давно шли споры о природе человека и процессе познания мира: с точки зрения дуализма, мысль не является материальной, и поэтому разум нельзя объяснить с помощью физических понятий; материализм утверждает, что разум можно понять физически как комбинацию единиц и нулей, оставляя возможность существования разумов, созданных искусственно. Нейрофизиологи и психологи разработали ряд теорий относительно работы человеческого мозга и мышления. Математики задавались вопросами оптимальных расчётов и представления знаний о мире в формализованном виде. И наконец зародился фундамент теории вычислений – теория алгоритмов, созданы первые компьютеры. Возможности машин в плане скорости вычислений оказались больше человеческих, поэтому появился естественный вопрос о границах возможностей компьютеров в плане уровня развития человека.

Тест А. Тьюринга

Одним из первых, кто задался целью понять, может ли машина мыслить, был английский ученый Алан Тьюринг. В 1950 г. в философском журнале «Mind» он опубликовал статью «Вычислительные машины и разум» (англ. Computing Machinery and Intelligence), в которой описал способ, с помощью которого можно определить момент, когда машина в плане разумности сравняется с человеком. Процедура получила название «Тест Тьюринга». В основе теста лежит имитационная игра: *человек (эксперт) взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы – ввести в заблуждение, заставив сделать неверный выбор*. Участники игры не видят друг друга. Если эксперт не может отличить ответы человека и машины, считается, что машина прошла тест. Общение ведётся в режиме «только текст» через фиксированные промежутки времени.

Широкую известность получили две программы, одни из первых успешно прошедшие тест Тьюринга: ELIZA, 1966 г. – виртуальный собеседник, имитирующий поведение психотерапевта, и PARRY, 1972 г. – программа, моделирующая поведение параноидального шизофреника. Обе программы реализуют схожий подход: анализируют введенные пользователем комментарии на наличие ключевых слов, по которым применяются

правила, возвращающие комментарий или предложение-результат. В первом случае программе удалось ввести в заблуждение обычных пользователей: они были уверены, что общаются с реально существующим специалистом; во втором – команде психиатров, изучавших стенограммы бесед, не удалось точно определить, кто из «пациентов» – человек, а кто – компьютерная программа.

При этом следует заметить, что эксперименты не являлись в полной мере тестами Тьюринга, так как для вынесения решения тест требует, чтобы вопросы можно было задавать в интерактивном режиме вместо чтения стенограммы беседы, но в то время интерактивные компьютеры были в новинку. Ведь только через 15 лет персональные компьютеры перестанут быть чем-то сверхъестественным, а через 20 лет люди познакомятся с естественно-языковыми сервисами. Сегодня мы также можем встретить различные варианты теста, например, КАПЧА – тест «распознавания разума», который используется для того, чтобы определить, кем является пользователь: человеком или компьютером. Несмотря на то, что прошло более 50 лет, тест Тьюринга благодаря простоте и универсальности не потерял значимости. Однако в настоящее время исследователи не стремятся решить задачу прохождения теста Тьюринга, считая, что важнее изучить основополагающие принципы интеллекта, чем продублировать один из его носителей. В частности, проблему «искусственного полета» удалось успешно решить лишь после того, как исследователи перестали имитировать птиц и приступили к изучению аэродинамики.

Мысленный эксперимент «китайская комната»

В 1980 г. американский ученый Джон Сёрль в своей статье «Разум, мозг и программы» (англ. Minds, Brains, and Programs) выдвинул аргумент против теста Тьюринга, известный как мысленный эксперимент «китайская комната». Суть его описывается следующим образом: *возьмём, какой-нибудь язык, которого вы не понимаете, например, китайский. Текст, написанный по-китайски, я воспринимаю как набор бессмысленных символов. Теперь предположим, что человека, не знающего китайский, поместили в комнату, в которой расставлены корзинки, полные китайских иероглифов. Допустим также, что человеку дали учебник на родном языке, в котором приводятся правила сочетания символов китайского языка, причем их можно применять, зная лишь форму символов.*

Например, правила могут гласить: возьмите такой-то иероглиф из корзинки номер один и поместите его рядом с таким-то иероглифом из корзинки номер два. Теперь представим, что находящиеся за дверью комнаты люди, понимающие китайский язык, передают в комнату наборы символов и что в ответ человек, манипулируя согласно правилам символами, передаёт обратно другие наборы символов. В данном случае книга правил

есть не что иное, как компьютерная программа; люди, написавшие ее, – программисты, а человек в комнате играет роль компьютера. Корзинки, наполненные символами, это база данных; наборы символов, передаваемых в комнату, это вопросы, а наборы, выходящие из комнаты, это ответы.

Предположим далее, что книга правил написана так, что ответы на вопросы не отличаются от ответов человека, свободно владеющего китайским языком. Например, люди, находящиеся снаружи, могут передать непонятные человеку символы, означающие «Какой цвет Вам больше всего нравится?» В ответ, выполнив предписанные правилами манипуляции, человек выдаст символы, также непонятные, но означающие, что «Мой любимый цвет синий». Таким образом, человек выдержит тест Тьюринга на понимание китайского языка, но на самом деле он не понимает ни слова по-китайски. Более того, даже нет способа научиться этому языку в рассматриваемой системе, так как не существует никакого способа, с помощью которого можно было бы узнать смысл хотя бы одного символа. Подобно компьютеру, человек манипулирует символами, но не может придать им какого бы ни было смысла.

В своей теории Дж. Сёрль вводит два понятия: «*сильный*» и «*слабый*» ИИ. «*Сильный*» утверждает, что мышление – это манипулирование формализованными символами, «*слабый*» рассматривает компьютерные модели как полезное средство для изучения разума. «Мысленный эксперимент» показывает, что если человек не может понять китайский язык, выполняя компьютерную программу для понимания китайского, то и никакой другой компьютер не сможет его понять таким образом. Цифровые компьютеры просто манипулируют формальными символами согласно правилам, зафиксированным в программе. Одного умения манипулировать символами недостаточно, чтобы гарантировать знание, восприятие, понимание и мышление. И поскольку компьютеры – это устройства, манипулирующие символами, наличия компьютерной программы недостаточно, чтобы можно было говорить о наличии знания их смыслового значения.

В 1984 г. Дж. Сёрль формулирует свою идею в виде набора аксиом и заключений:

• *Аксиома 1.* Компьютерные программы – это формальные (синтаксические) объекты.

• *Аксиома 2.* Человеческий разум оперирует смысловым содержанием (семантикой).

• *Аксиома 3.* Синтаксис сам по себе не составляет семантику, и его недостаточно для существования семантики.

• *Аксиома 4.* Мозг порождает разум.

• *Заключение 1.* Программы не являются сущностью разума, и их наличия недостаточно для наличия разума.

● *Заключение 2.* Любая система, способная порождать разум, должна обладать свойствами, эквивалентными соответствующим свойствам мозга.

● *Заключение 3.* Любой искусственный мозг должен иметь способность воспроизводить специфические свойства мозга, и наличия этих свойств невозможно добиться только за счёт выполнения формальной программы.

● *Заключение 4.* Способ, посредством которого человеческий мозг на самом деле порождает ментальные явления, не может сводиться лишь к выполнению компьютерной программы.

Мысленный эксперимент вызвал бурную дискуссию среди учёных. Одни считали, что тест Тьюринга выдержал не человек, а система, состоящая из комнаты, книги правил и человека, и именно эта система является разумной и понимает китайский язык. Другие придерживались мнения, что семантики не существует в принципе, а всё, что происходит в мозгу, это всего лишь манипулирование синтаксисом, которое осуществляется и в компьютерах. Было высказано и предположение, что, возможно, человеческий разум есть не что иное, как продуманная либо природой, либо «сверхсуществами», компьютерная программа, которую человечество пока ещё не осознало и не изучило.

Гипотеза о физической символьной системе

В 1976 г. два выдающихся американских ученых Герберт Саймон и Аллен Ньюэлл выдвинули теорию, получившую название гипотеза Ньюэлла-Саймона, или гипотеза о физической символьной системе. Они утверждали, что «*Физическая символьная система имеет необходимые и достаточные средства для произведения базовых интеллектуальных действий в широком смысле*» (под «широким смыслом» понимается «сильный» ИИ). *Без символьных вычислений невозможно выполнять осмысленные действия, а способность выполнять символьные вычисления вполне достаточна для того, чтобы стать способным выполнять осмысленные действия. Таким образом, если животное или человек, или машина действуют осмысленно, то значит, они каким-то образом выполняют символьные вычисления. И наоборот, так как компьютер способен к подобным вычислениям, то на его основе может быть создан искусственный интеллект*». Основанием для данной гипотезы послужило создание и успешное применение системы для автоматического доказательства теорем в исчислении высказываний – «Логик теоретик» (англ. Logic Theorist, 1956 г.) и системы для моделирования рассуждений человека – «Универсальный решатель задач» (англ. General Problem Solver, 1959 г.). Гипотеза Ньюэлла-Саймона также уязвима для критики, но так сложилось, что большая часть исследований ИИ пошла именно по пути создания символьных систем.

1.2.2. Основные этапы исторического развития

История развития ИИ как научного направления характеризуется чередой взлетов и падений: сначала зарождается блестящая идея, вызывающая эйфорию в научном мире, а затем она подвергается серьезной критике. В результате мнения исследователей разделяются, одни считают, что идея ни к чему не приведет и не стоит тратить время впустую, другие продолжают ее развивать и получают результаты. Учитывая наиболее значимые достижения и опираясь на десятилетний период, можно сформировать следующую хронологию этапов развития ИИ, хотя такое разграничение достаточно условное.

Э т а п 1 (1950-е годы)

Этап 1 связан с появлением первых машин последовательного действия с очень небольшими, по сегодняшним меркам, ресурсными возможностями по памяти, быстродействию и классу решаемых задач. Как правило, решались задачи вычислительного характера, для которых были известны схемы решений и которые можно было описать на некотором формальном языке.

Научные исследования в основном были сосредоточены на изучении мозга как носителя интеллекта. Стремление воспроизвести функции человеческого мозга позволило исследователям создать простые аппаратные (позже и программные) модели биологического нейрона и системы его соединений, которые получили название *перспептроны* (от лат. *perceptio* – восприятие). Развитие данного направления началось с работ Ф. Розенблатта, в 1957 г. он предложил модель электронного устройства, названного им персептроном, которое должно было имитировать процессы человеческого мышления. Персептрон передавал сигналы от «глаза», составленного из фотоэлементов, в блоки электромеханических ячеек памяти, которые оценивали относительную величину электрических сигналов. Двумя годами позже была продемонстрирована первая действующая машина *Марк-1*, которая могла распознавать буквы, написанные на карточках.

Чтобы «научить» персептрон способности строить догадки на основе исходных предпосылок, в нём предусматривалась некая элементарная разновидность автономной работы или «самопрограммирования». При распознавании той или иной буквы одни её элементы оказывались более существенными, чем другие. Персептрон мог «научиться» выделять такие характерные особенности буквы полуавтоматически, своего рода методом проб и ошибок, напоминающим процесс обучения. Персептроны вызвали большой интерес и оптимизм. Ф. Розенблатт доказал теорему об обучении персептронов, Б. Уидроу и У. Мак-Каллок дали ряд убедительных демонстраций систем персептронного типа, и исследователи во всем мире стремились изучить возможности этих систем. Однако первоначальная эйфория

сменилась разочарованием, когда оказалось, что персептроны не способны обучиться решению ряда простых задач.

М. Минский, проанализировав проблему, показал, что имеются жесткие ограничения на то, что могут выполнять однослойные персептроны, и, следовательно, на то, чему они могут обучаться. Так как в то время методы обучения многослойных сетей ещё не были известны, многие исследователи перешли в более многообещающие области, и исследования однослойных персептронов пришли в упадок. Позже открытие методов обучения многослойных сетей существенно повлияло на возрождение интереса и исследовательских усилий в области нейроинформатики. Работа Ф. Розенблатта сыграла значимую роль в развитии искусственного интеллекта, впервые показав, что ЭВМ может выполнять функции, которые до того казались присущи только человеку. Теория персептронов является основой для многих типов искусственных нейронных сетей, а сами персептроны являются логической исходной точкой для изучения искусственных нейронных сетей.

Э т а п 2 (1960-е годы)

Помимо развития бионического подхода к ИИ активно развивается прагматическое направление, связанное с моделированием способов мышления. Мозг представлялся в виде черного ящика, на входы которого поступали сигналы внешнего мира, а на выходе получали реакции. На их основе строились модели решений «интеллектуальных задач». Задачи этого класса характеризовались отсутствием априори известных схем решения. Поэтому для решения таких задач строили иерархические представления и программы, имитирующие механизмы мышления человека. Данный класс задач было принято считать интеллектуальным, а методы их решения стали развиваться в рамках научного направления, получившего название эвристического программирования. Сторонники этого направления (А. Ньюэлл, Г. Саймон, К. Шеннон, Дж Шоу и др.) считали, что интеллектуальной можно считать только ту программу, которая обладает механизмом логического вывода и которая на его основе способна самостоятельно формировать алгоритмы решения задач.

Под решением задач понимается такой алгоритм, выполнение которого переводит процесс из начального состояния в некоторое целевое состояние. При этом он должен формироваться автоматически на основе описания пространства состояния задачи. В качестве прототипа модели механизма мышления использовалась теория логического вывода, разработанная в математике.

На данном этапе были достигнуты фундаментальные для теории ИИ результаты: исследован и апробирован механизм логического вывода; созданы

программы автоматического доказательства теорем в логике и арифметике (Логик теоретик); получены новые методы и алгоритмы, ускоряющие процесс логического вывода (Универсальный решатель задач). Разработана теория пространства состояний и методы поиска в нем, наработаны универсальные эвристики разбиения задачи на подзадачи и сокращения пространства поиска решений (А. Ньюэлл, Г. Саймон, К. Шеннон). Был предложен новый язык программирования – Лисп (LISP, LISt Processing –алгоритм обработки списков, Дж. Маккарти), ставший первым языком реализации задач ИИ.

Другим важнейшим результатом данного этапа развития ИИ стало понимание возможности разделения данных и механизмов их обработки. Развитие этого направления стимулировалось специальным заказом в США, Японии и Европе на создание интеллектуальных роботов для исследования космического пространства, для работы в недоступных или опасных для человека средах. Построить искусственный разум для таких целей было невозможно на основе только вычислительного интеллекта, нужен был принципиально новый подход. Исследования проводились на простейших задачах: на кубиках (планирование поведения роботов при сборке простейших конструкций), построение пространства состояний и поиск в нем решений для игр (крестики-нолики, шашки, шахматы и т. д.).

Активно велись работы в области машинного перевода. Исследователи в основном ориентировались на использование синтаксического анализа и информации, получаемой из словарей. Потребовалось достаточно много времени, чтобы осознать, что автоматический перевод не является изолированной задачей и требует наличия такого этапа, как «понимание».

В 1965 г. Дж. Робинсон предложил метод резолюции – эффективный прием доказательства теорем в исчислениях высказываний и предикатов первого порядка, ставший основой для построения автоматических показателей теорем и решения прикладных задач ИИ. Немного позднее появился ещё один язык программирования для задач ИИ – Пролог (PROLOG, PROgramming in LOGic – язык логического программирования, А. Колмрауер).

В СССР исследования по всем приведенным направлениям велись достаточно активно. Одним из выдающихся отечественных результатов стал метод ситуационного управления и моделирования, а также семиотический подход к построению ситуационной модели управления для класса больших систем (Д. А. Поспелов, 1965). В это же время появился отечественный язык для задач ИИ – Рефал (алгоритмический язык рекурсивных функций, В. Ф. Турчин).

Э т а п 3 (1970-е годы)

По мере роста возможностей ЭВМ всё больше стали обращать внимание на рутинные виды деятельности, которые хотелось бы передать

машине. Так, в «интеллект» машины добавились механизмы поиска, сортировки, простейшие операции по обобщению информации, возникли первые СУБД.

В начале 1970-х гг. произошел качественный скачок – исследователи осознали, что всем ранее созданным программам не хватает самого важного – знаний. При этом имелись в виду опытные знания, знания специалистов в различных областях деятельности. Стало очевидно, что важно не просто усовершенствовать эвристики или какие-то числовые коэффициенты, с которыми работает программа, необходимо использовать методы рассуждения и знания, представленные в символьной форме. Такие опытные знания получили название *экспертных знаний*, и соответственно системы, работающие на основе экспертных знаний, стали называться *экспертными системами*. В рамках экспертных систем как важнейшего типа систем ИИ произошло соединение неформально описываемых знаний с механизмами логического вывода и другими методами, разработанными в рамках эвристического программирования (Е. Шортлифф, М. Минский, Дж. Маккарти). Это дало потрясающий эффект в виде экспертных систем для таких направлений деятельности, к которым ранее никакие математические формализации не подходили – медицина, химия, геология, управление и т. д.

Одной из первых систем, использовавших специфические для предметной области знания, была система DENDRAL. Эта система была задумана для определения строения органических молекул из химических формул и спектрографических данных о химических связях в молекулах. Поскольку органические молекулы очень велики, число возможных структур этих молекул также весьма внушительно. DENDRAL решает проблему большого пространства перебора, применяя эвристические знания экспертов-химиков к решению задачи определения структуры. Другая широко известная экспертная система, разработанная в середине 1970-х гг., – MYCIN. В ней использовались знания экспертов медицины для диагностики и лечения спинального менингита и бактериальных инфекций крови.

MYCIN одной из первых обратилась к проблеме принятия решений на основе ненадежной информации. Система выводит ясные и логические пояснения своих рассуждений, используя соответствующую специфике предметной области логику и критерии для оценки своей работы. К числу других классических экспертных систем относится программа PROSPECTOR, определяющая предполагаемые рудные месторождения и их типы, основываясь на геологических данных о местности. Кроме перечисленных можно отметить такие экспертные системы, как INTERNIST – диагностика в сфере медицины внутренних органов, Dipmeter Advisor – интерпретация протоколов бурения нефтяных скважин, SIAP – обнаружение и идентификация различных типов океанских судов, СПРИНТ – контроль за работой

электростанций, REACTOR – помощь диспетчерам атомного реактора, WILLARD – предсказание погоды, PLANT – оценка будущего урожая.

В период развития экспертных систем кроме формальных моделей представления знаний (исчисление высказываний, исчисление предикатов) появились неформальные модели (продукционные, семантические, фреймовые), развивались логические и программные средства вывода, были созданы первые инструментальные программные средства для разработки экспертных систем.

В СССР в этот период также наблюдался подъём в области разработки систем ситуационного управления и экспертных систем. Появились научные школы по ИИ (Г. С. Поспелов, Д. А. Поспелов, Э. В. Попов).

Э т а п 4 (1980-е годы)

Особую роль в развитии ИИ сыграл японский проект ЭВМ пятого поколения, появившийся в 1980 г. Главной стратегической задачей проекта было создание интеллектуальной машины вывода на основе языка Пролог. Работы во всех направлениях ИИ приобрели стратегический характер для каждой страны. США выдали свой проект интеллектуальной ЭВМ на основе языка Лисп (позже была построена Лисп-машина для символьных преобразований), СССР – на основе языка Рефал (позже реализован как символьный процессор в виде приставки к машинам серии ЕС ЭВМ). Японцы, встретившиеся с принципиальными трудностями схемной реализации Пролога, не смогли выполнить проект до конца.

На данном этапе большое внимание стало уделяться разработке инструментального программного обеспечения систем ИИ и экспертных систем, их связи с базами данных. Мощный толчок ИИ получил с появлением персональных компьютеров (резкое увеличение возможностей машин по памяти и быстродействию). Оживились работы по нейронным сетям (новые технологии открыли для них новые возможности). С появлением экспертных систем начался новый этап развития систем ИИ – эра интеллектуальных систем и технологий. Начали развиваться интеллектуальные системы и других типов: расчетно-логические, системы аналитических преобразований. Особенно популярными стали экспертные системы, которые предлагают человеку варианты решений, обосновывают их, способны к обучению (а, следовательно, развитию), общаются с человеком на привычном для него, хотя и ограниченном, естественном языке.

Одной из первых экспертных систем, способных к обучению, была система EURISCO (Д. Ленат, 1982). Программа на основе эвристических правил решала задачи по проектированию электронных интегральных микросхем. Расчёт микросхем требовал проверки множества вариантов,

и программа предложила удачные варианты многослойного размещения элементов, что явилось своего рода открытием.

В СССР к началу 1990-х гг. исследования шли вровень с США и Европой и даже опережали их по некоторым аспектам, особенно теоретическим. Именно в СССР впервые появилась многопроцессорная ЭВМ «ЭЛЬБРУС» с новейшей архитектурой, на основе которой решались сложнейшие задачи с применением систем ИИ.

Э т а п 5 (1990-е годы)

Постепенно усложнение решаемых задач потребовало нового уровня «интеллектуальности» программных систем. Решение было найдено в интеграции нескольких технологий. Стали создаваться сложные аппаратно-программные комплексы интеллектуальных систем, позволяющие создавать гибкие среды, в которых обеспечивалось решение всех необходимых задач. Одно из таких направлений получило название *мультиагентных систем*. Каждый агент имеет свою систему, свою область действий и ответственности, а взаимодействие между ними обеспечивается метаинтеллектом. В рамках такого понимания традиционные методы, алгоритмы и программы стали «кирпичиками», из которых строились решения для возникающих комплексных задач.

Э т а п 6 (2000-е годы)

Этап развития, начиная с 2000-х гг. и по настоящее время, можно охарактеризовать как время глобальной интеллектуализации машин. Элементы интеллектуальных технологий внедряются во все сферы человеческой деятельности. С огромной интенсивностью развиваются интеллектуальные методы, направленные на повышение уровня «понимания» машин до уровня человеческого общения. Системы не просто копируют, но и автономно используют знания человека. Становится возможным собирать, тиражировать, обрабатывать, проектировать знания, сделать их доступными для всех заинтересованных в них людей.

Что касается отечественных достижений, то масштабные перемены в экономической и политической жизни страны во второй половине 1980-х гг. и последующий распад СССР резко затормозили исследования. В результате рынок был быстро завоеван программными системами США и Западной Европы. В настоящее время ситуация заметно меняется в лучшую сторону – активно развиваются ставшие традиционными школы по ИИ: РГГУ (В. К. Финн), МИФИ (Г. В. Рыбина), ИПС РАН, ИСА РАН (Г. С. Осипов), СПб ГПУ (Т. А. Гаврилова), ИПИ РАН (В. Л. Стефанюк) и др.

На фоне всех этапов исторического развития исследователей постоянно привлекала идея создания человека целиком (интегрального робота).

Идея зародилась довольно давно, но особое звучание она получила в 1960-е гг., когда в 1961 г. человек впервые побывал в космосе. Однако реализация этой идеи требовала решения огромного количества задач, многие из которых не решены до сих пор. При этом сегодня достигнуты очевидные успехи, и миру представлены уникальные разработки человекоподобных роботов (андроидов), среди которых можно отметить:

ASIMO (Advanced Step in Innovative Mobility, Япония, 2011) – андроид, который имеет рост 130 см, массу 48 кг и способен передвигаться со скоростью 9 км/ч. Робот способен следить за перемещением большого числа объектов, определить дистанцию до них и направление; умеет распознавать жесты, предметы и поверхности, легко обходя предметы и людей, спускается по лестнице; различает звук, способен воспринимать речь сразу трех человек (откликается на собственное имя, реагирует на тревожные звуки); способен узнавать знакомые лица. Aiko (Канада, 2007) – гиноид с имитацией человеческих чувств: осязание, слух, речь, зрение, созданный для ухода за домом, больными людьми, детьми. Робот умеет разговаривать, читать текст, распознавать предметы и цвета, реагировать на внешние раздражители. Кожа робота состоит из мягкого силикона, способна «чувствовать» боль. TOPIO (TOSY Ping Pong Playing Robot, Вьетнам, 2005) – андроид, разработанный для игры в настольный теннис против человека. Робот обладает человеческой внешностью и перемещается на двух ногах. Робот HRP-4C (Миим, Япония, 2009), предназначен для демонстрации одежды, умеет распознавать слова и интерпретировать окружающие звуки, в 2010 г. продемонстрировал возможности мимики и танца, уникальные вокальные способности.

В результате на сегодняшний день сложилось несколько основных направлений ИИ.

Разработка систем, основанных на знаниях. Основной целью построения таких систем является выявление, исследование и применение знаний высококвалифицированных экспертов для решения сложных трудноформализуемых задач. В данной области исследований осуществляется разработка моделей представления, извлечения и структурирования знаний, изучаются проблемы создания баз знаний.

Разработка естественноязыковых систем и машинный перевод. Системы машинного перевода строятся на основе базы знаний о предметной области текста и сложных моделей, обеспечивающих перевод по схеме: исходный язык оригинала – язык смысла – язык перевода. Здесь используется структурно-логический подход, включающий последовательный анализ и синтез естественноязыковых сообщений. Кроме того в них осуществляется ассоциативный поиск аналогичных фрагментов текста и их переводов в специальных базах данных. Направление охватывает и исследования ме-

тодов, и разработку систем, обеспечивающих реализацию процесса общения человека с компьютером на естественном языке.

Генерация и распознавание речи. Системы речевого общения создаются в целях повышения скорости ввода информации в ЭВМ и возможности управления голосом.

Обработка визуальной информации. Решаются задачи обработки, анализа и синтеза изображений. Задача обработки изображений связана с трансформированием графических образов, результатом которого являются новые изображения.

Обучение и самообучение. Направление включает модели, методы и алгоритмы, ориентированные на автоматическое накопление и формирование знаний с использованием процедур анализа и обобщения данных. К данной области относятся системы добычи данных (Data Mining) и системы поиска закономерностей в данных (Knowledge Discovery).

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

Игры и машинное творчество. Машинное творчество охватывает сочинение компьютерной музыки, стихов, интеллектуальные системы для изобретения новых объектов. 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 искусственного интеллекта. Направление связано с созданием инструментальных средств для разработки интеллектуальных систем: специальных языков программирования, ориентированных на обработку символьной информации (LISP, SMALLTALK, РЕФАЛ), языков логического программирования (PROLOG), языков представления знаний (OPS 5, KRL, FRL), интегрированных программных сред (KE, ARTS, GURU, G2), оболочек экспертных систем, которые позволяют создавать прикладные системы, не прибегая к программированию (BUILD, EMYCIN, EXSYS Professional, ЭКСПЕРТ).

Интеллектуальные роботы. Создание интеллектуальных роботов составляет конечную цель робототехники. В настоящее время в основном используются программируемые манипуляторы с жесткой схемой управления. Основными сдерживающими факторами в разработке автономных роботов являются нерешенные проблемы в области интерпретации знаний, машинного зрения, хранения и обработки трехмерной визуальной информации.

1.3. Знания и их свойства

Основная функция интеллектуальных технологий – это обработка знаний. При этом традиционно возникает вопрос, *что такое знания и чем они отличаются от обычных данных.*

В области философии можно встретить следующее определение: *знание – идеальное выражение в знаковой форме объективных свойств и связей мира, природного и человеческого.* Поскольку знание идеально и первоначально связано с субъектом, то оно нуждается в объективизации, которая осуществляется средствами знаковых систем – естественных и искусственных языков.

В области ИИ приняты следующие определения:

знания – это закономерности предметной области (принципы, связи, законы), выявленные в результате практической деятельности или профессионального опыта и позволяющие решать задачи в этой области;

данные – это отдельные факты, характеризующие объекты, процессы и явления в предметной области, а также их свойства.

По содержанию знания включают:

- информацию об объектах и их свойствах. Объекты, которые рассматриваются в составе знаний, называются *концептуальными* (или *концептами*). В качестве концептов могут выступать предметы, понятия, явления, ситуации, процессы и т. д.;
- отношения между концептами. В настоящее время выделяют более 200 стандартных видов отношений, среди которых «род-вид», «элемент-класс», «часть-целое» и т. д.;
- действия или процедуры над концептами и отношениями;
- сценарии (последовательности действий) или стратегии построения последовательности действий, которые приводят к цели.

В системах ИИ знания являются основным объектом формирования, обработки и исследования. Они проходят путь от внутренних представлений в сознании человека к описанию на *языке представления знаний* (ЯПЗ) и *базам знаний* (БЗ) на различных носителях информации.

В отличие от данных знания обладают следующими свойствами:

Интерпретируемость. Наличие информации, позволяющей идентифицировать каждую единицу знания любой структуры в общем массиве знаний (*внутренняя интерпретируемость*). Наличие информации, позволяющей идентифицировать совокупность знаний, взаимно описывающих некоторую ситуацию в предметной области (*внешняя интерпретируемость*).

Структурированность. Наличие отношений между единицами знания (*внутренняя структурированность*) и наличие отношений между элемен-

тами предметной области, к которой относятся знания. Помимо отношений иерархии («род-вид», «часть-целое») это могут быть отношения временные («раньше», «позже»), пространственные («дальше», «ближе»), сравнения («больше», «меньше») и др. Одна единица знания может входить в состав любой другой, а также быть выделенной из любой другой составляющей ее единицы (отношение «класс-экземпляр»).

Семантическая метрика. Сопоставление с некоторыми смысловыми шкалами (смысловая интерпретация). Семантическая метрика позволяет выполнять оценку близости или различия концептов – силу ассоциативной связи между единицами знания и находить знания, близкие к уже имеющимся.

Конструктивность. Ориентированность на получение решения.

Активность. Способность инициировать действия в зависимости от ситуации (способность к саморазвитию).

Принято выделять следующие типы знаний:

Декларативные и *процедурные*. Декларативные (или описательные) знания – знания об объектах и отношениях. Процедурные знания – знания о действиях над объектами и отношениями.

Концептуальные и *эмпирические.* Концептуальные знания представляют собой сложные отношения между концептами, для их представления используются *метазнания* – знания о знаниях (о структуре, свойствах, организации, приоритетах и способах применения). Эмпирические знания – знания, полученные путем наблюдения или эксперимента.

Чёткие и нечёткие. Чёткие знания предполагают однозначность интерпретации и выбора действий по их обработке. Нечёткие знания представляют собой информацию, обладающую одним из таких свойств, как ненадёжность, неполнота, неоднозначность интерпретации, размытость, недетерминированность.

Решение задачи с применением знаний представляет собой процесс рассуждения, основанный на проверке гипотез путём сопоставления условий задачи с имеющимися знаниями. В области ИИ данная процедура называется логическим выводом.

Логический вывод – это пошаговая процедура, на каждом шаге которой выполняются следующие операции:

1. Сопоставление данных о задаче и знаний о предметной области. Результат: совокупность знаний, которые применимы на текущем этапе решения задачи.

2. Разрешение конфликта между найденными знаниями. Результат: выбор фрагмента знаний, которые по определенным критериям наиболее применимы на данном этапе задачи.

3. Применение выбранных знаний. Результат: получение новых фактов и нового состояния задачи.

Процесс логического вывода продолжается до тех пор, пока не будет найдено решение или не станет ясно, что при данном объеме знаний задачу решить невозможно.

Контрольные вопросы и задания

1. Объясните, что такое «искусственный интеллект», как менялось понятие в процессе развития представлений об искусственном интеллекте.
2. Объясните, в чём заключается тест Тьюринга.
3. Объясните, на что направлен мысленный эксперимент Дж. Сёрля «китайская комната».
4. Объясните, в чём состоит гипотеза Ньюэлла-Саймона о физической символьной системе.
5. Назовите основные исследования в области искусственного интеллекта в 1950–1960-е гг.
6. Укажите особенности развития искусственного интеллекта в 1970–1980-е гг.
7. Укажите тенденции развития искусственного интеллекта в 1990–2000-е гг.
8. Назовите современные направления искусственного интеллекта.
9. Назовите отличия знания от данных.
10. Укажите свойства знания.
11. Назовите основные типы знаний.
12. Назовите отличия декларативных и процедурных знаний.
13. Объясните, что понимается под «решением» задачи на основе знаний.
14. Объясните, что такое логический вывод.
15. Объясните, что такое интеллектуальная система и какими свойствами она обладает.

2. ЭКСПЕРТНЫЕ СИСТЕМЫ

2.1. Структура и принципы функционирования экспертных систем

Главным источником знаний является *эксперт* – человек, владеющий знаниями о структуре и законах функционирования предметной области. Наиболее распространенным видом интеллектуальных систем, основанных на знаниях, являются экспертные системы.

Экспертная система (ЭС) – это интеллектуальная система, которая решает задачи с применением специальной информации – знаний, причем решает так, как это делал бы специалист (эксперт) в данной предметной области.

ЭС представляют собой программные комплексы, аккумулирующие знания специалистов в определенных предметных областях и тиражирующие этот эмпирический опыт для консультаций менее квалифицированных пользователей. Они не призваны заменить эксперта в его непосредственной деятельности, напротив, позволяют расширить возможную сферу применения знаний авторитетных специалистов. При этом ЭС не обладают интуицией и общими знаниями о мире, их ход и метод решения проблемы не может выйти за рамки тех знаний, что в них заложены.

Традиционно знания существуют в двух видах: коллективный опыт и личный опыт. Если большая часть знаний в предметной области представлена в виде коллективного опыта (например, высшая математика), данная предметная область не нуждается в экспертных системах. Если же в предметной области большая часть знаний является личным опытом специалистов высокого уровня (экспертов) и если эти знания по каким-либо причинам слабо структурированы, такая предметная область, скорее всего, нуждается в экспертной системе. Сегодня ЭС широко используются во многих областях и вызывают большой интерес у экономистов, финансистов, преподавателей, инженеров, медиков, психологов, лингвистов.

Все ЭС имеют схожую архитектуру, в основе которой лежит разделение знаний, представленных в системе, и алгоритмов их обработки. Такая организация позволяет корректировать знания и изменять работу системы без перепрограммирования. Так, например, программа, решающая квадратное уравнение, использует знание о том, как следует решать этот

вид уравнений, но это знание «зашито» в программу и его нельзя прочитать или изменить, если исходный код программы недоступен. Такая программа удобна для тех, кто постоянно решает квадратные уравнения, но если пользователю захочется решить другой тип уравнения, ему придётся обратиться к программисту для написания новой программы.

Однако, если предположить, что программа при запуске считывает из некоторого файла тип уравнения и способ его решения и пользователь имеет возможность самостоятельно вводить новые способы решения уравнений, то тогда появляется возможность менять функционал программы без помощи программиста. При этом формат файла должен быть одинаково «понятен» как компьютеру, так и пользователю. Данный пример, несмотря на нетипичность предметной области для применения технологии ЭС (обычно для решения математических уравнений используются специализированные пакеты программ), иллюстрирует особенности архитектуры ЭС – наличие базы знаний. На рис. 2.1 представлена обобщённая структура ЭС. Реальные системы могут иметь более сложную разветвленную структуру, однако приведенные блоки являются обязательными для любой ЭС.

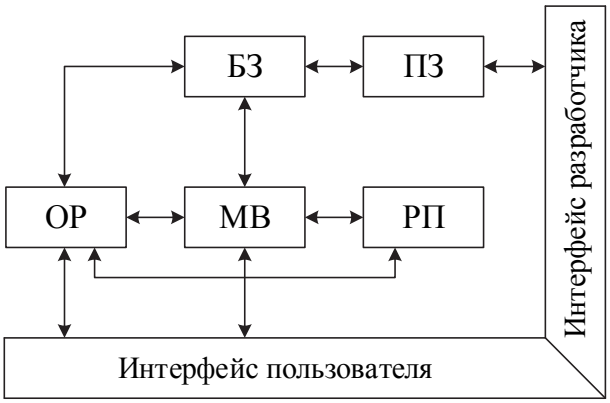


Рис. 2.1. Обобщенная структура
экспертной системы

В структуре ЭС выделяют следующие информационные и программные блоки:

База знаний (БЗ) – ядро системы, блок, куда в систематизированном виде помещается информация о предметной области – знания. Обычно знания в БЗ записываются в форме, приближенной к естественному языку. Форма записи знаний получила название *язык представления знаний*.

Блок приобретения знаний (ПЗ) – блок, основная функция которого поддержка пополнения и коррекции знаний. Данный блок включает в себя систему вложенных меню, шаблонов языка представления знаний, подсказок и других сервисных средств, облегчающих работу с базой знаний.

Рабочая память (РП) – блок, обеспечивающий хранение исходных и промежуточных данных решаемой в текущий момент задачи. Состояние РП характеризует состояние задачи. РП может содержать дополнительную информацию, позволяющую управлять процедурой логического вывода.

Машина вывода (МВ) – блок, моделирующий ход рассуждений эксперта на основании знаний и реализующий пошаговую процедуру логического вывода. Логический вывод реализуется по одной или нескольким стратегиям вывода. Стратегия вывода определяет тип цепочки вывода (прямая, обратная, смешанная); способ и порядок рассмотрения информации при сопоставлении данных и знаний; способ разрешения конфликта между знаниями; способ взаимодействия с пользователем; способы останова и формы выдаваемого системой решения.

Блок объяснения решений (ОР) – блок, обеспечивающий анализ и аргументацию решений системы. ОР позволяет пользователю получать ответы на вопросы, как было получено то или иное решение и почему система приняла такое решение.

Интерфейс – среда, в которой работает пользователь. *Интерфейс пользователя* реализует диалог пользователя с системой как на стадии ввода информации, так и получения результатов. *Интерфейс разработчика* служит для поддержки взаимодействия с блоком ПЗ и функций отладки знаний.

ЭС функционирует в двух основных режимах:

- в режиме приобретения знаний;
- в режиме решения задачи (в режиме консультации).

В *режиме приобретения знаний* эксперт, используя компонент приобретения знаний, в соответствии с определенной моделью представления знаний, наполняет систему знаниями, которые позволят в дальнейшем системе самостоятельно решать задачи из проблемной области. Эксперт описывает проблемную область в виде совокупности данных и правил. Данные определяют факты, описывающие состояние предметной области, ее объекты, их свойства и значения, правила определяют способы манипулирования данными, характерные для рассматриваемой предметной области.

В *режиме консультации* пользователь передает системе исходные данные о задаче, которые после обработки диалоговым компонентом поступают в рабочую память. Исходные данные могут поступать в систему посредством файлов или баз данных, от внешних датчиков или приборов. Машина вывода на основе данных рабочей памяти и знаний в базе знаний формирует решение задачи. При решении задачи ЭС не только исполняет предписанную последовательность операций, но и предварительно формирует ее. Если реакция системы непонятна пользователю, система может объяснить ход своих рассуждений.

Процесс логического вывода сводится к выполнению однотипных операций по перебору записей базы знаний и сравнению их с имеющимися фактами, пока не будет найдено решение или некоторый целевой факт. Однако управление процессом вывода, не зависящее от контекста проблемы, неэффективно. При решении реальных задач человек редко прибегает к перебору данных, он, как правило, использует некоторые эвристические правила, которые ограничивают пространство поиска решения и позволяют быстро и эффективно решать задачи. Эвристические знания имеют эмпирическую природу и формируются на базе опыта и интуиции эксперта.

Существует два основных типа логического вывода: прямой и обратный. Прямой вывод соответствует обычному ходу решения задачи – от исходных фактов к цели. Обратный вывод соответствует обратной задаче – определить, какие факты требуются для подтверждения цели. В реальных системах может применяться комбинация из прямого и обратного вывода. Для управления процессом логического вывода могут использоваться метаправила, которые позволяют упорядочить применение знаний в зависимости от конкретных значений фактов и текущего состояния базы знаний.

2.2. Классификация и область применения экспертных систем

Класс *экспертные системы* объединяет огромное множество программных комплексов, которые можно классифицировать по различным критериям: по типу решаемой задачи (области применения), способу формирования решения, виду используемых знаний, связи со временем, степени интеграции с другими программами и т. д. Рассмотрим основные из существующих классификаций.

Классификация по типу решаемой задачи (области применения):

• *Интерпретация данных*. Под интерпретацией понимается определение смысла данных, результаты которого должны быть согласованными и корректными. Пример ЭС: обнаружение и идентификация различных типов океанских судов по результатам аэрокосмического сканирования – SIAP; определение основных свойств личности по результатам психодиагностического тестирования – АВТАНТЕСТ, МИКРОЛЮШЕР.

• *Диагностика*. Под диагностикой понимается обнаружение неисправности. Неисправность – это отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать неисправность оборудования в технических системах, заболевания живых организмов и всевозможные природные аномалии. Пример ЭС: диагностика и терапия

сужения коронарных сосудов – ANGY; диагностика ошибок в аппаратуре и математическом обеспечении ЭВМ – CRIB.

• *Мониторинг*. Основная задача мониторинга – непрерывная интерпретация данных в реальном масштабе времени и сигнализация о выходе тех или иных параметров за допустимые пределы. Пример ЭС: контроль за работой электростанций – СПРИНТ; помощь диспетчерам атомного реактора – REACTOR; контроль аварийных датчиков на химическом заводе – FALCON.

• *Проектирование*. Проектирование состоит в подготовке спецификаций на создание «объектов» с заранее определенными свойствами. Пример ЭС: проектирование конфигураций ЭВМ – XCON; проектирование больших интегральных схем – CADHELP.

• *Прогнозирование.* Прогнозирование состоит в выводе вероятных последствий событий или явлений на основании анализа имеющихся данных с помощью параметрической динамической модели, в которой значения параметров «подгоняются» под заданную ситуацию. Пример ЭС: предсказание погоды –WILLARD; оценка будущего урожая – PLANT; прогнозы в экономике – ECON.

• *Планирование.* Под планированием понимается нахождение планов действий, относящихся к объектам, способным выполнять некоторые функции. Используются модели поведения реальных объектов с тем, чтобы логически вывести последствия планируемой деятельности. Пример ЭС: планирование поведения робота – STRIPS; планирование промышленных заказов – ISIS; планирование эксперимента – MOLGEN.

• *Обучение.* Обучение состоит в выявлении ошибок при изучении какой-либо дисциплины и подсказке правильного решения с учётом знаний о гипотетическом «ученике» и его характерных ошибках. Пример ЭС: обучение языку программирования ЛИСП – «Учитель ЛИСПа»; обучение языку Паскаль – PROUST.

• *Управление.* Управление состоит в поддержке определённых режимов деятельности и поведения сложных систем в соответствии с заданными спецификациями. Пример ЭС: помощь в управлении газовой котельной – GAS; управление системой календарного планирования – Project Assistant.

• *Поддержка принятия решений.* Поддержка принятия решений заключается в формировании и/или выборе альтернатив при принятии ответственных решений. Пример ЭС: выбор стратегии выхода фирмы из кризисной ситуации – CRYSIS; помощь в выборе страховой компании или инвестора – CHOICE.

Классификация ЭС по связи с реальным временем:

• *Статические.* Системы разрабатываются в предметных областях, в которых база знаний и интерпретируемые данные не меняются во времени. Пример: диагностика неисправностей.

• *Квазидинамические.* Системы интерпретируют ситуацию, которая меняется с некоторым фиксированным интервалом времени. Пример: периодическая регистрация показаний с технологического процесса и анализ динамики полученных показателей.

• *Динамические ЭС.* Системы работают в сопряжении с датчиками объектов в режиме реального времени с непрерывной интерпретацией поступающих в систему данных. Пример: управление производственными комплексами, мониторинг.

Классификация ЭС по степени интеграции с другими программами:

• *Автономные ЭС.* Узкоспециализированные системы, реализующие логический вывод без дополнительных методов обработки данных.

• *Гибридные ЭС.* Программные комплексы, агрегирующие пакеты прикладных программ и средства манипулирования знаниями.

2.3. Технология разработки экспертных систем

Разработка ЭС осуществляется коллективом, в состав которого входят: *эксперт* – специалист предметной области, задачи которой решает ЭС; *инженер по знаниям* – специалист по разработке ЭС; *программист* – специалист по разработке инструментальных средств; *пользователь* – специалист предметной области, для которого предназначена система. Эксперт определяет знания, характеризующие проблемную область, обеспечивает полноту и корректность знаний в ЭС. Инженер по знаниям помогает эксперту выявить и структурировать знания, необходимые для работы ЭС, осуществляет выбор наиболее подходящих для проблемной области инструментальных средств, определяет способ представления знаний, выделяет и формализует типичные для данной проблемной области функции. Программист разрабатывает инструментальные средства, содержащие основные компоненты ЭС, и осуществляет сопряжение системы с той средой, в которой она будет использоваться.

Процесс разработки ЭС можно разделить на шесть основных этапов: выбор проблемы, разработка прототипа ЭС, развитие ЭС, оценка ЭС, стыковка ЭС и поддержка ЭС.

Э т а п 1. Выбор проблемы

Данный этап определяет деятельность, предшествующую решению начать разрабатывать конкретную ЭС:

• определение проблемной области и задачи;

• нахождение эксперта и назначение коллектива разработчиков;

• определение предварительного подхода к решению проблемы;

• анализ расходов и прибылей от разработки;

● подготовка подробного плана разработки.

Правильный выбор проблемы представляет самую критическую часть разработки. Если выбрать неподходящую проблему, то можно быстро увязнуть в «болоте» проектирования задач, которые никто не знает как решать. Неподходящая проблема может также привести к созданию ЭС, которая стоит намного больше, чем экономит. Ещё хуже будет, если разработать систему, которая работает, но неприемлема для пользователей.

Критериями, указывающими на необходимость создания ЭС, являются следующие:

● нехватка высококвалифицированных специалистов в какой-либо узкой предметной области;

● потребность в многочисленном коллективе специалистов, поскольку ни один из них не обладает достаточным знанием;

● сниженная производительность, поскольку задача требует полного анализа сложного набора условий;

● расхождение между решениями самых опытных специалистов и новичков.

Для того чтобы знания, необходимые для решения поставленной задачи, могли быть успешно представлены в базе знаний, они должны быть достаточно узкоспециализированными и не зависеть в значительной степени от общечеловеческих знаний. Задачи, решаемые с помощью этих знаний, не должны быть для эксперта не слишком легкими, не слишком сложными. Также должен существовать способ каким-либо образом оценить результаты решения поставленной задачи, что позволит судить об адекватности создаваемой ЭС.

Способ, каким ЭС будет решать задачи, зависит от эксперта, знания которого будут заложены в базу знаний. Одни системы могут содержать стратегии одного эксперта, а другие представлять подходы коллектива специалистов. Выбор подходящего эксперта является одним из важнейших шагов в создании экспертной системы, который определяет ее авторитетность и дальнейшую эффективность работы.

После того как инженер по знаниям убедился, что данная задача может быть решена с помощью ЭС, имеется подходящий эксперт, предложенные критерии производительности являются разумными, затраты и срок их окупаемости приемлемы, составляется план разработки. План определяет шаги процесса разработки и необходимые затраты, а также ожидаемые результаты.

Э т а п 2. Разработка прототипа ЭС

Система-прототип является усечённой версией ЭС, спроектированной для проверки правильности кодирования фактов, связей и стратегий

рассуждения эксперта. Процесс разработки прототипа ЭС включает несколько этапов (рис. 2.2).

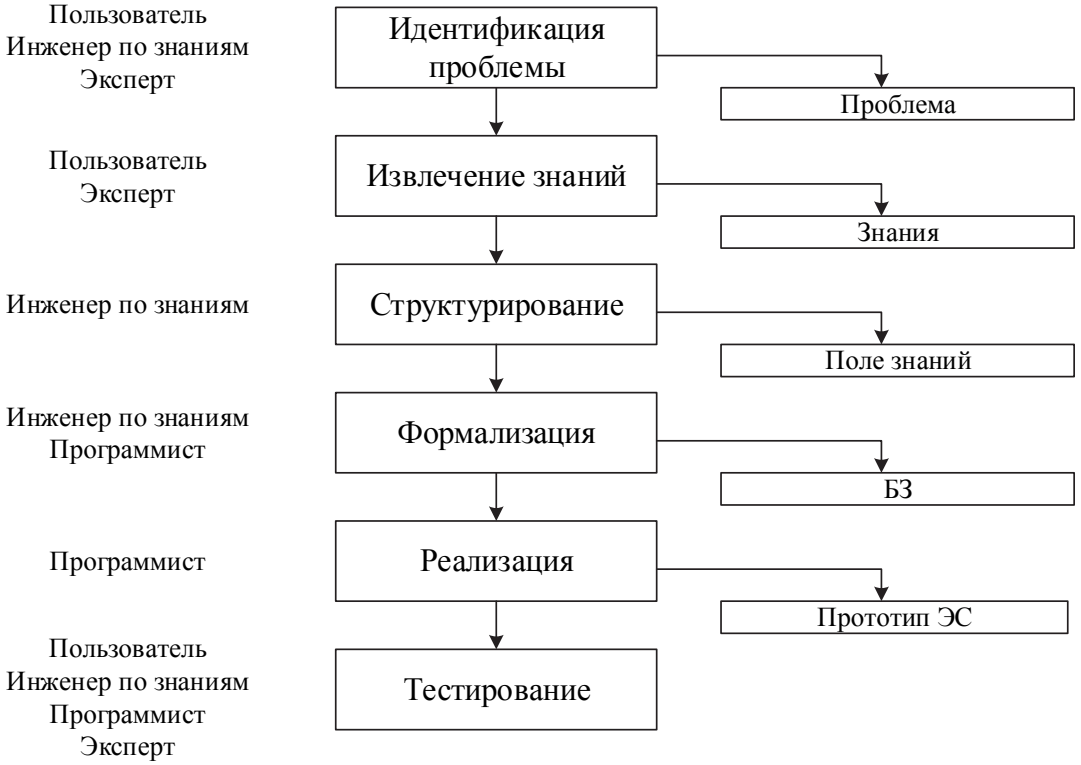


Рис. 2.2. Стадии разработки прототипа ЭС

Идентификация проблемы. Создание неформальной формулировки проблемы. На данной стадии уточняется задача, планируется ход разработки прототипа ЭС, определяются: цель создания ЭС (распространение опыта, автоматизация рутинных действий и др.); класс решаемых задач; исходные данные; природа и объём знаний; необходимые ресурсы (время, люди, ЭВМ и т. д.); источники знаний (эксперты, книги, методики); имеющиеся аналогичные экспертные системы и т. д.

Извлечение знаний. Получение инженером по знаниям наиболее полного представления о предметной области и способах принятия решения. Происходит перенос компетентности от эксперта к инженеру по знаниям с использованием различных методов: анализ текстов, лекции, интервью, диалоги, экспертные игры и др.

Структурирование (или концептуализация) знаний. Разработка неформального описания знаний о предметной области, которое отражает основные концепции предметной области и взаимосвязи между ними. Выявляется структура полученных знаний о предметной области, определяются: терминология, список основных понятий и их атрибутов, отношения между понятиями, структура входной и выходной информации, стратегии принятия решений и их ограничения. Такое описание называется *полем знаний*.

Формализация знаний. Формализованное представление концепций предметной области на основе выбранного языка представления знаний. Разработка базы знаний на языке, который, с одной стороны, соответствует структуре поля знаний, а с другой – позволяет реализовать прототип системы на следующей стадии программной реализации.

Реализация. Разработка компьютерной программы, демонстрирующей жизнеспособность подхода в целом – прототипа ЭС. Создаваемый прототип должен включать базу знаний и все основные блоки.

Тестирование. Выявление ошибок в подходе и реализации прототипа, выработка рекомендаций по доведению системы до промышленного варианта и доработки прототипа. Прототип проверяется на удобство и адекватность интерфейсов ввода-вывода (характер вопросов в диалоге, связность выводимого текста результата и др.), эффективность стратегии управления выводом на знаниях (порядок перебора, использование нечёткого вывода и др.), качество тестовых примеров, корректность базы знаний (полнота и непротиворечивость правил). При неудовлетворительном функционировании прототипа эксперт и инженер по знаниям имеют возможность оценить, что необходимо включить в разработку окончательного варианта системы и какие объекты и свойства необходимо изменить.

Э т а п 3. Развитие ЭС

Прототип системы постепенно совершенствуется и расширяется, происходит плавный переход от демонстрационного прототипа к коммерческой системе. Выделяют следующие этапы перехода: *демонстрационный прототип → действующий прототип → промышленная система → коммерческая система* (см. таблицу).

Таблица

Типы систем при развитии прототипа

Тип системы	Описание
Демонстрационный прототип ЭС	Система решает часть задач, демонстрируя жизнеспособность подхода
Исследовательский прототип ЭС	Система решает большинство задач, но неустойчива в работе
Действующий прототип ЭС	Система надежно решает все задачи на реальных примерах, но для сложной задачи требует много времени и памяти
Промышленная система	Система обеспечивает высокое качество решений при минимизации требуемого времени и памяти; переписывается с использованием более эффективных средств представления знаний
Коммерческая система	Промышленная система, пригодная к продаже, хорошо документирована и снабжена сервисом

Основная работа при развитии системы заключается в добавлении дополнительных эвристик, которые увеличивают глубину системы, обеспечивая большее число правил для трудноуловимых аспектов отдельных случаев. В то же время эксперт и инженер по знаниям могут расширить охват системы, включая правила, управляющие дополнительными подзадачами или дополнительными аспектами экспертной задачи (метазнания). Начинается постепенный перевод контроля над системой от инженера по знаниям эксперту для уточнения, детальной разработки и обслуживания.

Э т а п 4. Оценка ЭС

На этапе 4 выполняется проверка точности работы системы и её эффективность, оценка степени интегрированности со своим окружением (приборами, другими программными продуктами и системами). Оценка ЭС выполняется исходя из следующих критериев:

- критерии пользователей (понятность и удобство интерфейсов);
- критерии экспертов (адекватность и непротиворечивость базы знаний, корректность работы подсистемы объяснения решений);
- критерии разработчиков (эффективность реализации, производительность, анализ чувствительности программы к изменениям).

Э т а п 5. Стыковка ЭС

На этапе 5 выполняется стыковка модулей системы с другими программными средствами, проводится обучение пользователей системы. Для внесения изменений могут привлекаться разработчики. Стыковка также подразумевает совершенствование системных факторов, зависящих от времени, вычислительных ресурсов и платформы функционирования ЭС, в целях обеспечения её более эффективной работы и улучшения технических характеристик, если система работает в неоднородной среде (например, связь с измерительными устройствами).

Э т а п 6. Поддержка ЭС

Этап 6 предполагает поддержку системы в её инструментальной среде разработки при изменении знаний проблемной области. Как правило, для разработки ЭС используются следующие инструментальные средства:

- традиционные языки программирования, позволяющие создавать инструментальные средства и включать интеллектуальные подсистемы в крупные программные комплексы;
- языки искусственного интеллекта, обеспечивающие работу с символьными и логическими данными;
- специальный программный инструментарий, включающий библиотеки и надстройки над языком искусственного интеллекта, позволяющие

работать с заготовками экспертных систем на более высоком уровне, чем в обычных языках искусственного интеллекта;

- «оболочки» экспертных систем, позволяющие создавать готовые системы, не требуя программирования, путём наполнения базы знаний в соответствии с используемой моделью представления знаний.

Контрольные вопросы и задания

1. Объясните, что понимается под экспертными системами, в чём их особенность и назначение.
2. Приведите типовую архитектуру экспертных систем.
3. Приведите классификацию экспертных систем по типу решаемой задачи.
4. Приведите классификацию экспертных систем по связи со временем и степени интеграции с другими программами.
5. Назовите основные этапы технологии разработки экспертных систем.
6. Приведите этапы разработки прототипа экспертной системы.
7. Укажите проблемы инженерии знаний на каждом этапе разработки экспертных систем.
8. Назовите типы систем в процессе развития прототипа экспертной системы.
9. Назовите критерии оценки эффективности работы экспертной системы.
10. Назовите основные инструментальные средства разработки экспертных систем.

3. ПОИСК В ПРОСТРАНСТВЕ СОСТОЯНИЙ

3.1. Пространство состояний

В системах искусственного интеллекта задача поиска решений представляется как *поиск в пространстве состояний*. Ключевым понятием при формализации задачи в пространстве состояний является понятие *состояние* – некоторый момент решения задачи. Например, для игры «Пятнашки» состояние – это некоторая конкретная конфигурация фишек. Для игры в шахматы состояние – это шахматные позиции – расположение шахматных фигур на доске. Среди всех состояний выделяют *начальное состояние* $S_н$ и *целевое состояние* $S_ц$. Задача заключается в том, чтобы перевести начальное состояние в целевое.

Другим важным понятием для рассматриваемого представления является понятие *оператор* – действие или допустимый ход в задаче из множества возможных $G = (g_1, g_2, \ldots, g_k)$. Так, например, для игры в шахматы оператор – это допустимый по шахматным правилам ход, реализация которого преобразует текущее состояние в новое. Для игры в пятнадцать удобнее выделить четыре оператора, соответствующие перемещениям пустой клетки влево, вправо, вверх, вниз. В некоторых случаях оператор может оказаться неприменимым к какому-то состоянию: например, операторы вправо и вниз неприменимы, если пустая клетка расположена в правом нижнем углу.

Решение задачи представляет собой определённую последовательность операторов, преобразующих начальное состояние в целевое состояние: $Z = (S_н \xrightarrow{G} S_ц)$. Допустим, что к состоянию $S_н$ применяется оператор g_1. Его реализация ведёт состояние $S_н$ в S_1, но оно не совпадает с $S_ц$, т. е. $S_1 = g_1(S_н)$, $S_1 \neq S_ц$. Для перевода S_1 в S_2 используется другой оператор – $g_2 \in G$. Результатом его реализации становится состояние $S_2 = g_2(S_1)$, $S_2 \neq S_ц$ и т. д., пока не найдётся g_j, такое что $S_j = g_j(S_{j-1})$ и $S_j = S_ц$. Таким образом, решение задачи ищется в пространстве состояний – множестве состояний, достижимых из начального состояния при помощи применения операторов: $S_ц = g_j(g_{j-1}(\ldots(g_3(g_2(g_1(S_н))))\ldots))$. Последовательность ($g_1$, g_2, g_3, …, g_j) представляет собой алгоритм решения задачи, при этом важно отметить, что перевод $S_н \rightarrow S_ц$ возможен не единственным способом.

П р и м е р «*О наполнении ведра водой*»

Условия задачи: В начальный момент времени пустое ведро стоит рядом с раковиной, кран закрыт. В целевом состоянии необходимо, чтобы ведро было полно и стояло на полу у раковины, а кран был закрыт. Все операции выполняет робот. Требуется построить план его действий – алгоритм перевода начального состояния в целевое.

Введём обозначения, необходимые для формального описания задачи и решения.

Множество объектов и предметов (X):

робот (P),

ведро (B),

кран (K),

пол ($П$),

раковина ($РК$).

Множество свойств объектов (C):

состояние *ведра* – (*пусто*, *полно*);

состояние *крана* – (*открыт*, *закрыт*).

Множество отношений между объектами (R):

у (P, K) – робот у крана;

на (B, $П$) – ведро на полу;

в (B, $РК$) – ведро в раковине.

Множество действий $G = (g_1, g_2, g_3, g_4)$, где g_1 – *поставить* (P, B, $РК$) *или* (P, B, $П$); g_2 – *открыть* (P, K); g_3 – *наполнить* (P, B, $РК$); g_4 – *закрыть* (P, K).

Робота можно исключить из описания, так как он находится в одной точке. Тогда начальное $S_н$ и целевое $S_ц$ состояния могут быть описаны в виде

$S_н$ = <*пусто (В), закрыт (К), на (В, П)*>;

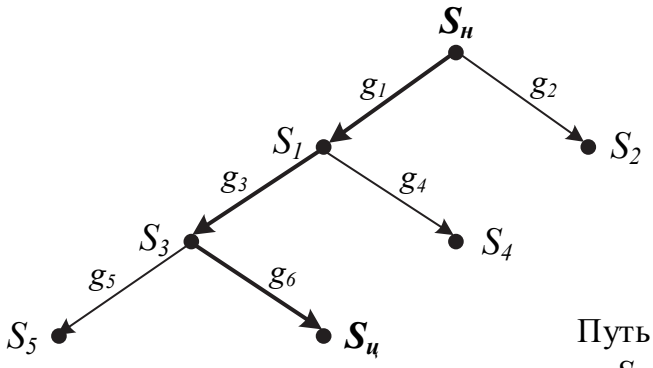
$S_ц$ = <*полно (В), закрыт (К), на (В, П)*>.

Решение задачи заключается в переходе из $S_н$ в $S_ц$ путем применения определенных действий. Отображение состояний в действия может быть представлено в следующем виде:

1. <*пусто (В), закрыт (К), На (В, П)*> → g_1 →
   → <*пусто (В), закрыт (К), В (В, РК)*>.
2. <*пусто (В), закрыт (К), В (В, РК)*> → g_2 →
   → <*пусто (В), открыт (К), В (В, РК)*>.
3. <*пусто (В), открыт (К), В (В, РК)*> → g_3 →
   → <*полно (В), открыт (К), В (В, РК)*>.
4. <*полно (В), открыт (К), В (В, РК)*> → g_4 →
   → <*полно (В), закрыт (К), В (В, П)*>.
5. <*полно (В), закрыт (К), В (В, РК)*> → g_1 →
   → <*полно (В), закрыт (К), На (В, П)*>.

Как видно, последняя ситуация является целевым состоянием. Решением задачи является алгоритм $S_ц = g_1 (g_4 (g_3 (g_2 (g_1 (S_н)))))$.

Процесс поиска решения может быть длинным и сложным, поэтому возникает необходимость в некотором едином методе представления множества состояний и поиска решений. Таким методом является построение графа-дерева: вершины дерева соответствуют состояниям, а дуги – применяемым операторам. Применение оператора к состоянию порождает новое состояние, образуя новую вершину дерева (рис. 3.1). Новая вершина может быть промежуточной или целевой. Если вершина промежуточная, то процесс порождения новых вершин будет продолжен, пока не найдётся целевая. Процесс применения оператора к некоторой вершине называется раскрытием вершины. От каждой порожденной вершины к породившей её расставляются указатели, которые позволяют найти путь назад, к начальной вершине, после того как обнаружена целевая. Длиной пути является количество вершин на этом пути.



Путь решения задачи:
$S_ц = g_6(g_3(g_1(S_н)))$

Рис. 3.1. Дерево решения задачи

Общая процедура построения дерева пространства состояний может быть представлена следующим образом:

1. К корню дерева $S_н$ применяются операторы g_j из множества G. Полученные вершины образуют первый уровень вершин.

2. Каждая из полученных вершин проверяется, не является ли она целевой. Если нет, то процесс продолжается по отношению к каждой из них. Образуется второй уровень вершин. Если к вершине неприменим ни один оператор из G, то вершина становится терминальной (конечной). Таким образом, на каждом шаге выполняется две операции: порождение новой вершины и проверка, не является ли вершина целевой.

3. Когда целевая вершина найдена, в обратном направлении (от цели к началу) просматриваются указатели дуг и определяется путь решения.

В общем случае количество вершин может быть достаточно большим, и тогда возникает проблема перебора вершин. Все известные алгоритмы

поиска в пространстве состояний различаются несколькими характеристиками:

- использование эвристической информации;
- порядок раскрытия вершин;
- полнота просмотра пространства;
- направление поиска.

В соответствии с первой характеристикой алгоритмы делятся на *слепые* и *эвристические*. В слепых алгоритмах местонахождение целевой вершины никак не влияет на порядок, в котором раскрываются вершины. В противоположность им эвристические алгоритмы для уменьшения возникающего перебора используют эвристическую информацию о том, где в пространстве состояний расположена цель, поэтому для раскрытия обычно выбирается более перспективная вершина. Два основных вида слепых алгоритмов поиска, различающихся порядком раскрытия вершин, – это алгоритмы *поиска в ширину* и *поиска в глубину*. Как слепые, так и эвристические алгоритмы могут отличаться полнотой просмотра пространства состояний.

Полные алгоритмы перебора осуществляют полный просмотр пространства, *неполные* алгоритмы реализуют просмотр лишь части пространства, и если искомая целевая вершина не находится в этой части, то решение этим алгоритмом не будет найдено.

В соответствии с направлением поиска алгоритмы можно разделить на *прямые* (поиск ведется от начальной вершины к целевой), *обратные* (поиск от целевой вершины в направлении к начальной) и *двунаправленные* (чередование прямого и обратного поиска или их одновременное проведение). Наиболее часто используемыми (отчасти в силу их простоты) являются прямые алгоритмы.

3.2. Методы полного перебора

В слепых алгоритмах поиска местонахождение целевой вершины никак не влияет на порядок, в котором рассматриваются вершины. Основными стратегиями поиска решающего пути, различающимися порядком раскрытия вершин, являются *поиск в ширину (breadth-first search)* и *поиск в глубину (depth-first search)*.

3.2.1. Стратегия поиска в ширину

В данном подходе вершины раскрываются в том порядке, в котором они строятся: сначала формируются «соседи», потом «потомки». Основной алгоритм состоит в выполнении следующих действий:

1) Раскрывается начальная вершина $S_н$, путём применения одного или нескольких операторов (в зависимости от условий задачи). В результате образуются вершины первого уровня. Затем полученные вершины также раскрываются, образуя вершины второго уровня и т. д. Пример дерева полного перебора в ширину, где S_1 и S_2 – вершины первого уровня; S_3, S_4, S_5 и S_6 – вершины второго уровня и т. д. см. на рис. 3.2. Наименования вершин соответствуют порядку их построения.

2) Расставляются указатели, ведущие от новых вершин к корню. Это могут быть условные имена и имена операторов, могут быть реальные величины, такие как расстояние, стоимость и т. д.

3) Проверяется наличие целевой вершины среди полученных вершин. Если обнаружена целевая вершина, то формируется решение на основе соответствующего оператора. Если целевой вершины нет, то рассматривается первая порожденная вершина и к ней применяется тот же алгоритм. Перебор вершин осуществляется до тех пор, пока среди получаемых вершин не окажется целевой.

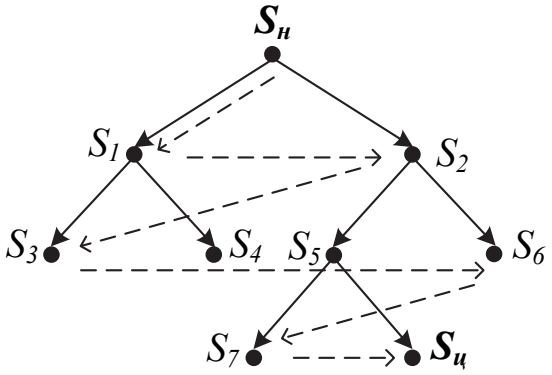


Рис. 3.2. Дерево полного перебора в ширину

Стратегия полного перебора в ширину гарантирует нахождение целевой вершины, если дерево пространства состояний не бесконечно. Путей достижения цели может быть много, в этом случае имеется возможность выбрать наикратчайший (в зависимости от критериев) путь. Данный метод имеет и другие названия, например метод грубой силы, метод проб и ошибок.

3.2.2. Стратегия поиска в глубину

В отличие от стратегии поиска в ширину в данном подходе сначала формируются дочерние вершины, а затем соседние. Алгоритм перебора в глубину состоит в следующем:

1) Раскрывается начальная вершина $S_н$, путём применения одного или нескольких операторов (в зависимости от условий задачи).

2) Раскрывается первая вершина, получаемая в результате раскрытия $S_н$. Ставится указатель.

3) Если вершина раскрывается, то следующей будет раскрываться вновь порожденная вершина. Если вершина не раскрывается, то процесс возвращается в предыдущую вершину.

4) При формировании целевой вершины процесс раскрытия заканчивается, и по указателям строится путь, ведущий к корню. Соответствующие дугам операторы образуют решение задачи.

5) Пример дерева полного перебора в глубину см. на рис. 3.3. Наименования вершин соответствуют порядку их построения. Сначала определяются дочерние вершины S_1 – S_2 и S_3, а затем соседняя вершина S_4 и её потомки. По сравнению с предыдущим примером с помощью данного подхода целевое состояние найдено за меньшее число шагов.

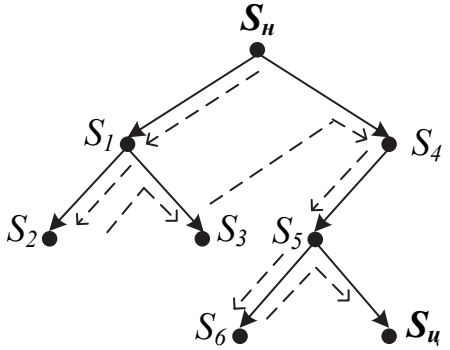


Рис. 3.3. Дерево полного перебора в глубину

Однако такой подход может привести к бесконечному процессу, если пространство состояний бесконечно или поиск пошёл по ветви дерева, не содержащей целевое состояние. Чтобы как-то ограничить перебор, вводится понятие *глубины просмотра*, тогда в первую очередь раскрытию подлежит вершина наибольшей глубины, но не превышающей эту границу. Если образующий путь оказывается бесполезным и при заданной глубине не получилось найти целевой вершины, осуществляется возврат в вершину, предшествующую раскрытой, и поиск продолжается по новой ветке дерева. Соответствующий алгоритм называется *ограниченным перебором в глубину*. Возврат выполняется с помощью указателей. Данный метод, как и поиск в ширину, относится к методам грубой силы, обеспечивая перебор всех состояний.

П р и м е р «*Миссионеры и каннибалы*»

На левом берегу реки находятся три миссионера и три каннибала. К этому же берегу причалена единственная лодка. На этой лодке нужно переправить всех миссионеров и всех каннибалов на правый берег при

условии, что лодка одновременно может перевозить не более двух человек, в обратный путь на лодке должен отправиться один человек. Если окажется, что каннибалов на любом из берегов больше, чем миссионеров, то несчастных просто съедят. Требуется найти решение с помощью стратегии полного перебора в ширину.

Поиск решения осуществляется в пространстве состояний задачи (рис. 3.4).

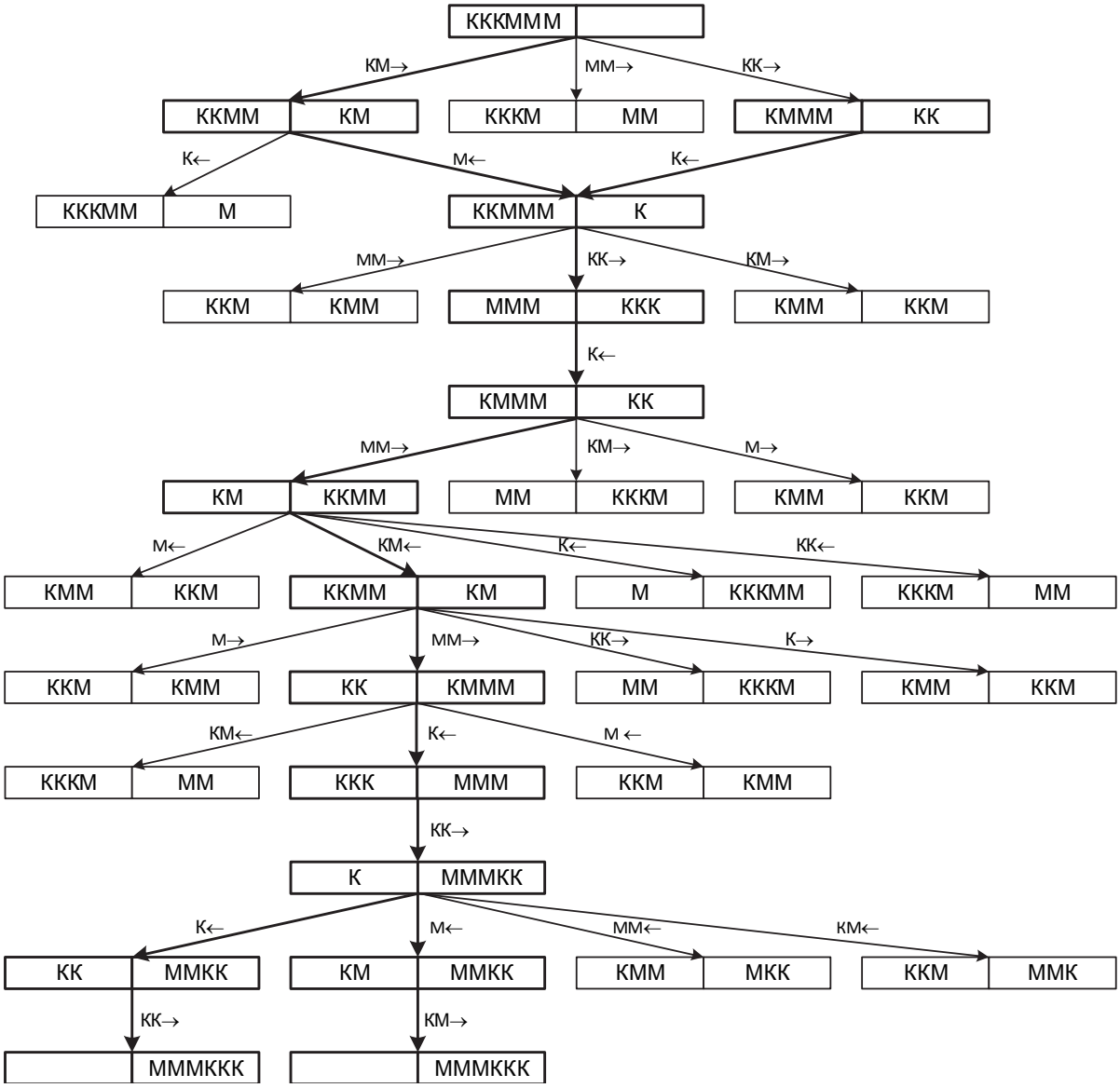


Рис. 3.4. Дерево поиска решения для примера «Миссионеры и каннибалы»

Для формального описания задачи введены следующие обозначения:
Начальное состояние: КККММ |
Целевое состояние: | МММККК

Операторы:

К→, К← – в путь отправляется один каннибал;

М→, М← – в путь отправляется один миссионер;

КК→, КК← – в путь отправляются два каннибала;

ММ→, ММ← – в путь отправляются два миссионера;

КМ→, КМ← – в путь отправляются каннибал и миссионер;

Стрелка обозначает направление, в котором движутся люди: с левого берега на правый или с правого на левый.

Лодка исключена из описания, так как представляет собой средство для передвижения людей, и её наличие условно. Условие, когда каннибалов становится больше, чем миссионеров, рассматривается как ограничение, и такое состояние является тупиковым. Чтобы исключить зацикливание, не выполняются повторные действия: если действие оператора ведет к состоянию, полученному на предыдущем шаге, он не применяется.

Для данного типа задач метод полного перебора является единственным способом поиска решения, других вариантов найти решение, кроме как перебрать все возможные варианты, не существует. В целом алгоритмы слепого перебора являются малоэффективными методами, приводящими в случае нетривиальных задач к проблеме комбинаторной сложности.

3.3. Методы эвристического поиска

Эвристические алгоритмы для уменьшения возникающего перебора используют априорную эвристическую информацию о том, где в пространстве состояний расположена цель, поэтому для раскрытия выбирается более перспективная вершина. *Эвристика* (от гр. *heurisko – отыскиваю*, *открываю*) в пространстве состояний поиска определяется как набор правил для выбора тех ветвей из пространства состояний, которые с наибольшей вероятностью приведут к приемлемому решению проблемы. Идея, лежащая в основе большинства эвристических алгоритмов, состоит в том, чтобы оценить с помощью количественных оценок перспективность нераскрытых вершин пространства состояний с точки зрения достижения цели и выбрать для продолжения поиска наиболее перспективную вершину.

Рассмотрим наиболее распространенные методы эвристического поиска на примере решения игровых задач, связанных с перебором большого числа состояний.

Общий алгоритм решения задач:

1. Выбрать из множества возможных действий некоторое действие.
2. Осуществить выбранное действие и изменить текущее состояние.
3. Оценить состояние.

4. Отбросить бесполезные состояния.

5. Если достигнуто целевое состояние – конец; если нет, то выбрать новое начальное состояние и начать заново.

Наиболее простой формой эвристического поиска является «*восхождение на гору*» (*hill climbing*). Для поиска выбирается наилучший «потомок», при этом о его «братьях» и «родителях» забывают. Поиск прекращается, когда достигается состояние, которое лучше, чем любой из его наследников. В процессе поиска используется некоторая оценочная функция, с помощью которой можно грубо оценить, насколько «хорошим» (или «плохим») является текущее состояние. Например, оценочная функция для программы игры в шахматы может включать очевидную оценку материала (количества и качества имеющихся на доске фигур) – своего и соперника. Затем программа перебирает возможные операторы перехода в новое состояние – ходы, и, сравнивая результаты вариантов, отыскивает вариант, характеризующийся максимальным значением оценочной функции. Однако иметь много фигур не значит иметь лучшую позицию и быть ближе к успеху.

Такая оценочная функция не учитывает многих особенностей этой игры. Кроме того, даже если оценочная функция позволяет адекватно оценить текущую ситуацию, разнообразные ситуации игры могут быть источником затруднений. Например, оказывается, что все возможные ходы одинаково хороши (или плохи), т. е. ни один из путей не влечет подъем. Другой источник затруднений – наличие локальных максимумов, из которых возможен только спуск, т. е. «ухудшение» состояния. Например, взять ферзя соперника и после этого проиграть партию. Другой пример: в головоломке «пятнашки», чтобы передвинуть фишку в нужную позицию, необходимо сдвинуть фишку, находящуюся в наилучшей позиции.

Лучшими свойствами обладает другая форма эвристического поиска – «*жадный*» (*first-best search*). На каждом шаге алгоритм сохраняет состояние с учётом эвристической оценки его «близости к цели». Если путь оказывается ошибочным, алгоритм возвращается к некоторому ранее сгенерированному лучшему состоянию и продолжает поиск в другой части пространства. Благодаря своей универсальности «жадный» алгоритм поиска может использоваться с различными эвристиками, начиная от субъективных оценок «хороших» состояний и заканчивая сложными критериями оценки состояний, ведущих к цели.

3.3.1. Алгоритм A*

Алгоритм A* относится к группе «*first-best search*».

Каждому состоянию *S*, полученному из начального состояния в результате определенной последовательности действий, ставится численная оценка:

$$f(S) = g(S) + h(S),$$

где $g(S)$ – реальная стоимость пути от начального состояния до состояния S; $h(S)$ – эвристическая функция, которая оценивает стоимость наилучшей последовательности действий, начинающейся в S и заканчивающейся целевым состоянием.

В алгоритме A^* к функции h предъявляется следующее требование: если $h^*(S)$ – минимальная стоимость всей последовательности действий, позволяющей получить решение, исходя из состояния S, то $h(S)$ нижняя граница $h^*(S)$; кроме того, $h(S)$ должна быть положительной. Таким образом: $0 \le h(S) \le h^*(S)$.

Функция $f(S)$ является оценочной для оптимальной функции:

$$f^*(S) = g^*(S) + h^*(S),$$

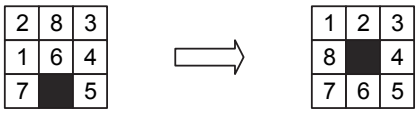
где $f^*(S)$ – наименьшая стоимость решений, проходящих через S; $g^*(S)$ – наилучшая стоимость пути от начального состояния до S; $h^*(S)$ – наилучшая стоимость пути от S до целевого состояния.

Алгоритм A^* приводит к цели за конечное число шагов, если существует конечная последовательность действий, которая ведёт от исходного состояния к решению.

П р и м е р «*Игра* “*Восьмерка*”»

Рассмотрим классический пример применения алгоритма A^* для игры «Восьмерка» – упрощенного варианта игры «Пятнашки».

На поле из 9 клеток имеется 8 фишек от 1 до 8. Требуется расставить их на определённые позиции путём последовательного передвижения одной из фишек на пустую позицию (рис. 3.5).



Начальное состояние Целевое состояние

Рис. 3.5. Условие поиска решения
в игре «Восьмерка»

Для данной задачи удобнее выделить четыре оператора, соответствующие перемещениям пустой клетки: влево, вправо, вверх и вниз. В некоторых случаях оператор может оказаться неприменимым к какому-то состоянию, например, операторы вправо и вниз неприменимы, если пустая клетка расположена в правом нижнем углу.

Дерево поиска с возможными вариантами ходов и значениями оценочной функции представлено на рис. 3.6. Оценочная функция имеет вид

$f(S) = g(S) + h(S)$, где за $g(S)$ примем количество выполненных шагов, а за $h(S)$ – эвристика – количество фишек, находящихся не на своих местах.

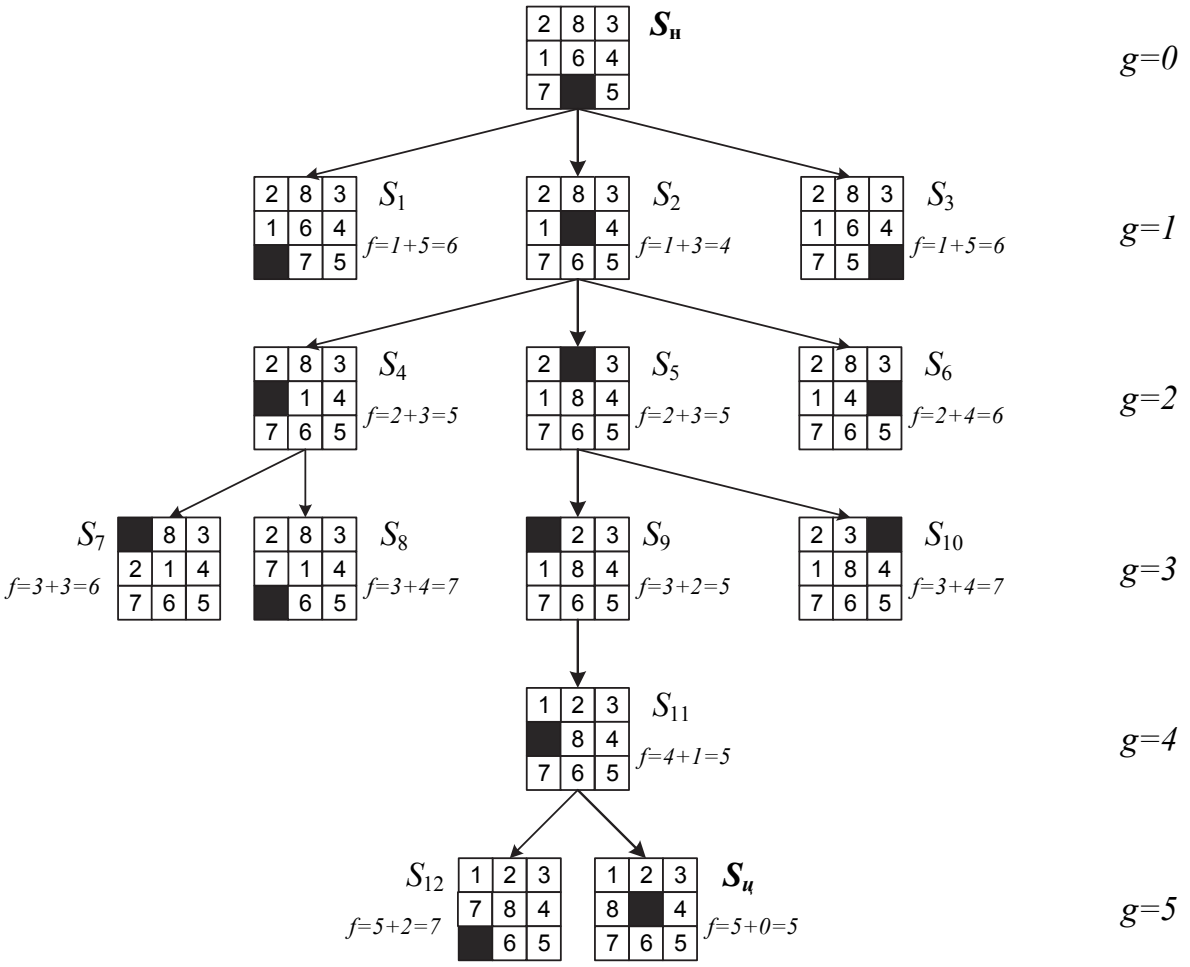


Рис. 3.6. Дерево поиска решения для игры «Восьмерка» с помощью алгоритма A^*

На первом шаге имеется три возможных передвижения пустой клетки: вверх (↑), влево (←) и вправо (→). На каждом шаге выбираем состояние с минимальным значением оценочной функции и выполняем возможные ходы, достраивая дерево поиска. Пока не достигнуто целевое состояние, повторяем цикл, каждый раз выбирая состояние с минимальным значением оценочной функции. Состояния S_4 и S_5 имеют эвристическое значение, равное 5. Состояние S_4 проверено первым, его «потомками» являются S_7 и S_8. Хотя S_7 является «потомком» S_4, для него число фишек, расположенных не на своих местах, соответствует состоянию S_5, при этом данное состояние находится на один уровень ниже в пространстве состояний. Мера глубины $g(S)$ заставляет алгоритм выбирать S_5 в качестве оценки. Алгоритм возвращается к состоянию S_5, находящемуся на один уровень

выше, и затем продолжает двигаться к цели по другой ветке. Компонент $g(S)$ функции оценки придаёт нашему поиску свойства поиска в ширину. Он предотвращает возможность заблуждения из-за ошибочной оценки: если эвристика непрерывно возвращает «хорошие» оценки для состояний на пути, не ведущем к цели, то значение $g(S)$ будет расти и доминировать над $h(S)$, возвращая процедуру поиска к кратчайшему пути. Это избавляет алгоритм от зацикливания.

Состояния, которые уже встречались на предыдущих шагах, повторно не рассматриваются, так как при повторном получении их оценочная функция будет принимать заведомо большее значение. Найденный путь решения задачи длиною в пять ходов может быть получен и другими методами перебора, но использование оценочной функции приводит к существенно меньшему числу раскрытий вершин.

В рассматриваемом примере применяется простая эвристика – «количество фишек, находящихся не на своих местах», однако эта эвристика не использует всю имеющуюся информацию, например, она не учитывает расстояние, на которое фишки должны быть перемещены. В этом случае более информативной является эвристика, суммирующая расстояния между фишкой, находящейся не на своём месте, и полем, в которое она должна быть перемещена для достижения целевого состояния. Однако и в первом, и во втором случае эвристики не учитывают трудности при перестановке фишек. Если необходимо переместить две рядом стоящие фишки, то это потребует более двух ходов, чтобы фишки могли обойти друг друга. Поэтому важно не просто сложить расстояния между фишками, но и как минимум удвоить расстояния между элементами, которые должны быть взаимно переставлены. Такие рассуждения показывают, насколько трудно подобрать хорошую эвристику: чем более обоснованной она является, тем более направленным становится поиск и тем меньше состояний придется проверить при поиске цели. Разработка хорошей эвристики – это эмпирическая проблема, в решении которой во многом помогают рассуждения и интуиция.

Игры всегда были прикладной областью для эвристических алгоритмов. Игры для двух человек более сложные, чем простые головоломки, из-за существования непредсказуемого противника. Классический подход, реализуемый мыслящим существом, состоит в прогнозировании последующих ходов – своих и ответных противника. Такие игры не только обеспечивают некоторые интересные возможности для развития эвристик, но и создают большие трудности в разработке алгоритмов поиска. В играх с небольшим пространством состояний проблема решается систематическим перебором всех возможных ходов и контрмер противника. В играх, где полный перебор либо невозможен, либо нежелателен и можно просмотреть только часть пространства состояний, игрок вынужден использовать эвристику.

3.3.2. Стратегия первого уровня

В сложных играх вместо нереальной задачи поиска полной игровой стратегии решается, как правило, более простая задача − поиск для заданной позиции игры достаточно хорошего первого хода.

Принимая во внимание, что партия должна закончиться выигрышем или проигрышем, процесс игры можно представить в виде шагов:

1. Если существует выигрывающий ход, то сделать его, иначе рассмотреть другие разрешенные ходы.

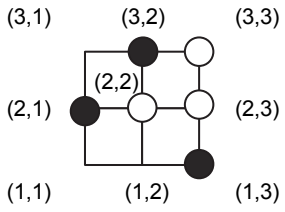
2. Если существует выигрывающий ответ противника на следующий ход, то исключить его, иначе учесть число возможных ответов противника после хода.

3. Сделать ход, на который противник имеет минимальное число ответных ходов.

П р и м е р «*Игра* "*Тик-Так-Ту*"»

Игра ведётся на поле из девяти клеток. Оба играющих имеют по три фишки. В первом туре игроки по очереди расставляют свои фишки на свободные клетки. Во втором туре, когда размещены все шесть фишек, игроки поочередно перемещают по одной фишке на свободные соседние клетки. Соседней считается клетка, расположенная в одном шаге от исходной. Выигрывает тот, кто первым расположит свои фишки по прямой линии.

Рассмотрим состояние игры, приведенное на рис. 3.7. Покажем, как найти наилучший ход из заданного игрового состояния, используя стратегию первого уровня.



Ход черных

Рис. 3.7. Состояние в игре «Тик-Так-ту»

По правилам игры у игрока, ходящего чёрными фишками, в данной ситуации существует четыре допустимых хода:

a) (3,2)→(3,1);

b) (2,1)→(3,1);

c) (2,1)→(1,1);

d) (1,3)→(1,2).

Ни один из ходов не является непосредственно выигрывающим, ход d) приводит к немедленному проигрышу, поэтому его сразу отбрасываем. Ход a) даёт возможность противнику ответить четырьмя ходами, а ходы b) и c) – тремя. Следуя стратегии первого уровня, необходимо выбрать ход, на который противник имеет минимальное число ответных ходов. Ходы b) и c) приведут к выигрышу, в чём можно убедиться, применяя дальше данную стратегию.

Рассмотренная стратегия первого уровня может применяться независимо от вида игры и к любому уровню. В случаях, когда нельзя построить полное дерево ходов, с её помощью можно проводить анализ ходов на любую выбранную глубину, используя числовые значения в возникающих ситуациях – число возможных ответных ходов противника.

3.3.3. Метод минимакса

Метод минимакса реализует поиск с учётом предположения, что противники использует одни и те же знания о пространстве состояний и применяют их для своей победы. В игре два противника: *MIN* и *MAX*. *MAX* представляет игрока, пытающегося выиграть или максимизировать своё преимущество, а *MIN* – противника, который пытается минимизировать счет своего оппонента. *MIN* использует ту же информацию и всегда пытается достичь наихудшего для *MAX* состояния.

Метод минимакса основан на следующей идее: если бы игроку *MAX* пришлось выбирать один из нескольких возможных ходов, то он выбрал бы наиболее сильный ход, т. е. ход, приводящий к состоянию с наибольшей оценкой; если бы игроку *MIN* пришлось выбирать ход, то он выбрал бы ход, приводящий к состоянию с наименьшей оценкой.

Принцип минимакса:

- вершине *MAX* дерева игры присваивается оценка, равная максимуму оценок её дочерних вершин;
- вершине *MIN* дерева игры присваивается оценка, равная минимуму оценок ее дочерних вершин.

П р и м е р «*Игра* “*Ним*”»

Рассмотрим применение метода минимакса для игры «*Ним*», пространство состояний которой допускает исчерпывающий поиск.

На стол между двумя противниками выкладывается некоторое число фишек. При каждом ходе игрок должен разделить фишки на два непустых набора разных размеров. Игрок, который первым не сможет сделать ход, проигрывает.

Пусть у игроков семь фишек. При реализации метода минимакса пометим каждый уровень области поиска именем игрока, выполняющего

очередной ход (рис. 3.8). В рассматриваемом примере первый ход делает игрок *MIN*. Каждому листовому узлу игрового дерева в зависимости от победителя *MAX* или *MIN* присваивается значение 1 или 0 соответственно. Затем значения передаются вверх согласно *принципу минимакса*. Полученные значения оценок используются для того, чтобы выбрать наилучший ход среди возможных. Наилучший ход выбирается для начального состояния (корневой вершины дерева) путём просмотра всех игровых позиций и возможного исхода игры.

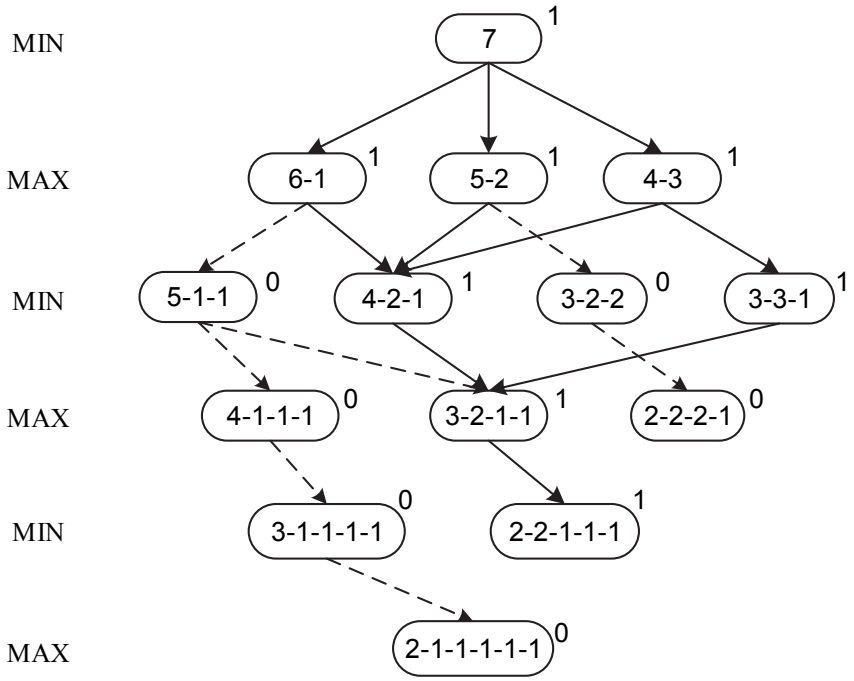


Рис. 3.8. Метод минимакса для игры «Ним»

Поскольку любой из возможных шагов *MIN* ведёт к вершинам со значением 1, второй игрок, *MAX,* может победить независимо от хода игрока *MIN*. При этом *MIN* может победить только в случае, если *MAX* допустит ошибку.

В сложных играх редко удаётся построить полный граф пространства состояний. Как правило, поиск выполняется в глубину на определённое число уровней. Данная стратегия называется *прогнозированием N-го слоя*, где *N* – число исследуемых уровней. Такой подграф не отражает всех состояний игры, и поэтому его узлам невозможно присвоить значения, отражающие победу или поражение игроков. Значения листовых узлов частичного дерева пространства состояний оцениваются *статической эвристической функцией* – эвристические значения оптимального состояния, которое может быть достигнуто за *N* шагов от корневой вершины. Для

оценки остальных вершин – промежуточных и корневой – используется *принцип минимакса* (рис. 3.9).

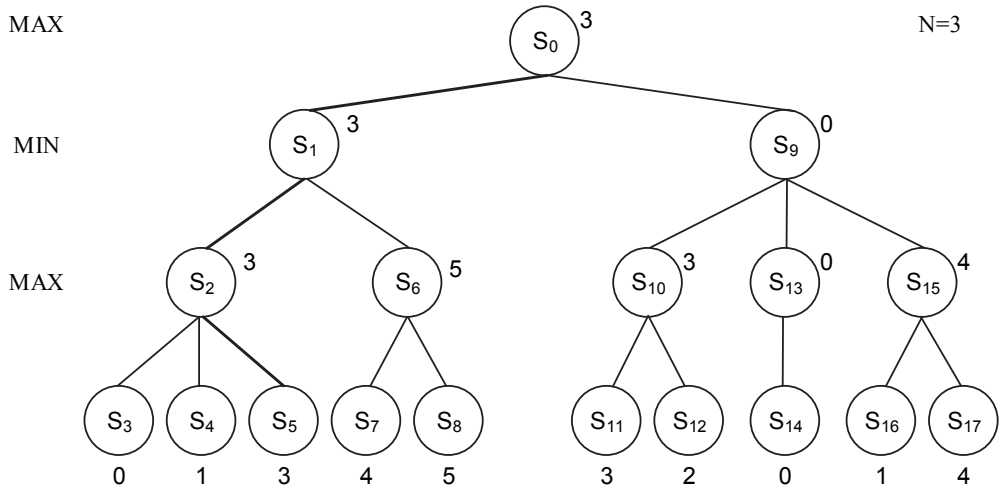


Рис. 3.9. Метод минимакса при фиксированной глубине поиска

Вершины глубины *N=3* обозначены оценками − значениями эвристической оценочной функции. Числовые индексы имён вершин показывают порядок, в котором эти вершины строились алгоритмом поиска в глубину. Рядом с вершинами находятся их минимаксные оценки. По полученным оценкам определяется наилучший ход от исходного состояния. В рассматриваемом примере наилучший ход для начального состояния – первый из двух возможных. Последовательность ходов S_0-S_1-S_2-S_5, при которой каждый из игроков выбирает наилучший для себя ход, называется *минимаксно-оптимальной игрой* (или *основным вариантом игры*). Стратегия минимакса интегрирует отдельные оценки в значение, которое присваивается родительскому состоянию. Прогнозирование увеличивает силу эвристики, позволяя применить её на большей области пространства состояний. Стратегия минимакса интегрирует отдельные оценки в значение, которое присваивается родительскому состоянию.

3.3.4. Процедура альфа-бета отсечения

Алгоритм, получивший название «Альфа-бета отсечение» (рис. 3.10) позволяет повысить эффективность поиска в играх с двумя игроками. В основе лежит очевидный факт: если есть два варианта хода одного игрока, то худший в ряде случаев можно отбросить, не выясняя, насколько он хуже. Дерево до глубины *N=3* строится алгоритмом поиска в глубину. Сразу, как только это становится возможно, вычисляются не только эвристические оценки листьев игрового дерева, но и предварительные минимаксные оценки промежуточных вершин. Предварительная оценка может

быть получена при наличии оценки хотя бы одной дочерней вершины. По мере получения новых (предварительных и точных) оценок в ходе построения дерева предварительные оценки постепенно уточняются по минимаксному принципу.

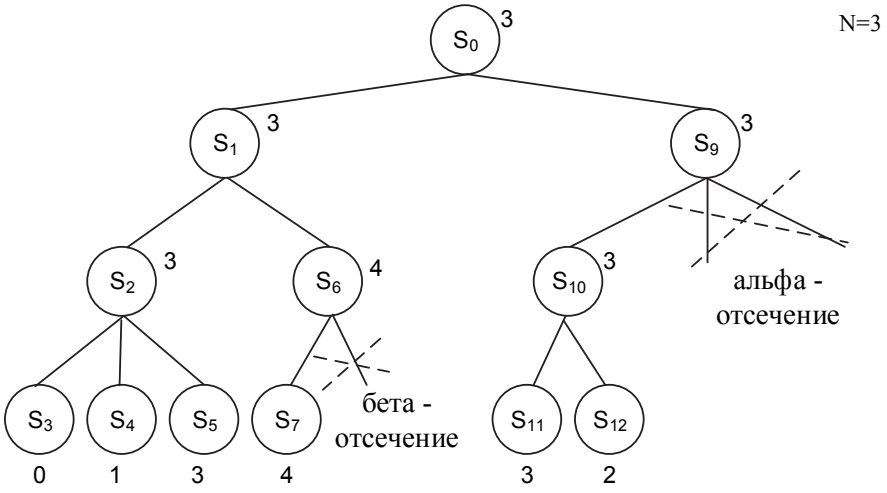


Рис. 3.10. Процедура альфа-бета усечения

Пусть построены вершины S_1, S_2 и первые три конечные вершины − S_3, S_4, S_5. Полученные листья оценены эвристической функцией, и вершина S_2 получила точную минимаксную оценку 3, а вершина S_1 − предварительную оценку 3. Далее при построении и раскрытии вершины S_6 эвристическая оценка первой ее дочерней вершины даёт значение 4, которое становится предварительной оценкой вершины S_6. Эта предварительная оценка позже будет пересчитана (после построения второй дочерней вершины), причем согласно минимаксному принципу она может только увеличиться (она равна максимуму оценок дочерних вершин), но даже если она увеличится, это не повлияет на оценку вершины S_1, поскольку последняя при уточнении по минимаксному принципу может только уменьшаться (она равна минимуму оценок дочерних вершин). Следовательно, можно пренебречь второй дочерней вершиной для S_6, не строить и не оценивать её (так как уточнение оценки вершины S_6 не повлияет на оценку вершины S_1). Такое сокращение процедуры поиска в дереве называется *отсечением ветвей дерева*.

Продолжим процесс поиска в глубину с одновременным вычислением предварительных (и точных, где это возможно) оценок вершин до момента, когда получены вершины S_9, S_{10} и две дочерних вершины, которые оцениваются эвристической функцией. Вершина S_{10} получила точную минимаксную оценку 3, а её родительская S_9 получила только предварительную оценку 3. Эта предварительная оценка вершины S_9 может быть уточнена

в дальнейшем, но в соответствии с минимаксным принципом возможно только её уменьшение, а это не повлияет на оценку начальной вершины S_0, так как она согласно минимаксному принципу может только увеличиваться. Таким образом, построение дерева можно прервать ниже вершины S_9, отсекая целиком вторую и третью ветви. На этом построение игрового дерева заканчивается, полученный результат − лучший первый ход − тот же самый, что и при методе минимакса. У некоторых вершин дерева осталась неуточнённая предварительная оценка, однако этих приближённых значений достаточно для того, чтобы определить точную минимаксную оценку начальной вершины и наилучший первый ход. В то же время произошло существенное сокращение: вместо семнадцати вершин построено одиннадцать, и вместо десяти обращений к эвристической оценочной функции понадобилось всего шесть.

Предварительную оценку *MAX*-вершины принято называть *альфа-величина*, а предварительную оценку *MIN*-вершины – *бета-величина*. Из правил вычисления оценок следует, что альфа-величины не могут уменьшаться, а бета-величины не могут увеличиваться. При этом в случае, когда отсекаются ветви дерева, начиная с вершин, которым приписана альфа-величина, имеет место *альфа-отсечение*; в случае, когда отсекаются ветви дерева, начиная с вершин, которым приписана бета-величина, имеет место *бета-отсечение*. В рассматриваемом примере на рис. 3.10 первое прерывание перебора является *бета-отсечением*, а второе − *альфа-отсечением*. Альфа-бета процедура за счёт сокращения перебора является более эффективной реализацией минимаксного принципа. Оценочная функция и альфа-бета процедура − две непременные составляющие подавляющего большинства компьютерных игровых программ.

П р и м е р «*Игра* “*Крестики-нолики*”»

Игра ведётся на поле из девяти клеток. Каждый игрок имеет свой знак – крестик или нолик (выбор определяет жребий). Суть игры заключается в следующем: игроки по очереди расставляют в свободные клетки свой знак, выигрывает тот, кто первый закроет линию.

В данной игре для оценки состояний используется более сложная эвристика, позволяющая измерить конфликт в игре: определяются все возможные победные строки для игрока *MAX* (всегда ходит крестиками), а затем из них вычитается общее количество возможных победных строк для игрока *MIN* (всегда ходит ноликами). Алгоритм поиска пытается максимизировать эту разницу. Если состояние приводит к победе игрока *MAX*, то оно оценивается как $+\infty$, если приводит к победе игрока *MIN* – $-\infty$ (рис. 3.11).

Пример определения лучшего хода для игрока *MAX* с использованием двухуровневого минимакса продемонстрирован на рис. 3.12.

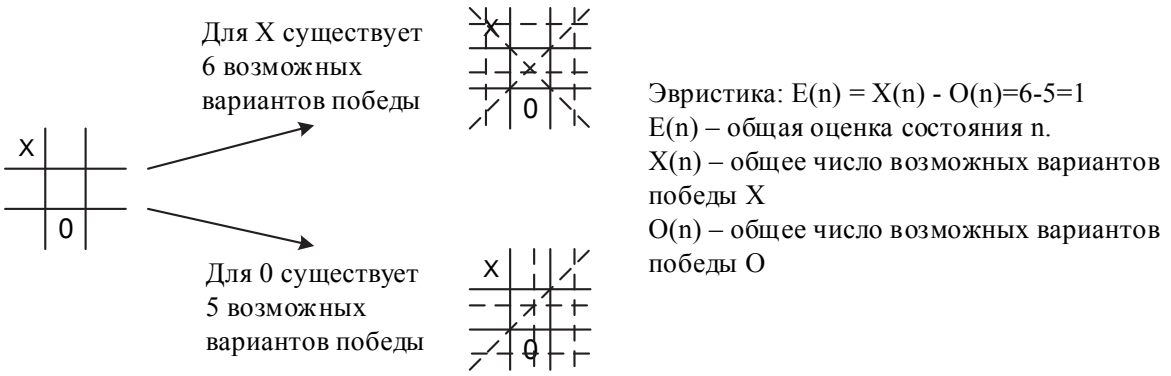


Рис. 3.11. Пример расчета эвристики в игре «Крестики-нолики»

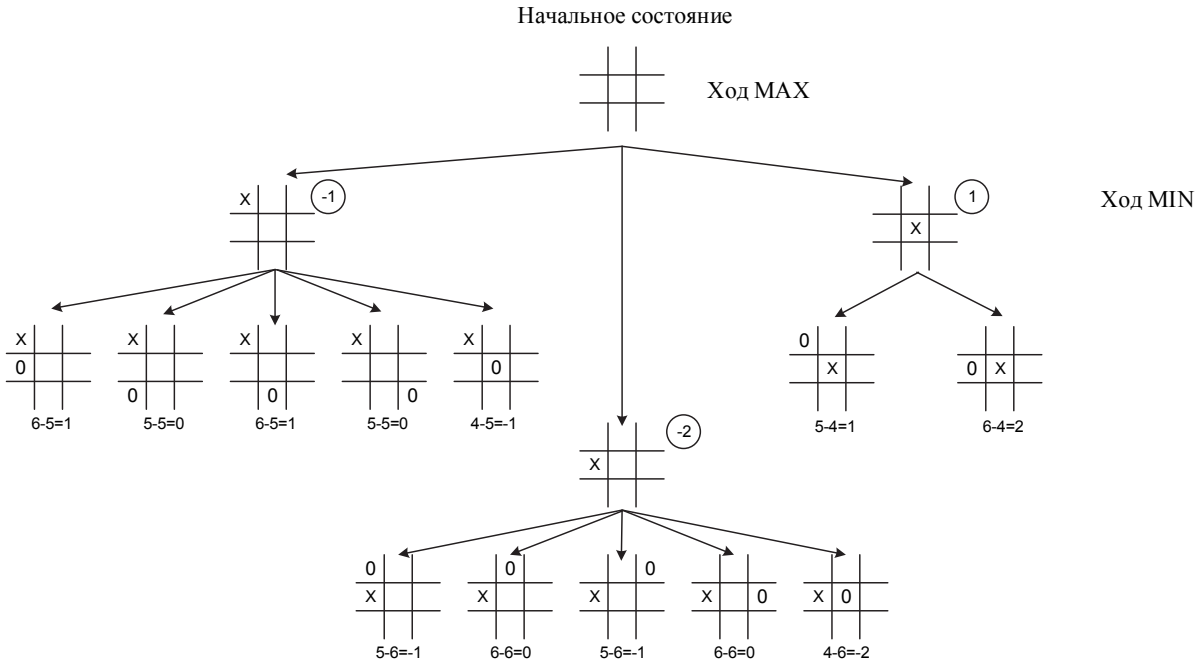


Рис. 3.12. Двухуровневый минимакс
для первого хода игры «Крестики-нолики»

Состояния второго уровня (листья игрового дерева) оценены с помощью эвристики, состояния первого уровня определены по минимаксному принципу: оценка равна минимуму оценок дочерних вершин (ход игрока *MIN*). Для игрока *MAX* лучший ход – ход, ведущий к состоянию с максимальной оценкой.

Контрольные вопросы и задания

1. Объясните, что представляет собой пространство состояний.
2. Объясните, как определяются начальное и целевое состояние.
3. Объясните, что представляет собой оператор.
4. Объясните, как определяется решение задачи в пространстве состояний.
5. Приведите форму пространства состояний.
6. Приведите классификацию алгоритмов поиска в пространстве состояний.
7. Объясните стратегию поиска в ширину.
8. Объясните стратегию поиска в глубину.
9. Назовите эвристические алгоритмы.
10. Назовите общий алгоритм эвристических задач.
11. Назовите виды эвристических алгоритмов, их преимущества и недостатки.
12. Опишите алгоритм A^*.
13. Укажите условия использования стратегии первого уровня.
14. Опишите метод минимакса.
15. Опишите схему реализации принципа минимакса, когда пространство состояний допускает исчерпывающий поиск.
16. Опишите схему реализации принципа минимакса при фиксированной глубине поиска.
17. Объясните понятие «минимаксно-оптимальная игра».
18. Опишите процедуру альфа-бета отсечения.
19. Назовите правила вычисления оценок вершин дерева игры при реализации процедуры альфа-бета отсечения.
20. Назовите правила отсечения ветвей дерева игры при реализации процедуры альфа-бета отсечения.

4. ПРЕДСТАВЛЕНИЕ ЗНАНИЙ

4.1. Продукционная модель

4.1.1. Понятие продукционной модели

Термин *продукция* был введён американским логиком Е. Постом в 1940-е гг. в работах по обоснованию и формализации алгоритмических систем. *Продукцией*, или *продукционным правилом*, называется символьная конструкция вида

S: ЕСЛИ < условие > ТО <действие>,

где *S* – спецификация правила; конструкция «*ЕСЛИ <условие > ТО <действие>*» – ядро правила. Спецификация правила содержит уникальное имя и условия применения этого правила, например: область применения, приоритет и любую другую полезную информацию. Ядро правила содержит служебные слова «*ЕСЛИ*» и «*ТО*», которые интерпретируются как причинно-следственная связь, <*условие*> – левая часть правила, или антецедент, <*действие*> – правая часть правила, или консеквент (заключение).

Условие чаще всего представляет собой логико-лингвистическое высказывание, т. е. такое символьное выражение, которому придаётся логический смысл – истина или ложь. Считается, что условие представляет некоторый сложный факт, описывающий текущую ситуацию задачи. Сложный факт строится из простых фактов с помощью логико-лингвистических связок «*и*», «*или*», «*не*», которые интерпретируются соответственно как логические связки &, $\vee$, $\neg$ (эти обозначения также будут использоваться для сокращения записи продукций). Простой факт определяется как некоторое утверждение, которое либо истинно, либо ложно.

Действие состоит из одной или нескольких операций, которые выполняются в порядке их перечисления. Элементарное действие представляет собой либо установление нового факта, который помещается в рабочую память, либо такое действие, как «выдать сообщение», «прочитать данные», «выполнить процедуру» и т. п. Виды действий определяются спецификой задачи.

Правило называется применимым, если при текущем состоянии рабочей памяти условие правила истинно. Процесс применения правила заключается в выполнении действий в его правой части.

Сущность продукционного подхода состоит в том, что все знания о предметной области представляются в виде совокупности правил, которые

помещаются в базу знаний (базу правил). Форма правила определяется тем, как представлены элементарные факты.

Существует несколько способов представления фактов:

• Использование *образцов текстов*

Пример правила:

ЕСЛИ <двигатель не заводится> & <стартер двигателя не работает> ТО <неполадки в системе электропитания стартера>.

Факт считается истинным, если представляющий его образец текста находится в рабочей памяти; если нет, то факт считается либо ложным, либо неопределенным. Преимущество использования образцов текстов состоит в том, что можно заранее их не описывать, а пополнять рабочую память по мере необходимости.

• Применение *кортежей отношений*

<объект; атрибут 1, значение 1; атрибут 2, значение 2; …>

Пример правила:

ЕСЛИ <собака; порода, такса; кличка, Тузик; масть, рыжий>

ТО <хозяин; имя, Петя; фамилия, Иванов>.

Факт считается истинным, если такой кортеж имеется в рабочей памяти; если нет, то факт считается либо ложным, либо неопределенным.

• Использование *логических переменных*

Пример правила:

ЕСЛИ <жарко> ТО <открыть окно>.

В отличие от образцов текстов логические переменные должны быть описаны заранее, например, в виде словаря.

• Использование *фактовых переменных*

Пример правила:

ЕСЛИ <температура > 0 & вещество = «вода»>

ТО <агрегатное состояние = («жидкое» ∨ «газообразное»)>.

Факты задаются как выражения сравнения фактовой переменной с другой фактовой переменной или со значением. Фактовые переменные могут иметь любой тип. Тип каждой переменной и множество допустимых значений должны быть заранее определены. В одной продукционной системе факты должны быть представлены каким-либо одним способом.

4.1.2. Логический вывод в продукционной системе

Логический вывод в продукционной системе основывается на взаимодействии следующих элементов: элементарных фактов, описывающих текущее состояние предметной области, продукционных правил и механизма интерпретации правил. Исходные факты задаются в рабочей памяти, формат фактов согласуется с форматом правил. Совокупность продукционных правил является базой знаний. Интерпретация правил представляет собой механизм вывода, реализуемый машиной вывода.

В продукционной системе механизм логического вывода, как правило, реализуется по прямой или обратной цепочке.

Прямая цепочка вывода

Прямая цепочка вывода осуществляется *от ситуации к цели*.

Предварительно в рабочую память помещаются исходные факты, и далее выполняется пошаговая процедура вывода, при этом на каждом шаге выполняются следующие операции:

1. Сопоставление содержимого рабочей памяти с левыми частями правил из базы знаний, результатом операции сопоставления является множество всех применимых для этой задачи правил (таких, у которых левая часть принимает значение «истина»). Это множество правил называется конфликтным набором.

2. Разрешение конфликта. Осуществляется выбор из множества применимых правил одного правила по каким-либо критериям.

3. Применение выбранного правила.

П р и м е р

Пусть в базе знаний имеется система правил:

Правило 1: $F \,\&\, B \rightarrow Z$

Правило 2: $C \,\&\, D \rightarrow F$

Правило 3: $A \rightarrow D$

Правило 4: $F \,\&\, B \rightarrow F$

Факты A, G, C, E, H, B являются истинными. Необходимо доказать истинность какого-либо целевого факта, представляющего решение задачи. Таких фактов может быть несколько. Заранее целевой факт может быть не известен. Однако известно, какие из фактов имеют статус целевых. Предположим, что в нашем примере целевым является факт Z.

Рассмотрим решение задачи по шагам логического вывода.

Известные факты помещаются в рабочую память. Содержимое рабочей памяти к началу логического вывода: $РП=\{A, G, C, E, H, B\}$.

Ш а г 1

В результате операции сопоставления правил и фактов находятся применимые правила и помещаются в конфликтный набор: $КН = \{$*Правило 3*, *Правило 4*$\}$.

В результате операции разрешения конфликта выбирается *Правило 3*. Используется критерий выбора правила с наиболее общим условием.

Применяется *Правило 3*. В результате операции содержимое рабочей памяти изменяется: $РП = \{A, G, C, E, H, B, D\}$. В рабочую память добавлен факт D как результат применения *Правила 3*.

Ш а г 2

В результате операции сопоставления правил и фактов находятся применимые правила и помещаются в конфликтный набор: *КН*={*Правило 2*, *Правило 4*}. *Правило 3* уже применено, поэтому в конфликтный набор не помещается.

В результате операции разрешения конфликта выбирается *Правило 2*. Критерий – первое попавшееся правило из конфликтного набора.

Применяется *Правило 2*. В результате операции применения правила содержимое рабочей памяти изменяется: *РП* = {*A*, *G*, *C*, *E*, *H*, *B*, *D*, *F*}.

Ш а г 3

В результате операции сопоставления правил и фактов находятся применимые правила и помещаются в конфликтный набор: *КН* = {*Правило 1*, *Правило 4*}. Примененные правила в набор не помещаются.

В результате операции разрешения конфликта выбирается *Правило 1*. Критерий – первое попавшееся правило из конфликтного набора.

Применяется *Правило 1*. В результате операции применения правила содержимое рабочей памяти изменяется: *РП* = {*A*, *G*, *C*, *E*, *H*, *B*, *D*, *F*, *Z*}. Как только целевой факт *Z* попадает в рабочую память, подтверждается его истинность. Задача решена.

Обратная цепочка вывода

Обратная цепочка вывода осуществляется *от цели к ситуации*.

Предварительно в рабочую память вместе с исходными фактами помещается факт, представляющий целевую гипотезу. Этот факт помечается как требующий подтверждения. Цель вывода – подтвердить или опровергнуть целевую гипотезу. На каждом шаге обратной цепочки вывода выполняются следующие операции:

1. Сопоставление содержимого рабочей памяти с правыми частями правил из базы знаний. Правила, правые части которых потенциально могли бы подтвердить какие-либо неподтвержденные факты, помещаются в конфликтный набор.

2. Разрешение конфликта. Осуществляется выбор из множества применимых правил одного правила по каким-либо критериям.

3. Применение выбранного правила:

- если выбранное правило применимо, т. е. если левая часть этого правила истинна, то это правило применяется обычным образом, в результате подтверждается факт, который стоит в правой части этого правила;
- если правило не применимо, т. е. если в левой части есть факты, которые на данный момент не известны, то эти факты помещаются в рабочую память и помечаются как требующие подтверждения.

П р и м е р

Рассмотрим базу знаний из предыдущего примера. Факты A, G, C, E, H, B являются истинными. Необходимо подтвердить истинность факта Z.

Известные факты и факт, требующий подтверждения, помещаются в рабочую память. Содержимое рабочей памяти: *РП* = {*A*, *G*, *C*, *E*, *H*, *B*, *Z*-?}.

Ш а г 1

В результате операции сопоставления правил и фактов в конфликтный набор помещается единственное правило, которое может подтвердить неизвестный факт *Z*: *РП* = {*Правило 1*}.

В результате операции разрешения конфликта выбирается единственное *Правило 1*.

Применяется *Правило 1*. В результате операции попытки применения правила в рабочую память помещается факт *F* – как факт, требующий подтверждения. Содержимое рабочей памяти изменяется: *РП* = {*A*, *G*, *C*, *E*, *H*, *B*, *Z*-?, *F*-?}.

Ш а г 2

В результате сопоставления правил и фактов формируется конфликтный набор *КН* = { *Правило 1*, *Правило 2*, *Правило 4*}.

В результате разрешения конфликта выбирается *Правило 4*. Использован критерий – правило применимо.

Применяется *Правило 4*. В результате применения правила содержимое рабочей памяти изменяется: *РП* = {*A*, *G*, *C*, *E*, *H*, *B*, *Z*-?, *F*}.

Ш а г 3

В результате сопоставления правил и фактов формируется конфликтный набор *КН* = {*Правило 1*};

Разрешение конфликта – выбирается единственное *Правило 1*;

Применяется *Правило 1*. В результате операции применения правила содержимое рабочей памяти изменяется: *РП* = {*A*, *G*, *C*, *E*, *H*, *B*, *Z*, *F*}.

Таким образом, истинность факта *Z* подтверждена. Задача решена.

Тип цепочки логического вывода, как правило, определяется следующими условиями: если целевая гипотеза заранее не известна или исходных фактов не так много, то выбирается прямая цепочка; если целевая гипотеза известна, то выбирается обратная цепочка как более направленная; если ни прямая, ни обратная цепочка непригодны, то выбирается смешанный тип.

Принципы управления правилами

Несмотря на «прозрачность» механизма вывода, в продукционных системах возникает проблема выбора правил из множества применимых. Используется несколько эвристических принципов организации продукционных правил в базе знаний:

1. *Принцип стопки книг*. Наиболее часто используемые продукции актуализируются первыми. Принцип целесообразно применять, если продукции относительно независимы друг от друга.

2. *Принцип наиболее длинного условия.* Первой выбирается продукция, у которой наиболее «длинное» условие. Принцип целесообразен, когда продукции хорошо структурированы по отношению «частное-общее».

3. *Принцип метапродукций.* В систему продукций вводятся специальные правила – метапродукции, направленные на разрешение ситуаций в условиях конфликтоного набора правил.

4. *Принцип приоритетного выбора.* Для продукций устанавливаются статические или динамические приоритеты. Статические приоритеты формируются априори, чаще всего на основе экспертных оценок. Динамические приоритеты вырабатываются в процессе функционирования системы продукций (например, учитывается время нахождения в конфликтном наборе). Выбор способа управления правилами определяется условиями решаемой задачи и имеющимися ресурсами.

Достоинствами продукционного подхода к представлению знаний являются естественность структуры базы знаний, простота модификации, однородность базы знаний (правила имеют единую синтаксическую форму), ясность механизма логического вывода и простота объяснения решений (трассировка выполненных правил). К недостаткам можно отнести неясность взаимных отношений правил, сложность оценки целостного образа знаний при большом количестве правил, низкую эффективность вывода и большое количество операций переборного типа. В связи с возникающими ограничениями актуальными становятся задачи проверки базы знаний на полноту и непротиворечивость (при одних и тех же исходных фактах система не должна выдавать противоречивые решения), отсечение избыточности, возникающей при сопоставлении и разрешении конфликтов, а также определения хорошего критерия разрешения конфликтов, обеспечивающего более целенаправленный вывод. Одним из решений, направленных на повышение эффективности логического вывода, является структурирование базы правил.

4.1.3. Структурирование базы правил

Структурирование в продукционной системе представляет собой процедуру упорядочения и определения отношений между правилами, в результате которой строится некоторая структурная модель базы правил. Рассмотрим основные типы структурных моделей.

И/ИЛИ граф

Структура *И/ИЛИ*-графа представляет собой отношение информационной зависимости правил. Построение *И/ИЛИ*-графа осуществляется сле-

дующим образом. Для правил специального вида, которые имеют в левой части только операцию конъюнкции: *S: A & B & C → D*, строится модель в виде подграфа, центральная вершина (*И – вершина*) помечается именем правила, остальные – именами фактов (рис. 4.1).

Такой подграф строится для каждого правила, затем подграфы собираются в один *И/ИЛИ*-граф путём отождествления одноименных вершин. В полученном графе вершины, поименованные фактами, называются *ИЛИ-вершинами*.

И/ИЛИ-граф может использоваться для повышения эффективности логического вывода следующим образом:

- с помощью отношения информационной зависимости выполнять отсечение при сопоставлении фактов и правил;
- при разрешении конфликта можно производить отсечение и выбирать те правила, которые по уровню ближе к целевой гипотезе.

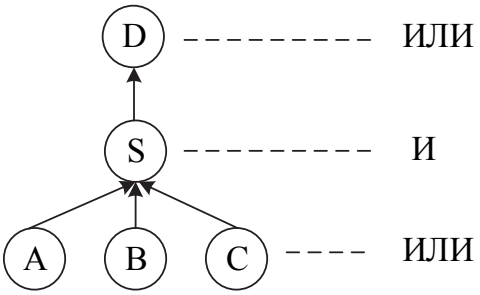


Рис. 4.1. Представление правила в виде подграфа

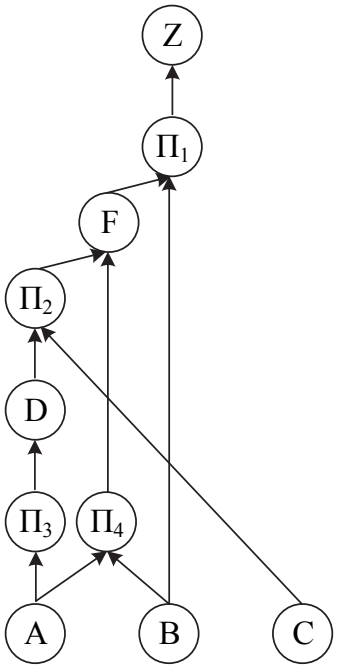


Рис. 4.2. И/ИЛИ-граф

П р и м е р

Необходимо структурировать в виде И/ИЛИ-графа базу правил:

П1: F&B → Z;

П2: C&D → F;

П3: A → D;

П4: C&E → Z;

Решением является И/ИЛИ-граф, изображенный на рис. 4.2.

Параллельно-последовательная структура

Построение параллельно-последовательной структуры основывается на отношении информационной зависимости, информационной независимости и альтернативности правил.

Правила называются *информационно зависимыми*, если для применения одного правила необходимо предварительное применение другого, иначе правила *информационно независимые*.

Правила называются *альтернативными*, если их условия логически противоречивы. Альтернативные правила не могут быть одновременно применимы.

Зависимые правила можно представить в виде последовательной цепочки, т. е. они будут выполняться последовательно (рис. 4.3).

Независимые правила можно представить в виде параллельной структуры, которая интерпретируется как параллельное выполнение правил (рис. 4.4).

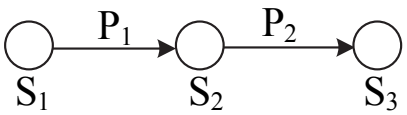


Рис. 4.3. Последовательная цепочка зависимых правил:
P_i – правила; S_j – состояния задачи

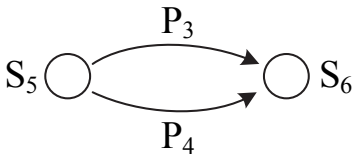


Рис. 4.4. Параллельная структура независимых правил

Альтернативные правила можно представить в виде параллельной структуры со специальными вершинами – триггерами, которая интерпретируется как параллельное выполнение правил (рис. 4.5).

На рис. 4.6 представлен пример структурирования базы правил.

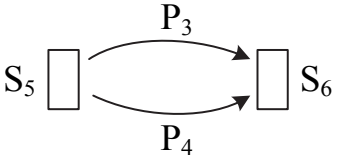


Рис. 4.5. Параллельная структура альтернативных правил

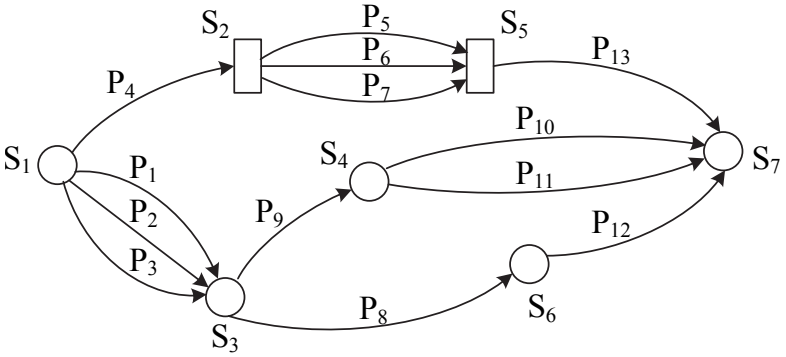


Рис. 4.6. Параллельно-последовательное структурирование базы правил

Логический вывод в параллельно-последовательной структуре выполняется как процедура поиска пути от начального состояния к целевому.

Несмотря на возможные механизмы повышения эффективности логического вывода в продукционной системе, при большом числе продукций сложной становится проверка непротиворечивости системы продукций. Считается, что если в системе число продукций достигает тысячи, то мало шансов, что система во всех случаях будет правильно функционировать. Поэтому число продукций, с которыми, как правило, работают современные интеллектуальные системы, не превышает тысячи. Тем не менее продукционная модель является наиболее используемой в системах искусственного интеллекта. В различных вариантах она также используется в композиции с другими моделями представления знаний (семантические сети, фреймы).

4.2. Семантические сети

4.2.1. Понятие семантической сети

Семантическая сеть применяется для описания метода представления знаний, основанного на сетевой структуре. *Семантическая сеть* представляет собой ориентированный граф, вершины которого отождествляются с сущностями (объектами) предметной области, а дуги – с отношениями между ними. Семантическая сеть является удобным способом графического описания объектов предметной области, при этом под объектом может пониматься процесс, состояние, какая-либо конкретная или обобщенная (класс) сущность.

Две вершины и связь между ними определяют элементарный факт. Примеры элементарных фактов представлены на рис. 4.7.

Рис. 4.7. Представление элементарных фактов
в семантической сети:
слева – факт «Ваня – мальчик»;
справа – факт «Ваня любит Таню»

Более сложные факты представляются более сложными графами. Такие графы включают дополнительную информацию, для того чтобы более полно раскрыть смысл описываемой ситуации. Семантическая сеть в целом (или ее целостный фрагмент) представляет собой совокупность взаимосвязанных фактов.

В конкретной предметной области могут быть различные типы объектов и множество типов связей между ними. Выделяют следующие основные типы объектов и связей.

Типы объектов в семантической сети:

• *обобщённый объект* (класс объектов) – это некоторое известное, широко используемое в моделируемой области понятие, например: вещество, орудие труда, профессия, планета;

• *конкретный объект* (экземпляр класса) – это единичная сущность, экземпляр из класса объектов, например: сульфат натрия, молоток, преподаватель;

• *агрегатный объект* – это объект, состоящий из других объектов, явлений, процессов. Агрегатный объект может быть обобщенным и конкретным.

Понятия обобщённого и конкретного объектов относительны, они зависят от рассматриваемой задачи. Обобщённый объект не может быть частью конкретного объекта.

Типы связей в семантической сети:

• *родовидовая* (*is kind of*). Вид наследует все свойства родового понятия, родовое понятие не охватывает всех свойств видового. Используемое обозначение – *S* (подмножество). Эту связь можно рассматривать как отношение подкласса к классу;

• *является представителем* (*is a*). Связь устанавливается между обобщенным и конкретным объектом. Один и тот же конкретный объект может быть представителем нескольких обобщенных объектов. Используемое обозначение – *e* (элемент);

• *является частью* (*is part of*). Связь устанавливается между агрегатным объектом и каким-либо другим объектом, который является его частью. Используемое обозначение – *p* (часть);

• *атрибутивная* (*has-property*). Например: *имеет значение*, *имеет свойство* и т. д. Используемое обозначение – *h-p* (свойство);

• *функциональная*. Выражается глаголом. Например: *любит*, *владеет* и т. п.;

• *количественная*. Например: *больше*, *меньше* и т. п.;

• *пространственная*. Например: *далеко*, *близко* и т. п.;

• *временная*. Например: *раньше*, *позже* и др.

В экспертных системах, использующих рассматриваемый подход, база знаний представляется семантической сетью. Пополнение или модификация базы знаний происходит путём добавления или удаления вершин и ребер графа. Сложность модификации состоит в подборе подходящего фрагмента для замены или подходящей вершины для дополнения, так,

чтобы не нарушить общую структуру базы знаний. Другая сложность – трудность описания семантической сетью обширной предметной области.

Рассмотрим пример построения семантической сети, представляющей факт «*Иван Иванович владеет автомобилем* “*Волга*”», где «Волга» – это конкретный автомобиль, которым владеет Иван Иванович и который является экземпляром автомобилей ВАЗ (рис. 4.8).

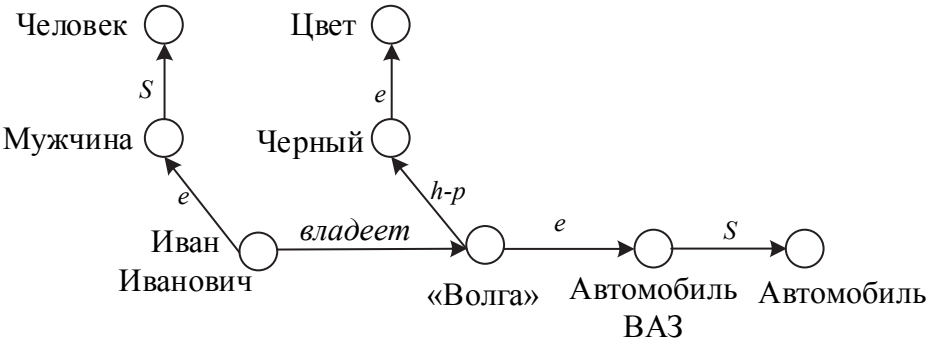


Рис. 4.8. Семантическая сеть факта
«Иван Иванович владеет автомобилем “Волга”»

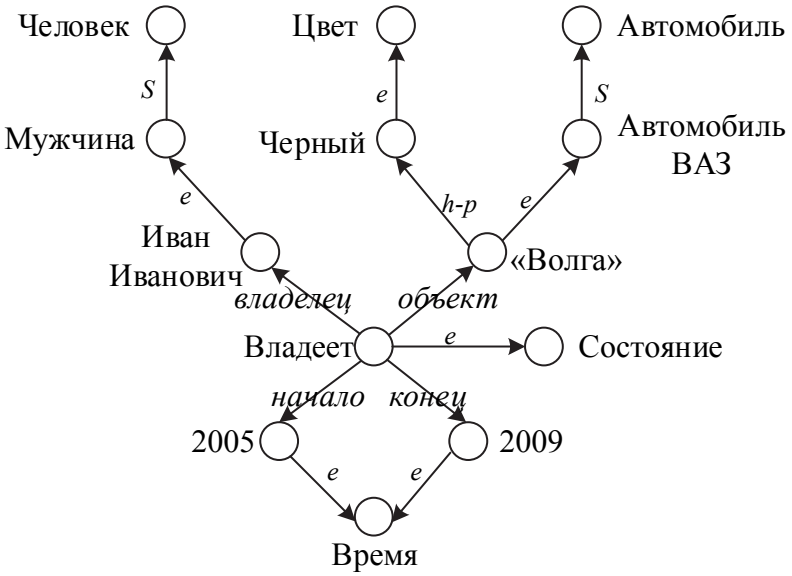


Рис. 4.9. Семантическая сеть факта
«Иван Иванович владел автомобилем “Волга” с 2005 по 2009 г.»

Если дополнить представленную на рис. 4.8 семантическую сеть информацией, что *Иван Иванович владел автомобилем “Волга” с 2005 по 2009 г.*, тогда вершинами необходимо представить не только объекты, но и ситуации, и действия (рис. 4.9). Для вершины ситуации «*Владеть*» определено несколько связей. Такая вершина называется *падежным фреймом* (*case frame*).

4.2.2. Логический вывод в семантической сети

Для организации логического вывода в экспертной системе, использующей семантическую сеть, применяются следующие механизмы.

1. *Механизм наследования со свойством транзитивности*. Данный способ позволяет сжимать базу знаний (рис. 4.10). Представленная пунктиром дуга в явном виде в базе знаний не задаётся.

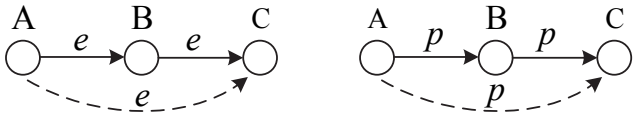


Рис. 4.10. Использование транзитивности
в семантических сетях

2. *Вывод, основанный на базовых операциях.* К базовым операциям относятся: поиск вершины или ребра по имени; переход от одной вершины к другой по связям. Цель поисков и переходов – найти вершину с заданными свойствами или найти свойства заданной вершины.

3. *Вывод, основанный на сопоставлении с образцом.* Данный механизм включает следующую последовательность шагов:

• строится запрос в виде фрагмента семантической сети, при этом некоторые вершины могут быть незаполненными (рис. 4.11);

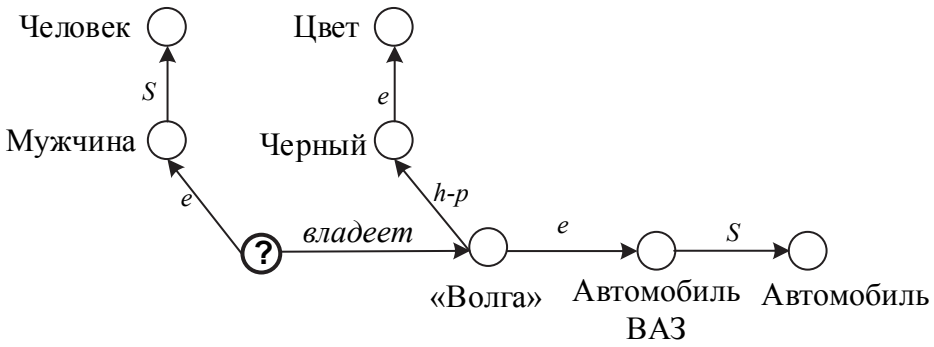


Рис. 4.11. Семантическая сеть запроса

• выполняется сопоставление семантических сетей запроса и базы знаний, в результате чего в семантической сети базы знаний определяется фрагмент, изоморфный фрагменту сети запроса;

• заполняются вершины семантической сети запроса, в которых отсутствует информация, в соответствии со значениями вершин семантической сети базы знаний.

4. *Вывод, основанный на использовании присоединенных процедур.* Процедуры активизируются при прохождении по соответствующей дуге или через соответствующую вершину.

Преимуществами семантического подхода являются естественность представления знаний, простота и прозрачность описания, однако с увеличением размеров сети существенно увеличивается время поиска, теряется наглядность. К недостаткам подхода можно отнести неоднозначность описания, отсутствие формального аппарата установления противоречивости описания, сложность разработки сети для обширных предметных областей и сложность технической реализации вывода. Основное применение семантические сети находят в системах обработки естественных языков, а также распознавания образов, в которых семантические сети используются для хранения знаний о структуре, форме и свойствах физических объектов.

4.3. Фреймы

4.3.1. Понятие фрейма и фреймовой системы

Фреймовая модель представления знаний – форма представления знаний, основанная на объектно-ориентированном подходе. Впервые понятие фрейма (*англ*. *frame – каркас*, *рамка, структура*) было введено М. Минским в 1975 г. В основу понятия положены представления о восприятии человеком внешнего мира в форме целостных фрагментов. Таким фрагментом может быть объект внешнего мира с его наиболее характерными свойствами, ситуация, процесс и т. п. По Минскому *фрейм* – это структура данных, содержащая минимально необходимую информацию для представления класса объектов (явлений или процессов), которая однозначно определяет эти объекты. *Фрейм* – структура данных для описания концептуальных объектов (понятий, явлений, ситуаций).

Спецификация фрейма включает:

- структуру фрейма;
- описание связей с другими фреймами.

Графически структуру фрейма можно представить в виде таблицы (рис. 4.12). Информация, относящаяся к фрейму, содержится в слотах. *Слот* (*slot* – место для вставки данных) – это единица информации фрейма, элемент данных для фиксации знаний о свойстве объекта. Спецификация слота включает:

- имя слота;
- указатель типа атрибута слота;
- указатель наследования;
- значение слота «по умолчанию»;
- присоединенные процедуры и другую информацию, относящуюся к данному свойству объекта.

Различают два типа фреймов: фрейм-описание и фрейм-экземпляр. *Фрейм-описание* определяет общее понятие – структуру для заполнения информацией. Фрейм, слоты которого заполнены, является *фреймом-экземпляром*. В экспертной системе, основанной на фреймах, база знаний представляется совокупностью фреймов-описаний, рабочая память – совокупностью фреймов-экземпляров.

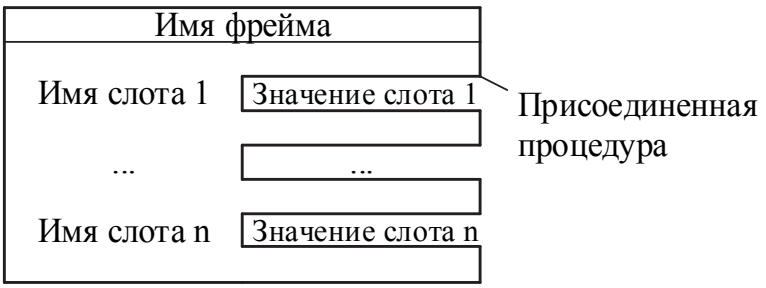


Рис. 4.12. Графическое представление фрейма в виде таблицы

Фреймы, как правило, взаимосвязаны и образуют единую фреймовую систему. *Фреймовая система* – структура, узлами которой являются фреймы. Чаще всего фреймовые системы имеют иерархическую структуру, которая характеризует степень абстракции понятия. В иерархической структуре могут использоваться, например, такие отношения, как класс-подкласс (*is kind of*); абстрактное-конкретное (*is a*); целое-часть (*is part of*).

Особенностью фреймовой системы является возможность наследования значений слотов и использование значений по умолчанию.

Рассмотрим описание структуры данных фрейма на примере фреймовой модели для класса объектов «Аудитория» (рис. 4.13).

Имя фрейма. Фрейм должен иметь имя, единственное в данной фреймовой системе. В рассматриваемом примере имена фреймов: *Аудитория*, *Номер*, *Вид аудитории*, *Занятие*, *Время* и др.

Каждый фрейм состоит из некоторого числа слотов, причем значения одних слотов задаются пользователем, значения других определяются системой в процессе логического вывода.

Имя слота. Слот должен иметь уникальное имя во фрейме, к которому он принадлежит. В рассматриваемом примере имена слотов: *Количество мест*, *Кафедра*, *Номер корпуса*, *Преподаватель, Полугодие* и др.

Тип данных. Указывается, что слот имеет некоторое значение либо служит указателем другого фрейма. Могут использоваться следующие типы данных: frame (указатель); integer (целый); real (действительный); bool (булев); text (текст); list (значение из списка); table (таблица); expression (выражение) и т. п. В рассматриваемом примере тип frame имеют слоты

Номер, *Занятие*, *Время*; тип list (значение из списка) имеют слоты *Полугодие*, *Неделя*; тип integer имеют слоты *Номер аудитории*, *Номер корпуса*, *Количество мест*; тип текст – слоты *Кафедра*, *Предмет*, *Преподаватель*, *Группа* и др.

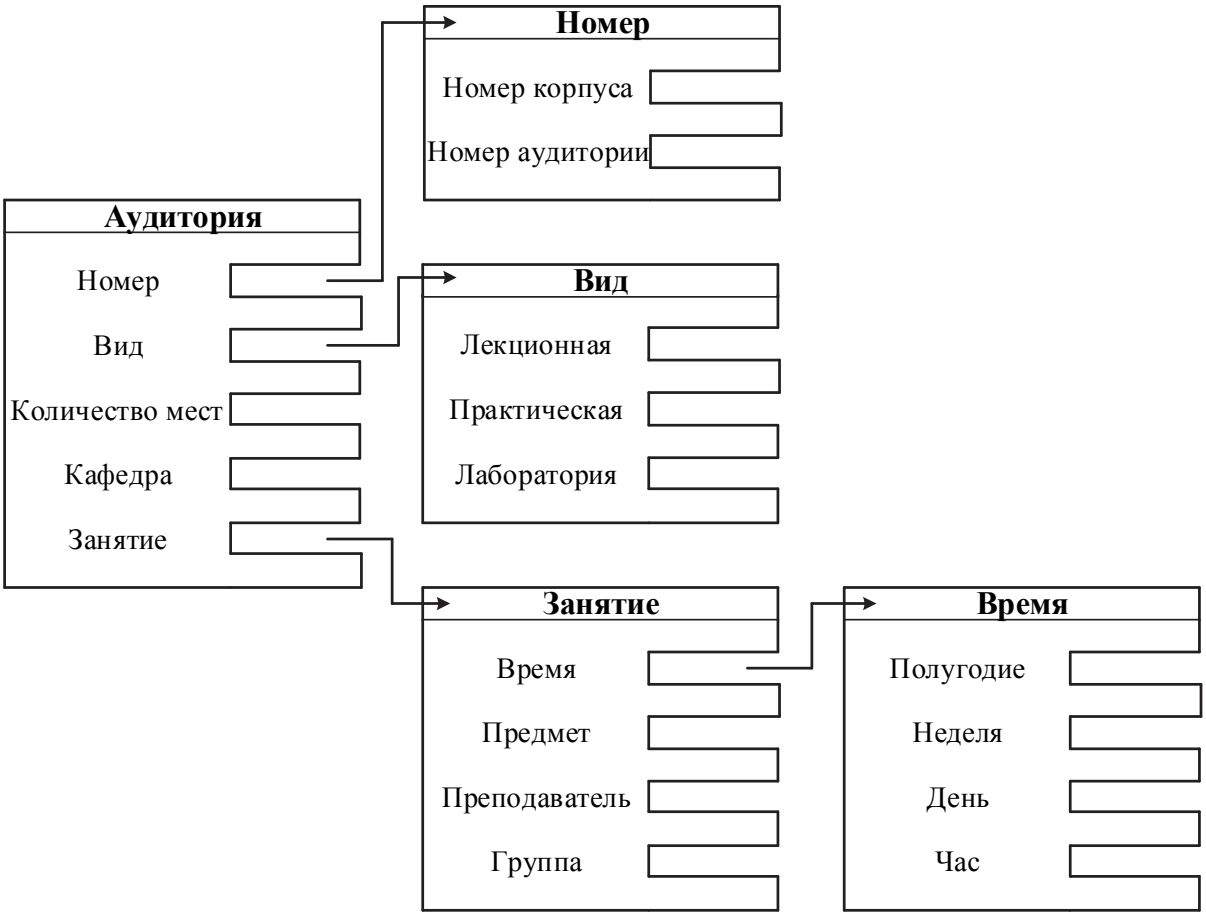


Рис. 4.13. Пример фреймовой модели для класса объектов «Аудитория»

Значение слота. Значение слота должно совпадать с указанным типом данных этого слота и соответствовать условию наследования.

Присоединённая процедура. С каждым слотом могут быть связаны процедуры, выполнение которых зависит от значения слота. Это *присоединённые процедуры*. Они обеспечивают процесс логического вывода.

Для стандартизации обработки присоединенных процедур используются процедуры специального типа – *демоны* (demon) – стандартные процедуры, которые автоматически запускаются при обращении к соответствующему слоту и выполнении некоторого условия. К таким процедурам относятся:

If needed (*если нужно*): процедура выполняется, когда запрашивается информация из слота, а он пуст;

If added (*если добавлено*): процедура выполняется, когда новая информация помещается в пустой слот;

If changed (*если изменено*): процедура выполняется при изменении значения слота;

If removed (*если удалено*): процедура выполняется при удалении информации из слота.

Для рассматриваемого примера можно задать процедуры:

If needed: если для слота *Номер* во фрейме *Аудитория* значение не указано, то выводится сообщение, что его нужно указать;

If added: при занесении значения в слот *Номер корпуса* проверяется, существует ли такой корпус; если нет, выводится сообщение об ошибке;

If removed: вывод подтверждения при удалении значения.

4.3.2. Логический вывод во фреймовой системе

Во фреймовом подходе к представлению знаний отсутствует специальный механизм управления логическим выводом, и для его организации применяются следующие способы:

- эстафета присоединённых процедур;
- наследование;
- использование значений по умолчанию.

Рассмотрим реализацию логического вывода на примере модели системы, помогающей корпорации в подготовке отчетов (рис. 4.14).

Принцип работы представленной модели следующий. Если системе задан запрос: «Мне нужен отчет о выполнении проекта ”Новые технологии”», то интерфейсная программа анализирует его, выделяя ключевые слова, и вносит *Новые технологии* в слот *Название отчета* следующего пустого узла *Отчет о выполнении проекта* (узел *№ 15*). Предполагается, что программный интерфейс позволяет общаться с системой на языке, близком к естественному.

Далее автоматически выполняется последовательность действий:

1. Процедура *If added*, связанная со слотом *Название отчета*, осуществляет поиск руководителя этого проекта. Допустим, что его фамилия Иванов. Процедура вписывает фамилию в слот *Автор* узла *№ 15*. Если руководитель этого отчета не будет найден, в слот *Автор* будет наследовано значение класса, а именно текст Руководитель.

2. Выполняется процедура *If added*, связанная со слотом *Автор*, так как в слот только что было вписано новое значение. Эта процедура начинает составлять сообщение, чтобы отправить его Иванову, но обнаруживает, что значение объема отсутствует. Слот *Объем* не связан с процедурами, однако выше узла *№ 15* существует узел общей концепции отчета о вы-

полнении проекта, содержащий значение объема. Процедура путём наследования использует значение объема – 3 страницы.

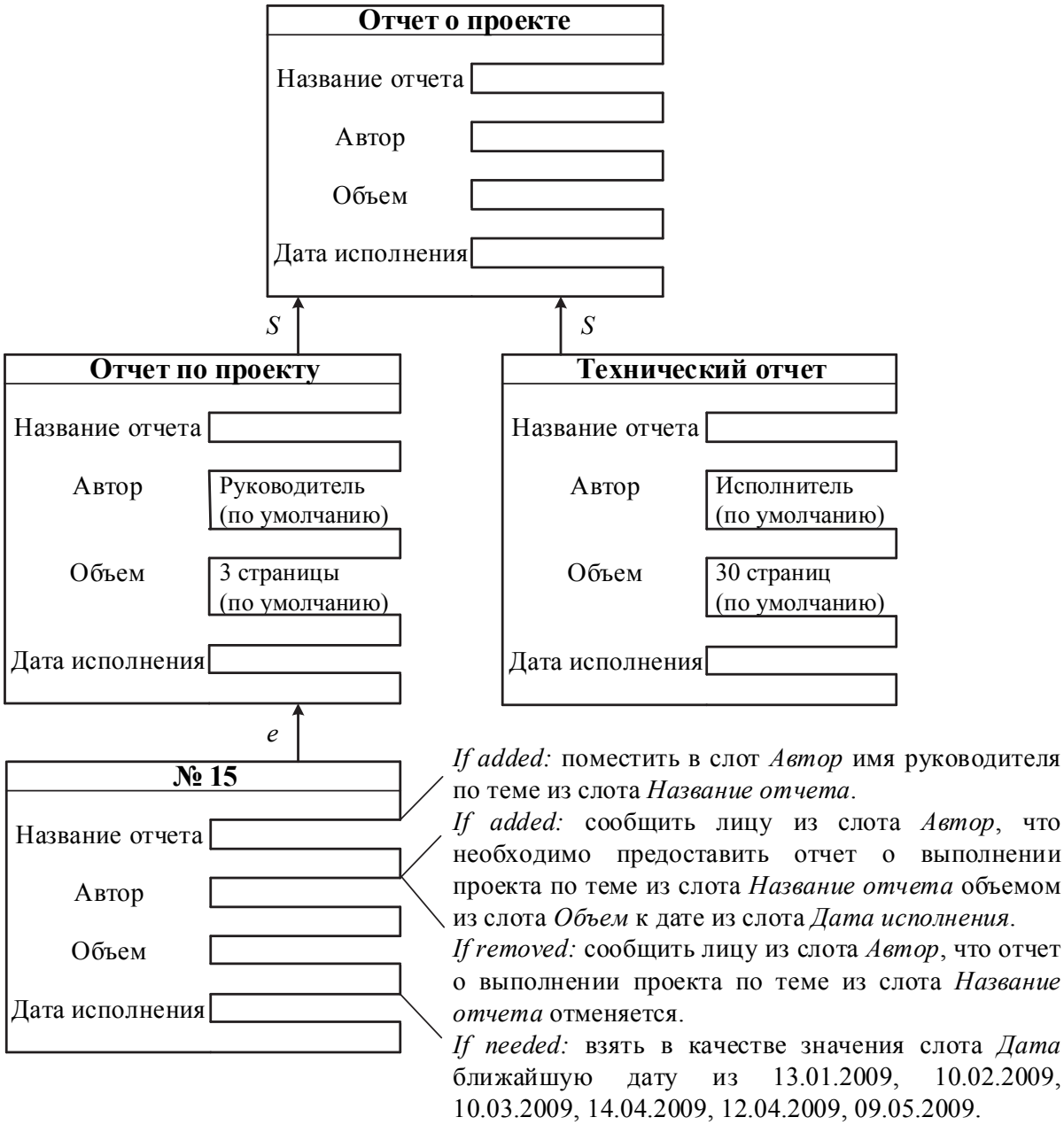


Рис. 4.14. Фреймовая система для организации отчетности

3. Затем процедура *If added*, связанная со слотом *Автор*, найдет ещё одно значение, которое нужно включить в сообщение: дата. Происходит активация процедуры *If needed*, связанной с этим слотом. Анализируя те-

кущую дату (например, 01.02.2009), *If needed* выберет ближайшую к ней (например, 10.02.2009) и впишет её в слот *Дата*.

4. Получив всю недостающую информацию, процедура *If added*, связанная со слотом *Автор*, составляет следующее сообщение: «*Господин Иванов, пожалуйста, подготовьте отчет по проекту "Новые технологии" к 10.02.2009 объемом 3 страницы*».

Если в какой-то момент имя Иванов будет удалено из слота *Автор*, то сработает присоединённая процедура *If removed*, и система автоматически отправит ему сообщение, что его отчет о выполнении проекта по теме из слота *Название отчета* отменяется.

К достоинствам фреймового подхода следует отнести простоту и естественность представления знаний, однородность и высокую степень структуризации знаний, способность отражать концептуальную основу организации памяти человека, совмещать декларативные и процедурные знания. К недостаткам подхода можно отнести сложность отладки системы и управления выводом. Каждый фрейм представляет собой достаточно сложный фрагмент знаний. Удаление или добавление нового фрейма – достаточно «болезненная» для фреймовой системы процедура, так как должна предусмотреть удаление всех составляющих элементов, которые могут быть составными частями других фреймов.

Фреймовый подход позволяет комбинировать различные модели представления знаний, образуя гибридные модели, такие как семантико-фреймовая, продукционно-фреймовая и семантико-продукционно-фреймовая. Гибридные модели позволяют объединять достоинства классических моделей и компенсировать их недостатки. Например, недостатком продукционной модели является низкая эффективность вывода при больших базах знаний. Этот недостаток хорошо компенсируется фреймовой моделью, когда декларативные знания представляются в виде фреймов, а процедурные знания – в виде слотов обращения к присоединенным процедурам. В итоге формируется иерархия, вершинами которой являются фреймы, моделирующие левую часть правил, а правая часть правил моделируется слотами – присоединенными процедурами. Если знания имеют слабую иерархичность или вложенность, то может использоваться семантико-фреймовая модель, в которой вершины задаются фреймами, а связи между ними описываются семантической сетью. В продукционно-фреймовой модели база знаний состоит из продукций, задающих причинно-следственные отношения между простыми и сложными объектами предметной области. При этом в качестве сложных объектов выступают фреймы. На основе объектов-условий определяется значение выделенного объекта-цели, представляющего собой одну из гипотез решения. Вывод истинной (наилучшей с точки

зрения имеющейся базы знаний) гипотезы-решения может осуществляться как по прямой, так и по обратной цепочке.

4.4. Формальные логические модели

4.4.1. Понятие формальной системы

Логический подход основан на представлении знаний и фактов формулами формальной системы. *Формальная система* представляет собой совокупность абстрактных объектов – формул, в которых представлены правила оперирования множеством символов в синтаксической трактовке без учета смыслового содержания. Например, к формальным системам относятся логика высказываний, логика предикатов, формальная арифметика, нечёткая логика, модальная логика и др. В инженерии знаний чаще всего используется логика предикатов первого порядка.

Формальная система F определена, если задано ее исчисление, которое включает:

- конечный алфавит A (множество символов);
- определение процедуры построения формул V;
- выделенное множество формул W, называемых аксиомами;
- конечное множество правил вывода R, позволяющее получать из некоторого конечного множества формул другое множество формул.

Таким образом, формальная система F определяется совокупностью элементов исчисления: $F=<A, V, W, R>$.

Определим элементы исчисления:

Алфавит:

Константы: a, b, c, d, …;

Переменные: t, u, v, w, x, y, z, …;

Предикаты: A, B, C, D, …

Знаки логических операций:

& ($\wedge$) – конъюнкция, логическое И;

$\vee$ – дизъюнкция, логическое ИЛИ;

$\neg$ – логическое отрицание;

$\rightarrow$ – импликация (логическое следование «если – то»);

$\forall$ – квантор всеобщности;

$\exists$ – квантор существования;

true, *false* – значения истинности: «истина» и «ложь» соответственно.

Правила построения формул. Если F_1 и F_2 – логические формулы, тогда правильно построенными логическими формулами являются:

$$\neg F_1,\ F_1 \vee F_2,\ F_1\ \&\ F_2,\ F_1 \rightarrow F_2,\ (\forall x)F_1(x),\ (\exists x)F_1(x).$$

Все выражения, получаемые суперпозицией конечного числа формул, логические формулы. Например, фразу *«Все люди смертны»* можно записать в виде формулы: $(\forall x)[Человек(x) \rightarrow Смертен(x)]$.

Аксиомы – логические формулы, истинность которых не требует доказательства. Одну и ту же формальную систему можно задавать, используя разные системы аксиом. Например, для исчисления предикатов можно использовать одну из следующих систем аксиом.

Система аксиом 1:

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

$(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C)$;

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

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

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

$A \rightarrow (A \vee B)$;

$B \rightarrow (A \vee B)$;

$(A \rightarrow C) \rightarrow ((C \rightarrow C) \rightarrow ((A \vee B) \rightarrow C))$;

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

$\neg\neg A \rightarrow A$.

Система аксиом 2:

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

$(A \rightarrow (B \rightarrow)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$;

$(\neg A \rightarrow \neg B) \rightarrow ((\neg A \rightarrow B) \rightarrow A)$.

Приведённые системы аксиом равносильны в том смысле, что порождают одно и то же множество формул.

Правила вывода. Вывод представляет собой процедуру, которая из заданной группы формул выводит новую формулу. При этом группа заданных формальных выражений (посылок) и выведенное выражение (следствие или заключение) являются семантически эквивалентными, т. е. если посылка истинна, то и заключение истинно.

В логике предикатов в качестве правила вывода используется так называемое *правило отделения* (*modus ponens*). Из двух формул F_1 и $(F_1 \rightarrow F_2)$ выводится формула F_2. Для записи этого правила используется следующее обозначение:

$$\frac{F_1, (F_1 \rightarrow F_2)}{F_2} \quad \text{или} \quad F_1, (F_1 \rightarrow F_2) \vdash F_2.$$

Например, сумма внутренних углов многоугольника равна 180^o (F_1). Если сумма внутренних углов многоугольника равна 180^o, то многоугольник есть треугольник ($F_1 \rightarrow F_2$). Следовательно, дан треугольник (F_2).

Одну и ту же формальную систему можно задать разными исчислениями за счет вариации набора аксиом и правил вывода.

Переменные, ограниченные кванторами, называются *связанными*. Несвязанные переменные называются *свободными*. Действие кванторов формальной системы предикатов первого порядка распространяется только на предметные переменные, а не предикатные.

Представим в виде формул логики предикатов следующие выражения на естественном языке:

Все люди любят всех людей.

$(\forall x)(\forall y)$ *Любит* (x, y).

Существует человек, который любит всех людей.

$(\exists x)(\forall y)$ *Любит* (x, y).

Для каждого человека существует человек, который его любит.

$(\forall x)(\exists y)$ *Любит* (y, x).

Существует человек, который кого-нибудь любит.

$(\exists x)(\exists y)$ *Любит* (x, y).

Простое представление условия задачи на формальном языке позволяет ее решить и даёт возможность объяснить решение. В качестве примера рассмотрим следующую загадку.

П р и м е р ***«Загадка»***

Человек смотрит на портрет и говорит: «*У меня нет братьев и сестер, но отец этого человека – это сын моего отца*». Кто изображен на портрете?

В такой словесной постановке задача достаточно сложно воспринимается, и ее решение кажется неочевидным. Введем следующие обозначения: A – человек, изображенный на портрете; B – человек, который смотрит на портрет. Предикат *Сын* (y, x) истинен, если y является сыном x. Также определим функцию $y =$ *Отец* (x), возвращающую для x его отца y. Тогда истинное утверждение *«Отец этого человека – это сын моего отца»* запишется следующим образом: *Сын* (*Отец* (A), *Отец* (B)) – отец человека на портрете (A) приходится сыном человеку, смотрящему на портрет (B). Высказывание *«У меня нет братьев и сестер»* запишется так: $\forall y$ *Сын* $(y,$ *Отец* $(B)) \rightarrow y = B$ – любой сын отца B и является самим B.

Пусть $y =$ *Отец (A)*, тогда получаем *Сын* (*Отец* (A), *Отец* (B)) является истинным по условию 1. Из условия 2 получаем *Отец* $(A) = B$. В результате несложных операций получаем, что человек, который смотрит на портрет, приходится отцом человеку, изображенному на портрете, т. е. отец смотрит на сына.

Основным недостатком формальных систем считается их «закрытость», так как они не позволяют учитывать изменения предметной области: модификация и расширение всегда связаны с перестройкой всей системы. Формальные системы как модели представления знаний используются

в тех предметных областях, которые мало зависят от внешних факторов и не требуют вмешательства пользователя в процесс логического вывода.

Определим важнейшие свойства формальных систем, о которых необходимо знать при построении интеллектуальных систем, основанных на логическом подходе.

Разрешимость. Формальная система называется *разрешимой*, если существует алгоритм, позволяющий отличить теорему от нетеоремы. Логический вывод в формальных системах можно представить в виде последовательности формул (F_1,…, F_n, G_1, …, G_m, H), в которой (F_1, …, F_n) – исходные формулы; (G_1, …, G_m) – формулы, которые являются аксиомами либо непосредственно выводимы из предыдущих формул; H – гипотеза. Если исходные формулы – аксиомы, то процесс логического вывода называется *доказательством*, а выражение (F_1, ..., F_n) $\vdash H$ *– теоремой*. Исчисление предикатов относится к полуразрешимой системе, т. е. имеется алгоритм, который позволяет доказать теорему в том случае, когда она действительно является теоремой.

Непротиворечивость. Формальная система называется *непротиворечивой*, если в ней нельзя одновременно доказать H и $\neg H$. Логика предикатов обладает свойством непротиворечивости.

Полнота. В полной формальной системе любая теорема представляет собой тождественно истинное утверждение во всех интерпретациях.

Интерпретация – распространение исходных положений формальной системы на реальный мир. Она придает смысл каждому символу формальной системы и устанавливает взаимно однозначное соответствие между символами и реальными объектами. Теоремы после интерпретации становятся утверждениями.

По организации логического вывода интеллектуальные системы, основанные на логике предикатов, делятся на дедуктивные системы и системы, основанные на принципе резолюций.

4.4.2. Естественные дедуктивные системы

Интеллектуальные системы данного типа естественны в смысле самого логического исчисления: в качестве логического вывода интеллектуальной системы используется процедура логического вывода формальной системы. Множество знаний и фактов представляется в виде формул.

Пусть знания представляются множеством формул $\{Z_1, Z_2, \ldots, Z_m\}$, где Z_i – аксиомы или теоремы. $\{F_1, F_2, \ldots, F_n\}$ – совокупность формул, описывающих данные о задаче.

Чтобы показать справедливость некоторого факта (гипотезы) H, строится цепочка:

$$\underbrace{F_1, F_2, ..., F_n,\ Z_1, Z_2, ..., Z_m}_{\text{Исходные формулы}},\ \underbrace{G_1, ..., G_2, ..., G_k}_{\text{Промежуточные факты}},\ \underbrace{H}_{\text{Решение}}$$

Суть проблемы логического вывода состоит в том, чтобы доказать выводимость $F_1, F_2, \ldots, F_n \vdash H$.

Логический вывод в такой системе представляет собой процесс построения последовательности формул. Если цепочка вывода строится в естественном порядке, то она называется *прямой*. При обратном направлении вывода образуется *обратная* цепочка. Применяется также смешанная цепочка.

Проблемы, возникающие в естественных дедуктивных системах:

1. Проблема комбинаторного взрыва – лавинообразное нарастание промежуточных формул.

2. Проблема унификации формул, т. е. приведения их к одному виду: только унифицировав формулы, можно определить применимость заданного правила вывода или теоремы к заданной формуле.

Унификация формул

Чтобы унифицировать две формулы, т. е. привести их к идентичному виду, применяются подстановки. Переменные, которые стоят на одних и тех же позициях, заменяются на соответствующие подформулы. Для этого обе формулы просматриваются параллельно в соответствии с их структурой. Несовпадения ликвидируются за счёт подстановок. Унификация невозможна, если в разных формулах на одном и том же месте оказываются разные операции.

П р и м е р 1

Выполним унификацию формул $F_1 = (a + b)^2$ и $F_2 = (\sqrt{2} + c^3)^2$.

Представим F_1 и F_2 в виде дерева (рис. 4.15). Очевидные подстановки: a заменяется на $\sqrt{2}$ и b на c^3. Обозначим подстановки: $S(a,\sqrt{2})$ и $S(b, c^3)$ соответственно.

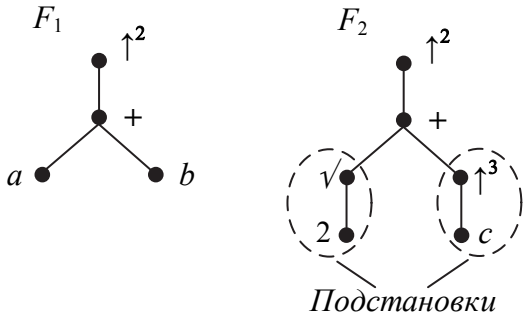


Рис. 4.15. Представление формул F_1 и F_2 в виде дерева

С помощью подстановок нельзя заменять константы, а также заменять переменную на формулу, зависящую от этой переменной. Алгоритм унификации можно использовать, чтобы автоматизировать процесс применения некоторой теоремы к некоторой формуле.

П р и м е р 2

Пусть имеются формула F и теорема T: $E \vdash H$.

Чтобы применить T к F, выполняется унификация E и F. Затем те же подстановки применяются к правой части теоремы. Результатом является применение T к F.

$F = (x \to (y \to x))$, T: $((P \to Q) \to R) \vdash (Q \to R)$.

Здесь $E = (P \to Q) \to R$, $H = (Q \to R)$.

Последовательно применяя подстановки, унифицируем F и E (рис. 4.16).

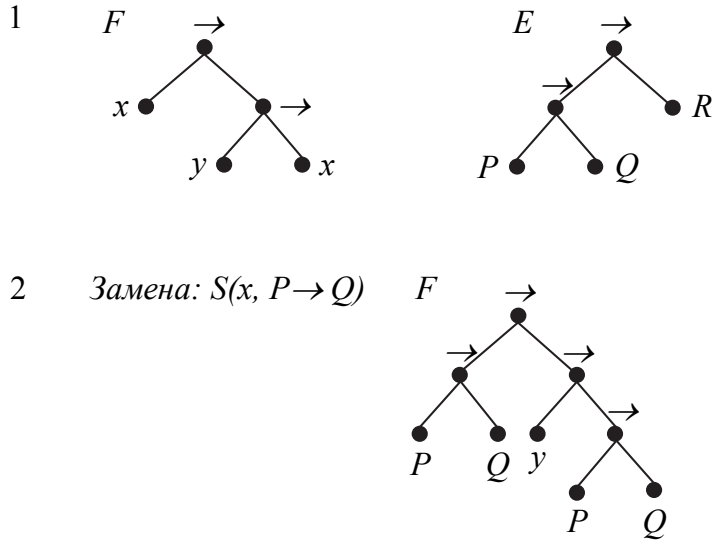


Рис. 4.16. Представление формул F и E в виде дерева

Проведя замену $S(R, y \to (P \to Q))$, получим окончательный результат $(x \to (y \to x)) \vdash (Q \to (y \to (P \to Q)))$, это и есть новая теорема. Следует иметь в виду, что при унификации формулы меняются, т. е. это нетождественное преобразование.

П р и м е р *«Геометрический тест»*

Используя алгоритм унификации, требуется среди рисунков K, L, M (рис. 4.17) найти такой рисунок X, который удовлетворял бы следующему условию: C так соотносится с X, как A соотносится с B.

Имеем теорему $A \to B$ и формулу C. Формализуем условие задачи. Обозначим отношение «*фигура X внутри фигуры Y*» как $X \subset Y$; «фигура X

вне фигуры Y» как $X \supset Y$; «фигура X расположена слева от фигуры Y» как $X \llcorner Y$. Опишем рисунки A, B, C в принятых обозначениях:

$$A = (\bullet\ \triangle\ \square) \,\&\, (\bullet \supset \triangle) \,\&\, (\square \subset \triangle) \,\&\, (\bullet \supset \square);$$

$$B = (\square\ \triangle) \,\&\, (\triangle \llcorner \square) \,\&\, (\square \supset \triangle);$$

$$C = (\bullet\ \bigcirc\ \triangle) \,\&\, (\bullet \supset \triangle) \,\&\, (\triangle \subset \bigcirc) \,\&\, (\bullet \supset \bigcirc).$$

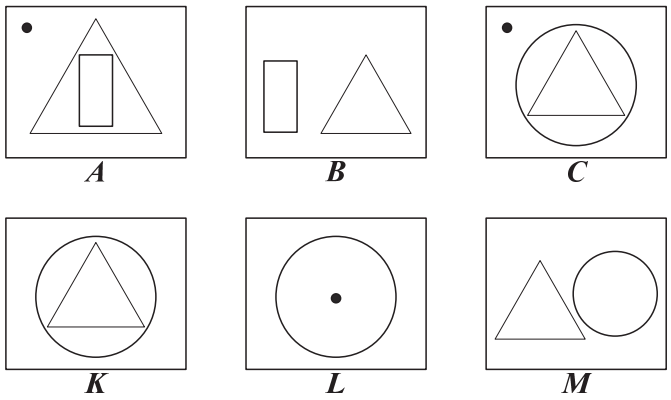


Рис. 4.17. Геометрический тест

Рисунки описаны с использованием констант. Для применения алгоритма унификации введем соответствующие переменные и перепишем формулы. Обозначим • – a; △ – b; □ – c; ○ – d. Теорема имеет вид

$$((a\ b\ c) \,\&\, (a \supset b) \,\&\, (c \subset b) \,\&\, (a \supset c)) \rightarrow ((c\ b) \,\&\, (c \llcorner b) \,\&\, (c \supset b)).$$

Для унификации A и C необходимо провести замены:

$$S(a, \bullet);\ S(b, \bigcirc);\ S(c, \triangle);\ S(d, \square).$$

Применим полученные подстановки к левой части теоремы. Получим $(\triangle\ \bigcirc) \,\&\, (\triangle \llcorner \bigcirc) \,\&\, (\triangle \supset \bigcirc)$.

Полученное описание соответствует рисунку M. Таким образом, решение найдено.

4.4.3. Системы, основанные на методе резолюций

В системах, основанных на методе резолюций, не используется напрямую логический вывод формальной системы. Вывод в них базируется на принципе доказательства «от противного».

Теорема Эрбрана

Чтобы доказать теорему

$$T: F_1, F_2, \ldots, F_n \vdash H$$

в исчислении предикатов, *достаточно* доказать логическую противоречивость (тождественную ложность) формулы

$$F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n \,\&\, \neg H.$$

Интеллектуальные системы, основанные на этом подходе, строятся следующим образом.

Знания представляются в виде набора фактов $\{F_1, F_2, \ldots, F_n\}$ и совокупности формул $\{Z_1, Z_2, \ldots, Z_m\}$. Чтобы показать справедливость гипотезы H, доказывается теорема

$$F_1, F_2, \ldots, F_n, Z_1, Z_2, \ldots, Z_m \vdash H.$$

Для этого достаточно доказать, что

$$F_1 \,\&\, F_2 \,\&, \ldots, \&\, F_n \,\&\, Z_1 \,\&\, Z_2 \,\&\, \ldots \,\&\, Z_m \,\&\, \neg H \equiv false.$$

Теорема о резолюции

Пусть имеется некоторая формула, представленная в конъюнктивной нормальной форме (КНФ)

$$F = C_1 \,\&\, C_2 \,\&, \ldots, \&\, (p \vee L_i) \,\&, \ldots, \&(\neg p \vee L_j) \,\&, \ldots, \&\, C_k$$

и некоторая формула

$$E = F \,\&\, (L_i \vee L_j).$$

Тогда, если E противоречива, то и F противоречива. Другими словами, для доказательства противоречивости F *достаточно* доказать противоречивость E.

Предположим, что E противоречива. Тогда либо F противоречива (и теорема доказана), либо $(L_i \vee L_j)$ противоречива. Покажем, что в последнем случае F также противоречива: $L_i \vee L_j \equiv false \Rightarrow L_i = false$ и $L_j = false$. Подставляя $L_i = false$ и $L_j = false$ в выражение F, путем равносильных преобразований получим:

$$F = C_1 \,\&\, C_2 \,\&, \ldots, \&\, p \,\&\, \neg p \,\&, \ldots, \&\, C_k \equiv false.$$

Теорема доказана.

Член $(L_i \vee L_j)$ называется *резольвентой*, а $C_i = (p \vee L_i)$, $C_j = (\neg p \vee L_j)$ – *родительскими предложениями.* Операция нахождения резольвенты называется *резолюцией.*

Следствие из теоремы: чтобы доказать противоречивость формулы, *достаточно* найти её пустую резольвенту.

П р и м е р 1

Пусть имеется формула исчисления высказываний

$$F = \neg s \,\&\, q \,\&\, (p \vee \neg q) \,\&\, (\neg q \vee s).$$

Покажем ее противоречивость.

Родительские предложения: $C_1 = \neg s$; $C_2 = q$; $C_3 = p \vee \neg q$; $C_4 = \neg q \vee s$.

По C_1 и C_4 построим резольвенту $C_5 = \neg q$. Из C_2 и C_5 получаем пустую резольвенту C_6. Следовательно, формула F противоречива.

П р и м е р 2

Рассмотрим формулу логики предикатов:

$$F = P_1(x) \,\&\, (P_2(a) \vee P_3(x)) \,\&\, (\neg P_1(x) \vee \neg P_2(x)) \,\&\, (\neg P_1(x) \vee \neg P_3(a)).$$

Покажем ее противоречивость.

Родительские предложения: $C_1 = P_1(x)$; $C_2 = P_2(a) \vee P_3(x)$; $C_3 = \neg P_1(x) \vee \neg P_2(x)$; $C_4 = \neg P_1(x) \vee \neg P_3(a)$. Выполнив резолюцию для C_1 и C_4 по P_1, получим $C_5 = \neg P_3(a)$. Применим метод унификации. В результате подстановки $S(x, a)$ и резолюции C_2 и C_3 по P_2 получаем $C_6 = P_3(a) \vee \neg P_1(a)$. Для C_5 и C_6 по P_3 получаем $C_7 = \neg P_3(a)$. Подстановка $S(x, a)$ и резолюция C_1 и C_7 по P_1 даёт пустую резольвенту. Таким образом, формула F противоречива при $x = a$.

Организация доказательства по принципу резолюции

Порядок организации доказательства включает три этапа.

Этап 1. Исходная формула представляется в КНФ. Для этого выполняются следующие преобразования:

а) приведение формулы к предваренной форме. Для этого все кванторы выносятся в префиксную часть с сохранением их порядка и области действия;

б) удаление кванторов существования. Квантор существования можно отбросить, заменив все вхождения связываемой переменной на константу: $(\exists x)\, A(x)$ заменяется на $A(c)$, где с – константа – значение переменной, для которого утверждение A имеет место. Это возможно только в том случае, когда квантору $\exists$ не предшествует квантор всеобщности $\forall$. В противном случае при отбрасывании квантора $\exists$ все вхождения связываемой переменной заменяются на так называемою *функцию Сколема*, т. е. $(\forall y)(\exists x)\, A(x)$ заменяется на $(\forall y) A(f(y))$, где $f(y)$ – функция Сколема, которую можно интерпретировать как псевдоконстанту, имеющую тот же смысл, что и с, с той разницей, что для каждого значения переменной y имеется своя константа, для которой имеет место утверждение A. Соответственно, если квантору существования предшествует несколько кванторов всеобщности, функция Сколема будет иметь несколько переменных;

в) после удаления всех кванторов существования все кванторы всеобщности $\forall$ отбрасываются. Таким образом, получим бескванторную формулу;

г) приведение формулы к базису &, ∨, ¬ выполняется путем равносильных преобразований. Например, исключить операцию импликации можно, используя $(A \to B) = (\neg A \vee B)$;

д) опускание отрицаний по законам Моргана: $\neg(A \vee B) = (\neg A \& \neg B)$ и $\neg(A \& B) = (\neg A \vee \neg B)$.

е) получение КНФ с применением закона дистрибутивности дизъюнкции относительно конъюнкции $A \vee (B \& C) = (A \vee B) \& (A \vee C)$.

Этап 2. Все члены КНФ представляются в виде массива родительских предложений, куда потом помещаются и резольвенты, причем каждое предложение хранится в единственном экземпляре; допускается применение закона поглощения (менее общие предложения отбрасываются).

Этап 3. Операция резолюции выполняется до тех пор, пока не получится пустая резольвента.

Основное ограничение метода резолюций состоит в том, что, поскольку все доказательства ведутся по условию *достаточности*, этот метод гарантирует получение пустой резольвенты только в том случае, если исходная формула действительно противоречива.

П р и м е р ***«О таможенных чиновниках»***

Необходимо формализовать следующее описание задачи и решить ее с применением метода резолюций: «*Таможенные чиновники обыскивают каждого, кто въезжает в страну, кроме высокопоставленных лиц. Некоторые люди, способствующие незаконному провозу наркотиков, въезжали в страну и были обысканы исключительно людьми, способствующими провозу наркотиков. Никто из высокопоставленных лиц не способствовал провозу наркотиков*». Требуется оценить, правильным ли является заключение «*Некоторые из таможенных лиц способствовали провозу наркотиков*».

Введем следующие обозначения:

$E(x)$ – «x – человек, въезжающий в страну»;

$V(x)$ – «x – высокопоставленное лицо»;

$C(x)$ – «x – таможенник»; $S(y, x)$ – «y обыскивает x»;

$P(x)$ – «x способствует провозу наркотиков»;

y, x – предметные переменные.

Запишем известные факты в виде формул логики предикатов:

1. $F1$: $(\forall x)((E(x) \& \neg V(x)) \to (\exists y)(S(y, x) \& C(y)))$;
2. $F2$: $(\exists x)(E(x) \& P(x) \& (\forall y)(S(y, x) \to P(y)))$;
3. $F3$: $(\forall x)(P(x) \to \neg V(x))$;
4. Гипотеза H: $(\exists x)\,(C(x) \& P(x))$.

По теореме Эрбрана построим формулу $F1 \& F2 \& F3 \& F4 \& \neg H$.

Приведем формулу к КНФ:

1. $(\forall x)((E(x) \,\&\, \neg V(x)) \rightarrow (\exists y)(S(y, x) \,\&\, C(y)))$;
$(\forall x)(\exists y)((E(x) \,\&\, \neg V(x)) \rightarrow (S(y, x) \,\&\, C(y)))$;
$(E(x) \,\&\, \neg V(x)) \rightarrow (S(f(x), x) \,\&\, C(f(x)))$;
$\neg(E(x) \,\&\, \neg V(x)) \vee (S(f(x), x) \,\&\, C(f(x)))$;
$(\neg E(x) \vee V(x)) \vee (S(f(x), x) \,\&\, C(f(x)))$;
$((\neg E(x) \vee V(x)) \vee S(f(x), x)) \,\&\, ((\neg E(x) \vee V(x)) \vee C(f(x)))$.
2. $(\exists x)\,(E(x) \,\&\, P(x) \,\&\, (\forall y)\,(S(y, x) \rightarrow P(y)))$;
$E(a) \,\&P(a) \,\&\, (S(y, a) \rightarrow P(y))$;
$E(a) \,\&\, P(a) \,\&\, (\neg S(y, a) \vee P(y))$.
3. $(\forall x)\,(P(x) \rightarrow \neg V(x))$;
$\neg P(x) \vee \neg V(x)$.
4. $\neg(\exists x)\,(C(x) \,\&\, P(x))$;
$(\forall x)\,\neg(C(x) \,\&\, P(x))$;
$\neg C(x) \vee \neg P(x)$.

Получим следующий массив родительских предложений:

C_1: $(\neg E(x) \vee V(x)) \vee S(f(x), x)$;
C_2: $(\neg E(x) \vee V(x)) \vee C(f(x))$;
C_3: $E(a)$;
C_4: $P(a)$;
C_5: $\neg S(y, a) \vee P(y)$;
C_6: $\neg P(x) \vee \neg V(x)$;
C_7 $\neg C(x) \vee \neg P(x)$.

Доказательство противоречивости осуществляем, порождая новые родительские предложения:

C_8: $\neg V(a)$ – в результате резолюции C_4 и C_6 при $S(a, x)$;
C_9: $V(a) \vee C(f(a))$ – в результате резолюции C_2 и C_3 при $S(a, x)$;
C_{10}: $\neg C(a)$ – в результате резолюции C_4 и C_7 при $S(a, x)$;
C_{11}: $V(a)$ – в результате резолюции C_{10} и C_9.

В результате резолюции C_{11} и C_8 получается пустая резольвента. Следовательно, можно сделать вывод, что некоторые из таможенных лиц способствовали провозу наркотиков.

Важной проблемой, связанной с «комбинаторным взрывом» числа резолюций, остается проблема выбора предложений и литералов (переменных или их логических отрицаний) для унификации. Этот выбор должен выполняться так, чтобы пустое предложение (пустая резольвента) определялось наискорейшим образом. Разработаны следующие стратегии выбора, приводящие к ограничению числа возможных резолюций.

Стратегия унитарности. Одно из родительских предложений должно всегда иметь единственный литерал.

Стратегия исходных данных. В качестве одного из предложений берется гипотеза или отрицание заключения.

Стратегия поддержки. Используется одно из родительских предложений и доказываемое заключение или одно из предложений, вытекающих из них.

Линейная стратегия. Одно из предложений является предложением предшествующей резольвенты, а другое – одним из ее «предков» или исходным предложением.

Стратегия замка. Литералы выбираются вне родительских предложений, когда возможны многочисленные унификации. Литералы задаются заранее, и из них выбирается первая пара, использование которой приводит к успешному результату. Нумерация передаётся от «родителей» к резольвенте.

Контрольные вопросы и задания

1. Объясните суть продукционного подхода к представлению знаний.
2. Объясните, что понимается под понятием «продукционное правило».
3. Опишите «правую» и «левую» части продукционного правила.
4. Назовите способы представления фактов в продукционном правиле.
5. Опишите принцип реализации прямой цепочки вывода в продукционной системе.
6. Опишите принцип реализации обратной цепочки вывода в продукционной системе.
7. Назовите принципы организации правил в базе знаний для повышения эффективности вывода.
8. Опишите процесс структурирования базы правил с помощью И/ИЛИ графов.
9. Опишите процесс структурирования базы правил с помощью параллельно-последовательной структуры.
10. Назовите достоинства и недостатки продукционной модели представления знаний.
11. Объясните суть семантического подхода к представлению знаний.
12. Дайте определение понятия «семантическая сеть».
13. Назовите типы объектов в семантической сети.
14. Назовите типы связей между объектами в семантической сети.
15. Опишите механизмы организации логического вывода в семантической сети.
16. Назовите достоинства и недостатки семантической модели представления знаний.
17. Объясните суть фреймового подхода к представлению знаний.
18. Дайте определение понятий «фрейм» и «фреймовая система».
19. Укажите отличия понятий «фрейм-описание» и «фрейм-экземпляр».

20. Поясните, что включает в себя спецификация фрейма.

21. Объясните, что представляют собой присоединенные процедуры.

22. Опишите механизмы организации логического вывода во фреймовой системе.

23. Назовите достоинства и недостатки фреймовой модели представления знаний.

24. Перечислите гибридные модели представления знаний.

25. Объясните суть формального подхода к представлению знаний.

26. Поясните, каким образом задаётся формальная система.

27. Назовите наиболее часто используемые в инженерии знаний формальные системы.

28. Назовите основные свойства формальных систем.

29. Опишите механизм реализации логического вывода в естественных дедуктивных системах.

30. Объясните суть унификации формул.

31. Опишите механизм реализации логического вывода в системах, основанных на методе резолюции.

32. Объясните суть теоремы Эрбрана.

33. Объясните суть теоремы о резолюции.

34. Перечислите основные этапы доказательства по принципу резолюции.

35. Назовите достоинства и недостатки формальной модели представления знаний.

5. ПРЕДСТАВЛЕНИЕ НЕЧЁТКИХ ЗНАНИЙ

5.1. Методы представления ненадёжных знаний

В инженерии знаний под нечёткостью понимаются виды знаний и данных, обладающие какими-либо свойствами неопределённости: ненадёжность данных; неполнота информации; многозначность интерпретации; размытость информации; недетерминированность процедур вывода решений.

Ненадёжность как наиболее часто встречаемое свойство информации делится на два типа: объективную и субъективную.

Объективная ненадёжность – это свойство информации, когда те или иные знания не всегда применимы, а данные не всегда наблюдаются по объективным причинам. Например, при одних и тех же клинических данных диагноз болезни может быть разный. Объективная ненадёжность может быть представлена с помощью вероятностного подхода.

Субъективная ненадёжность – это свойство информации, когда имеет смысл говорить о достоверности данных или знаний с субъективной точки зрения. Например, один врач ставит больному один диагноз, а другой врач, исходя из своего опыта, ставит этому же больному другой диагноз. Субъективная ненадёжность может быть представлена с помощью вероятностного подхода или коэффициентов уверенностей, которые позволяют оценивать достоверность информации.

5.1.1. Метод Криса Нейлора

Метод К. Нейлора используется для представления как объективной, так и субъективной ненадёжности. В основе метода лежит теорема Байеса. Формула Байеса вытекает из определения условной вероятности (вероятность наступления одного события при условии, что другое событие уже произошло). Вероятность совместного события AB двояко выражается через условные вероятности:

$$P(AB) = P(A|B)\cdot P(B) = P(B|A)\cdot P(A),$$

$$\text{тогда } P(AB) = \frac{P(AB)}{P(B)} = \frac{P(B|A)\cdot P(A)}{P(B)}$$

где $P(A|B)$ – вероятность наступления гипотезы A при наступлении события B; $P(B|A)$ – вероятность наступления события B при истинности гипо-

тезы A; $P(A)$ – априорная вероятность гипотезы A; $P(B)$ – вероятность наступления события B.

Безусловная вероятность справедливости гипотезы называется *априорной* (насколько вероятна причина вообще), а *условная* – с учётом факта произошедшего события – *апостериорной* (насколько вероятна причина с учётом данных о событии).

Формула Байеса позволяет определить вероятность, взяв в расчёт как ранее известную информацию, так и данные новых наблюдений, а также переставить причину и следствие: по известному факту события вычислить вероятность того, что оно было вызвано данной причиной. Байесовский подход лежит в основе формирования базы знаний и организации логического вывода с помощью метода К. Нейлора.

Знания представляют собой совокупность гипотез и характеризующих их признаков. База знаний состоит из двух частей:

1-я часть – гипотезы – кортежи вида $(H; p; n; (j, p^+, p^-); \ldots)$,

где H – наименование гипотезы; $p = P(H)$ – априорная вероятность наступления гипотезы H; j – номер признака; $p^+ = P(j|H)$ – вероятность наступления события j при наступлении гипотезы H; $p^- = P(j|\neg H)$ – вероятность наступления события j в случае, если гипотеза H не наблюдается;

2-я часть – характеризующие признаки – кортежи вида $(j; g; q)$,

где j – номер признака; g – название признака; q – вопрос, задаваемый относительно j-го признака.

П р и м е р

1-я часть базы знаний:

(Грипп; 0,01; 4; (1; 0,7; 0,01); (2; 0,5; 0,02); (3; 0,8; 0,05); (5; 0,7; 0,03));

(Ангина; 0,001; 3; (1; 0,9; 0,01); (4; 0,9; 0,06); (5; 0,6; 0,04));

2-я часть базы знаний:

(1; Температура; «У Вас повышенная температура?»);

(2; Кашель; «У Вас есть кашель?»);

(3; Насморк; «У Вас есть насморк?»);

(4; Боль в горле; «Испытываете ли Вы боль в горле?»);

(5; Головная боль; «Испытываете ли Вы головную боль?»).

Идея логического вывода состоит в следующем. На каждом шаге рассматривается один из признаков (из тех, что еще не были рассмотрены) и задаётся вопрос пользователю. В зависимости от ответа пересчитываются апостериорные вероятности гипотез по следующим правилам:

Если получен ответ «*да*»:

$$P(H|j) = \frac{p^+ p}{p^+ p + p^-(1-p)};$$

Если получен ответ «*нет*»:

$$P(H|\neg j) = \frac{(1-p^+)p}{(1-p^+)p + (1-p^-)(1-p)}.$$

После того как пройдены все признаки, формируется решение – гипотеза с максимальным значением апостериорной вероятности.

С целью повышения эффективности логического вывода применяется следующая стратегия:

1. Сокращение количества признаков.

Чтобы решить, в каком порядке задавать вопросы пользователю, особенно если вопросов очень много, на каждом шаге логического вывода для каждого признака рассчитывается коэффициент «Цена свидетельства»:

$$\text{ЦС}(j) = \sum_{i=1}^{m} |P(H_i|j) - P(H_i|\neg j)|.$$

Коэффициент характеризует степень влияния признака на вероятность гипотезы.

2. Формирование досрочного решения

Чтобы понять, какую гипотезу выбрать в качестве решения и при этом избежать перебора большого количества признаков, рассчитываются дополнительные значения вероятностей:

$P_{\max}(H)$ – значение $P(H)$, когда все признаки работают в пользу гипотезы (все ответы «*да*»);

$P_{\min}(H)$ – значение $P(H)$, когда все признаки сработают против гипотезы (все ответы «*нет*»).

В процессе логического вывода величины $P_{\max}(H)$ и $P_{\min}(H)$ сходятся, следовательно:

• если на некотором шаге оказывается, что $P_{\max}(H)$ какой-либо гипотезы становится меньше $P_{\min}(H)$ других гипотез, то такую гипотезу можно отбросить;

• если на некотором шаге оказывается, что $P_{\min}(H)$ какой-либо гипотезы становится больше $P_{\max}(H)$ других гипотез, то гипотеза может быть досрочно принята в качестве решения.

3. Обработка ненадежности ответов пользователя

В случае, если ответы пользователя обладают ненадёжностью (используются ответы: *скорее да*, *скорее нет*, *не знаю*), то ответ на вопрос по

j-му признаку задаётся некоторой величиной $R(j) \in [0, 1]$, и вероятность гипотезы рассчитывается следующим образом:

$$P\left(H\middle|R(j)\right) = P(H|j) \cdot P\left(j\middle|R(j)\right) = P(H|\neg j) \cdot P\left(\neg j\middle|R(j)\right).$$

Метод К. Нейлора учитывает субъективную и объективную ненадёжность данных, позволяет естественным образом организовать логический вывод в режиме вопрос-ответ.

5.1.2. Коэффициенты уверенности Шортлиффа

Коэффициенты уверенности Шортлиффа применяются для представления субъективной ненадёжности. В основе данного метода лежит продукционный подход, когда знания представляются правилами, для каждого правила и факта задаётся коэффициент уверенности (*КУ*), при этом $КУ \in [-1; 1]$:

$КУ(x) = 1$, x – истинно;
$КУ(x) = -1$, x – ложно;
$КУ(x) > 0$, x – скорее истинно;
$КУ(x) < 0$, x – скорее ложно;
$КУ(x) = 0$, x – неопределенно.

При наполнении базы знаний экспертом задаются коэффициенты уверенности всех правил (субъективизм со стороны эксперта). В режиме консультации пользователем вместе с исходными фактами задаются значения их коэффициентов уверенности (субъективизм со стороны пользователя).

Логический вывод на основе коэффициентов уверенности реализуется следующим образом. В отличие от классического подхода для формирования решения применяются все применимые правила (отсутствует этап разрешения конфликта). Итоговый факт определяется с коэффициентом уверенности на основе результирующих фактов применимых правил и их коэффициентов уверенности. Расчет коэффициента уверенности итогового факта включает три этапа:

1. Вычисляются значения коэффициентов уверенности сложных фактов, представленных в условиях правил. Способ расчёта определяется типом логической операции:

• если факты соединены операцией «*И*», то
$КУ(X\&Y) = КУ(X)\&КУ(Y) = \min(КУ(X), КУ(Y))$;
• если факты соединены операцией «*ИЛИ*», то
$КУ(X \vee Y) = КУ(X) \vee КУ(Y) = \max(КУ(X), КУ(Y))$;
• если факт отрицается и используется операция «*НЕ*», то
$КУ(\neg X) = -КУ(X)$.

2. Вычисляются значения коэффициентов уверенности результирующих фактов в правых частях правил:

$$КУ(\textit{результата}) = КУ(\textit{условия}) \cdot КУ(\textit{правила}).$$

3. Если применимо несколько правил, то вычисляется коэффициент уверенности итогового результирующего факта:

$$\text{КУ}(Z|\text{П}_1, \text{П}_2)=\begin{cases} 1, \text{если КУ}(Z|\text{П}_1)=1 \text{ или КУ}(Z|\text{П}_2)=1; \\ -1, \text{если КУ}(Z|\text{П}_1)=-1 \text{ или КУ}(Z|\text{П}_2)=-1; \\ \text{КУ}(Z|\text{П}_1)+\text{КУ}(Z|\text{П}_2), \text{если КУ}(Z|\text{П}_1)\cdot\text{КУ}(Z|\text{П}_2)\le 0; \\ \big(\text{КУ}(Z|\text{П}_1)+\text{КУ}(Z|\text{П}_2)\big)-\big(\text{КУ}(Z|\text{П}_1)\cdot\text{КУ}(Z|\text{П}_2)\big), \\ \text{если КУ}(Z|\text{П}_1)>0 \text{ и КУ}(Z|\text{П}_2)>0; \\ \big(\text{КУ}(Z|\text{П}_1)+\text{КУ}(Z|\text{П}_2)\big)+\big(\text{КУ}(Z|\text{П}_1)\cdot\text{КУ}(Z|\text{П}_2)\big), \\ \text{если КУ}(Z|\text{П}_1)<0 \text{ и КУ}(Z|\text{П}_2)<0. \end{cases}$$

П р и м е р

Дана база правил:

П$_1$: *ЕСЛИ <падает давление> и <дует ветер> ТО <пойдёт дождь>*;

П$_2$: *ЕСЛИ <на небе черная туча> ТО <пойдёт дождь>*;

П$_3$: *ЕСЛИ <низкая облачность> и <высокая влажность> ТО <пойдёт дождь>*.

Требуется рассчитать коэффициент уверенности факта *<пойдёт дождь>*.

Для удобства записи введем обозначения для фактов и запишем правила в виде

П$_1$: $X\&Y \rightarrow Z$

П$_2$: $W \rightarrow Z$

П$_3$: $F\&K \rightarrow Z$

Пусть *КУ*(*X*) = 0.4; *КУ*(*Y*) = 0.9; *КУ*(*W*) = 0.8; *КУ*(*F*) = 0.3; *КУ*(*K*) = 0.2;

КУ(*П*$_1$) = 0.6; *КУ*(*П*$_2$) = 0.5; *КУ*(*П*$_3$) = 0.7

Чтобы рассчитать коэффициенты уверенности факта *Z* на основе трех правил, сначала вычисляются коэффициенты уверенности результирующих фактов для каждого правила отдельно:

КУ(*Z*|*П*$_1$) = *КУ*(*X*&*Y*) · *КУ*(*П*$_1$) = (*КУ*(*X*) & *КУ*(*Y*)) · *КУ*(*П*$_1$) = 0.4 · 0.6 = = 0.24;

КУ(*Z*|*П*$_2$) = *КУ*(*W*) · *КУ*(*П*$_2$) = 0.8 · 0.5 = 0.4;

КУ(*Z*|*П*$_3$) = *КУ*(*F*&*K*) · *КУ*(*П*$_3$) = (*КУ*(*F*) & *КУ*(*K*)) · *КУ*(*П*$_3$) = 0.2 · 0.7 = = 0.14.

Затем рассчитывается итоговый факт с учётом коэффициентов уверенностей результирующих фактов, полученных по отдельным правилам. Расчёт выполняется по всем применимым правилам попарно.

$$КУ(Z|П_1,П_2) = (КУ(Z|П_1) + КУ(Z|П_2)) – (КУ(Z|П_1) \cdot КУ(Z|П_2)) =$$
$$= (0.24 + 0.4) – (0.24 \cdot 0.4) = 0.54.$$

$$КУ(Z|П_1, П_2, П_3) = (КУ(Z|П_1, П_2) + КУ(Z|П_3)) – (КУ(Z|П_1, П_2) \cdot КУ(Z|П_3)) =$$
$$= (0.54 + 0.14) – (0.54 \cdot 0.14) = 0.6.$$

Таким образом, коэффициент уверенности факта <*пойдёт дождь*> равен 0,6, что соответствует достаточно высокой уверенности.

Применение коэффициентов уверенности позволяет учитывать ненадежность данных, сохраняя естественность структуры и простоту модификации базы знаний, и обеспечивает ясный механизм логического вывода.

5.2. Методы представления размытых знаний и нечёткий вывод

Теория нечётких множеств (или нечёткая логика) являются обобщениями классической теории множеств и классической формальной логики. Данные понятия впервые были предложены американским ученым Лотфи Заде в 1965 г. Основной причиной появления новой теории стало наличие нечётких и приближенных рассуждений при описании человеком процессов, явлений, объектов: маленький-большой, далеко-близко, горячий-холодный и т. д. Для создания интеллектуальных систем, способных адекватно взаимодействовать с человеком, необходим новый математический аппарат, который переводит *неоднозначные жизненные утверждения* в язык *чётких и формальных математических формул*.

5.2.1. Понятие нечёткого множества и нечёткого отношения

Характеристикой *нечёткого множества F* на некотором базовом заданном множестве *U* выступает функция принадлежности $\mu_F(u)$, $u \in U$ – степень принадлежности элемента базового множества нечёткому множеству *F*. Нечётким множеством *F* называется множество упорядоченных пар вида $F = \{\mu_F(u)/u\}$, $\mu_F(u) \in [0,1]$. Значение $\mu_F(u) = 0$ означает отсутствие принадлежности к множеству, $\mu_F(u)=1$ – полную принадлежность.

Нечёткое множество можно представить в следующем виде:

$$F(\sqsubseteq U) = \frac{\mu_F(u_1)}{u_1} + \frac{\mu_F(u_2)}{u_2} + \cdots + \frac{\mu_F(u_n)}{u_n} = \sum_{i=1}^{n} \frac{\mu_F(u_i)}{u_i}.$$

Если базовое множество непрерывно, то нечёткое множество можно записать в виде

$$F(\subseteq U) = \int \frac{\mu(u)}{u}$$

В обозначениях нечётких множеств знаки суммы и интеграла обозначают перечисление элементов базового множества, а знак дроби используется как обозначение соответствия. Значение функции принадлежности можно интерпретировать как вероятность.

Над нечёткими множествами выполняются операции, соответствующие обычным операциям, принятым в теории множеств и формальной логике. К базовым операциям над нечёткими множествами, применяемым в инженерии знаний, относятся:

пересечение (нечёткое «И»):

$$\mu_{F\cap G}(u) = \mu_F(u) \wedge \mu_G(u) = \min\left(\mu_F(u), \mu_G(u)\right);$$

объединение (нечёткое «ИЛИ»):

$$\mu_{F\cup G}(u) = \mu_F(u) \vee \mu_G(u) = \max\left(\mu_F(u), \mu_G(u)\right);$$

отрицание:

$$\mu_{\overline{F}}(u) = 1 - \mu_F(u).$$

Следует отметить, что нечёткие множества не всегда удовлетворяют законам, справедливым для чётких множеств.

П р и м е р 1

Требуется на базовом множестве U – «возраст» построить аналитически и графически три нечётких множества: F – «молодой», G – «средних лет» и H – «пожилой». Показать, что «молодой» ∪ «ср. лет» ≠ ¬«старый».

Если базовое множество $U = \{0; 10; 20; 30; 40; 50; 60; 70; 80\}$ (годы), тогда аналитическое представление нечётких множеств имеет вид

$$\textit{«Молодой»}\ F(\sqsubseteq U) = \frac{1}{10} + \frac{1}{20} + \frac{0{,}7}{30} + \frac{0{,}2}{40} + \frac{0}{50} + \frac{0}{60} + \frac{0}{70} + \frac{0}{80};$$

$$\textit{«Средних лет»}\ G(\sqsubseteq U) = \frac{0}{10} + \frac{0}{20} + \frac{0{,}2}{30} + \frac{0{,}8}{40} + \frac{1}{50} + \frac{0.5}{60} + \frac{0{,}2}{70} + \frac{0}{80};$$

$$\textit{«Пожилой»}\ H(\sqsubseteq U) = \frac{0}{10} + \frac{0}{20} + \frac{0}{30} + \frac{0}{40} + \frac{0{,}3}{50} + \frac{1}{60} + \frac{1}{70} + \frac{1}{80}.$$

Графическое представление нечётких множеств имеет вид (рис. 5.1):

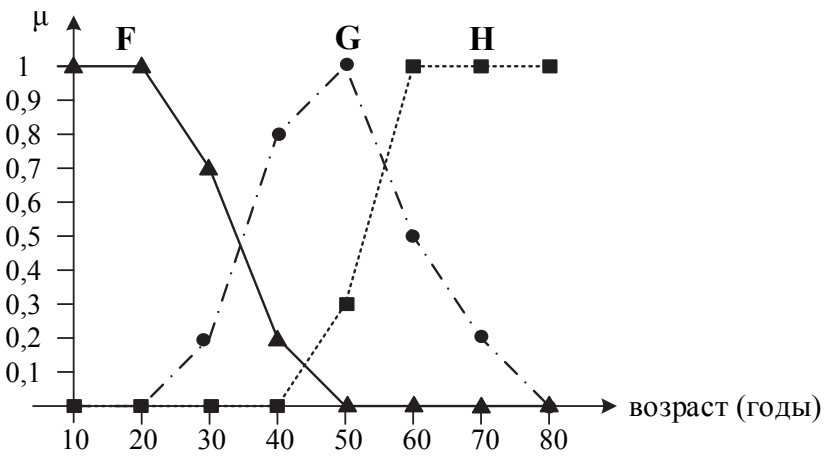


Рис. 5.1. Графическое представление нечётких множеств *F*, *G* и *H*

Покажем, что «молодой» ∪ «ср. лет» ≠ ¬«старый». Это можно установить с помощью непосредственных вычислений:

$$F(\sqsubseteq U) \cup G(\sqsubseteq U) = \frac{1}{10} + \frac{1}{20} + \frac{0{,}7}{30} + \frac{0{,}8}{40} + \frac{1}{50} + \frac{0{,}5}{60} + \frac{0{,}2}{70} + \frac{0}{80};$$

$$\overline{H}(\sqsubseteq U) = \frac{1}{10} + \frac{1}{20} + \frac{1}{30} + \frac{1}{40} + \frac{0{,}7}{50} + \frac{0}{60} + \frac{0}{70} + \frac{0}{80};$$

Из сравнения полученных нечётких множеств по значениям функции принадлежности видно, что множества не совпадают, что подтверждает истинность приведенного выражения. Убедиться в истинности выражения можно также, представив нечёткие множества графически (рис. 5.2).

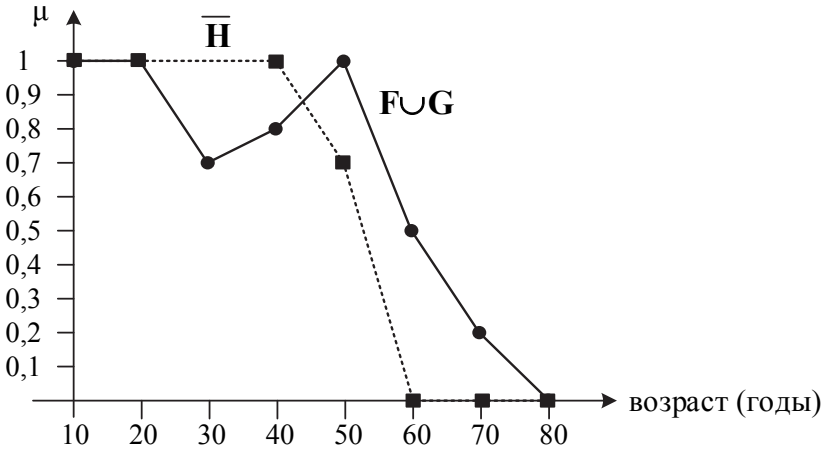


Рис. 5.2. Графическое представление нечётких множеств *F*∪*G* и ¬*H*

Нечёткое отношение – это нечёткое множество, базовым множеством которого является декартово произведение множеств: если

$U = \{u_1, \dots, u_n\}$ и $V = \{v_1, \dots, v_m\}$ – базовые множества, то $S(\sqsubseteq U \times V)$ – нечёткое отношение, которое может быть представлено в следующем виде:

$$S(\sqsubseteq U \times V) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\mu(u_i, v_j)}{(u_i, v_j)}.$$

Нечёткие отношения используются для представления нечётких правил: $R: F \to G$, где $F(\sqsubseteq U)$ и $G(\sqsubseteq V)$ – нечёткие множества (факты), заданные на базовых множествах U и V соответственно. Тогда R представляется в виде следующего бинарного отношения:

$$R(\sqsubseteq U \times V) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\mu_F(u_i) \wedge \mu_G(V_j)}{(u_i, v_j)}.$$

П р и м е р 2

Рассмотрим нечёткое правило $R: F \to G$ – «*Если студент часто прогуливает занятия, то оценка у него низкая*».

Определим базовые множества: $U = \{0; 5; 10; 15\}$ (количество занятий), $V = \{2; 3; 4; 5\}$ (оценка).

На базовых множествах зададим нечёткие множества:

$$\textit{«Часто прогуливает занятия»}\ F(\sqsubseteq U) = \frac{0}{0} + \frac{0{,}3}{5} + \frac{0{,}7}{10} + \frac{1}{15};$$

$$\textit{«Низкая оценка»}\ G(\sqsubseteq V) = \frac{1}{2} + \frac{0{,}8}{3} + \frac{0{,}2}{4} + \frac{0}{5}.$$

Вычислим нечёткое отношение R, выполняя поэлементно операцию минимум. Нечёткое отношение может быть представлено в виде

$$R(\sqsubseteq U \times V) = \begin{array}{c} U \backslash V \\ 0 \\ 5 \\ 10 \\ 15 \end{array} \begin{array}{c} \begin{array}{cccc} 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0{,}3 & 0{,}3 & 0{,}2 & 0 \\ 0{,}7 & 0{,}7 & 0{,}2 & 0 \\ 1 & 0{,}8 & 0{,}2 & 0 \end{bmatrix} \end{array}$$

Над нечёткими отношениями можно выполнять те же операции, что и над нечёткими множествами. Дополнительно вводится операция *свертки (композиции)*.

Свертка минимакса двух нечётких отношений

Пусть заданы два нечётких бинарных отношения, которые соответствуют правилам $R: F \to G$ и $S: G \to H$:

$$R(\sqsubseteq U \times V) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\mu_R(u_i, v_j)}{(u_i, v_j)} \quad \text{и} \quad S(\sqsubseteq V \times W) = \sum_{j=1}^{m} \sum_{l=1}^{k} \frac{\mu_S(v_j, w_l)}{(v_j, w_l)}$$

Свертка двух нечётких отношений R и S соответствует правилу $R \circ S: F \to H$ и определяется следующим образом:

$$R \circ S(\sqsubseteq U \times W) = \sum_{i=1}^{n}\sum_{l=1}^{k}\frac{\mu_{R\circ S}(u_i, w_l)}{(u_i, w_l)} = \sum_{i=1}^{n}\sum_{l=1}^{k}\frac{\bigvee(\mu_R(u_i, v_j)\wedge\mu_S(v_j, w_l))}{(u_i, w_l)}.$$

Свертка минимакса нечёткого отношения и нечёткого множества:

Пусть задано нечёткое бинарное отношение, которое соответствует правилу $R: F \to G$, и нечёткое множество F'':

$$R(\sqsubseteq U \times V) = \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\mu_R(u_i, v_j)}{(u_i, v_j)} \quad \text{и} \quad F''(\sqsubseteq U) = \sum_{i=1}^{n}\frac{\mu_{F'}(u_i)}{(u_i)}.$$

Свертка нечёткого отношения и нечёткого множества соответствует применению правила к факту и определяется следующим образом:

$$R(\sqsubseteq U \times W) \circ F''(\sqsubseteq U) = \sum_{j=1}^{m}\frac{\bigvee(\mu_R(u_i, v_j)\wedge\mu_F(u_i))}{(v_j)}.$$

П р и м е р 3

Применим нечёткое правило $R: F \to G$ к факту F'' – *«студент прогуливает занятия не очень часто»*.

Зададим нечёткое множество F'' на базовом множестве $U = \{0; 5; 10; 15\}$ (оценка):

«Не очень часто прогуливает занятия» $F''(\sqsubseteq U) = \frac{0}{0} + \frac{0{,}5}{5} + \frac{1}{10} + \frac{0{,}8}{15}$.

Выполним операцию свертки нечёткого отношения и нечёткого множества:

$$G''(\sqsubseteq V) = R(\sqsubseteq U \times V) \circ F''(\sqsubseteq U) =$$

$$= \begin{array}{c} U\backslash V \\ 0 \\ 5 \\ 10 \\ 15 \end{array} \begin{array}{c} \begin{array}{cccc} 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0{,}3 & 0{,}3 & 0{,}2 & 0 \\ 0{,}7 & 0{,}7 & 0{,}2 & 0 \\ 1 & 0{,}8 & 0{,}2 & 0 \end{bmatrix} \end{array} \circ \frac{0}{0} + \frac{0{,}5}{5} + \frac{1}{10} + \frac{0{,}8}{15} = \frac{0{,}8}{2} + \frac{0{,}8}{3} + \frac{0{,}2}{4} + \frac{0}{5}.$$

5.2.2. Логический вывод в нечётких системах

В нечётких системах в качестве модели знаний используются нечёткие правила в форме нечётких отношений. Нечёткие факты представляют собой нечёткие множества. Логический вывод основывается на нечётком *правиле заключения* (*modus ponens*):

$$\frac{F',\ F \to G}{G'}.$$

Здесь F и F' должны быть определены на одном и том же базовом множестве U. Тогда F и G' будут также определены на одном и том же базовом множестве V. Согласно нечёткому *modus ponens*: G' строится похожим на G в такой же степени, в какой F' похож на F.

Чтобы применить правило $R: F \to G$ к факту F' выполняется операция свертки нечёткого множества и нечёткого отношения:

$$G''(\sqsubseteq V) = R(\sqsubseteq U \times V) \circ F''(\sqsubseteq U) = \sum_{j=1}^{m} \frac{\bigvee(\mu_R(u_i, v_j) \wedge \mu_F(u_i))}{(v_j)}.$$

Результатом нечёткого вывода является нечёткий факт в виде нечёткого множества. Как правило, в нечётких экспертных системах после завершения логического вывода выполняется процедура *дефазификации* – преобразование в единственное (чёткое) значение. Существует достаточно большое количество методов перехода к точным значениям. Наиболее распространенными методами являются: метод полной интерпретации или «центра тяжести» и метод максимума функции принадлежности.

П р и м е р 1

Дано знание в виде нечёткого правила: $R: F \to G$ – «*Если груз тяжелый, то скорость машины низкая*» и исходный факт F' – «*груз не очень тяжелый*». Необходимо получить вывод G'.

Определим базовые множества:

$U = \{10; 12; 14; 16; 18; 20\}$ (тонны), $V = \{30; 40; 50; 60; 70\}$ (км/ч).

Зададим нечёткие факты:

$$\textit{«Груз тяжелый»}\ F(\sqsubseteq U) = \frac{0{,}1}{10} + \frac{0{,}4}{12} + \frac{0{,}6}{14} + \frac{0{,}8}{16} + \frac{1}{18} + \frac{1}{20};$$

$$\textit{«Скорость низкая»}\ G(\sqsubseteq V) = \frac{0{,}1}{30} + \frac{0{,}4}{40} + \frac{0{,}6}{50} + \frac{0{,}8}{60} + \frac{1}{70};$$

$$\textit{«Груз не очень тяжелый»}\ F'(\sqsubseteq U) = \frac{0}{10} + \frac{0{,}4}{12} + \frac{1}{14} + \frac{1}{16} + \frac{0{,}6}{18} + \frac{0}{20}.$$

Вычислим нечёткое отношение, соответствующее правилу $R: F \to G$:

$$R(\sqsubseteq U \times V) = \begin{array}{c} U\backslash V \\ 10 \\ 12 \\ 14 \\ 16 \\ 18 \\ 20 \end{array} \begin{array}{c} \begin{array}{ccccc} 30 & 40 & 50 & 60 & 70 \end{array} \\ \begin{bmatrix} 0{,}1 & 0{,}1 & 0{,}1 & 0{,}1 & 0 \\ 0{,}4 & 0{,}4 & 0{,}4 & 0{,}1 & 0 \\ 0{,}6 & 0{,}6 & 0{,}5 & 0{,}1 & 0 \\ 0{,}8 & 0{,}8 & 0{,}5 & 0{,}1 & 0 \\ 1 & 0{,}8 & 0{,}5 & 0{,}1 & 0 \\ 1 & 0{,}8 & 0{,}5 & 0{,}1 & 0 \end{bmatrix} \end{array}.$$

Применим правило R к факту F', выполнив операцию свертки нечёткого отношения и нечёткого множества:

$$G'(\sqsubseteq V) = R(\sqsubseteq U \times V) \circ F'(\sqsubseteq U) =$$

$$= \begin{matrix} U\backslash V \\ 10 \\ 12 \\ 14 \\ 16 \\ 18 \\ 20 \end{matrix} \begin{matrix} 30 & 40 & 50 & 60 & 70 \end{matrix} \begin{bmatrix} 0{,}1 & 0{,}1 & 0{,}1 & 0{,}1 & 0 \\ 0{,}4 & 0{,}4 & 0{,}4 & 0{,}1 & 0 \\ 0{,}6 & 0{,}6 & 0{,}5 & 0{,}1 & 0 \\ 0{,}8 & 0{,}8 & 0{,}5 & 0{,}1 & 0 \\ 1 & 0{,}8 & 0{,}5 & 0{,}1 & 0 \\ 1 & 0{,}8 & 0{,}5 & 0{,}1 & 0 \end{bmatrix} \circ \frac{0}{10} + \frac{0{,}4}{12} + \frac{1}{14} + \frac{1}{16} + \frac{0{,}6}{18} + \frac{0}{20} =$$

$$= \frac{0{,}8}{30} + \frac{0{,}8}{40} + \frac{0{,}2}{50} + \frac{0}{60} + \frac{0}{70}.$$

В качестве чёткого ответа выбирается значение базового множества, которому соответствует наибольшая уверенность – максимальное значение функции принадлежности. В данном примере два значения базового множества (30 и 40) имеют максимальное значение функции принадлежности (0,8), поэтому в качестве чёткого ответа берется их среднее значение – 35 км/ч.

Рассмотренный нечёткий вывод представляет прямой (восходящий) вывод от предпосылок к заключению. В системах также может использоваться и обратный (нисходящий) вывод.

П р и м е р 2

Рассмотрим упрощенную модель диагностики неисправности автомобиля: u_1 – неисправность аккумулятора; u_2 – отработка машинного масла; v_1 – затруднения при запуске; v_2 – ухудшение цвета выхлопных газов; v_3 – недостаток мощности.

Между u_i и v_j существуют нечёткие причинно-следственные отношения $r_{ij}: u_i \to v_j$, которые могут быть заданы в виде нечёткого правила $R: F \to G$. Предпосылки и заключения можно представить как нечёткие множества $F(\sqsubseteq U)$ и $G(\sqsubseteq V)$. Тогда $G'(\sqsubseteq V) = R(\sqsubseteq U \times V) \circ F'(\sqsubseteq U)$.

Пусть имеются знания эксперта-автомеханика:

$$R(\sqsubseteq U \times V) = \begin{matrix} U\backslash V \\ u_1 \\ u_2 \end{matrix} \begin{matrix} v_1 & v_2 & v_3 \end{matrix} \begin{bmatrix} 0{,}9 & 0{,}1 & 0{,}2 \\ 0{,}6 & 0{,}5 & 0{,}5 \end{bmatrix}$$

и наблюдаются симптомы неисправности автомобиля, полученные в результате его осмотра:

$$G'(\sqsubseteq V) = \frac{0{,}9}{30} + \frac{0{,}1}{40} + \frac{0{,}2}{50}.$$

Необходимо определить причины состояния автомобиля

$$F'(\sqsubseteq U) = \frac{a_1}{u_1} + \frac{a_2}{u_2}.$$

Отношение введенных нечётких множеств можно представить равенством

$$[0{,}9 \quad 0{,}1 \quad 0{,}2] = [a_1 \quad a_2]^{\circ} \begin{bmatrix} 0{,}9 & 0{,}1 & 0{,}2 \\ 0{,}6 & 0{,}5 & 0{,}5 \end{bmatrix}.$$

При использовании свертки минимакса отношение преобразуется к виду

$$0{,}9 = (0{,}9 \wedge a_1) \vee (0{,}6 \wedge a_2);$$
$$0{,}1 = (0{,}1 \wedge a_1) \vee (0{,}5 \wedge a_2);$$
$$0{,}2 = (0{,}2 \wedge a_1) \vee (0{,}5 \wedge a_2).$$

При решении данной системы заметим, что в правой части второй член не влияет на левую часть (операция ИЛИ), поэтому

из первого уравнения:

$$0{,}9 = 0{,}9 \wedge a_1, \qquad a_1 \geq 0{,}9;$$

из второго уравнения:

$$0{,}1 \geq 0{,}5 \wedge a_2, \qquad a_2 \leq 0{,}1;$$

полученное решение удовлетворяет третьему уравнению, таким образом, имеем

$$0{,}9 \leq a_1 \leq 1{,}0, \qquad 0 \leq a_2 \leq 0{,}1.$$

Учитывая полученные значения функции принадлежности, для u_1 – неисправность аккумулятора и u_2 – отработка машинного масла соответственно, можно сделать вывод, что проблема в аккумуляторе.

Использование логики нечётких множеств в инженерии знаний потребовало введения нового типа переменных в моделях представления знаний. Такой новый тип переменной называется *лингвистической*, что подчеркивает нечёткий (лингвистический) характер используемых знаний. Значения лингвистической переменной представляют собой словесные оценки, выраженные словами или фразами, заданными нечёткими множествами. Рассмотрим в качестве примера нечёткое множество «*слабый дождь*»:

$$\text{«Слабый дождь»}\ F(\sqsubseteq U) = \frac{0}{\text{нет}} + \frac{0{,}7}{\text{накрапывает}} + \frac{0{,}9}{\text{моросит}} + \frac{0}{\text{льет}}.$$

Элементы базового множества здесь представляют собой также нечёткие множества на базовом множестве «*количество осадков*» (мм). В этом случае понятие «*слабый дождь*» является *лингвистической переменной.*

Аппарат нечёткой логики показал достаточную эффективность в системах управления. Круг задач, при решении которых используется нечёткая логика, постоянно расширяется: автомобильная промышленность, аэрокосмическая промышленность, приборостроение и производство бытовой техники, системы управления производством и транспортом, анализ и прогнозирование в сфере политики и экономики, финансы, анализ данных и др.

Контрольные вопросы и задания

1. Перечислите виды нечёткости знаний.
2. Дайте определение термина «ненадёжные знания».
3. Назовите тип ненадёжности знаний, для которого применяется метод К. Нейлора.
4. Объясните принцип формирования базы знаний в методе К. Нейлора.
5. Объясните способ реализации логического вывода в методе К. Нейлора.
6. Назовите стратегии для повышения эффективности логического вывода в методе К. Нейлора.
7. Назовите тип ненадежности знаний, для которого применяются коэффициенты уверенности Шортлиффа.
8. Объясните принцип формирования базы знаний с помощью коэффициентов уверенности Шортлиффа.
9. Объясните способ реализации логического вывода на основе коэффициентов уверенности Шортлиффа.
10. Дайте определение термина «размытые знания».
11. Дайте определение понятия «нечёткое множество».
12. Поясните значение функции принадлежности.
13. Объясните, как задаётся нечёткое множество.
14. Перечислите базовые операции над нечёткими множествами в инженерии знаний.
15. Объясните, как задаётся нечёткое отношение.
16. Объясните, как выполняется операция свертки (композиции) нечёткого отношения и нечёткого множества.
17. Объясните, как выполняется операция свертки (композиции) двух нечётких отношений.
18. Опишите способ организации нечёткого логического вывода.
19. Объясните цель и способ выполнения процедуры дефазификации.
20. Дайте определение понятия «лингвистическая переменная».

ЗАКЛЮЧЕНИЕ

В учебном пособии «Модели и методы искусственного интеллекта» систематизировано изложены основы теории представления знаний в системах искусственного интеллекта; дано описание наиболее значимых в настоящее время моделей и технологий проектирования систем, основанных на знаниях; представлены основные теоретические положения и примеры решения практических задач.

Вопросы, связанные с моделированием знаний в системах искусственного интеллекта, охватывают такие актуальные на сегодняшний день области знаний, как технология инженерии знаний, эвристический поиск, теория алгоритмов, логика предикатов и нечёткая логика, которые являются необходимыми элементами в современной подготовке специалистов в сфере информационных технологий.

Материал данного учебного пособия сформирован на основе анализа современных достижений науки как отечественных, так и зарубежных исследователей в области искусственного интеллекта. Методический уровень материала позволяет использовать пособие при самостоятельной работе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Болотова, Л. С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях / Л. С. Болотова. – М. : Финансы и статистика, 2012. – 663 с.

2. Гаврилова, Т. А. Извлечение и структурирование знаний для экспертных систем / Т. А. Гаврилова, К. Р. Червинская. – М. : Радио и связь, 1992. – 200 с.

3. Гаврилова, Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб. : Питер, 2000. – 384 с.

4. Гаскаров, Д. В. Интеллектуальные информационные системы / Д. В. Гаскаров. – М.: Высшая школа, 2003. – 431 с.

5. Джарратано, Дж. Экспертные системы: принципы разработки и программирование: пер. с англ. / Дж. Джарратано, Г. Райли. – М.: Вильямс, 2006. – 1152 с.

6. Джексон, П. Введение в экспертные системы.: учеб. пособие; пер. с англ. / П. Джексон. – М. : Вильямс, 2001. – 624 с.

7. Жданов, А. А. Автономный искусственный интеллект / А. А. Жданов. – М. : Бином. Лаборатория знаний, 2015. – 259 с.

8. Искусственный интеллект: в 3 кн. Кн. 2. Модели и методы: справочник / под ред. Д. А. Поспелова. – М.: Радио и связь, 1990. – 304 с.

9. Кофман, А. Введение в теорию нечётких множеств / А. Кофман. – М. : Радио и связь, 1982. – 432 с.

10. Лорьер, Ж.-Л. Системы искусственного интеллекта: пер. с фр. / Ж.-Л. Лорьер. – М. : Мир, 1991. – 568 с.

11. Люгер, Д. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем: пер. с англ. / Д. Ф. Люгер. – 4-е изд. – М. : Вильямс, 2003. – 863 с.

12. Муромцев, Д. И. Введение в технологию экспертных систем / Д. И. Муромцев. – СПб. : СПбГУИТМО, 2005. – 126 с.

13. Нейлор, К. Как построить свою экспертную систему / К. Нейлор; пер. с англ. – М. : Энергоатомиздат, 2001. – 422 с.

14. Нильсон, Н. Искусственный интеллект. Методы поиска решений: пер. с англ. / Н. Нильсон ; под ред. С. В. Фомина – М. : Мир, 1973. – 270 с.

15. Нильсон, Н. Принципы искусственного интеллекта: пер. с англ. / Н. Нильсон. – М. : Радио и связь,1985. – 373 с.

16. Поспелов, Д. А. Моделирование рассуждений. Опыт анализа мыслительных актов / Д. А. Поспелов. – М. : Радио и связь, 1989.

17. Построение экспертных систем / под ред. Ф. Хейес-Рота, Д. Уотермена, Д. Лената. – М. : Мир, 1987. – 44 с.

18. Рассел, С. Искусственный интеллект. Современный подход / С. Рассел, П. Норвиг – М. : Вильямс, 2006. – 1407 с.

19. Сидоркина, И. Г. Системы искусственного интеллекта: учеб. пособие / И. Г. Сидоркина. – М.: КНОРУС, 2011. – 248 с.

20. Станкевич, Л. А. Интеллектуальные системы и технологии: учебник и практикум для бакалавриата и магистратуры / Л. А. Станкевич. – М. : Юрайт, 2018. – 397 с.

21. Уотерман, Д. Руководство по экспертным системам: пер. с англ. / Д. Уотерман. – М. : Мир, 1989.

22. Уэно, Х. Представление и использование знаний: пер. с яп. / Х. Уэно, М. Исидзука. – М. : Мир, 1989. – 220 с.

23. Яковлев, С. А. Экспертные системы: учеб. пособие / С. А. Яковлев. – СПб. : ГУАП, 2010. – 124 с.

24. Ярушкина, Н. Г. Основы теории нечётких множеств и гибридных систем / Н. Г. Ярушкина. – М. : Финансы и статистика, 2004. – 320 с.

25. Ясницкий, Л. Н. Интеллектуальные системы: учебник / Л. Н. Ясницкий. – М. : Лаборатория знаний, 2016. – 221 с.

ПРИЛОЖЕНИЕ

Задачи и упражнения

Поиск в пространстве состояний

1. К реке подошли три рыцаря с оруженосцами. У берега оказалась одна двухместная лодка. Нужно организовать переправу при условии, что ни один оруженосец нигде не должен оставаться в обществе других рыцарей без своего хозяина. Решить задачу методом полного перебора, построить дерево поиска решения.

2. Пять очень ревнивых мужей вместе со своими женами желают переправиться через реку. Посередине реки есть остров. Лодка вмещает не более двух человек, и ни один из мужей не может оставить свою жену в обществе других мужчин. Решить задачу методом полного перебора, построить дерево поиска решения.

3. Имеется квадрат, разделенный на девять клеток. В клетках, помеченных буквами, находятся восемь фишек. За один ход можно переместить одну фишку, расположенную рядом с пустой клеткой, в эту клетку. Цель задачи – достигнуть расположения фишек, показанного на рисунке, за минимальное количество ходов из любой начальной расстановки фишек или прийти к выводу, что решения не существует. Решить задачу «Хамелеон» методом полного перебора, построить дерево поиска решения.

Х	**А**	**М**
Е	**Л**	**Е**
О	**Н**	

Целевое состояние

4. Решить задачу «Три монеты» методом полного перебора, построить дерево поиска решения. На столе лежат три монеты: две крайние – кверху решкой, средняя – кверху гербом. Ход игры состоит в переворачивании двух соседних монет. Необходимо за минимальное количество ходов перевернуть все монеты так, чтобы они легли либо решкой, либо гербом кверху.

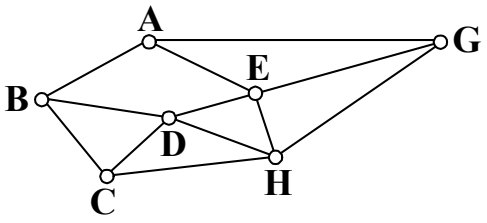
5. Решить задачу методом поиска в ширину, построить дерево поиска решения. Необходимо расставить восемь ферзей на шахматной доске таким образом, чтобы ни один ферзь не находился под боем другого. В шахматах ферзь может бить любую фигуру, стоящую с ним в одном ряду, столбце или по диагонали.

6. Решить задачу «Волк, коза и капуста» методом поиска в глубину, построить дерево поиска решения. Необходимо перевезти с одного берега

на другой волка, козу и капусту. Предполагается, что вместе с человеком в лодке помещается только один объект, и человеку приходится охранять козу от волка и капусту от козы.

7. Решить головоломку «Ханойская башня» (упрощенный вариант) методом поиска в ширину, построить дерево поиска решения. Имеется три стержня 1, 2, 3 и три кольца a, b, c (a – самое большое, c – самое маленькое). В начальный момент все кольца надеты на стержень 1 так, что самое маленькое находится наверху. Необходимо переложить все кольца на стержень 3. За один ход можно переместить одно верхнее кольцо с одного стержня на другой; при этом запрещено помещать большее кольцо на меньшее.

8. Решить задачу методом поиска в ширину, построить дерево поиска решения. Путешественник имеет карту дорог нескольких городов, необходимо выстроить маршрут поездки так, чтобы побывать в каждом городе, но не более одного раза. На рисунке показана карта дорог семи городов.



Целевое состояние

9. Решить головоломку «Прыгающие шарики» методом поиска в ширину. Имеется с одной стороны три белых шарика, с другой – три черных и пустое поле. Необходимо белые шарики переместить вправо, а черные – влево. Допустимые ходы: можно двигать шарик на пустое соседнее поле; можно перепрыгивать через шарик, если следующее за ним поле пустое. Проанализируйте сложность пространства состояний и наличие циклов. Какую эвристику можно применить в данной задаче.


Начальное состояние


Целевое состояние

10. Решить задачу «Обезьяна и бананы». Обезьяна и ящик находятся в комнате, к потолку комнаты привязана связка бананов. Обезьяна может перемещаться по комнате, переносить ящик и забираться на него. Достать бананы она сможет только в том случае, если заберется на ящик, расположенный в точности под бананами. Предполагается, что обезьяна может ходить по комнате, двигать по полу ящик, взбираться на него и доставать бананы. Процесс решения задачи сводится к уменьшению различий между исходным состоянием и целевым. Необходимо найти последовательность действий, которая позволит обезьяне достать бананы.

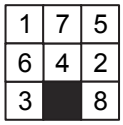
11. Применить алгоритм A* для поиска решения головоломки «Восьмерка» для следующих ситуаций:

a)

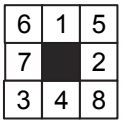
Начальное состояние

Целевое состояние

b)



Начальное состояние



Целевое состояние

c)

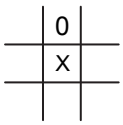
Начальное состояние

Целевое состояние

12. Решить задачу «Тик-так-ту», применяя стратегию первого уровня, построить дерево поиска. У двух игроков по три фишки. В первом туре играющие по очереди расставляют свои фишки (за один ход) на свободных клетках. Во втором туре, когда размещены все шесть фишек, играющие поочередно перемещают по одной фишке на свободные соседние клетки. Соседней считается клетка, расположенная в одном шаге от данной. Выигрывает тот, кто первым расположит свои фишки по прямой линии.

13. Выполнить двухуровневый минимакс для игры в «крестики-нолики» для следующих ситуаций:

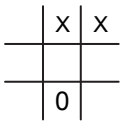
a)



Ход делает X (игрок MAX)

Начальное состояние

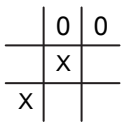
b)



Ход делает 0 (игрок MIN)

Начальное состояние

c)



Ход делает X (игрок MAX)

Начальное состояние

Представление знаний

1. Сформировать базу знаний продукционной системы для управления движения лифтом в трехэтажном доме с учетом местоположения лифта, наличия запросов на остановки и вызовов с разных этажей. Необходимо предусмотреть кнопки вызова на каждом этаже, кнопки с указанием этажей внутри лифта, кнопки для открытия и закрытия дверей, сенсорный датчик для обнаружения препятствий при закрытии дверей, таймер фиксации времени ожидания. Пример продукции:

ЕСЛИ <срок_действия_таймера_истек> И <дверь_открыта> И <ничто_не_препятствует_закрытию двери> ТО <закрыть_дверь>.

2. Сформировать базу знаний продукционной системы для успешного решения задачи «Волк, коза и капуста», где человеку необходимо перевезти с одного берега на другой волка, козу и капусту. Предполагается, что вместе с человеком в лодке помещается только один объект, и человеку приходится охранять козу от волка и капусту от козы.

3. Сформировать базу знаний продукционной системы для управления автомобилем: вождение (увеличение / уменьшение скорости, останов, пропуск другого транспорта) осуществляется в зависимости от типа дороги (главная, второстепенная), указателей на дороге (ограничение на скорость, переезд, пешеходный переход) и т. д.

4. Сформировать базу знаний продукционной системы по диагностике и лечению заболеваний (грипп, ОРЗ, воспаление легких) в зависимости от симптомов (температура, боль в горле, насморк, хрипы в легких).

5. Сформировать базу знаний продукционной системы, позволяющей идентифицировать животных по ряду признаков (длине зубов, цвету шести, длине хвоста, остроте когтей и т. д.).

6. Сформировать базу знаний продукционной системы по определению типа растения, используя дерево решений.

7. Дана база правил. Необходимо построить прямую цепочку вывода для доказательства истинности гипотезы Z и обратную цепочку вывода, чтобы подтвердить или опровергнуть гипотезу Z. A,B,C – известные факты.

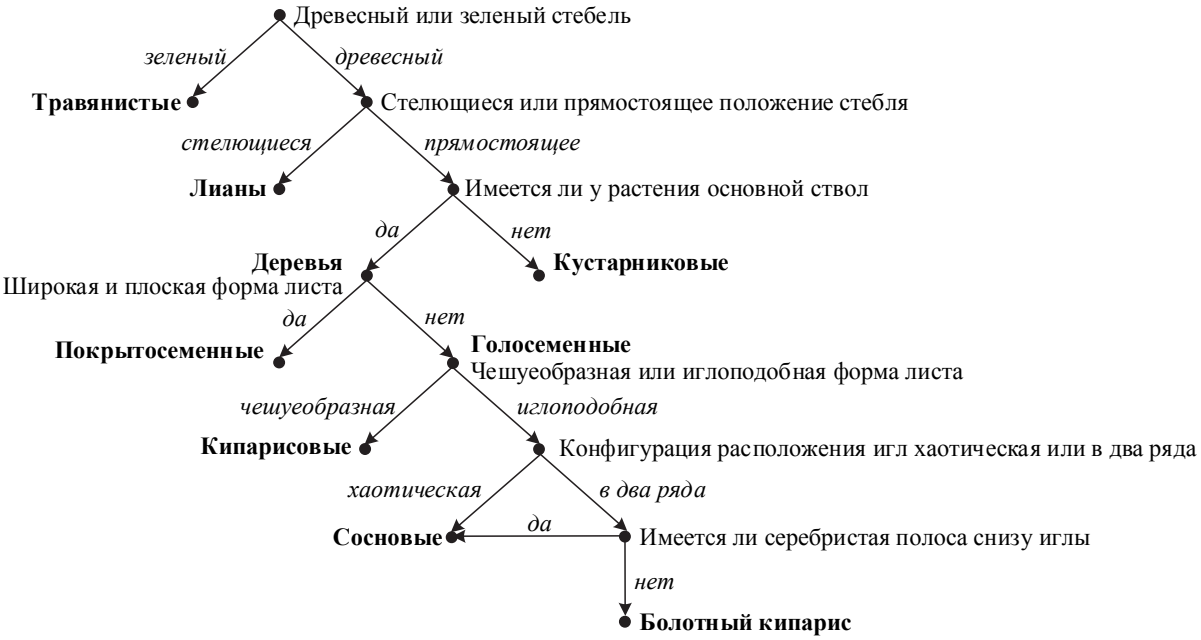
База правил: $П_1$: $A \rightarrow M$; $П_2$: $F \rightarrow Y$; $П_3$: $E \rightarrow P$; $П_4$: $B \rightarrow P$; $П_5$: $B\&C \rightarrow L$; $П_6$: $M\&C \rightarrow X$; $П_7$: $D\&A \rightarrow U$; $П_8$: $X\&A \rightarrow Z$; $П_9$: $P\&Y \rightarrow W$.

8. Дана база правил. Необходимо построить прямую цепочку вывода для доказательства истинности гипотезы C и обратную цепочку вывода, чтобы подтвердить или опровергнуть гипотезу C. H, K – известные факты.

База правил: $П_1$: $A \rightarrow E$; $П_2$: $B \rightarrow D$; $П_3$: $H \rightarrow A$; $П_4$: $E\&G \rightarrow C$; $П_5$: $E\&K \rightarrow B$; $П_6$: $D\&E\&K \rightarrow C$; $П_7$: $G\&K\&F \rightarrow A$.

9. Дана база правил. Необходимо построить прямую цепочку вывода для доказательства истинности гипотезы Z и обратную цепочку вывода, чтобы подтвердить или опровергнуть гипотезу Z. C, E – известные факты.

База правил: $П_1$: $F\&D\&E\rightarrow Z$; $П_2$: $C\rightarrow L$; $П_3$: $D\&E\rightarrow A$; $П_4$: $D\&K\rightarrow Z$; $П_5$: $A\rightarrow F$; $П_6$: $L\rightarrow D$; $П_7$: $K\&E\&G\rightarrow L$.



10. Дана база правил. Необходимо построить И/ИЛИ граф для цели «*пожар*». База правил: $П_1$: ЕСЛИ <*топливо*> И <*выделение тепла*> И <*кислород*> ТО <*пожар*>; $П_2$: ЕСЛИ <*жидкость*> И <*воспламеняемая*> ТО <*топливо*>; $П_3$: ЕСЛИ <*твердое*> И <*горючее*> ТО <*топливо*>; $П_4$: ЕСЛИ <*газ*> И <*горючее*> ТО <*топливо*>; $П_5$: ЕСЛИ <*пламя*> И <*открытое*> ТО <*выделение тепла*>; $П_6$: ЕСЛИ <*электричество*> ТО <*выделение тепла*>;

$П_7$: ЕСЛИ <*трение*> ТО <*выделение тепла*>.

11. Построить И/ИЛИ граф для базы правил: $П_1$: $F\&H\&D\rightarrow C$; $П_2$: $E\&F\rightarrow B$; $П_3$: $B\&C\rightarrow Z$; $П_4$: $A\&G\rightarrow B$; $П_5$: $A\&G\rightarrow H$; $П_6$: $E\rightarrow G$.

12. Построить И/ИЛИ граф для базы правил: $П_1$: $A\&G\rightarrow E$; $П_2$: $E\&F\rightarrow B$; $П_3$: $F\&A\&D\rightarrow C$; $П_4$: $B\&K\rightarrow Z$; $П_5$: $C\rightarrow K$; $П_6$: $E\&C\rightarrow W$; $П_7$: $C\&D\rightarrow W$.

13. Построить И/ИЛИ граф для базы правил: $П_1$: $C\rightarrow M$; $П_2$: $F\&C\&A\rightarrow R$; $П_3$: $A\&E\rightarrow M$; $П_4$: $H\&G\&B\rightarrow E$; $П_5$: $T\&G\rightarrow F$; $П_6$: $H\&T\rightarrow A$; $П_7$: $E\&D\rightarrow L$.

14. Преобразовать знания эксперта в виде суждений в базу знаний продукционной системы. Знания эксперта: «*Я консультирую сотрудников моей фирмы в вопросах одежды. Например, меня спрашивают, нужно ли надевать костюм на встречу. Я отвечаю, что костюмы бывают разные:*

официальный (тройка), деловой (двойка) и спортивный. Спортивный костюм носится только в свободное от работы время: либо дома, если нет халата, либо во время занятий спортом. Официальный костюм надевают на работу и только в том случае, если предусмотрена деловая встреча. Этот костюм должен быть обязательно темного цвета и того же цвета выбирается галстук, в крапинку или горошек. В двойке любых цветов радуги, кроме чёрного, можно щеголять в гостях, но на работу нужно надевать только серый, синий или коричневый костюмы. Галстук с деловым костюмом также обязателен: если костюм светлый, то и галстук тоже светлый, хотя может быть и другого цвета, причем неважно в крапинку, полоску или горошек. Но только глупец может надеть галстук со спортивным костюмом. Спортом можно заниматься в костюме какого угодно цвета, хоть в малиновом».

15. Построить семантическую сеть, отражающую приведенные факты таксономии животных: *млекопитающие относятся к позвоночным*; *плотоядные – млекопитающие с острыми зубами*; *копытные – млекопитающие, имеющие копыта*; *все млекопитающие кормят детей молоком*; *корова – представитель копытных*, *корова умеет мычать*; *семейство псовых и семейство кошачьих относятся к плотоядным*; *кошка – представитель семейства кошачьих*; *собака – представитель семейства псовых*; *собака лает, а кошка умеет мяукать.*

16. Используя различные типы связей, построить семантическую сеть, содержащую в качестве объектов следующие понятия: *аккумулятор*, *концерт*, *лекция*, *казарма*, *медведь*, *начальник*, *обувь*, *отец*, *принтер*, *программа*, *студент*, *компьютер*, *зачет*, *комод*, *корабль*, *платье*.

17. Построить семантическую сеть, представляющую факт *«Ласточка живет в гнезде летом».*

18. Построить семантическую сеть, представляющую факт *«Если закончилась обработка детали станком, робот грузит детали на машину, которая перевозит их на склад».*

19. Построить семантическую сеть, представляющую факт *«Жил-был у бабушки серенький козлик».*

20. Построить семантическую структуру знаний о событии *«Директор завода “Салют” 30.03.2009 г. остановил цех № 4, чтобы заменить оборудование»*.

21. Построить семантическую сеть, отражающую подчиненность сотрудников организации.

22. Построить семантическую сеть «Распределение полномочий и обязанностей между менеджерами различного уровня» на основе следующих знаний: «*Управление осуществляется аппаратом, включающим менеджеров различного уровня управления. Высшее звено управления*

включает генерального директора и главных специалистов (главный конструктор, главный технолог). Среднее звено управления включает начальников цехов и отделов. Низшее звено – мастера и бригадиры. Распределение полномочий, ответственности и обязанностей между менеджерами различного уровня может быть следующим: высшие менеджеры – определение цели, формирование организационной структуры, подбор кадров среднего уровня, распределение прибыли; менеджеры среднего уровня – планирование работ, подбор кадров низшего уровня; менеджеры низшего уровня – организация работ, распределение производственных заданий».

23. Построить семантическую сеть, содержащую информацию о владельцах автомобилей (имя, марка машины, год выпуска, год покупки и др.) и семантическую сеть запроса, позволяющую находить владельцев автомобилей по определенному году покупки машины или году выпуска.

24. Построить семантическую сеть базы знаний, содержащую информацию о супружеских парах (имя, год рождения, место проживания и др.), и семантическую сеть запроса, позволяющую найти супружеские пары с годом рождения раньше 1980.

25. Построить модель фреймовой системы, упрощающую работу агента по недвижимости. Задача агента – оценить примерную стоимость земельных участков, полной информацией о которых он не располагает (исходными данными могут служить форма и размер участка).

26. Построить модель фреймовой системы, помогающей в планировании конференций по различным направлениям: информационным технологиям, экономике, экологии и т. д. Обязательно должны быть учтены: дата, место проведения, тема, организаторы, участники.

27. Построить модель фреймовой системы, которая помогала бы студенту, обучающемуся по индивидуальному плану, организовать свой учебный процесс. Входными данными для фреймовой системы могут быть сведения о специальности, специализации, а также собственных предпочтениях студента по углубленному изучению тех или иных дисциплин. Результатом работы системы может быть перечень рекомендаций по выбору дисциплин индивидуального плана обучения.

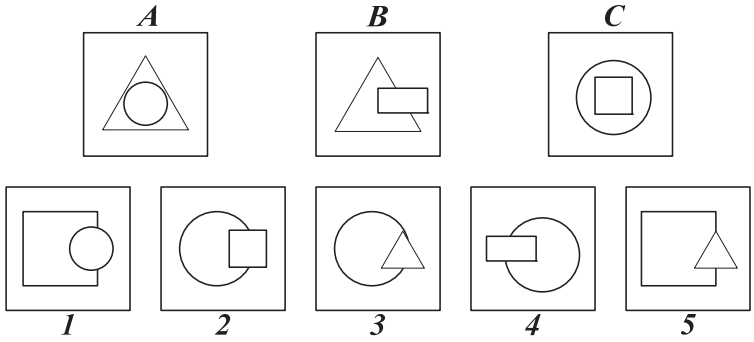
28. Построить модель фреймовой системы по выбору автомобиля, позволяющую, исходя из требований покупателя, его финансовых возможностей, наличия товаров в торговых фирмах и т. п., определить марку автомобиля и рекомендовать наиболее подходящие фирмы.

29. Построить модель фреймовой системы по выбору программного обеспечения для персонального компьютера пользователя. Входными данными могут выступать: цели использования компьютера, доступные ресурсы и пределы стоимости требуемых приложений.

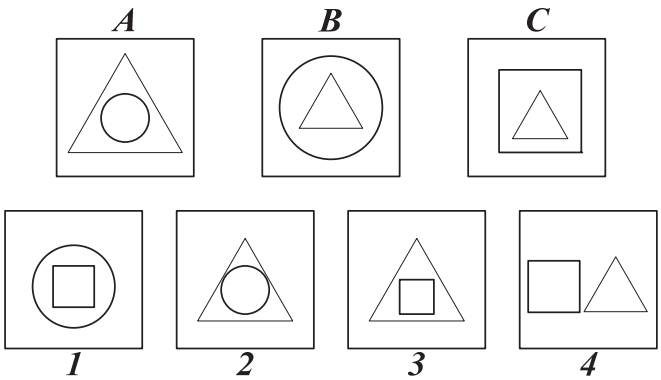
30. Построить модель фреймовой системы по выбору литературы (для написания реферата, курсовой работы и т. п.) в библиотеке. Входные данные: предметная область, ключевые слова, вид литературы (справочная, учебная, художественная, статьи из журналов, методические указания).

31. Используя исчисление предикатов, создать набор выражений, описывающих приведенную загадку. Добавить выражения, описывающие основные отношения в семье, в том числе определение тестя (или свекра), и, используя правило *modus ponens*, доказать заключение «*Я сам себе дедушка*». Загадка: «*Я женился на вдове (назовем ее W), которая имела взрослую дочь (D). Мой отец, который часто навещал нас, влюбился в мою падчерицу и женился на ней. Поэтому мой отец (F) стал моим зятем, а моя падчерица стала моей мачехой. Спустя несколько месяцев моя жена родила сына (S_1), который стал шурином (зятем) моему отцу, а потому моим дядей. Жена моего отца, т. е. моя падчерица, тоже родила сына (S_2)*».

32. Используя алгоритм унификации, требуется среди рис. *1 – 5* найти такой рисунок *X*, который удовлетворял бы условию: *C* так соотносится с *X*, как *A* соотносится с *B*. Необходимо учитывать такие существенные признаки, как размер, форма и относительное местоположение.



33. Используя алгоритм унификации, требуется среди рис. *1 – 4* найти такой рисунок *X*, который удовлетворял бы условию: *C* так соотносится с *X*, как *A* соотносится с *B*. Необходимо учитывать такие существенные признаки, как размер, форма и относительное местоположение.



34. Доказать противоречивость формулы исчисления высказываний:

$$(x \vee \neg y) \,\&\, \neg x \,\&\, (x \vee t \vee y) \,\&\, (x \vee \neg t)$$

35. Доказать противоречивость формулы логики предикатов:

$$G = (P(x) \vee \neg S(x) \vee H(x)) \,\&\, (\neg R(y) \vee S(y)) \,\&\, R(z) \,\&$$
$$\&\, \neg P(z) \,\&\, (\neg H(v) \vee E(v)) \,\&\, \neg E(w).$$

36. Доказать противоречивость формулы логики предикатов:

$$G = (\neg E(x) \vee V(x) \vee S(y, x)) \,\&\, (\neg E(x) \vee V(x) \vee C(y)) \,\&\, P(a) \,\&\, E(a) \,\&$$
$$\&\, (\neg S(y, a) \vee P(y)) \,\&\, (\neg P(x) \vee \neg V(x)) \,\&\, (\neg P(x) \vee \neg C(x)).$$

37. Доказать противоречивость формулы логики предикатов:

$$G = P1(x) \,\&\, (P2(a) \vee P3(x)) \,\&\, (\neg P1(x) \vee \neg P2(x)) \,\&\, (\neg P1(x) \vee \neg P3(a)).$$

38. Дано следующее описание: «*В совершении преступления подозреваются четверо человек: A, B, C, D. Установлено следующее: если A или B виновны, то и подозреваемый C виновен; если A виновен, то B или C тоже виновны; если C виновен, то D виновен; если A виновен, то D виновен*». Используя принцип резолюции, необходимо проверить, «*виновен ли D*».

39. Дано следующее описание: «*Любой студент, который сдаёт экзамен и выигрывает в лотерею, счастлив. Любой удачливый или старательный студент сдаёт экзамен. Любой удачливый студент выигрывает в лотерею. Ваня не относится к числу старательных студентов, но удачлив*». Используя принцип резолюции, необходимо оценить, «*счастлив ли Ваня*».

40. Дано следующее описание: «*Полиция разыскивает всех въехавших в страну, за исключением дипломатов. Шпион въехал в страну, однако распознать личность шпиона может только шпион. Шпион не является дипломатом*». Используя принцип резолюции, необходимо при таких посылках оценить правильность заключения «*среди полицейских имеется шпион*».

41. Дано следующее описание: «*Если иск предъявлен недееспособным лицом, то суд оставляет иск без рассмотрения. Иск предъявлен недееспособным лицом*». Используя принцип резолюции, необходимо при таких посылках оценить правильность заключения «с*уд оставляет иск без рассмотрения*».

42. Дано следующее описание: «*Если подозреваемый совершил эту кражу, то она была тщательно подготовлена или он имел соучастника. Если бы кража была тщательно подготовлена, то если бы он имел соучастника, был бы украден дорогой компьютер. Компьютер остался на месте*». Используя принцип резолюции, необходимо при таких посылках оценить правильность заключения «*подозреваемый не виновен*».

43. Дано следующее описание: «*Всякий, кто может решить эту задачу, – математик. Олег – математик, но не может решить эту задачу*». Используя принцип резолюции, необходимо при таких посылках оценить правильность заключения «*задачу не может решить никто*».

44. Дано следующее описание: «*Каждый из первокурсников знаком с кем-либо из спортсменов. Некоторые из первокурсников не знакомы ни с одним любителем подледного лова*». Используя принцип резолюции, необходимо при таких посылках оценить правильность заключения «*ни один спортсмен не является любителем подледного лова*».

45. Дано следующее описание: «*Если деталь обрабатывалась на токарном станке, то она обрабатывалась и на фрезерном. Деталь обрабатывалась на токарном станке*». Используя принцип резолюции, необходимо при таких посылках оценить правильность заключения «*деталь обрабатывалась на фрезерном станке*».

46. Перед нами три девушки: Лиза, Мария и Диана. Предположим, что юноша говорит следующее: «*Я люблю, по крайней мере, одну из этих трех девушек. Если я люблю Лизу, а не Диану, то я также люблю Марию. Я люблю либо Диану и Марию, либо не люблю ни одну из них. Если я люблю Диану, то я также люблю Лизу*». Используя принцип резолюции, необходимо определить, любит ли юноша Диану.

Представление нечётких знаний

1. Дана база правил и коэффициенты уверенности: $П_1$: $X\&Y\to Z$; $П_2$: $W\to Z$; $КУ(X) = 0{,}4$; $КУ(Y) = 0{,}9$; $КУ(W) = 0{,}8$; $КУ(П_1) = 0{,}6$; $КУ(П_2) = 0{,}5$, где X – «*падает давление*»; Y – «*дует ветер*»; Z – «*пойдет дождь*»; W – «*на небе большая черная туча*». Необходимо, используя коэффициенты уверенности Шортлиффа для правил и фактов, найти коэффициент уверенности факта Z.

2. Дана база правил и коэффициенты уверенности: $П_1$: $A\vee(\neg B\&C) \to D$; $КУ(П_1) = 0{,}9$; $КУ(A) = 0{,}5$; $КУ(B) = 0{,}1$; $КУ(C) = 0{,}7$. Необходимо, используя коэффициенты уверенности Шортлиффа для правил и фактов, найти коэффициент уверенности факта D.

3. Дана база правил и коэффициенты уверенности: $П_1$: *Если давление падает* (*0,2*) *и дует ветер* (*0,1*), *то пойдет дождь* (*0,18*); $П_2$: *Если на небе большая черная туча* (*0,9*), *то пойдет дождь* (*0,36*); $П_3$: *Если низкая облачность* (*0,3*) *и влажность высокая* (*0,2*), *то пойдет дождь* (*0,42*). $КУ(П_1)=0{,}6$; $КУ(П_2)=0{,}4$; $КУ(П_3)=0{,}8$. Необходимо, используя коэффициенты уверенности Шортлиффа для правил и фактов, найти коэффициент уверенности факта «*пойдет дождь*».

4. Дана база правил и коэффициенты уверенности: $П_1$: $A\vee\neg C\to K$; $П_2$: $K\&(B\vee A)\to W$; $КУ(П1) = 0{,}6$; $КУ(П2) = 0{,}7$; $КУ(A) = 0{,}2$; $КУ(B) = 0{,}6$;

КУ(*C*) = –0,5. Необходимо, используя коэффициенты уверенности Шортлиффа для правил и фактов, найти коэффициент уверенности факта *W*.

5. Дана база правил и коэффициенты уверенности: $П_1$: $\neg F\&D \rightarrow R$; $П_2$: $\neg A\&K \rightarrow R$; *КУ*(*П*1) = 0,4; *КУ*(*П*2) = 0,8; *КУ*(*F*) = 0,3; *КУ*(*D*) = 0,7; *КУ*(*A*) = –0,1; *КУ*(*K*) = 0,6. Необходимо, используя коэффициенты уверенности Шортлиффа для правил и фактов, найти коэффициент уверенности факта *R*.

6. Дана база правил и коэффициенты уверенности: $П_1$: $A \vee (\neg B\&C) \rightarrow D$; $П_2$: $B\&\neg C \rightarrow D$; *КУ*(*П*1) = 0,9; *КУ*(*П*2) = 0,5; *КУ*(*A*) = 0,1; *КУ*(*B*) = 0,7; *КУ*(*C*) = –0,2. Необходимо, используя коэффициенты уверенности Шортлиффа для правил и фактов, найти коэффициент уверенности факта *D*.

7. На основе метода К. Нейлора разработать структуру базы знаний системы по определению профессии исходя из особенностей характера человека, его интересов, образа жизни и др.

8. На основе метода К. Нейлора разработать структуру базы знаний системы по предсказанию продолжительности жизни исходя из показателей состояния здоровья, характеристик образа жизни, вредных привычек и т. д.

9. На основе метода К. Нейлора с использованием существующих тестов разработать структуру базы знаний системы по психодиагностическому тестированию, позволяющую строить личностный портрет человека, его поведенческие особенности, стиль отношений в коллективе и т. п.

10. На основе метода К. Нейлора разработать структуру базы знаний системы по выбору компьютерной техники (устройств, конфигураций) исходя из требований к решаемым задачам, финансовых возможностей потребителя, наличия товаров.

11. На основе метода К. Нейлора разработать структуру базы знаний системы по выбору автомобиля», позволяющую, исходя из требований покупателя, его финансовых возможностей, определить марку автомобиля.

12. На основе метода К. Нейлора разработать структуру базы знаний системы по выбору места отдыха исходя из предпочитаемого вида отдыха, климатических особенностей, удаленности места, продолжительности пребывания, финансовых возможностей и т. д.

13. Пусть эксперт определяет толщину изделия с помощью понятий: «*маленькая толщина*», «*средняя толщина*», «*большая толщина*», при этом минимальная толщина десять миллиметров, а максимальная восемьдесят миллиметров. Необходимо аналитически и графически продемонстрировать, что «*средняя толщина*» $\cup$ «*большая толщина*» $\neq \neg$ «*маленькая толщина*».

14. Пусть *A* и *B* – нечёткие множества (факты), заданные на базовых множествах *X* и *Y* соответственно:

$$A = \frac{0{,}3}{x_1} + \frac{0}{x_2} + \frac{1}{x_3} + \frac{0{,}5}{x_4} + \frac{0{,}9}{x_5};\ B = \frac{0}{y_1} + \frac{0{,}3}{y_2} + \frac{0{,}4}{y_3} + \frac{0{,}7}{y_4} + \frac{1}{y_5} + \frac{0{,}8}{y_6} + \frac{0{,}6}{y_7}.$$

Требуется вычислить нечёткое отношение, заданное правилом $S: A \to B$.

15. Пусть дано правило $R: F \to Q$, заданное нечётким отношением:

$$R(\sqsubseteq U \times V) = \begin{array}{c} X\backslash Y \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \end{array} \begin{array}{c} \begin{array}{cccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \end{array} \\ \begin{bmatrix} 0{,}1 & 0{,}1 & 0{,}1 & 0{,}1 & 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}2 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 \\ 0{,}1 & 0{,}2 & 0{,}5 & 0{,}7 & 0{,}7 & 0{,}7 \\ 0{,}1 & 0{,}2 & 0{,}5 & 0{,}1 & 0{,}8 & 0{,}8 \\ 0{,}1 & 0{,}2 & 0{,}5 & 0{,}8 & 0{,}9 & 1 \end{bmatrix} \end{array}$$

F и Q – нечёткие множества (факты), заданные на базовых множествах X и Y соответственно. Исходный факт:

$$F' = \frac{0}{x_1} + \frac{0{,}6}{x_2} + \frac{1}{x_3} + \frac{0{,}8}{x_4} + \frac{0{,}4}{x_5}.$$

Требуется найти Q', согласно нечёткому *modus ponens*.

16. Дано знание в виде нечёткого правила $R: F \to G$: «*Если уровень воды в резервуаре высокий, то открыть пробку*» и исходный факт F' «*уровень воды не очень высокий*». Необходимо, используя нечёткий *modus ponens*, получить нечёткий вывод G'.

17. Дано знание в виде нечёткого правила $R: F \to G$: «*Если температура воды высокая, то расход воды низкий*» и исходный факт F' «*температура воды не очень высокая*». Необходимо, используя нечёткий *modus ponens*, получить нечёткий вывод G'.

18. Дано знание в виде нечёткого правила $R: F \to G$: «*Если груз тяжелый, то скорость машины низкая*» и исходный факт F' «*груз не очень тяжелый*». Необходимо, используя нечёткий *modus ponens*, получить нечёткий вывод G'.

Учебное издание

Пенькова Татьяна Геннадьевна
Вайнштейн Юлия Владимировна

МОДЕЛИ И МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебное пособие

Редактор *Н. А. Варфоломеева*
Компьютерная верстка *Н. Г. Дербенёвой*

Подписано в печать 28.05.2019. Печать плоская. Формат 60×84/16
Бумага офсетная. Усл. печ. л. 7,25. Тираж 100 экз. Заказ № 7060

Библиотечно-издательский комплекс
Сибирского федерального университета
660041, Красноярск, пр. Свободный, 82а
Тел. (391) 206-26-16; http://bik.sfu-kras.ru
E-mail: publishing_house@sfu-kras.ru