Министерство образования и науки Республики Казахстан

Павлодарский государственный университет им. С. Торайгырова

Факультет физики, математики и информационных технологий

Опорный конспект лекции

по дисциплине «Системы искуственного интеллекта»

Для специальности 050602 «Информатика»

Павлодар

УТВЕРЖДАЮ
Декан факультета ФМ иИТ
_____ Тлеукенов С.К.
«__»_____2008г.

Составитель: старший преподаватель Аканова А.С.

Кафедра информатики и информационных систем

Опорный конспект лекции

по дисциплине Системы искуственного интеллекта
для специальности 050602 «Информатика»

форма обучения: дневная и заочная на базе ОСО, СПО, ВПО

Опорный конспект лекции разработан на основании рабочей программы дисциплины

Рекомендована на заседании кафедры от «__» _____200 протокол №__

Зав. кафедрой _____Ж.К. Нурбекова
(подпись, Ф.И.О.)

Одобрено методическим советом факультета ФМиИТ

«_____»_____200__г., протокол № _____

Председатель МС_____ А.Т.Кишубаева
(подпись)

Тема 1 Концептуальные основы искуственного интеллекта. Базовые понятия ИИ. Задачи и методы ИИ.

Терминология. Философские проблемы систем ИИ.
История развития систем ИИ.
Место искуственного интеллекта в информационных технологиях.
Тенденция развитя систем ИИ.
Распознавание образов.
Постановка задачи распознавания образов.
Методы распознавания образов.
Структурный подход к анализу образов.
Выделение признаков.

Многие виды умственной деятельности человека, такие, как написание программ для вычислительной машины, игра в шахматы, занятие математикой, ведение рассуждений на уровне здравого смысла, требуют наличия определенного "интеллекта". На протяжении последних десятилетий было построено немало компьютерных систем, способных решать подобные задачи. Имеются системы, способные диагностировать заболевания, планировать синтез сложных химических соединений, решать дифференциальные уравнения в символьном виде, анализировать электронные схемы, понимать ограниченный объем человеческой речи и естественного языкового текста. Можно сказать, что такие системы обладают в некоторой степени искусственным (созданным человеком) интеллектом. Поэтому область исследований, где проводится работа по построению таких систем, получила название *искусственный интеллект* (ИИ). При реализации интеллектуальных функций непременно присутствует и используется информация, называемая знаниями, поэтому системы искусственного интеллекта можно рассматривать и как *системы обработки знаний*.

Интерес к созданию искусственного разума имеет более чем 2000-летнюю историю, но, чтобы перейти от фантазий (подобных легенде о Пигмалионе) и мистификаций (вроде механического шахматиста) к их воплощению в реальность, надо было подготовить надлежащий математический и технический инструментарий. Решение этих задач обеспечили успехи в развитии математической логики, теории алгоритмов и изобретение электронных вычислительных машин

Математическая логика – это формальная логика, применяющая математические методы для исследования законов мышления, это качественно новая ступень в развитии формальной логики, изучающей закономерности получения выводимого знания. Первой ступенью можно считать традиционную логику, основы которой были заложены величайшим мыслителем античного мира Аристотелем (384 - 322 г.г. до н.э.). Ее путь к современной математической логике можно (весьма неполно) проследить, отметив хотя бы некоторые, наиболее важные достижения, имеющие отношение к математизации логики.

Р. Луллий (1235 - 1315) - испанский философ, высказал мысль о возможности математизации логических операций и сконструировал "логическую машину" из семи кругов, которая позволяла производить простейшие логические вычисления.

Р. Декарт (1596 - 1650) - французский философ и математик, ввел понятия переменной и функции, без которых была бы невозможна современная математика.

Дж. Буль (1815 - 1864) - английский математик, построил первый вариант алгебры логики, позволивший сводить процесс получения умозаключений к решению логических уравнений.

Г. Фреге (1848 - 1925) - немецкий логик и математик, ввел в аппарат логики понятия кванторов всеобщности и существования.

Дж. Пеано (1858 - 1932) - итальянский математик и логик, ввел современную символику теории множеств и применил математическую логику для обоснования арифметики, создав систему аксиом для арифметики натуральных чисел.

К. Гёдель (1906 - 1978) - австрийский математик и логик, с 1940 г. работавший в США, в 1931 г. доказал теорему о неполноте формализованного знания, из которой следует, что не все содержательно истинные предложения могут быть доказаны или опровергнуты в формальных аксиоматических системах.

Современное состояние математической логики связано с именами Б.Рассела, Я. Лукасевича, Е. Поста, А.А. Маркова, П.С. Новикова и других. Основоположник кибернетики, Норберт Винер считал, что возникновение кибернетики было бы невозможно без математической логики. С этим

трудно не согласиться: сегодня математическая логика находит практическое применение при создании как технических средств, так и программного обеспечения разнообразных современных кибернетических систем.

Понятие алгоритма также является не только одним из основных, но и достаточно древним понятием. Уже на ранних этапах развития математики производились вычисления, носившие механический характер и выполнявшиеся по заранее определенным правилам. Сам термин "алгоритм" (или "алгорифм") происходит от латинской формы Algorithmi имени выдающегося средневекового узбекского математика Мухамеда бен Муса аль-Хорезми, предложившего в 825 г. правила выполнения четырех арифметических действий в десятичной системе счисления.

Начиная с двадцатых годов прошлого века, задача точного определения понятия алгоритма становится одной из центральных проблем в математике, поскольку в этот период были сформулированы математические задачи, возможность построения алгоритмов решения которых вызывала сомнения. Дело в том, что для доказательства существования алгоритма решения некоторого класса задач достаточно найти хотя бы один алгоритм, в то время как невозможность построения такого алгоритма можно установить лишь с привлечением математических методов доказательства, что невозможно сделать без математически точного понятия алгоритма. Однако до 30-х годов прошлого века понятие алгоритма определялось интуитивно и имело больше методологическое, чем математическое значение. Математически строгие формулировки понятия алгоритма были даны в середине 30-х годов прошлого века в двух формах: на основе особого класса функций, названных рекурсивными (Д. Гильберт, К. Гёдель, А. Чёрч, С. Клини), и на основе специального класса гипотетических машин, названных машинами Тьюринга (Э. Пост, А. Тьюринг). В 50-х годах А.А. Марковым было предложено также математически строгое понятие нормального алгоритма как особого соответствия между словами, описывающими исходные данные и результаты решения задачи. В дальнейшем была установлена равносильность этих трех способов формализации понятия алгоритма в том смысле, что любой алгоритм, если он описан одним из трех способов, всегда может быть также описан и каждым из двух остальных способов.

С использованием этих определений было доказано, что существуют математические проблемы, для решения которых в принципе не могут быть построены алгоритмы. Такие проблемы были названы алгоритмически неразрешимыми. Среди них оказались многие проблемы, не поддающиеся решению, в том числе древние геометрические проблемы на построение с помощью циркуля и линейки: о квадратуре круга (построить квадрат, равновеликий данному кругу), о трисекции угла (разделить произвольный данный угол на три равные части), об удвоении куба (по стороне данного куба построить сторону куба, объем которого вдвое больше данного). С 50-х годов прошлого столетия (в связи с появлением и началом широкого применения электронных вычислительных машин) математическая логика и теория алгоритмов получают интенсивное развитие и находят применение всюду, где возникает необходимость автоматизации процессов решения каких-либо задач, в том числе и в моделировании интеллекта.

Эволюция первых исследований по ИИ

Здесь мы очень кратко рассмотрим историю первых работ в области искусственного интеллекта в век компьютеров, сократив её до четырех этапов (табл. 1.1). Тем самым мы её, конечно, существенно упростим, но ключевые моменты развития этих исследований в ней будут отражены.

Таблица 1.1

Годы	Парадигма	Исполнители	Система
50-е	Нейронные сети	Розенблатт (Винер, Маккаллок)	PERCEPTRON
60-е	Эвристический поиск	Ньюэл , Саймон (Шеннон, Тьюринг)	GPS
70-е	Представление знаний	Шортлифф (Минский, Маккарти)	MYCIN
80-е	Обучающиеся машины	Ленат (Сэмюэл, Холланд)	EURISCO

В колонке "Парадигма" помещен тот ответ, который вы получили бы в то время, если бы

спросили специалиста по искусственному интеллекту, чему посвящены исследования в этой области.

В колонке "Исполнители" показаны одна-две основные фамилии ученых, которые, по-видимому, характеризуют собой дух искусственного интеллекта в этот период (под основными исполнителями в круглых скобках помещены фамилии теоретиков, заложивших основы исследований в рамках соответствующей парадигмы).

Наконец, в колонке "Система" приведены типичные системы (далеко не все и не обязательно лучшие из них), в которых нашли отражение тенденции или модные в соответствующий период истории течения.

Нейронные сети. В 50-х годах исследователи в области искусственного интеллекта пытались строить разумные машины, имитирующие мозг. Разумеется, эти попытки оказались тщетными, поскольку аппаратные и программные средства были ещё непригодны для такого дела. Типичным примером систем такого типа является указанный в табл.1.1 PERCEPTRON. Он представлял собой самоорганизующийся автомат, который можно считать грубой моделью работы сетчатки глаза человека. Его можно было научить узнавать образы, но, как позже показали Минский и Пейперт, это был лишь ограниченный класс зрительных образов.

Энтузиазм в отношении систем, подобных системе Розенблатта, был основан на теоретических работах кибернетиков Норберта Винера и Уоррена Маккалока по абстрактным нейронным сетям. Считалось, что если взять сильно связанную систему модельных нейронов, которой вначале ничего не известно, применить к ней программу тренировки из поощрений и наказаний, то в конце концов она будет делать всё, что требуется ее создателю. При этом не принималось во внимание, что мозг человека содержит 10^{10} нейронов, каждый из которых по сложности соответствует примерно микрокомпьютеру, реализованному на одном кристалле и способному выполнять сравнительно небольшую программу.

Поскольку экспериментальные результаты, полученные Розенблаттом, были недостаточно хороши, воображением специалистов по искусственному интеллекту овладела новая идея.

Эвристический поиск. Новые рубежи в конце 50-х - начале 60-х годов наметили Аллен Ньюэлл и Герберт Саймон из университета Карнеги-Меллона (США). Они решили, что моделировать следует не мозг, а мышление человека. Ньюэл и Саймон считали, что мышление человека основано на операциях над символами (сравнение, поиск, модификация символов и т.п.), которые могут выполняться и компьютером. Решение задач им представлялось как поиск путем перебора вариантов применения возможных операций. Начали Ньюэл и Саймон с разработки программы доказательства теорем математической логики, затем перешли к программам моделирования шахматной игры. Необозримость множества возможных вариантов развития каждой шахматной позиции заставила их обратиться к изучению человеческих способов сокращения перебора за счет применения эвристик (основанных на опыте правил выбора приоритетных направлений поиска решения), что сделало эвристический поиск центральным в их подходе. В процессе этих исследований был найден ряд приемов сокращения перебора: выдвижение гипотез и возврат при неудаче ("поиск в глубину"), локальная оптимизация ("подъем на холм"), иерархическая декомпозиция (разбиение) сложных задач на всё более простые подзадачи, рекурсия.

Полученные результаты позволили Ньюэлу и Саймону переключить внимание на поиск общих методов, применимых к обширному спектру задач, что привело их к созданию системы GPS (General Problem Solver - общий решатель задач). Универсальность системы GPS заключалась в том, что "не было конкретного указания, к какой области относится задача", пользователь должен был сам задать *проблемную среду* в терминах ее объектов и применимых к этим объектам операций. Но на деле эта универсальность относилась лишь к ограниченной области математических головоломок и различных игр с относительно небольшим множеством состояний и четко определенных формальных правил. Как и большинство ее современниц, система GPS функционировала в таком формализованном микромире, где возникающие проблемы (например, задача "Ханойская башня" или задача о миссионерах и людоедах), с точки зрения людей, проблемами и не являются.

Авторы проекта создания универсального решателя задач также стали жертвой неоправданного оптимизма, но работа над ним привнесла в исследования по ИИ ряд упомянутых выше ценных идей.

Представление и использование знаний. Система GPS не была способна решать задачи

реальной сложности. Поэтому в 70-х годах группа ученых во главе с Эдвардом Фейгенбаумом из Станфордского университета отказалась от безнадежного поиска эффективных универсальных эвристик и поставила задачу добиться эффективности за счет специализации систем ИИ и использования знаний и навыков (приемов и неформальных правил) специалистов. На этом пути были созданы первые экспертные системы. Хотя система анализа масс-спектрограмм химических соединений DENDRAL и является прототипом всех экспертных систем, но её дочерняя система MYCIN, созданная под руководством медика Шортлиффа, оказала гораздо большее влияние на идеологию и развитие экспертных систем.

Система MYCIN - это компьютерная система, способная диагностировать бактериальные инфекции крови и давать рекомендации по лекарственной их терапии. Она положила начало созданию целой серии медико-диагностических систем, используемых в рутинной клинической практике. Ряд характеристик системы MYCIN стали отличительной чертой всех экспертных систем.

Во-первых, медицинские знания MYCIN представляют собой сотни правил-продукций, подобных следующему:

ЕСЛИ (1) инфекция представляет собой первичную бактеримию,

И (2) место взятия культуры является стерильным,

И (3) предполагается, что организм проник через желудочно-кишечныи тракт,

ТО (с уверенностью 0,7) этот организм носит бактериальный характер.

Во-вторых, Шортлифф разработал схему с использованием "коэффициентов уверенности", позволяющую системе на основе даже ненадежных данных приходить к правдоподобным заключениям.

В-третьих, система MYCIN была способна объяснять пользователю (лечащему врачу) ход своих рассуждений и причины задаваемых ему вопросов, благодаря чему существенно снималось недоверие к машинному диагнозу. Достигнутая системой степень "дружественности" по отношению к пользователю была прямым следствием представления медицинских знаний в виде правил-продукций: если пользователь хотел узнать, почему ему был задан данный вопрос, система просто сообщала ему последовательность правил, приведшую к этому вопросу.

В-четвертых, и самое главное, система MYCIN действительно эффективно работала, выполняя работу, которой люди обучаются годами.

В-пятых, на основе системы MYCIN была создана первая "оболочка" экспертных систем. Так система PROSPECTOR для разведки полезных ископаемых была построена путем замены медицинских знаний (системы правил-продукций) на геологические. То, что при этом перешло в новую экспертную систему неизменным, было названо "оболочкой" (от английского термина shell - скорлупа, оболочка).

Приобретение знаний путем обучения машин. Итак, 80-е годы оказались периодом очередного раунда необузданного оптимизма, охватившего теперь не только замкнутую в себе область ИИ, но и людей, занятых информатикой, вычислительной техникой и обработкой данных. На этот раз магическим катализатором стали знания, потому что от обширности и качества знаний зависит успех экспертной системы. Сейчас можно констатировать, что на этот раз энтузиазм был оправданным, хотя и не вполне, т.к. на пути к массовому производству знаний для компьютера были выявлены значительные трудности. Пытаясь снять эти трудности, Дуг Ленат (Станфордский университет), использовал алгоритмы обучения Сэмюэла и Холланда, основанные на принципах естественного отбора и генетики, для создания машинной обучающейся системы EURISCO, способной автоматически улучшать и расширять свой запас эвристических правил.

Нет сомнения, что программы машинного обучения станут важнейшим этапом в развитии искусственного интеллекта в последующие десятилетия, так как до сих пор перенесение знаний и навыков специалистов в базу знаний экспертной системы остается сложной и долгой процедурой.

Созданием EURISCO завершился первый виток спирали в развитии искусственного интеллекта, поскольку машинное обучение и было той проблемой, с которой начинали кибернетики 50-х годов прошлого столетия.

Тема 2 Методы представления знаний. Логика предикатов первого порядка. Знания и данные. Способы структурирования и классификация знаний. Логические и эвристические методы представлений знаний. Понятие предика, формулы, кванторов общности и существования. Интерпретация формул в логике предикатов первого порядка.

Правила продукции. Структура правил-продукции. Методы логического вывода: прямой и обратный. Стратегии выбора правил при логическом выводе.

Семантические сети и фреймы. Основные понятия семантических сетей.Типы отношений в семантических сетях. Принципы обработки информации в семантических сетях. Основные понятия фрейма, наследование свойств. Сети фреймов.

Человек для решения какой-либо задачи использует собственные и другие знания. Для выполнения той же работы с помощью компьютера необходимо этим знаниям придать определенную форму, чтобы представить их в компьютере, а также составить программу для компьютера, решающую задачу с использованием знаний. В самом общем плане формы представления знаний делят на императивные, декларативные и комбинированные.

Императивные формы представления знаний - это традиционные (процедурные) способы описания процессов решения задач в виде последовательностей операций над данными, совершаемых согласно заданным алгоритмам или формулам (как, например, последовательность операций вычисления выражения $y:=x*(z+v$)). В процедурах знания (связи, зависимости, законы) представлены (учтены) неявно: в организации вычислительного процесса, в структуре программы решения задачи, в характере и последовательностях операций. По этой причине императивная форма представления знаний наиболее эффективна с вычислительной точки зрения (по затратам времени и памяти на решение задачи), поскольку в процедурах поиска решения глубоко учитывается специфика конкретной проблемной области (ПО) и решаемой задачи, что является важным фактором при создании систем, работающих в реальных условиях (в системах реального времени). Главный недостаток этой формы представления знаний - сложность внесения изменений, что делает ее непригодной для использования в слабо изученных и изменяющихся ПО.

Декларативные формы представления знаний разработаны в рамках исследований по искусственному интеллекту. Их отличительная особенность в том, что знания относительно ПО в этом случае описываются в виде совокупности утверждений, характеризующих состав, свойства, законы строения и поведения (например, закон Ома $I=U/R$ в электротехнике связывает зависимостью величины тока, напряжения и сопротивления, позволяющей вычислить каждую из этих величин, если заданы две другие). Знания в этой форме можно использовать для решении любых задач, связанных с данной ПО. Постановка задачи в этом случае сводится к описанию свойств искомого решения (цели), способ же поиска решений (механизм поиска, "машина" вывода) универсален и не зависит ни от поставленной задачи, ни даже от ПО, что весьма важно при описании слабо изученных и изменяющихся ПО. Главный недостаток этой формы представления знаний - низкая вычислительная эффективность (по затратам времени и памяти), поскольку в процедурах поиска решения не учитывается специфика решаемой задачи и ПО, что делает эту форму непригодной для применения в системах реального времени.

Комбинированные формы описания знаний создаются, чтобы преодолеть недостатки и сохранить достоинства императивной и декларативной форм. Достигается это за счет того, что хорошо обоснованная, устойчивая и формализованная часть знания воплощается в эффективных процедурах, а слабо изученная и изменчивая составляющая знания представляется в декларативной форме. Главный недостаток комбинированных форм представления знаний - трудность их теоретизации из-за их составного характера, что препятствует созданию теоретически обоснованных методов построения баз знаний и методов поиска решений с использованием таких форм представления знаний.

Императивные формы представления знаний широко используются в различных языках программирования, изучаются при освоении этих языков. Их рассматривать мы не будем, а познакомимся с типичными для систем искусственного интеллекта и экспертных систем моделями представления знаний. Основными моделями представления знаний в интеллектуальных системах в настоящее время принято считать:

- *логические модели*;
- *продукционные модели*;
- *сетевые модели*;
- *фреймовые модели*.

Чисто декларативные формы представления знаний реализовать весьма сложно, но наиболее близкими к ним вариантами являются логические и продукционные модели. В сетевых и фреймовых моделях находит воплощение идея комбинированных форм представления знаний.

Логические модели

Все предметы, взаимосвязи, события и процессы, составляющие основу необходимой для решения задачи информации, называют *предметной областью*. Мысленно предметная область представляется состоящей из реальных компонент, называемых *сущностями*. Сущности каждой конкретной предметной области характеризуются свойствами и находятся между собой в определенных *отношениях*. Свойства и отношения между сущностями описывают посредством *суждений*. Суждения в каждом языке (как естественном, так и формальном) выражают в виде *предложений*.

Языки, предназначенные для описания предметных областей, называются *языками представления знаний*. Универсальным языком представления знаний является естественный язык, но применение естественного языка для машинного представления знаний наталкивается на ряд препятствий, главным из которых является отсутствие формальной семантики естественного языка. *Семантика* – это смысловое значение единиц языка. В естественном языке смысл слов и выражений часто зависит от текста, в котором они встречаются. Кроме того, естественные языки - это "живые", постоянно меняющиеся и развивающиеся языки.

Для представления математического знания пользуются формальными логическими языками – исчислением высказываний и исчислением предикатов. Эти языки имеют ясную формальную семантику, и для них разработаны формальные методы логического вывода. Поэтому исчисление предикатов было первым логическим языком, который стали применять для формального описания пригодных для этого предметных областей. Описания предметных областей, выполненные в логических языках, называются *логическими моделями*. Логические модели, построенные с применением языков логического программирования, довольно широко применяются в базах знаний систем искусственного интеллекта и экспертных систем.

Формальные системы. Многие научные теории (не обязательно математические) строятся по следующему принципу. Сначала предлагаются некоторые основные понятия и некоторые исходные законы (аксиомы), присущие основным понятиям. Далее формулируются производные понятия и по определенным правилам доказываются некоторые утверждения (теоремы), относящиеся к основным и производным понятиям. Совокупность основных и производных понятий, аксиом и теорем, построенная таким способом, называется *аксиоматической системой*.

Часто аксиомы (а значит, и теоремы) аксиоматической системы сохраняют истинность при замене одних основных понятий другими (как, например, в теории колебаний, которая находит применение в механике, электронике, оптике). Это позволяет рассматривать аксиоматические системы с двух позиций: синтаксически (принципы построения правильных и истинных предложений) и семантически (связь смысла правильных и истинных предложений со смыслом основных понятий).

Для исследования синтаксиса аксиоматической системы требуется ее полная формализация, т.е. символическое представление основных и производных понятий, аксиом, правил вывода и теорем. Поэтому *формальная аксиоматическая теория (формальная система)* - это синтаксический аспект (сторона) аксиоматической системы. Точное же определение понятия формальной аксиоматической теории включает следующие компоненты.

Во-первых, каждая формальная аксиоматическая теория должна иметь свой *формальный язык*. Формальный язык считается полностью определенным, когда задано (счётное) множество его символов и описаны формулы языка. Любая конечная последовательность символов языка называется выражением этого языка. Среди всех возможных выражений выделяются формулы языка, под которыми подразумеваются правильно построенные, утверждающие нечто осмысленное предложения языка.

Во-вторых, каждая формальная аксиоматическая теория должна иметь свою *систему аксиом* - подмножество заведомо истинных формул, из которых по правилам теории могут быть выведены все истинные предложения этой теории (обычно к системе аксиом предъявляются требования непротиворечивости, независимости и полноты, среди которых обязательным является лишь требование непротиворечивости).

В-третьих, каждая формальная аксиоматическая теория должна располагать конечным множеством *правил вывода*. Каждое правило вывода содержит формулы-посылки и формулу-заключение, выводимую при определенных этим правилом условиях из формул-посылок. Формула-заключение называется непосредственным следствием формул-посылок по данному правилу вывода.

Доказательством в формальной аксиоматической теории называется всякая последовательность формул, в которой каждая формула есть либо аксиома теории, либо непосредственное следствие каких-либо предыдущих формул этой последовательности по одному из правил вывода данной теории. Формула *A* называется *теоремой* теории (обозначается $\vdash A$) тогда и только тогда, когда существует доказательство, в котором *A* является заключительной формулой.

Выводом из множества гипотез (посылок) *Г* в формальной аксиоматической теории называется всякая последовательность формул, в которой каждая формула есть либо аксиома теории, либо гипотеза из множества *Г*, либо непосредственное следствие каких-либо предыдущих формул этой последовательности по одному из правил вывода данной теории. Формула *A* называется *следствием множества гипотез Г* (обозначается $Г \vdash A$) тогда и только тогда, когда существует вывод из множества гипотез *Г*, в котором *A* является заключительной формулой.

Предлагаемый далее в этом разделе материал представляет собой описание исчисления высказываний и исчисления предикатов как конкретных примеров формальных аксиоматических теорий.

Исчисление высказываний - наиболее простой пример формальной аксиоматической теории в логике. Существует несколько исчислений высказываний (Д. Гильберта, С. Клини, Ж. Фреге, П.С. Новикова). Мы рассмотрим исчисление высказываний Я. Лукасевича (исчисление L), использующее функционально полную систему связок, состоящую из отрицания ($\neg$) и импликации ($\rightarrow$). Исчисление L описывает множество тождественно истинных формул (тавтологий), что соответствует реальному положению вещей в математике, где теоремы как высказывания являются тождественно истинными формулами.

Язык исчисления L включает перечень употребляемых символов и определение понятия формулы. *Символами исчисления* L являются (счётное) множество символов пропозициональных (логических) переменных $x_i, i = 1,2,K$, символы логических операций отрицания $\neg$ и импликации $\rightarrow$, а также вспомогательные символы скобок (,). Знаки других логических операций рассматриваются здесь как обозначения формул исчисления L, выражающих эти операции (например, формулу $(A \vee B)$ можно считать обозначением эквивалентной ей формулы $(\neg A \rightarrow B)$, построенной с помощью лишь отрицания и импликации). *Понятие формулы исчисления* L определяется индуктивно: формулами полагаются сначала переменные $x_i, i = 1,2,K$, а затем все выражения вида $(\neg A)$ и $(A \rightarrow B)$, где в качестве A и B могут выступать любые из уже имеющихся формул.

Систему аксиом исчисления L составляет бесконечное (счетное) множество формул исчисления, построенных с использованием следующих трех *схем аксиом*:

$(A1)\ (Z_1 \rightarrow (Z_2 \rightarrow Z_1))$;

$(A2)\ ((Z_1 \rightarrow (Z_2 \rightarrow Z_3)) \rightarrow ((Z_1 \rightarrow Z_2) \rightarrow (Z_1 \rightarrow Z_3)))$;

$(A3)\ ((\neg Z_1 \rightarrow \neg Z_2) \rightarrow ((\neg Z_1 \rightarrow Z_2) \rightarrow Z_1))$.

Аксиомой считается каждая формула, полученная подстановкой в схему аксиом любых формул исчисления L вместо переменных Z_1, Z_2, Z_3.

Правилом вывода в исчислении L является *modus ponens* (правило заключения), согласно которому из данных формул вида A и $(A \rightarrow B)$ выводится формула B.

Исчисление предикатов первого порядка и теории первого порядка. Предикат - это логическая функция от нелогических аргументов (например, функция *sum(x,y,z)* принимает истинные значения, когда *x+y=z*, и ложные - в противном случае). Исчисление предикатов первого порядка и теории первого порядка отличаются от теорий более высоких порядков тем, что в них допускается применение кванторов только лишь к переменным. Однако установлено, что большинство теорий более высоких порядков сводимо к теориям первого порядка. Каждая теория первого порядка располагает системой аксиом, включающей логические (общие) и собственные (частные) аксиомы. Исчисление предикатов первого порядка - это теория первого порядка,

использующая только логические аксиомы. Дополнение исчисления предикатов аксиомами некоторой предметной области превращает его в частную теорию первого порядка этой предметной области.

Исчисление предикатов первого порядка является определенным расширением исчисления высказываний, поэтому на основе каждого исчисления высказываний может быть построено соответствующее ему исчисление предикатов. Здесь будет рассматриваться исчисление предикатов (ИП), включающее систему связок и аксиом исчисления высказываний Лукасевича. Большинство обсуждаемых здесь результатов (кроме особо оговоренных для ИП) относится к любым теориям первого порядка.

Язык теорий первого порядка богаче языка исчисления высказываний благодаря использованию (нелогических) предметных переменных, что влечет за собой необходимость рассмотрения логических и нелогических функций от нелогических переменных (наряду с логическими переменными и логическими связками исчисления высказываний). По этой причине *множество символов* теорий первого порядка включает подмножества:

$\Sigma_1 = \{a_1, a_2, \mathbf{K}\}$ - символов предметных констант;

$\Sigma_2 = \{f_1^1, f_2^1, \mathbf{K}, f_j^1, \mathbf{K}\}$ - функциональных символов (функторов);

$\Sigma_3 = \{p_1^1, p_2^1, \mathbf{K}, p_j^n, \mathbf{K}\}$ - предикатных символов (предикатов);

$\Sigma_4 = \{x_1, x_2, \mathbf{K}\}$ - символов предметных переменных;

$\Sigma_5 = \{\neg, \rightarrow, \forall\}$ - логических символов;

$\Sigma_6 = \{,(,)\}$ - вспомогательных символов.

Множество $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$ символов называется *сигнатурой*. Символы, входящие в сигнатуру, выделены в особое подмножество по той причине, что наделение этих символов предметным содержанием связывает формальную аксиоматическую теорию с конкретной предметной областью (формальная аксиоматическая теория получает предметную интерпретацию), благодаря чему одна и та же формальная аксиоматическая теория оказывается применимой в различных предметных областях.

Множество формул исчисления предикатов и теорий первого порядка определяется в два этапа: сначала описывается понятие терма (как нелогической функции от нелогических переменных), через которое затем определяется понятие логической формулы.

Терм (сигнатуры Σ) определяется индуктивно: термами полагаются сначала предметные переменные и предметные константы, а затем - все выражения вида $f_j^n(t_1, \mathbf{K}, t_n)$, где f_j^n - любой из функциональных символов, а $t_1, \mathbf{K}, t_n$ - любые из уже имеющихся термов.

Формула (сигнатуры Σ) определяется также индуктивно: простейшими формулами (атомами) полагаются выражения вида $p_j^n(t_1, \mathbf{K}, t_n)$, где p_j^n - предикатный символ и $t_1, \mathbf{K}, t_n$ - термы, а затем формулами считаются все выражения вида $(\neg A), (A \rightarrow B), (\forall x_i A)$, где A и B - атомы или любые из уже построенных так формул, а x_i - любая из предметных переменных.

В выражении $(\forall x A)$ формула A называется *областью действия квантора* $\forall x$. Если формула A не содержит переменной x, считается, что смысл A и $(\forall x A)$ одинаков. Чтобы уменьшить число используемых логических символов, выражение вида $(\exists x A)$, использующее

квантор существования $\exists$, рассматривается как обозначение формулы $(\neg(\forall x(\neg A)))$, выражающей тот же смысл, что и формула $(\exists x A)$.

Пара $S=(D,I)$ называется *алгебраической системой сигнатуры* Σ при условии, что D - непустое множество, а I - отображение, ставящее в соответствие каждой предметной константе $a_i \in \Sigma_1$ - элемент $d_i \in D$, каждому функциональному символу $f_j^n \in \Sigma_2$ - некоторую *n*-местную операцию g_j^n в D, каждому предикатному символу $p_j^n \in \Sigma_3$ - некоторое *n*-местное отношение R_j^n в D. Множество D называется *областью интерпретации*, а отображение I - *интерпретацией* сигнатуры Σ в D.

Алгебраическая система путем интерпретации опредмечивает абстрактные символы, придавая им конкретный смысл в предметной области, характеризуемой множеством D. Задание алгебраической системы позволяет вычислять значения термов и определять выполнимость формул.

Например, формула $A=\exists x_1 \forall x_2 P_1^2\left(f_1^2(a_1,x_1),f_2^2(a_1,x_2)\right)$ в любой алгебраической системе либо тождественно истинна, либо тождественно ложна, так как каждая из входящих в нее переменных связана квантором. Пусть a_1 интерпретируется как число 1, f_1^2 - как умножение (*), f_2^2 - как сложение (+), p_1^2 - как неравенство (<). Тогда в привычной записи формула A принимает вид $A=\exists x_1 \forall x_2((1*x_1)<(1+x_2))$. Эта формула истинна, если областью интерпретации D (множеством констант и возможных значений переменных) является множество натуральных чисел (т.е. D={1,2,3,…}), и ложна, если областью интерпретации D является множество всех целых (положительных и отрицательных) чисел (т.е. D={…,-3,-2,0,1,2,3,…}).

Формула исчисления предикатов может быть *выполнимой и опровержимой* в одной и той же или в разных алгебраических системах, *истинной* в одних и *ложной* в других алгебраических системах, *логически общезначимой* (истинной в любой алгебраической системе) или *противоречием* (ложной в любой алгебраической системе). Таким образом, истинность формулы в одной алгебраической системе еще не гарантирует ее истинности в других, а логически общезначимые формулы исчисления предикатов, тождественно истинные в любых алгебраических системах, являются аналогом и обобщением тавтологий исчисления высказываний..

Существуют и способы записи формул исчисления предикатов, подобные нормальным формам исчисления высказываний. Известна процедура преобразования формул исчисления предикатов к виду $Q_1 x_1,\ldots,Q_n x_n A$, в котором все кванторы предшествуют формуле A, не содержащей кванторов. Причем здесь Q_i- означает квантор ($\forall$ или $\exists$), x_i и x_j различны при $i \neq j$, A - бескванторная формула. Это так называемая *префиксная* (предваренная) *нормальная форма* записи формулы. Случай, когда все Q_i суть $\exists$, подобен дизъюнкции по наборам значений предметных переменных, а случай, когда все Q_i суть $\forall$, подобен конъюнкции по наборам значений предметных переменных. Когда же все $Q_1,\ldots,Q_i$ суть $\exists$, а все $Q_{i+1},\ldots,Q_n$- суть $\forall$, получаем *нормальную форму Сколема*, подобную дизъюнктивной нормальной форме исчисления высказываний. На этом мы завершаем рассмотрение *языка* теорий (и исчисления предикатов) первого порядка.

Аксиомы теорий первого порядка разбиваются на логические и собственные. *Логические аксиомы* (аксиомы исчисления предикатов) - это бесконечное множество формул, построенных с помощью следующих пяти схем аксиом:

$(A1): (Z_1 \to (Z_2 \to Z_1))$;

$(A2): ((Z_1 \to (Z_2 \to Z_3)) \to ((Z_1 \to Z_2) \to (Z_1 \to Z_3)))$;

$(A3): ((\neg Z_1 \to \neg Z_2) \to ((\neg Z_1 \to Z_2) \to Z_1))$;

$(A4): (\forall x_i Z(x_i) \to Z(t))$, где $Z(x_i)$ суть формула и t - терм, свободный для x_i в $Z(x_i)$;

$(A5): (\forall x_i (Z_1 \to Z_2) \to (Z_1 \to \forall x_i Z_2))$, если формула Z_1 не содержит свободных вхождений x_i.

Легко заметить, что первые три схемы аксиом совпадают со схемами аксиом исчисления высказываний Лукасевича, а последние две - специфичны для исчисления предикатов. Их назначение - учесть особенность формул с кванторами и обеспечить логическую общезначимость теорем ИП.

Собственные аксиомы теорий первого порядка не могут быть сформулированы в общем виде, так как каждая теория первого порядка имеет свой набор собственных аксиом. Таким образом, исчисление предикатов первого порядка - это теория первого порядка, не имеющая собственных аксиом.

Правилами вывода в теориях первого порядка являются следующие два основных правила: *modus ponens* (правило заключения), по которому из формул A и $(A \to B)$ выводится формула B; *правило обобщения* (связывания квантором общности), по которому из формулы A выводится формула $(\forall x A)$.

Наряду с названными основными правилами вывода, как обычно, используются и другие правила, в качестве которых выступают многие выведенные в исчислении предикатов теоремы.

Моделью теории первого порядка является алгебраическая система, в которой истинны все аксиомы этой теории.

Таким образом, при построении логической модели ПО ее сущности выступают в качестве области интерпретации D, а функциональные символы, предикатные символы и собственная система аксиом специфичны для каждой ПО и определяются особенностями операций, отношений и законов, характерных для данной ПО.

В качестве примера простейшей теории первого порядка опишем теорию частичного упорядочивания. Пусть рассматриваемая теория содержит один предикатный символ P_1^2 и не содержит функциональных символов. Вместо $P_1^2(x_1, x_2)$ и $\neg P_1^2(x_1, x_2)$ будем писать соответственно $(x_1 < x_2)$ и $(x_1 \geq x_2)$. Собственными аксиомами теории частичного упорядочивания являются аксиомы иррефлексивности и транзитивности:

а) $(\forall x_1 (x_1 \geq x_1))$;

б) $(\forall x_1 \forall x_2 \forall x_3 ((x_1 < x_2) \& (x_2 < x_3) \to (x_1 < x_3)))$.

Всякая модель этой теории называется частично упорядоченной структурой. Так, любой ациклический граф можно рассматривать в качестве модели этой теории, если в качестве области интерпретации считать множество его вершин, а $(x_1 < x_2)$ считать выполненным, если из x_2 в x_1 существует ориентированная цепь.

Продукционные модели [4,9,14,18]

Этот способ представления знаний основан на понятии формальной продукционной системы, которое в 1943 г. ввел в обращение американский логик Е. Пост. Он показал, что любая формальная система (в том смысле, как она определяется в разделе о логических моделях) может быть представлена как система продукций, т.е. как совокупность правил $\{R_i : \Phi_i \rightarrow \Psi_i\}$, где выражение $R_i : \Phi_i \rightarrow \Psi_i$ означает, что в данной формальной системе по правилу R_i из посылки Φ_i непосредственно выводится заключение Ψ_i .

В 1972-1973 гг. Ньюэл и Саймон (авторы идеи экспертных систем) показали, что человек в ходе рассуждений и деятельности использует правила, подобные продукциям, т.е. делает заключения по схеме "условие→следствие" и действует по схеме "ситуация→действие". Тогда же Ньюэл предложил использовать продукционные системы для моделирования процессов принятия решений в системах искусственного интеллекта. Сегодня системы продукций (наряду с сетевыми моделями) являются наиболее популярными средствами представления знаний в системах искусственного интеллекта.

В общем виде под продукцией понимают утверждение типа "если *A*, то *B*", обозначаемое логическим выражением вида $A \rightarrow B$. Продукция $A \rightarrow B$ может истолковываться в логическом смысле (как следование истинности *B* из истинности *A*). Возможны и другие интерпретации продукции $A \rightarrow B$ (например, *A* может быть описанием некоторого условия, выполнение которого необходимо, чтобы можно было совершить действие *B*).

При использовании продукционных моделей для представления знаний создается *продукционная система* - набор продукций, организованный по определенному принципу. Для систем продукций создаются также определенные процедуры управления их функционированием, на которые обычно возлагается решение следующих задач:

а) выявление совокупности активных правил - продукций, условия для применения которых выполнены (правил, которые *могут действовать*);

б) разрешение конфликтов между правилами (с учетом приоритетов, эффективности, эвристик) и выбор правила для применения (правила, которое *должно действовать*);

в) применение выбранного правила, т.е. выполнение действий, предписываемых правой частью продукции (исполнение *действия*).

В различных оболочках (пакетах программ, предназначенных для проектирования экспертных систем) используются различные способы управления системами продукций, определяющие также способы оформления самих продукций и их систем. В самом общем виде продукционное правило представляет собой набор $(I,Q,P,A \rightarrow B,N)$ следующих компонент:

I - имя (или порядковый номер) правила, имя может выражать суть правила, смысл выполняемого им действия;

Q - область применения правила - часто система продукций может быть разделена на подсистемы, каждая из которых связана с решением задачи определенного типа в комплексе задач;

P - предусловие применения правила - общие условия, при которых разрешено или имеет смысл обращаться к данному правилу;

$A \rightarrow B$ - ядро правила ("если *A*, то *B*"), где *A* и *B* соответственно либо посылка и заключение, либо ситуация и действие;

N - постусловие преименения правила - другие последствия и изменения, обусловленные применением данного правила, не учтенные в правой части *B* ядра продукции.

Наиболее ориентированной на применение продукционных моделей представления знаний является *концепция* систем искусственного интеллекта, основанных на использовании *глобальной базы данных*. В состав такой системы входит база правил (множество продукций), глобальная база данных и система управления. База правил представляет собой совокупность знаний в форме правил вида $A \rightarrow B$ ("если *A*, то *B*"). Глобальная база данных содержит знания, представляющие собой данные или уже установленые факты. Система управления формирует заключения, используя базу правил и глобальную базу данных.

Как уже отмечалось при рассмотрении архитектуры экспертных систем, существуют два способа вывода заключений: путем построения прямой или обратной цепочки рассуждений. В концепции глобальной базы данных может быть реализован каждый из этих способов. При этом в прямых цепочках рассуждений вывод очередного заключения осуществляется путем сопоставления данных, содержащихся в базе данных, с левыми частями правил, и если находится правило, где сопоставление происходит, правая часть этого правила в качестве заключения, полученного по этому правилу, помещается в базу данных (или исполняется предписываемое

правой частью этого правила действие, соответствующим образом изменяющее содержимое базы данных). В обратных цепочках рассуждений вывод начинается от поставленной цели. Если находится правило, правая часть (заключение) которого сопоставляется с этой целью, то левая часть (посылка) правила принимается за подцель. Этот процесс обратного вывода повторяется до тех пор, пока не будет получено совпадение подцелей с данными.

Таким образом, структура такой продукционной системы существенно отличается от структуры традиционной вычислительной системы: глобальная база данных доступна для всех правил продукций и ни одна ее часть не ориентирована преимущественно на какое-либо из них; одни правила не вызывают другие, связь между правилами осуществляется только через глобальную базу данных; конструкция системы продукций является гораздо более модульной в сравнении с традиционной вычислительной системой и потому изменения в базе данных, системе управления или правилах могут производиться относительно независимо. Концепция глобальной базы данных проста и тем удобна для теоретического исследования, но при большом числе продукций в такой базе знаний быстро усложняется анализ и разрешение противоречий между правилами. Поэтому число продукций, с которыми работают современные системы искусственного интеллекта такого типа, обычно не превышет тысячи.

Задачи представления. Для решения задачи с помощью системы продукций необходимо определить глобальную базу данных, набор правил и задать стратегию управления. Преобразование исходной формулировки задачи в такие три компоненты некоторой системы продукций называют решением ***проблемы представления***. Обычно существует несколько способов такого представления. Искусство выбора хорошего представления очень существенно для применения методов искусственного интеллекта к практическим задачам. Для наглядности и простоты обсудим решение задачи представления на примере головоломки, известной как игра в 8. На рис. 2.1 приведены две конфигурации фишек. Рассмотрим задачу преобразования начальной конфигурации в указанную целевую. Решением служит правильная последовательность ходов (например, передвинуть фишку 6 вниз, передвинуть фишку 8 вниз, передвинуть фишку 2 вправо, передвинуть фишку 1 вверх, передвинуть фишку 8 влево).

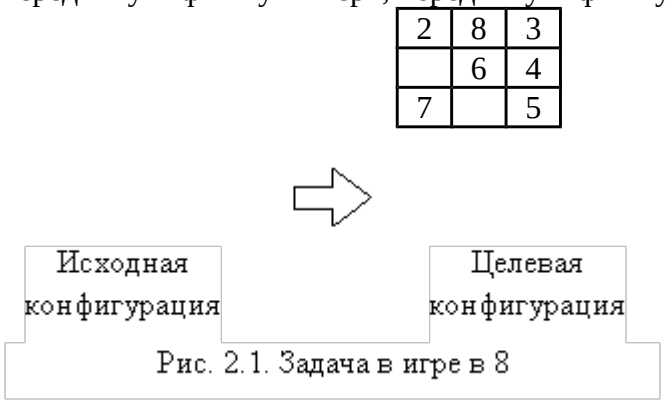

Рис. 2.1. Задача в игре в 8

Определим основные компоненты задачи: состояния, ходы и цель задачи.

В игре в 8 каждая конфигурация фишек является состоянием задачи. Множество всех возможных конфигураций образует пространство состояний задачи. Многие задачи имеют пространство состояний очень большого размера. У игры в 8 оно относительно невелико: 9! (362880) различных конфигураций.

Очевидно, что желательны представления с малыми пространствами состояний. Иногда имеющееся пространство состояний можно сократить, отбросив ряд правил и объединив некоторые из них, образовав более мощные. Можно прийти к меньшему пространству состояний и в результате переформулировки задачи (например, изменением самого понятия о том, что такое состояние).

Определив на смысловом уровне состояния задачи, необходимо сконструировать их машинное представление (описание), которое затем будет использоваться как глобальная база данных системы продукций. Для игры в 8 простым описанием является сам массив, или матрица чисел размером 3х3.

В принципе для описания состояния можно использовать любую из подходящих структур данных (строки символов, векторы, множества, деревья, списки). Исходная глобальная база данных представляет собой описание начального состояния задачи.

Каждый ход преобразует одно состояние задачи в другое. Удобно интерпретировать игру в 8 как игру, в которой имеется 4 хода: передвижение пустой клетки влево, вверх, вправо и вниз. Эти ходы моделируются правилами (продукциями), которые соответствующим образом применяются к описаниям состояний. Каждое правило имеет предусловие, которому должно удовлетворять описание состояния, чтобы это правило можно было применить к данному описанию состояния. Так, предварительным условием для передвижения пустой клетки вверх является требование, чтобы эта клетка не находилась в верхнем ряду.

В игре в 8 задача состоит в достижении целевого состояния (рис. 2.1). Решением задачи является последовательность ходов, которая переводит исходное состояние в целевое. Описание целевого состояния задачи служит основой для формулировки терминального условия (условия остановки) для системы продукций. В некоторых задачах на решение могут накладываться дополнительные ограничения (например, чтобы решение задачи об игре 8 достигалось за минимальное число ходов). В общем случае каждому ходу может быть приписана некоторая цена, а затем осуществляться поиск решения, имеющего минимальную цену.

Гибридные системы и системы типа "доска объявлений". Обычно в продукционных системах все правила однородны и взаимосвязаны. Именно в таких случаях, т.е. в тех случаях, когда система продукций неразложима, наиболее уместно использование концепции глобальной базы данных. Противоположная ситуация возникает, когда решаемая проблема распадается на несколько разнородных подпроблем. Даже если подпроблемы полностью разделить не удается, смешивать в одной продукционной системе правила, относящиеся к разным подпроблемам, нежелательно, особенно в тех случаях, когда подпроблемы относятся к различным предметным областям. Продукционные системы, состоящие из разнородных подсистем продукций, называют *гибридными*.

Одним из способов организации взаимодействия подсистем продукций гибридной системы является механизм, получивший название "доска объявлений", впервые применный в системе распознавания человеческой речи HEARSAY-II, разработанной в американском университете Карнеги-Меллона (рис. 2.2).

Рис. 2.2. Структура системы HEARSAY-II

В этой системе для каждой из подпроблем, образующих в совокупности единую проблему, создается отдельная система знаний. Общение между разнородными системами знаний

осуществляется через рабочую память ("доску объявлений") так, что все знания используются согласованно относительно автономно работающими продукционными подсистемами, именуемыми в HEARSAY-II "источниками знаний". Состав подсистем продукций в HEARSAY-II соответствует этапам процесса распознавания речи:

а) анализ сигнала с голосовых детекторов;

б) анализ фонем (электрических сигналов, соответствущих звукам);

в) анализ (выделение и распознавание) слогов;

г) анализ (выделение и распознавание) слов;

д) анализ (выделение и распознавание) последовательностей слов;

е) анализ (выделение и распознавание) предложений.

Таким образом, HEARSAY-II на каждом уровне (этапе) распознавания речи имеет и использует специфическую базу знаний в виде подсистемы однородных продукций, будучи в целом неоднородной.

Каждая из таких подсистем работает одновременно с другими и независимо от них, используя лишь "доску объявлений" для получения знаний, выработанных другими подсистемами, и отображая на "доску объявлений" знания, полученные в результате собственной работы. Общую координацию функционирования "источников знаний" осуществляет специальный механизм, состоящий из планировщика и монитора "доски объявлений".

Важно отметить, что задачи координации функционирования "источников знаний" в данном случае оказались сложнее подзадач, решаемых подсистемами продукций. В итоге произошел отход (на уровне координации) от продукционного принципа построения базы знаний в целом. Это свидетельствует об ограниченности области эффективного применения моделей знания в виде систем продукций.

Если попытаться охарактеризовать эту область, то о ней можно сказать, что продукционные системы эффективны в тех случаях, когда процесс решения проблемы может быть представлен как поведение, опирающееся на ограниченный контекст. При необходимости учета сложных смыловых связей для решения проблемы продукционные системы быстро теряют свою эффективность из-за быстрого роста их объема и сложности управления ими.

2.3. Сетевые модели [4,14,18]

В самом общем случае сетевая модель - это информационная модель предметной области. В сетевой модели представляются множество информационных единиц (объекты и их свойства, классы объектов и их свойств) и отношения между этими единицами. В зависимости от типов отношений между информационными единицами различают сети:

а) *классификационные* (отношения типа часть-целое, род, вид, индивид);

б) *функциональные* (преобразование информационных единиц);

в) *каузальные* (причинно-следственные отношения);

г) *смешаннные* (использующие разноообразные типы отношений).

В классификационных сетях используются отношения, позволяющие описывать структуру предметной области, что позволяет отражать в базах знаний разные иерархические отношения между информационными единицами. Функциональные сети часто называют вычислительными моделями, так как они позволяют описывать процедуры вычислений одних информационных единиц через другие. В каузальных сетях, называемых также сценариями, используются причинно-следственные отношения, а также отношения типа "средство–результат", "орудие–действие" и т.п. Если в сетевой модели допускаются отношения различного типа, то ее обычно называют семантической сетью. Обычно сетевая модель представляется в виде графа, вершины которого соответствуют информационным единицам, а дуги – отношениям между ними.

Наибольшую известность в системах искусственного интеллекта получили сети смешанного типа (семантические сети и их разновидность - сети фреймов), использующие, в зависимости от области применения, самые разные типы отношений. Семантические сети находят применение в системах понимания естественного языка, в вопросно-ответных системах, в других различных предметно–ориентированных системах.

Важной чертой семантических сетей является возможность представлять знания более естественным и структурированным образом, чем это делается с помощью других формализмов.

Семантические сети. Изначально семантические сети были задуманы в психологии как модель для представления структуры долговременной памяти человека через ассоциации между понятиями. Само понятие "ассоциативная память" появилось еще во времена Аристотеля, а в информатику вошло в связи с работами по использованию простых ассоциаций для представления значения слов в базе данных. Семантическая сеть по сути - это представление простой структуры данных, поэтому весьма важными являются прикладные вопросы, методы использования этих сетей в интеллектуальных системах.

Исследования по семантическим сетям начались с работ *М.Р. Куиллиана*, который в качестве структурной модели долговременной человеческой памяти предложил модель, получившую название *TLC-модели* (Teachable Language Comprehender - доступный механизм понимания языка). В этой модели для описания долговременной памяти была использована сетевая структура как способ представления семантических (смысловых) отношений между словами (концептами).

Стандартного определения семантической сети не существует. Обычно под семантической сетью подразумевают систему знаний в виде целостного образа сети, узлы (вершины) которой соответствуют понятиям, а дуги - отношениям между ними.

Базовым элементом семантической сети служит пара понятий и связывающее их отношение, что обычно представляется в виде пары соответствующих понятиям вершин графа, соединенных соответствующей отношению дугой. В качестве простого примера представим предложения "Куин Мэри является океанским лайнером" и "Каждый океанский лайнер является кораблем" в виде семантической сети, приведенной на рис. 2.3.

Рис. 2.3. Пример простой семантической сети

В этом примере используются понятия "корабль", "океанский лайнер", "Куин Мери" и важный тип отношения "является" между этими понятиями. Каждая из таких пар понятий, связанных отношением, представляет в семантической сети некоторый простой факт, а сеть в целом или ее целостный фрагмент представляет совокупность взаимосвязанных между собой фактов. В качестве логического эквивалента базового элемента семантической сети можно рассматривать бинарный предикат (предикат с двумя аргументами), аргументы которого представляются в сети вершинами, а сам предикат - направленной дугой, связывающей эти вершины.

На рис. 2.4 показана простая семантическая сеть, представляющая концептуальный объект "чайник". В этой сети определены операторы отношений типа *класс*, *свойства*, *пример* и описаны их значения.

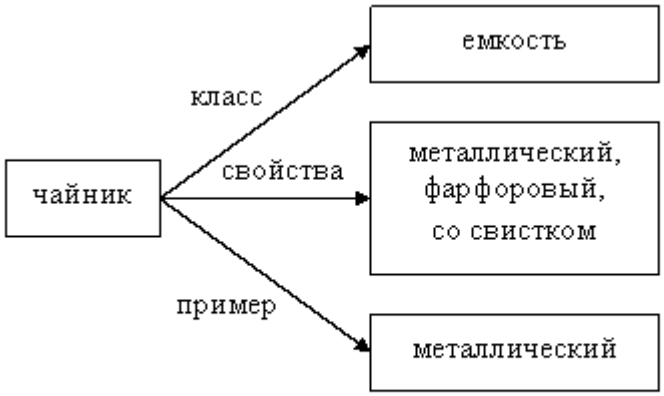

Рис. 2.4. Семантическая сеть концептуального объекта

Таким образом, в модели Куиллиана концептуальные объекты представлены ассоциативными сетями, состоящими из вершин, представляющих концепты, и дуг, показывающих отношения между концептами.

Дедуктивные возможности модели Куиллиана определяются отношением *подкласс* и отношением *модификация*. Частное понятие можно определить через более общее понятие

(первое как подкласс второго) и с помощью модифицирующего свойства, которое является комбинацией *атрибут–значение атрибута*. При этом очевидно, что свойство, присущее элементам класса, также присуще и элементам любого его подкласса.

Таким образом, семантическая сеть Куиллиана представляет собой комбинацию двух принципов: таксономической иерархии, основанной на отношении *класс–подкласс*, и описания свойств элементов класса посредством пар *атрибут–значение атрибута*. Таксономия – теория классификации и систематизации сложноорганизованных областей действительности, имеющих иерархическое строение.

Структурирование знаний в семантической сети. Хотя семантическая сеть Куиллиана сама по себе является моделью памяти, в ней не раскрывается, каким образом осуществляется представление знаний. В процессе решения проблемы представления разбиение на *блоки*, позволяет группировать вершины и дуги семантической сети в отдельные структуры. Эти структуры отождествляются с важными объектами в предметной области системы. Если системе нужна информация об одном из этих объектов, то открывается доступ к соответствующему блоку и отыскиваются сразу все могущие оказаться полезными сведения об этом объекте. Для таких схем представления используется термин *структурированные объекты*, поскольку основной упор в них делается на структуру представления. Наиболее важным в осуществлении блочной организации семантических сетей является использование отношений *"множество–подмножество"* и *"целое–часть"*. Рассмотрим эти отношения.

Важным инструментом структурирования семантических сетей является *иерархия*, или *классификация*. Для создания иерархической структуры объекты, относящиеся к проблемной области, классифицируются на некоторое число категорий или классов на основании их общих свойств. Например, множество людей можно классифицировать на мужчин, женщин, взрослых, детей. Детей можно тоже классифицировать на мальчиков, девочек и по возрастным категориям.

Такого рода классификации представляются в семантических сетях с помощью отношения isa (от английского is a – есть некоторый). Одной из важных черт isa–иерархии является то, что свойства вышележащих типов автоматически переносятся на нижележащие. Например, если свойство "разумный" присуще человеку, то оно присуще также мужчине, женщине, ребенку (как подклассам класса "человек"). Это позволяет избежать значительной части дублирования информации в сети. Так, если некоторый факт имеет место для каждого подмножества некоторого множества, то его можно хранить в структуре знаний только для этого множества.

Не менее важным является также отношение "целое–часть". Оно носит название part of (часть чего-либо). Это отношение позволяет разбивать информацию по уровням детализации. Part of – структура может представлять собой дерево, в котором каждая родительская вершина является part of – структурой для ее потомков.

Утверждение "все собаки – животные" можно представить сетью вида рис. 2.5, а, используя вершины "собака", "животное" и дугу, показывающую отношение между ними. В сети рис. 2.5, б из представленных в ней фактов ("Шарик – собака" и "собака – животное") можно вывести новый факт ("Шарик – животное"), используя наследование по иерархии, присущее отношению isa. В сети можно представить также знания, касающиеся атрибутов объекта. Например, факт "все собаки имеют хвост" показан в сети на рис 2.5, в. Если сеть на рис.2.5, в дополнить фактом "Шарик имеет конуру", то сеть приобретет вид, представленный на рис 2.5, г, где конура i – это конкретная конура, которой владеет Шарик, она является экземпляром понятия "конура".

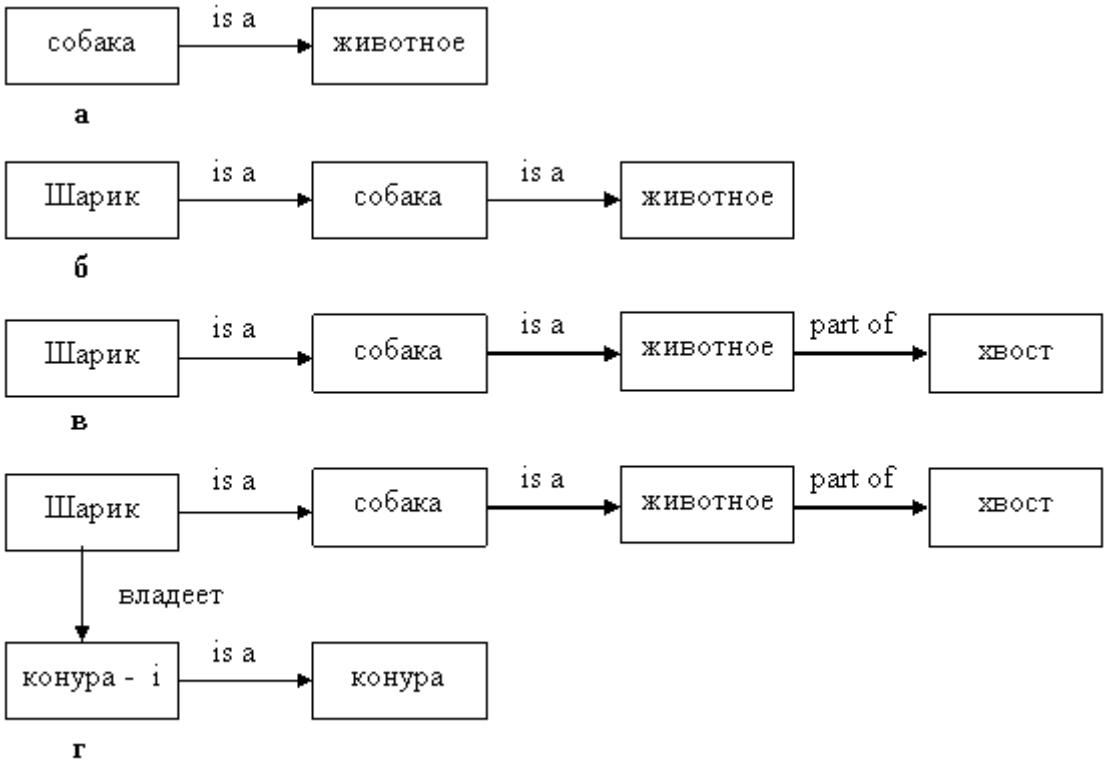

Рис. 2.5

При необходимости сеть можно дополнить информацией "Шарик владеет конурой с весны по осень", тогда вершинами надо представить не только объекты, но также ситуации, действия и события. На рис.2.6 показана такая семантическая сеть.

В этой сети для вершины-ситуации "владение" определено несколько связей. Такая вершина называется падежной рамкой (case frame). Семантические падежи обозначаются метками agt (агент, действующее лицо) и obj (объект, подвергающийся действию). Преимущества использования такой структуры в вершинах сети заключаются в возможности наследования ожидаемых значений и значений, используемых по умолчанию.

Таким образом, основными средствами и принципами структурирования знаний являются: локализация представления информации, обобщение и специализация понятий, семантические падежи, наследование свойств, иерархия (таксономия) понятий.

Сети фреймов. Фреймы - это фрагменты знания, предназначенные для представления стандартных ситуаций. Характерными для этого подхода являются: представление знаний в виде достаточно крупных, содержательно завершенных единиц, называемых фреймами; иерархическая структура фреймов, где иерархия основана на степени абстрактности фреймов; совмещение в фреймах декларативных и процедурных знаний.

Автором теории фреймов является М. Минский. В основе этой теории лежат психологические представления о том, как мы видим, слышим и концентрируем воспринимаемое. Сам Минский считал теорию фреймов скорее теорией постановки задач, чем продуктивной теорией, и суть ее излагал следующим образом. Каждый раз, попадая в некую ситуацию, человек вызывает из своей памяти соответствующую ситуации структуру, именуемую фреймом (frame - рамка). Фрейм - это единица представления знания, заполненная в прошлом, детали которой по необходимости изменяются и уточняются применительно к ситуации. Каждый фрейм может быть дополнен различной информацией, касающейся способов применения данного фрейма, последствий этого применения и т.п. Например, образ жизни каждого человека - это, большей частью, череда типовых ситуаций, различающихся каждый раз в деталях, но в общем и целом повторяющихся.

Фрейм имеет иерархическую структуру: на верхнем уровне располагаются фиксированные характеристики ситуации, на последующих уровнях (в так называемых "слотах" - отсеках, "щелях") - уточняющая и конкретизирующая информация. Различают также пользовательское и машинное представление фреймов.

С точки зрения пользователя, различают три уровня общности фреймов:

а) скелетный, пустой фрейм (*шаблон*), превращаемый после его заполнения в общее или конкретное понятие;

б) фрейм общего понятия (*прототип*) - шаблон, заполненный не конкретными значениями, константами, а переменными;

в) фрейм конкретного понятия (*экземпляр*) - прототип, заполненный конкретными значениями, константами.

Особенности фреймовых моделей видны на примере скелетного фрейма для понятия "РУКОВОДИТЕЛЬ" (рис. 2.7).

Во-первых, каждому фрейму присваивается имя, которое должно быть единственным во всей фреймовой системе. Во-вторых, его описание состоит из ряда описаний, именуемых слотами, которым также присвоены имена (они должны быть различны в пределах фрейма). Каждый слот предназначен для заполнения определенной структурой данных (в скелетном фрейме все они пусты, кроме первого, который имеет значение СЛУЖАЩИЙ, являющееся в данном случае именем фрейма, описывающего понятие "служащий").

На рис. 2.8 представлен тот же фрейм, что и на рис. 2.7, но только с заполненными слотами. При этом часть из них заполнена не простыми именами значений, а некими объектами.

В данном примере фигурируют три типа таких заполнителей слотов: имя другого фрейма (например ЗАРПЛАТА); агрегат (например (фамилия, имя, отчество)); интервал (например (производство, администрация)). Имя другого фрейма служит в качестве ссылки на фрейм, в котором дается описание соответствующего понятия. Обозначение "агрегат" указывает на то, какими конкретными объектами должен быть заполнен слот. Обозначение "интервал" указывает, что конкретное значение слота должно быть выбрано из представленного списка значений. Если значение слота не задается самим пользователем, то оно заполняется значением "по умолчанию". Обозначения "агрегат", "интервал", "по умолчанию" называют фасетами (fasets) слота.

имя: РУКОВОДИТЕЛЬ
категория: СЛУЖАЩИЙ
имя: агрегат (фамилия, имя, отчество)
возраст: агрегат (годы)
адрес: АДРЕС
отдел: интервал (производство, администрация)
заработная плата: ЗАРПЛАТА
дата начала: агрегат (месяц, год)
до: агрегат (месяц, год) (по умолчанию: теперь)

Рис. 2.8. Фрейм для общего понятия РУКОВОДИТЕЛЬ

Значением слота может быть практически все, что угодно (числа или математические соотношения, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов). При конкретизации фрейма ему и его слотам присваиваются конкретные имена и происходит заполнение слотов их значениями. Переход от исходного фрейма-прототипа к фрейму-экземпляру может быть многошаговым (за счет постепенного уточнения значений слотов).

Внутреннее (машинное) представление фрейма имеет более сложную организацию и содержит средства для создания иерархии фреймов, их взаимодействия, обмена информацией, порождения конкретных фреймов из общих и общих из скелетных. Структура данных фрейма (внутреннее представление) имеет вид, представленный на рис. 2.9.

ИМЯ ФРЕЙМА

Имя слота	Указатель наследования	Указатель атрибутов	Значение слота	Демон
Имя слота 1				
Имя слота 2				
……	……	……	…….	…….
Имя слота n				

Рис. 2.9. Структура данных (внутренняя структура) фрейма

Имя фрейма - это идентификатор, присваиваемый фрейму, уникальный во всей фреймовой системе. Фрейм состоит из произвольного числа слотов, часть из которых определяется системой.

Имя слота - это идентификатор слота, уникальный в пределах фрейма. Обычно имя слота смысловой нагрузки не имеет, кроме некоторых случаев: IS-A (родительский фрейм); DDESENDANT (указатель прямого дочернего фрейма); FINEDBY (пользователь, определивший фрейм); DEFINEDON (дата определения); MODIFINEDON (дата модификации); COMMENT (комментарий) и т.п. А также имена, используемые для представления структурированных объектов: HASPART (имеет часть); RELATION (отношение) и др. Все это системные слоты, используемые для редактирования базы знаний и управления выводом.

Указатель наследования. Эти указатели создаются только во фреймовых системах иерархического типа, основанных на отношениях "абстрактное-конкретное", и показывают, какую информацию об атрибутах слотов верхнего уровня наследуют слоты с такими же именами во фреймах нижнего уровня. Типичные указатели: U (unique - уникальный, показывает, что каждый фрейм может иметь слоты с различными значениями); S (same - такой же, показывает, что все слоты должны иметь одинаковые значения); R (range - в пределах границ, показывает, что значения слотов фреймов нижнего уровня должны быть в пределах значений слотов фрейма верхнего уровня); O (overridge - игнорировать, при отсутствии указаний ведет себя как указатель типа S, а при наличии значения - как указатель типа U) и т.п.

Указатель атрибутов слота - это указатель типа данных слота. К таким типам относятся FRAME (указатель); INTEGER (целое); REAL (вещественное); BOOL (булево); LISP (присоединенная процедура); TEXT (текст); LIST (список); TABLE (таблица); EXPRESSION (выражение) и др.

Значение слота - значение, соответствующее типу данных слота и удовлетворяющее условиям наследования.

Демон - процедура, автоматически запускаемая при обращении к слоту при выполнении некоторого условия: IF-NEEDED - "если нужно", запускается, если в момент обращения к слоту его значение не было установлено; IF-ADDED - "если добавлено", запускается при подстановке значения в слот; IF-REMOVED - "если удалено", запускается при стирании значения слота. Демон - это разновидность присоединенной процедуры.

Присоединенная процедура - это служебная программа процедурного типа, используемая в качестве значения слота, запускается по сообщению, переданному из другого фрейма (например, если поступает сообщение об изменении значения слота "зарплата" во фрейме ПЕТРОВ, то автоматически должна быть запущена процедура пересчета налога на Петрова во фрейме НАЛОГ).

В моделях представления знаний фреймами объединяются знания декларативные и процедурные (демоны и присоединенные процедуры). В языках представления знаний фреймами нет специального механизма управления выводом, и пользователь должен создавать его сам. Для этого используется три способа управления выводом: с помощью демонов, присоединенных процедур и механизма наследования. Механизм наследования является единственным основным механизмом, которым оснащаются фреймовые системы. Объединяя работу демонов и присоединенных процедур, можно реализовать любую схему управления выводом, но это требует тщательного его проектирования. Часть специалистов полагают, что нет необходимости специально выделять фреймовые модели в представлении знаний, так как в них объединены все основные особенности моделей остальных типов.

Общий вывод заключается в том, что на некотором глубинном уровне все формы представления знания равносильны (знания, представленные в одной форме, могут быть преобразованы в другую), но не раноценны (для различных предметных областей и различных задач более удобными и эффективными в вычислительном отношении оказываются различные формы представления знания).

3. Представление и обработка нечетких знаний

До сих пор мы не принимали во внимание тот факт, что в реальных условиях знания, которыми располагает человек, всегда в какой-то степени неполны, приближенны, ненадежны. Так, в медицине всегда остаются сомнения в диагнозе заболевания, но отсутствует возможность ждать абсолютно точных свидетельств. В геологии тоже оценка месторождений из-за большой стоимости полномасштабного их изучения всегда имеется лишь приближенная. Тем не менее

людям на основе таких знаний все же удается делать достаточно обоснованные выводы и принимать разумные решения. Следовательно, чтобы интеллектуальные системы были действительно полезны, они должны быть способны учитывать неполную определенность знаний и успешно действовать в таких условиях.

Неопределенность (не-фактор) может иметь различную природу. Наиболее распространенный тип недостаточной определенности знаний обусловлен объективными причинами: действием случайных и неучтенных обстоятельств, неточностью измерительных приборов, ограниченными способностями органов чувств человека, отсутствием возможности получения необходимых свидетельств. В таких случаях люди в оценках и рассуждениях прибегают к использованию вероятностей, допусков и шансов (например шансов победить на выборах). Другой тип неопределенности обусловлен субъективными причинами: нечеткостью содержания используемых человеком понятий (например "толпа"), неоднозначностью смысла слов и высказываний (например "ключ" или знаменитое "казнить нельзя помиловать"). Неоднозначность смысла слов и высказываний часто удается устранить, приняв во внимание контекст, в котором они употребляются, но это тоже получается не всегда или не полностью.

Таким образом, неполная определенность и нечеткость имеющихся знаний - скорее типичная картина при анализе и оценке положения вещей, при построении выводов и рекомендаций, чем исключение. В процессе исследований по искусственному интеллекту для решения этой проблемы выработано несколько подходов.

Самым первым, пожалуй, можно считать использование эвристик в решении задач, в которых достаточно отдаленный прогноз развития событий невозможен (как, например, в шахматной игре). Но самое серьезное внимание этой проблеме стали уделять при создании экспертных систем, и первым здесь был применен вероятностный подход (PROSPECTOR), поскольку теория вероятностей и математическая статистика в тот период были уже достаточно развиты и весьма популярны. Однако проблемы, возникшие на этом пути, заставили обратиться к разработке особых подходов к учету неопределенности в знаниях непосредственно для экспертных систем (коэффициенты уверенности в системах MYCIN и EMYCIN). В дальнейшем исследования в этой области привели к разработке особой (нечеткой) логики, основы которой были заложены Лотфи Заде.

В решении рассматриваемой проблемы применительно к экспертным системам, построенным на основе правил (систем продукций), выделяются четыре основных вопроса:

а) как количественно выразить достоверность, надежность посылок?

б) как выразить степень поддержки заключения конкретной посылкой?

в) как учесть совместное влияние нескольких посылок на заключение?

г) как строить цепочки умозаключений в условиях неопределенности?

На языке продукций эти вопросы приобретают следующий смысл. Будем обозначать $ct(A)$ степень уверенности в A (от англ. certainty - уверенность).

Тогда первый вопрос заключается в том, как количественно выразить степень уверенности $ct(A)$ в истинности посылки (свидетельства) A.

Второй вопрос связан с тем, что истинность посылки A в продукции $A \to C$ может не всегда влечь за собой истинность заключения C (так высокая температура вызывает лишь определенное подозрение на заболевание гриппом, но не гарантирует правильности диагноза "грипп"). Степень поддержки заключения C посылкой A в продукции $A \to C$ обозначим через $ct(A \to C)$.

Третий вопрос обусловлен тем, что одно и то же заключение C может в различной степени поддерживаться несколькими посылками (например, заключение C может поддерживаться посылкой A посредством продукции $A \to C$ с уверенностью $ct(A \to C)$ и посылкой B посредством продукции $B \to C$ с уверенностью $ct(B \to C)$). В этом случае возникает необходимость учета степени совместной поддержки заключения несколькими посылками.

Последний вопрос вызван необходимостью оценки степени достоверности вывода, полученного посредством цепочки умозаключений (например, вывода C, полученного из посылки A применением последовательности продукций $A \to B$, $B \to C$, обеспечивающих степени поддержки соответственно $ct(A \to B)$ и $ct(B \to C)$).

Тема 3 Экспертные системы. Архитектура экспертынх систем. Характеристики ЭС. Функции ЭС. Средства построения ЭС. Назначение компонент ЭС.

Технология построения экспертных систем. Этапы разработки ЭС. Модификация ЭС при ее разработке. Иструментальные средства разработки ЭС.

Успех первых экспертных систем стимулировал их разработку и применение в самых различных областях, что обязывает к более внимательному изучению их особенностей и возможностей.

Основные черты экспертных систем. Отметим характеристики, особо выделяющие экспертные системы (ЭС) из всего многообразия систем искусственного интеллекта и обеспечивающие их способность к решению сложных задач и, значит, их практическую полезность.

1. ЭС ограничены определенной сферой экспертизы - узкой проблемной областью (ПО), что позволяет систематизировать и загрузить в компьютер знания, достаточные для качественного решения реальных задач.

2. ЭС, благодаря использованию нечеткой логики, способны делать надежные экспертные заключения при ненадежных и неполных данных.

3. ЭС способны объяснять понятным пользователю способом ход своих рассуждений и причины своих запросов к нему, устанавливая тем самым с ним профессиональные и дружественные отношения.

4. ЭС выдают в качестве ответа на запрос не результаты вычислений, а результаты рассуждений в форме экспертного заключения или совета.

5. ЭС строится так, чтобы сохранялась возможность совершенствования и расширения ее знаний с развитием ПО и знаний о ней. Поэтому созданные экспертные системы до сих пор существуют, развиваются и используются.

6. ЭС экономически выгодны. Это еще одна причина того, что созданные экспертные системы живут и развиваются. И это пока единственный тип систем ИИ, приносящих положительный экономический эффект.

Области применения экспертных систем. Создание ЭС - весьма сложный, трудоемкий и дорогостоящий процесс. Поэтому каждый, кто замышляет её построение, должен задать себе вопрос: "Нужна ли мне экспертная система?" Ответ на него зависит от типа задачи, которую вы хотите решить. В табл. 1.2 представлен контрольный список характеристик областей применения с точки зрения пригодности подхода с использованием знаний.

Таблица 1.2

Подходит	*Не подходит*
Диагностика	Вычислительные задачи
Нет установившейся теории	Есть хорошо развитая теория
Данные неполны, неточны, ненадежны	Есть полные, точные, надежные данные
Мало специалистов	Нет недостатка в специалистах
Опасная, агрессивная среда	Нормальные условия работы

Если имеющееся вами в виду приложение больше относится к левой части таблицы, чем к правой, то следует всерьез рассмотреть перспективу использования экспертной системы.

К диагностическому типу задач относится не только задача постановки медицинского диагноза, но любая ситуация, в которой имеется множество возможных ответов, а требуется выбрать из них один верный или, по крайней мере, отбросить заведомо неверные. Сюда попадают многие задачи классификации и предсказания (например диагностика неисправностей в вычислительной машине или выявление причин экономического спада).

Проблемные области, не имеющие твердо установленной теории, отличаются большим числом переменных величин, затрудняющих создание полной и цельной теории, в силу чего искусные практики больше полагаются на опыт и интуицию, чем на теоретические выводы (вроде экономических или политических прогнозов). Следовательно, сюда не относятся задачи, для решения которых можно вывести и применить некоторую формулу (как, например, задача о движении небесных тел, где законов ньютоновской механики достаточно для управления полетом космического корабля).

Область с малым числом специалистов обычно легко узнается по высокой зарплате, спросу на специалистов и очередям на курсы переквалификации. Ясно, что в этом случае будет экономически оправданным машинизировать навыки, на которые имеется высокий спрос.

Наконец, если имеющаяся информация ненадежна, нечетко задана или "замусорена", то экспертные системы - это как раз то, что вам нужно. Тогда в построении экспертных заключений начнет играть ключевую роль какая-нибудь нечеткая, многозначная или вероятностная логика.

Архитектура экспертных систем. Подход к конструированию систем, основанных на использовании знаний, представляет собой новшество с последствиями революционного характера не только в содержании, но и в организации вычислительных процессов. В новой архитектуре традиционная вычислительная система превращается в основу системы качественно нового типа, ядро которой составляют база знаний и "машина логического вывода".

Чтобы понять суть принципиальных изменений, рассмотрим простой пример. Пусть требуется вычислить выражения и проверить их равенство: $x*(y+z)=x*y+x*z$. Не составляет труда написать на любом из языков программирования процедуру, проверяющую это равенство и выдающую результат для заданных значений переменных x,y,z. Мы знаем, что ответ при любых значениях x,y,z будет положительным, так как это арифметический закон дистрибутивности умножения относительно сложения, но компьютер этого не знает и всякий раз при запуске процедуры будет требовать задания конкретных значений этих переменных.

Возникает вопрос: как должна быть устроена и функционировать компьютерная система, способная определять истинность выражений, подобных $\forall x \forall y \forall z (x*(y+z)=x*y+x*z)$? Вычислительная система традиционного типа не в состоянии это сделать, если множества значений переменных x,y,z бесконечны, поскольку такое выражение представляет собой не формулу, которую следует применить, чтобы получить ответ, а утверждение, которое требуется доказать. Для построения же доказательств требуются не столько вычисления на основе данных, сколько рассуждения на основе знаний. Таким образом, вместо программ вычислений в системах нового типа на первый план выходят программы рассуждений, доказательств, логического вывода.

Полностью оформленная экспертная система обязательно имеет следующие четыре основные компоненты (рис.1.1), присущие сегодня всем системам, использующим ИИ:

а) база знаний;

б) машина (механизм) вывода;

в) модуль извлечения (приобретения) знаний;

г) система объяснения (интерфейс).

Рис. 1.1. Архитектура экспертной системы

Все четыре модуля являются важными, и, хотя система, основанная на знаниях, может обойтись без одного-двух из них, настоящая экспертная система обязана иметь их все.

База знаний. База знаний содержит факты и правила. Факты - это утверждения, характеризующие текущую ситуацию в проблемной области. Они имеют временный характер и могут изменяться в ходе консультации. Правила представляют собой законы (общего характера или присущие лишь данной проблемной области), позволяющие выводить новые утверждения (следствия, заключения), соответствующие имеющимся фактам и гипотезам.

Наиболее употребительным способом представления неформальных знаний являются правила в виде продукций, имеющих уже знакомый нам формат "ЕСЛИ…ТО…". Но продукции - не единственный способ представления знаний. В зависимости от специфики проблемной области более удобными могут оказаться исчисление предикатов, семантические сети или сети фреймов. Однако на некотором глубинном уровне все типы представления знания равносильны, поэтому лучшей будет рекомендация: выбирать для построения конкретной экспертной системы простейший из тех способов представления знания, что позволяют работать.

Употребление в обоих случаях слова "база" может вызвать вопрос: чем отличается база знаний от базы данных?. Для ответа на него сопоставлять надо данные и факты, программу и систему правил базы знаний. Подобно тому, как в вычислительных системах данные пассивны, а программы - операциональны, факты в базе знаний играют пассивную, констатирующую роль, а система правил - активную, процессуальную. Различие же между вычислительной и интеллектуальной системами, как уже отмечалось, состоит в том, что первая осуществляет вычисления по программе на основе данных, а вторая - рассуждения по правилам на основе фактов.

Машина вывода. Подобно тому, как в вычислительной системе операционнная система организует процесс вычислений, так в интеллектуальной системе машина вывода организует процесс рассуждений. Существует две основных стратегии построения логического вывода: "прямая цепочка рассуждений" и "обратная цепочка рассуждений".

Прямая цепочка связана с рассуждениями, ведущимися от данных и фактов к гипотезам (целям), тогда как обратная цепочка рассуждений представляет собой поиск данных и фактов, доказывающих или опровергающих некоторые гипотезы (цели).

Чисто прямая цепочка рассуждений при неполных и ненадежных данных ведет к необозримому множеству, в том числе невероятных следствий. Кроме того, ее применение в чистом виде порождает к пользователю множество вопросов, не имеющих отношения к поставленной им цели.

Чисто обратная цепочка рассуждений при неполных и ненадежных данных, напротив, может вести к навязыванию неадекватных объяснений фактам, "притягиванию" фактов к гипотезам. Кроме того, ее применение, как правило, приводит к настойчивому повторению вопросов к пользователю, касающихся поставленной им цели.

По этим причинам наиболее удачные системы комбинируют эти два способа рассуждений. Так, К. Нейлор [8] описал метод, известный как подход с оценкой правил, который сочетает в себе достоинства обеих стратегий и смягчает их недостатки.

Но работает ли процедура вывода в прямом или в обратном направлении, ей приходится сталкиваться с ненадежными данными, фактами, правилами и заключениями, так как экспертные системы имеют дело не с идеализированными, а с реальными ситуациями. Для работы с неполными данными и ненадежными фактами, правилами и заключениями используются различные типы логик (вероятностная логика, многозначная логика, нечеткая логика, коэффициенты уверенности, если назвать только четыре из них). На практике были испробованы разные виды систем нечетких логических операций, но существенной разницы между ними не обнаружено. Объяснение этому, видимо, в том, что организация знаний играет более важную роль в рассуждениях, чем связанные со знаниями числовые значения. Кроме того, большинству баз знаний присуща избыточность, позволяющая экспертной системе прийти к правильному заключению несколькими различными путями. Числа, измеряющие степень доверия к получаемым в процессе поиска результатам, служат лишь ориентирами. По этой причине можно применять любой из удобных в каждом конкретном случае способ измерения ненадежности.

Модуль усвоения знаний. В простейшем варианте модуль усвоения знаний представляет собой редактор базы знаний, и это - первое, о чем следует позаботиться разработчикам экспертной системы. Современные оболочки экспертных систем обычно им располагают. Однако подобно тому, как в вычислительной системе самым сложным вопросом является создание алгоритма и программы решения вычислительной задачи, так в интеллектуальных системах самым сложным вопросом является извлечение и представление в виде фактов и правил знаний и опыта экспертов. До настоящего времени усвоение знаний и представление их в понятной машине форме остается самым узким местом в развитии экспертных систем. Эксперты известны своей неспособностью объяснить, каким образом они приходят к своим решениям, а объяснения, которые они сами дают, часто оказываются чисто внешними.

Как можно описать и формализовать их опыт? Традиционный способ состоит в длительной совместной работе инженера по знаниям с одним-двумя экспертами, в процессе которой должно быть выявлено и формализовано все то, что знает эксперт. Поэтому остро ощущается потребность автоматизировать процесс получения знаний, что станет возможным лишь тогда, когда мы поймем, как мы их сами получаем.

Программа EURISCO, созданная Д.Б. Ленатом на основе эволюционных алгоритмов Сэмюэля и Холланда, стала предвестником нового поколения обучающихся машин: Р.С. Михальский создал систему, которая обучалась классификации болезней зерновых культур; Дж. Квинлан разработал алгоритм обучения понятиям на основе анализа примеров, содержащихся в базе данных; Р. Форсайт написал программу BEAGLE (Biological Evolutionary Algorithm Generatial Expressions - биологический эволюционный алгоритм, порождающий логические выражения), в которой используется дарвиновская схема естественного отбора. Но особо примечательным в программе EURISCO Лената было то, что используемый ею язык описаний (средство хранения правил и понятий) оказался достаточно выразительным для представления зачаточной формы самосознания в виде "метаправил". Эта система тратит массу времени на самоанализ и управление своим поведением, запоминая обнаруженные правила и применяя их к себе.

Интерфейс. Четвертой важной компонентой экспертной системы являются средства, обеспечивающие возможность ее объяснения с человеком. Одним из самых замечательных свойств, присущих классическим экспертным системам, подобным системе MYCIN, является то

внимание, которое было уделено в ней пользовательскому интерфейсу. В любой момент эту систему можно спросить, как (how) она пришла к такому заключению или почему (why) она задала пользователю такой вопрос? В системе, основанной на использовании правил, ответ обычно формируется путем повторного прослеживания тех шагов рассуждения, которые привели к данному вопросу или к данному заключению. Легкость, с которой это можно делать, является важным доводом в пользу систем, основанных на правилах.

Средства объяснений не следует считать специфической чертой, лишь экспертных систем. Доналд Мичи (1982) и другие авторы указывали на обреченность систем, в которых не предусмотрено "когнитивное окно для человека", т.е. действия которых носят скрытый или непонятный человеку характер. Поэтому метод рассуждения, который не может быть объяснен человеку, неприемлем, даже если он работает лучше, чем специалист.

Общая типология систем ИИ

До появления экспертных систем результаты исследований по проблеме ИИ представляли интерес, главным образом, для тех, кто ими занимался. Создание практически полезных и экономически эффективных экспертных систем вызвало широкий интерес к созданию подобных систем в самых различных областях, имеющих дело с решением сложных и трудно формализуемых проблем. Сегодня экспертные системы и нейронные сети востребованы и широко используются.

Многие авторы полагают, что термины экспертные системы и системы, основанные на знаниях, являются синонимами. Это не совсем верно. Они так считают потому, что знания (база знаний) и инструмент манипулирования знаниями (машина вывода) разделены в экспертной системе. Благодаря этому разделению база знаний создается независимо от машины вывода. Но в действительности любая компьютерная система основана на знаниях. Часть из них используют теоретические знания, другие - эмпирические знания, часть из них получают знания от человека (эксперта), другие приобретают знания сами путем изучения, обобщения и логического анализа фактов.

Классификация компьютерных систем, основанная на перечисленных видах знания и способностях, приведена в табл. 1.3.

Таблица 1.3

Тип знания	Источник знания	
	Человек	Познание
Теоретическое	Системы поддержки решений (DSS)	Системы искусственного интеллекта (AIS)
Эмпирическое	Экспертные системы (ES)	Нейронные сети (NN)

Классификация, приведенная в табл. 1.3, говорит о следующем:

- экспертные системы получают эмпирическое знание от эксперта в виде правил, основанных на его опыте;
- нейронные сети приобретают эмпирическое знание сами, путем тренировки с использованием больших объемов данных;
- системы поддержки решений основаны на теоретическом знании, которое включено исследователями и разработчиками в алгоритмы и программы, помогающие пользователю принимать эффективные решения;
- системы действительно интеллектуальные должны быть способны сами приобретать существующее и создавать новое теоретическое знание (благодаря способности создавать метаправила, этому требованию в некоторой степени удовлетворяет система EURISCO Лената).

Тема 4 Методы поиска решения в пространстве состояний. Классификация методов поиска решений. Простейшие методы поиска в одном пространстве состояний. Поиск методом полного перебора в глубину. Механизм возврата. Поиск методом полного перебора в ширину. Методы эвристического поиска в пространстве с большим числом состояний.

Представление знаний в условиях неопределенности. Класссификация видов неопределенности. Формула Байеса и логический вывод на основе теории вероятности. Логический метод на основе коэфициентов неуверенности. Отношения правдоподобия и логический вывод на их основе . Форматы правил экспкртных систем с вероятностным выводом.
Дедуктивный вывод, основанный на нечетких знаниях. Нечеткие знания. Нечеткое множество. Основные операции над нечеткикми множествами. Нечеткое отношение. Лингвистическая переменная.Логический вывод на основе нечеткой логики. Нечеткие высказывания и операции над ними. Этапы нечетких логических выводов.

Методы поиска решений на основе знаний

Вывод заключений в логических моделях

В основу вывода заключений из имеющихся фактов или гипотез в логических моделях положены приемы, получившие название "доказательство теорем на основе резолюции". Метод резолюции позволяет справиться со многими проблемами, осложняющими выбор правил вывода. Чтобы воспользоваться приемами решения задач на основе резолюции (позже они будут рассмотрены более детально), предложения исчисления предикатов в несколько этапов приводят к упрощенному виду записи, называемому "клаузальной формой" (от англ. clause - утверждение, предложение). Процесс упрощения включает освобождение выражения от кванторов и сведение его к списку предикатов, соединенных связкой ИЛИ.

***Префиксная нормальная форма*.** Приведение формулы ИП к префиксной нормальной форме является первым этапом на пути к освобождению выражения от кванторов. Для приведения формулы ИП к этому виду используется ряд равносильных в исчислении предикатов формул.

Имеем (при условии, что P не содержит свободно *x* и, значит, не подвержено действию кванторов) следующие пары равносильных формул:

$$\forall x F(x) \vee P = \forall x (F(x) \vee P);$$
$$\forall x F(x) \,\&\, P = \forall x (F(x) \,\&\, P);$$
$$\exists x F(x) \vee P = \exists x (F(x) \vee P);$$
$$\exists x F(x) \,\&\, P = \exists x (F(x) \,\&\, P).$$

Необходимы также следующие пары равносильных формул:

$$\forall x F(x) \,\&\, \forall x P(x) = \forall x (F(x) \,\&\, P(x));$$
$$\exists x F(x) \vee \forall x P(x) = \exists x (F(x) \vee P(x)).$$

Однако следует иметь в виду неравносильность следующих формул:

$$\forall x F(x) \vee \forall x P(x) \neq \forall x (F(x) \vee P(x));$$
$$\exists x F(x) \,\&\, \forall x P(x) \neq \exists x (F(x) \,\&\, P(x)).$$

Поэтому в последних двух случаях для обеспечения равносильности необходимо произвести переименование связанных переменных, используя в качестве *y* переменную, отсутствующую в $F(x)$:

$$\forall x F(x) \vee \forall x P(x) = \forall x F(x) \vee \forall y P(y) = \forall x \forall y (F(x) \vee P(y)),$$
$$\exists x F(x) \,\&\, \forall x P(x) = \exists x F(x) \,\&\, \exists y P(y) = \exists x \exists y (F(x) \,\&\, P(y)).$$

Возможность переименования связанных переменных в двух последних отношениях обусловлена тем, что связанная квантором переменная является "немой" (не влияет на истинность формулы) и ее можно заменять на любой другой символ переменной, еще не встречавшийся в выражении.

С учетом приведенных формул алгоритм преобразования произвольной заданной формулы ИП в равносильную ей формулу в префиксной нормальной форме состоит в выполнении следующих шагов.

Шаг 1. Выражение логических связок импликации и эквиваленции через связки отрицания, дизъюнкции и конъюнкции с помощью известных правил алгебры высказываний.

Шаг 2. Продвижение связки отрицания до атомарных предикатов с использованием законов де Моргана. В результате выполнения этого шага получается формула, в которой знаки отрицания могут стоять только перед атомарными предикатами.

Шаг 3. Переименование связных переменных с использованием символов переменных, еще не встречавшихся в выражении.

Шаг 4. Вынесение кванторов в префикс с использованием приведенных выше отношений равносильности формул. После выполнения этого шага формула приобретает префиксный вид.

Рассмотрим простой пример. Пусть дано утверждение *"Каждый программист имеет свой пароль"*. Формула ИП, соответствующая этому предложению, имеет вид

$$\forall x(\textit{программист}(x) \to \exists y(\textit{пароль}(y) \,\&\, \textit{имеет}(x,y)))$$,

где *программист*(*x*), *пароль*(*y*) - одноместные, а *имеет*(*x*,*y*) - двухместный атомарные предикаты.

Шаг 1 выполняется для исключения импликации с помощью отношения равносильности формул $(A \to B) = (\neg A \vee B)$. В итоге получим выражение

$$\forall x(\neg \textit{программист}(x) \vee \exists y(\textit{пароль}(y) \,\&\, \textit{имеет}(x,y)))$$.

Шаг 2 и Шаг 3 для данной формулы выполнять не требуется.

Шаг 4 применительно к полученной формуле связан с использованием отношения равносильности формул $\exists x F(x) \vee P = \exists x(F(x) \vee P)$, на основании которого можно вынести $\exists y$ в префикс и получить предложение требуемого вида: $\forall x \exists y(\neg \textit{программист}(x) \vee \textit{пароль}(y) \,\&\, \textit{имеет}(x,y))$.

Нормальная форма Сколема. После того как кванторы вынесены в префиксную часть формулы, необходимо полностью исключить кванторы из предикатных формул, чтобы обеспечить возможность применения законов алгебры высказываний в машинных процедурах обработки предикатных формул. Если все переменные в некотором выражении подвергнуты квантификации, кванторы общности можно опустить, не нарушая смысла выражения. Однако в конкретном случае квантором общности могут быть квантифицированы не все переменные и тогда проблема решается путем введения *функций Сколема*.

Для иллюстрации процесса сколемизации предикатных выражений продолжим рассмотрение примера, использованного для получения префиксной нормальной формы:

$$\forall x \exists y(\neg \textit{программист}(x) \vee \textit{пароль}(y) \,\&\, \textit{имеет}(x,y))$$.

Как уже говорилось, если все переменные в некотором выражении подвергнуты квантификации, то мы можем опустить кванторы общности. Однако в данном случае квантор общности относится только к *x*, но не к *y* (т.е. не все *y* удовлетворяют высказыванию). Эта проблема решается путем введения функциональной зависимости *y* от *x* вида *y*=*g*(*x*), называемой *функцией Сколема*. В нашем примере *g*(*x*) - это принцип присвоения пароля *y* каждому программисту *x*. Перепишем с учетом *y*=*g*(*x*) наше выражение:

$$\forall x(\neg \textit{программист}(x) \vee \textit{пароль}(g(x)) \,\&\, \textit{имеет}(x,g(x)))$$.

Теперь квантор общности распространяет свое действие на всё выражение и, следовательно, знак квантора общности можно опустить:

$$\neg \textit{программист}(x) \vee \textit{пароль}(g(x)) \,\&\, \textit{имеет}(x,g(x))$$.

Общее правило исключения квантора существования из формул ИП состоит в замене каждого вхождения переменной, относящейся к квантору существования, на сколемовскую функцию, аргументы которой являются переменными, связанными кванторами общности, и область действия этих кванторов общности включает область действия исключаемого квантора существования.

Приоритетность действия кванторов, имеющихся в префиксной форме представления, определяется слева направо. Например, формулу $\forall x \exists y \forall z F(x,y,z)$ можно переписать в следующем виде: $\forall x \forall z F(x,g(x),z)$, так как переменной, влияющей на связанную квантором $\forall$ переменную *y*, является только *x*. Если же исходная логическая формула имеет вид $\forall x \forall z \exists y F(x,y,z)$, то переменными, влияющими на переменную с квантором существования, являются *x* и *z* и тогда получаем $\forall x \forall z F(x,g(x,z),z)$. Если исключаемый квантор существования не входит в область действия никакого из кванторов общности, то применяется сколемовская функция без аргументов, т.е. просто константа. Так $\exists x P(x)$ становится $P(A)$, где

символ константы *A* используется для ссылки на элемент, который, как известно, существует. Важно, чтобы *A* был новым символом константы и не совпадал с другими символами, взятыми для других формул.

Если исключить подобным образом связанные квантором переменные, то любые другие переменные, которые встречаются в формуле, будут связаны только квантором . Порядок кванторов общности роли не играет, поэтому можно не указывать явно вхождения кванторов, а предположить по соглашению, что все переменные связаны кванторами общности.

Клаузальная форма (форма предложений). На данной стадии полученное для рассматриваемого примера выражение

$$\neg программист(x) \vee пароль(g(x)) \,\&\, имеет(x, g(x))$$

может быть представлено в общем виде как

$$A \vee (B \,\&\, C)$$.

Так как $A \vee (B \,\&\, C) = (A \vee B) \,\&\, (A \vee C)$, то выражение $A \vee (B \,\&\, C)$ на самом деле содержит два разных предложения со связкой "или":

$$(A \vee B),$$
$$(A \vee C).$$

Поэтому полученное для рассматриваемого примера выражение может быть записано так:

$$\neg программист(x) \vee пароль(g(x)),$$
$$\neg программист(x) \vee имеет(x, g(x)).$$

О представленном в таком виде пропозициональном выражении говорят, что оно имеет "клаузальную форму". Любая, представленная в клаузальной форме, фраза состоит из серии высказываний (с отрицаниями ¬ или без них), соединенных связкой ("или").

В общем случае любое пропозициональное выражение можно привести к конъюнктивной нормальной форме (КНФ) многократным применением дистрибутивных правил. Чтобы затем преобразовать КНФ в клаузальную форму (в форму предложений), исключают символы &, заменяя конъюнкцию выражений $(A_1 \,\&\, A_2 \,\&\, ... \,\&\, A_n)$ на множество выражений $\{A_1, A_2, ..., A_n\}$. Результатом осуществления таких замен будет некоторое конечное множество выражений, каждое из которых является дизъюнкцией литералов (с отрицаниями ¬ или без них). Любая формула, состоящая только из дизъюнкций литералов, называется *предложением*. Множество предложений, полученных в результате исключения символа &, называется клаузальным множеством.

Процедура резолюции. Как уже говорилось ранее, метод резолюции используется для доказательства теорем и вывода заключений в исчислении предикатов и в теориях первого порядка.

Задача в исчислении предикатов определяется совокупностью формул, считающихся истинными в условиях задачи (фактами, посылками), и формулой (целью, заключением), истинность которой требуется вывести (доказать или опровергнуть), исходя из предположения об истинности условий.

Решение задачи в исчислении предикатов равносильно доказательству (или опровержению) истинности формулы $(A_1 \,\&\, A_2 \,\&\, ... \,\&\, A_n) \to C$, где формулы $A_1, A_2, ..., A_n$ считаются посылками, а формула *C* - требующим доказательства (или опровержения) заключением.

Учитывая, что $(A \to C) = (\neg A \vee C)$, исходную формулу задачи перепишем в виде $(\neg(A_1 \,\&\, A_2 \,\&\, ... \,\&\, A_n) \vee C)$, что можно прочитать так: *либо неверны посылки, либо истинно заключение*. Но так как, по условиям задачи, посылки верны, то более удобной является форма постановки этой задачи от противного, а именно - попробуем опровергнуть истинность последней формулы, т.е. попытаемся доказать истинность ее отрицания:

$$\neg(\neg(A_1 \,\&\, A_2 \,\&\, ... \,\&\, A_n) \vee C) = ((A_1 \,\&\, A_2 \,\&\, ... \,\&\, A_n) \,\&\, \neg C).$$

В клаузальной форме формулы $A_1, A_2, \ldots, A_n$ представляют собой дизъюнкции литералов (с отрицаниями $\neg$ или без них). Если формула *C* - не литерал, то формула $\neg C$ тоже должна быть приведена к этому виду.

Итак, в последней формулировке мы должны, полагая истинными посылки, попытаться вывести истинность формулы $\neg C$. Отрицательный результат этой попытки будет свидетельствовать об истинности формулы *C*.

В основе метода резолюции лежит правило вывода "обобщенное modus ponens": ($A \vee B$), ($\neg A \vee C$) $\vdash$ ($B \vee C$), реализуемое как операция исключения высказываний из предложений, если эти высказывания в одних предложениях присутствуют без отрицаний, а в других - с отрицаниями.

Рассмотрим сначала абстрактный пример. Прежде всего предположим, что в клаузальной форме нам даны факты (посылки):

1) $\neg B \vee C$,

2) $A \vee B$,

3) $\neg A$.

Далее предположим, что мы хотим доказать, что из истинности этих посылок следует истинность формулы *C*. Чтобы это сделать, дополним заданный набор предложений отрицанием *C*:

1) $\neg B \vee C$,

2) $A \vee B$,

3) $\neg A$,

4) $\neg C$.

Исключим теперь вновь введенное предложение и его отрицание в другом предложении. Этот процесс называется *резолюцией*. Например, в данном случае мы можем исключить $\neg C$ в четвертом предложении и *C* - в первом, оставив в них без изменения совокупность остальных высказываний. Такую совокупность остатков предложений называют *резольвентой* (в нашем случае это $\neg B$). В результате осуществления первой резолюции остаются предложения:

1) $\neg B$,

2) $A \vee B$,

3) $\neg A$.

После выполнения второй резолюции (например, исключением $\neg B$ из первого предложения и *B* - из второго) останутся предложения:

2) *A*,

3) $\neg A$.

Наконец, исключив последние два предложения, получим пустое предложение, изображаемое символически ( ). В этом случае говорят, что получена "пустая резольвента". Если удается вывести пустое предложение, то истинность заключения (в данном случае *C*) считается доказанной.

Теперь рассмотрим конкретное содержательное рассуждение.

Предположим установленной истинность двух утверждений:

1) *если экономика развивается, то курс валюты не падает*,

2) *курс валюты падает*.

Рассмотрим далее, как из этих утверждений сделать вывод, что экономика не развивается. Для этого представим сначала данные нам утверждения в виде формул исчисления предикатов:

1) $(развивается(экономика) \rightarrow (\neg падает(курс))$,

2) $падает(курс)$.

Переведя данные выражения в клаузальную форму и дополнив их отрицанием искомого заключения $(\neg развивается(экономика))$, получим:

1) $(\neg развивается(экономика) \vee (\neg падает(курс))$,

2) $падает(курс)$,

3) $развивается(экономика)$.

Из полученных предложений видно, что использование в резолюции третьего и первого из них дает в итоге следующие два предложения:

1) $\neg падает(курс)$,

2) $падает(курс)$.

Резолюция этих предложений приводит к пустой резольвенте, что свидетельствует об истинности заключения $(\neg развивается(экономика))$.

Введение клаузальной формы выражений и применение метода резолюции предоставило ЭВМ простую процедуру вывода заключений. К сожалению, сам по себе метод резолюции не указывает, какие предложения следует выбирать для резолюции. В приведенных примерах, по существу, в этом не было необходимости, поскольку предложений было очень мало. Но даже при умеренном их количестве мы сталкиваемся с проблемой комбинаторного взрыва. Так, если мы будем проверять все возможные комбинации предложений, то для 10 предложений будет 50 комбинаций при первой попытке применения метода резолюции, а к десятой попытке их число возрастет до 10^{17} (для сравнения: в столетии около 10^{10} секунд). Таким образом, необходимость применения специальных способов предотвращения комбинаторного взрыва очевидна. В последнее время разработан достаточно эффективный метод решения этой проблемы, положенный в основу языка логического программирования Пролог.

Стратегии поиска решений для систем продукций

Логическая модель, представленная в виде совокупности предложений Хорна, являет собой, по сути, систему продукций. Поэтому принципы поиска решений, разрабатываемые для продукционных систем, во многом сходны с принципами поиска решений в языке Турбо-Пролог. Отличия же связаны с наличием у продукций, рассматривавшихся в предыдущей главе, дополнительных условий и атрибутов, обеспечивающих возможность более гибкого управления применением продукций и реализации различных стратегий поиска решений.

Стратегия управления для системы продукций - это принципы выбора правил вывода, запоминания уже опробованных последовательностей правил и баз данных, порожденных их применением. В большинстве приложений искусственного интеллекта информация, доступная стратегии управления, недостаточна для того, чтобы выбрать наиболее подходящее правило при каждом обращении к множеству правил. Поэтому работу систем продукций можно представить как ***процесс поиска***, в котором правила подвергаются испытанию до тех пор, пока не обнаружится, что некоторая их последовательность порождает базу данных, удовлетворяющую терминальному условию (условию, фиксирующему достижение цели). Эффективные стратегии управления предполагают наличие достаточной информации о решаемой задаче, чтобы то правило, которое будет выбрано из множества правил, с большой вероятностью было наиболее подходящим.

Различают *два основных типа стратегий управления*: безвозвратный и пробный. В *безвозвратном* режиме управления выбирается применимое правило и используется необратимо, исключая возможность пересмотра сделанного выбора в дальнейшем. В *пробном* режиме управления выбирается (произвольно или на каком-то разумном основании) применимое правило. Это правило применяется, но резервируется возможность возврата к данной ситуации для применения и оценки эффективности других правил.

Различают также *два типа пробных режимов управления*: с возвращением и с поиском на графе. В режиме управления *с возвращением* при выборе правила определяется некоторая точка возврата. Если последующие вычисления приводят к трудностям в построении решения, то процесс вычисления переходит к предыдущей точке возврата, где применяется другое правило, и процесс продолжается. В режим управления *с поиском на графе* предусмотрено запоминание результатов применения одновременно нескольких последовательностей правил. Здесь используются различные виды графовых структур и процедур поиска на графе.

Выбор стратегии управления. Важной характеристикой вычислительного процесса для выбора правил является объем имеющейся информации или "знаний" о задаче, который используется при вычислениях. В ситуации полной неинформированности выбор делается

абсолютно произвольно, без учета какой-либо имеющейся о данной задаче информации. В ситуации полной информированности стратегия управления руководствуется знанием, достаточным для того, чтобы каждый раз выбрать "верное" правило. В целом вычислительная эффективность системы продукций зависит от того, в каком положении между этими двумя крайними случаями находится данная стратегия управления. Можно разделить вычислительные затраты в системе продукций на две категории:

а) затраты на применение правил (стоимость пути к цели);

б) затраты на управление (стоимость поиска правил, ведущих к цели).

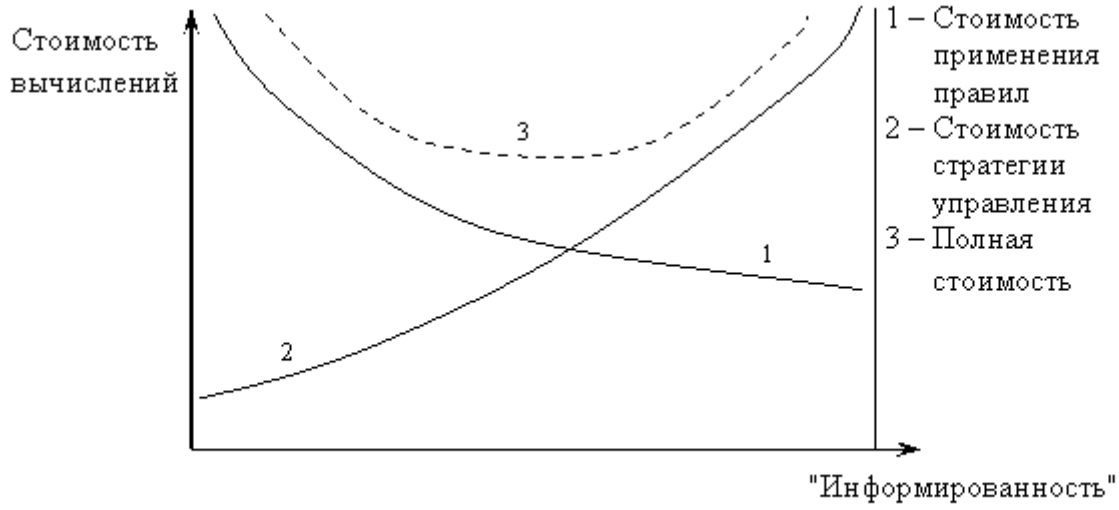

Рис. 4.2. Усредненная полная стоимость вычислений

Полностью неинформированная система управления несет небольшие расходы по пункту б, но это приводит к существенному увеличению расходов по пункту а. Полная информация о решаемой задаче дает обратное
распределение расходов. Эти тенденции отражены на рис 4.2.

В большинстве практических задач стремятся к тому, чтобы минимизировать некоторую комбинацию стоимости пути к цели (применений правил) и стоимости поиска, необходимого для нахождения этого пути (управления), *усредненную* по всем ожидаемым задачам.

Обратные и двусторонние системы продукций. Система продукций для решения игры в 8, при всех рассмотренных режимах управления, работает *непосредственно* от исходного состояния к целевому (т.е. в прямом направлении) и ее можно назвать *прямой системой продукций*.

Эта задача (игра в 8) могла бы быть решена и в обратном направлении – продвижением от целевого состояния к исходному и с применением *обратных ходов*. Каждый ход породил бы *подцелевое* состояние, из которого непосредственно очередное целевое состояние можно достичь за один ход вперед. Система продукций для решения игры в 8 таким способом просто изменила бы назначения состояний и целей и использовала бы правила, соответствующие обратным ходам.

Хотя между системами продукций, работающими над решением задачи в прямом и обратном направлениях, нет формального различия, часто оказывается удобным ввести его в явном виде.

Если в задаче можно выделить четкие на интуитивном уровне *состояния и цели* и если принято решение пользоваться описаниями этих состояний как глобальной базой данных системы продукций, эту систему называют *прямой системой продукций*. Правила применяются к описаниям состояний для порождения новых состояний, и эти правила называются *П-правилами*. Наоборот, если используются описания целей задачи как глобальная база данных, система называется *обратной системой* продукций. Тогда правила применяются к описаниям целей для порождения описания подцелей, и эти правила называются *О-правилами*. Наиболее эффективное направление решения определяется структурой пространства состояний (для игры в 8 начальное и целевое состояния единственны и направление решения не имеет существенного значения).

Часто удачной является попытка решить задачу путем двустороннего поиска – одновременно в прямом и обратном направлениях. Для этого в глобальную базу данных включают описания как состояний, так и целей. К части описания состояний применяются П-правила, а к части описания целей применяются О-правила. Управляющая система при этом должна на каждом шаге поиска

принимать решение об использовании правила того либо другого типа. При таком поиске терминальное условие для управляющей системы, позволяющее сделать вывод о том, что задача решена, должно быть определено как условие соответствия между описаниями состояний, достигнутых с помощью П-правил и О-правил.

Разложимые системы продукций. При использовании режима управления с поиском на графе часто возникает множество путей, приводящих к одной и той же базе данных. Избежать рассмотрения значительной части лишних путей можно, если исходная база данных может быть разбита на отдельные компоненты, которые можно обрабатывать независимо. В таких случаях правила продукций применимы к каждой составляющей независимо (возможно, параллельно). Результаты этих операций также могут подвергаться декомпозиции.

В системах искусственного интеллекта базы данных часто допускают подобное разбиение. Каждое применение правил воздействует только на ту компоненту глобальной базы данных, которая используется для проверки выполнения соответствующего предварительного условия этого правила. Поскольку некоторые из правил применяются, по существу, параллельно, порядок их использования не имеет значения.

Чтобы разложить базу данных на независимые составляющие, необходимо уметь разложить и соответствующее терминальное условие. Если каждая составляющая обрабатывается отдельно, то и глобальное терминальное условие должно быть выражено через терминальные условия каждой составляющей. Наиболее важным является случай, когда глобальное терминальное условие можно выразить как конъюнкцию терминальных условий для каждой составляющей баз данных. Системы продукций, глобальные базы данных и терминальные условия которых допускают такую декомпозицию, называются *разложимыми*.

Хотя возможна параллельная обработка составляющих баз данных, обычно ищется стратегия управления, позволяющая обрабатывать их в некотором последовательном порядке. Существует два основных способа упорядочивания этих компонент: расположить их в каком-то жестком порядке в момент их порождения или динамически переупорядочивать их в ходе обработки. При обработке составляющей может возникнуть разложимая база данных. В таком случае компоненты этой базы данных также обрабатываются по очереди. Обычно для отбора правил используется стратегия с возвращением в сочетании со стратегией жесткого упорядочивания обработки компонент.

4.4. Методы поиска решений в сложных пространствах [4,18]

Специфика представления знаний, организации базы знаний и поиска решений зависят от особенностей проблемной области (ПО), от характера имеющихся знаний о ПО, от сложности поставленной задачи. Что, более кокретно, здесь имеется в виду? Не претендуя на полноту и окончательность характеристик, расшифруем эти положения следующим образом.

Особенности ПО - это степень сложности объекта, его количественная сложность (размер, количество элементов) и его качественная сложность (сложность его организации и поведения - статический, динамический, развивающийся объект).

Особенности знаний о ПО - это полнота, определенность, точность знаний о ПО (полнота, определенность, точность данных об объекте, а также полнота и адекватность модели объекта).

Особенности поставленной задачи - это сложность подлежащей разрешению задачи, количество искомых решений (одно, несколько или все) и качество требуемых решений (наличие ограничений, оптимальность, время, память, точность и т.п.).

По этим признакам наиболее простая проблемная ситуация возникает в том случае, когда ПО мала, статична и однородна; имеются полные, точные данные и единая адекватная модель; задача состоит в отыскании любого одного решения без каких-либо особенных ограничений. Примером такого рода проблемы может быть игра в "крестики и нолики" или проверка наличия заданного типа товара на складе небольшой торговой фирмы.

Напротив, наиболее сложная проблемная ситуация возникает, если ПО велика, неоднородна, имеет сложную и меняющуюся организацию; данные о ПО неполны, неточны и сомнительны, а модель ПО неоднородна и не вполне адекватна; задача же состоит в отыскании всех решений в условиях множества различных ограничений. Примером такого рода проблем могут служить

проблемы распознавания человеческой речи, перевода с одного естественного языка на другой, оптимального планирования развития большого города, региона или страны.

Хороших универсальных методов поиска решений не существует и не может существовать, так как эффективность метода достигается как раз благодаря учету специфики ПО и решаемой проблемы. Поэтому в разных случаях оказываются применимыми различные методы, которые условно можно классифицировать следующим образом:

1. *Методы поиска в одном пространстве* - применяются в случаях, когда ПО статична и невелика, возможно построение единой полной модели и получение полных и точных данных;
2. *Методы поиска в иерархии пространств* - применяются в случаях, когда ПО велика, но возможно представление знаний о ней в виде иерархии моделей первого типа и получение полных и точных данных;
3. *Методы поиска в условиях неопределенности* - применяются в случаях, когда ПО недостаточно изучена, имеющиеся данные неполны, ненадежны и недостаточно точны;
4. *Методы поиска в динамических пространствах* - применяются в случаях, когда ПО представляет собой эволюционирующий, изменяющийся во времени и в пространстве, развивающийся объект;
5. *Методы поиска на основе нескольких моделей* - применяются в случаях, когда ПО неоднородна, агрегирована из объектов различной природы, представление которых требует использования специфических моделей.

Методы поиска в одном пространстве

***Поиск в пространстве состояний*.** В системах искусственного интеллекта, особенно на начальных этапах их развития, задача поиска представлялась как "поиск в пространстве состояний". Эта концепция хорошо согласуется с представлением знаний системой продукций, каждая из которых описывает действие или заключение, которые можно сделать в сложившейся ситуации (состоянии).

При решении задач на основе знаний, как правило, существует не единственный способ достижения цели поиска, обычно требуется опробовать ряд путей и, значит, желательно иметь возможность оценивать шансы на успех при движении по каждому из них.

Подход на основе "поиска в пространстве состояний" сложился при автоматизации решения игровых задач, отличающихся тем, что каждая такая игра может быть представлена как множество характеризуемых конечным набором признаков ситуаций, смена которых в процессе игры происходит по строго определенным правилам, число которых тоже конечно. Типичными примерами являются игры в шахматы, в шашки, в 8, в крестики-нолики и многие другие. Рассмотрим подробнее этот подход.

Игру в *шахматы* можно теоретически представить тройкой (S_0,F,S_T), где S_0 - начальное состояние (расположение шахматных фигур на доске); S_T - множество заключительных состояний (расположений фигур, означающих мат или ничью); F - множество допустимых по шахматным правилам ходов, реализация каждого из которых преобразует текущее состояние (расположение фигур) в новое. Заключительные состояния множества S_T в шахматах явно перечислить невозможно, поэтому множество S_T можно лишь описать через свойства заключительных состояний (расположений фигур, означающих мат или ничью). Задача состоит в том, чтобы найти последовательность ходов (операторов), преобразующих S_0 в одно из состояний (каждый из игроков желает свое) множества S_T. Описать процесс поиска можно с помощью графа, вершины которого соответствуют состояниям (шахматным позициям), а дуги - переходам из состояния в состояние под действием операторов (выполняемых ходов).

Аналогичным образом может быть описана игра в *крестики-нолики*. В этой игре позиция задается матрицей 3х3. Исходной позицией является пустая матрица. Два игрока (один из которых играет фишками X, а другой O) поочередно помещают свои фишки в свободные клетки матрицы. Цель игрока - установить в линию 3 своих фишки. Побеждает тот, кто это сделает первым. В этой игре, в отличие от шахмат, есть лишь один тип оператора - помещение фишки в клетку матрицы.

Методы поиска без информации дают решение задачи, но в процессе поиска создается слишком много вариантов. Поскольку на практике всегда существуют ограничения на время и память для реализации процесса поиска, такие методы оказываются весьма неэффективными. Поэтому не только в шахматах, но и в такой простой игре, как крестики-нолики, возникает вопрос о выборе наилучшего в данной позиции хода. Действительно, простейшая стратегия - перебрать все варианты продолжения игры и выбрать ход, соответствующий выигрышному варианту. Однако

подсчитано, что в шахматах просмотр позиции лишь на 5 ходов вперед уже дает 10^{15} вариантов, а в такой простой игре, как крестики-нолики существует 362880 (9!) позиций, число которых за счет учета симметрии можно сократить до 60000, но это тоже немало. Поэтому вместо полного перебора вариантов используют стратегии с применением "эвристических правил", позволяющих существенно сократить объем перебора за счет снижения гарантии успешности поиска.

Информацию о задаче, которая позволяет сократить поиск решения, называют *эвристической*, а процедуры поиска, использующие ее, – *методами эвристического поиска*. Эвристическая информация используется таким образом, чтобы процесс поиска распространялся только по наиболее перспективным направлениям. Для применения эвристического поиска нужна оценка эффективности вариантов, которую обычно получают с помощью некоторой "оценочной функции". Например, в игре крестики-нолики можно использовать достаточно простую оценочную функцию. Значение оценки доступных игроку ходов можно получить, подсчитывая число открытых игроку линий и вычитая число таких линий у противника при выборе того или иного хода. На рис. 4.3 приведен пример текущей позиции и варианты ходов игрока X с их оценками. Наилучшим считается ход, дающий наиболшее значение разности числа открытых линий (в данном примере это вариант 3).

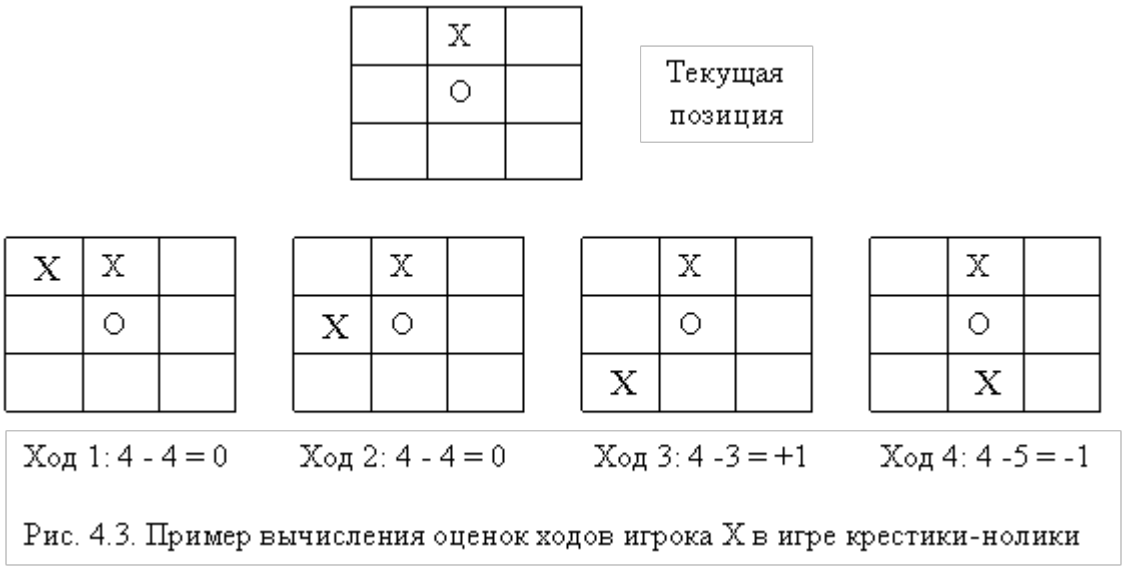

Рис. 4.3. Пример вычисления оценок ходов игрока X в игре крестики-нолики

Во второй главе было также рассмотрено применение концепции поиска в пространстве состояний на примере игры в 8, тоже ориентированное на использование эвристической оценочной функции эффективности ходов. Для вычисления "перспективности" вершин можно применить, например, следующую оценочную функцию. Обозначим эту функцию символом f. Тогда f(n) будет давать ее значение в вершине n дерева поиска. Будем считать, что вершина дерева с меньшей оценкой имеет большую вероятность оказаться на оптимальном пути. Тогда в качестве оценочной функции можно взять выражение f(n) = d(n) + W(n), где d(n) –глубина вершины n на дереве поиска и W(n) – число не находящихся на нужном месте клеток в базе данных, связанной с вершиной n. Таким образом, в процессе поиска варианты его продолжения следует рассматривать в порядке возрастания значения функции f(n). Использование этого правила будет способствовать сокращению среднего числа ходов, затрачиваемых на решение задачи.

Поиск методом редукции. Поиск методом редукции организуется в тех случаях, когда на множестве задач может быть выявлено отношение включения: "часть-целое", "задача-подзадача", "общий случай-частный случай". Цель состоит в том, чтобы представить сложную задачу как совокупность более простых относительно независимо решаемых задач.

Наиболее предпочтительным является случай, когда удается представить процесс решения сложной задачи в виде уже рассматривавшегося нами в разделе 4.2 дизъюнктивно-конъюнктивного дерева (ДК-дерева), которое обычно изображается в форме графа с вершинами двух типов: & и $\vee$. Конъюнктивная вершина (&) требует решения всех своих дочерних подзадач, дизъюнктивная же вершина ($\vee$) удовлетворяется решением хотя бы одной из дочерних подзадач. Цель поиска в этом случае состоит в нахождении при заданных условиях задачи "решающего подграфа" (части ДК-дерева, удовлетворяющей все его вершины типов & и $\vee$). На рис. 4.4

приведен абстрактный пример ДК-дерева, на котором жирными линиями выделен "решающий подграф", представляющий собой тоже ДК-дерево.

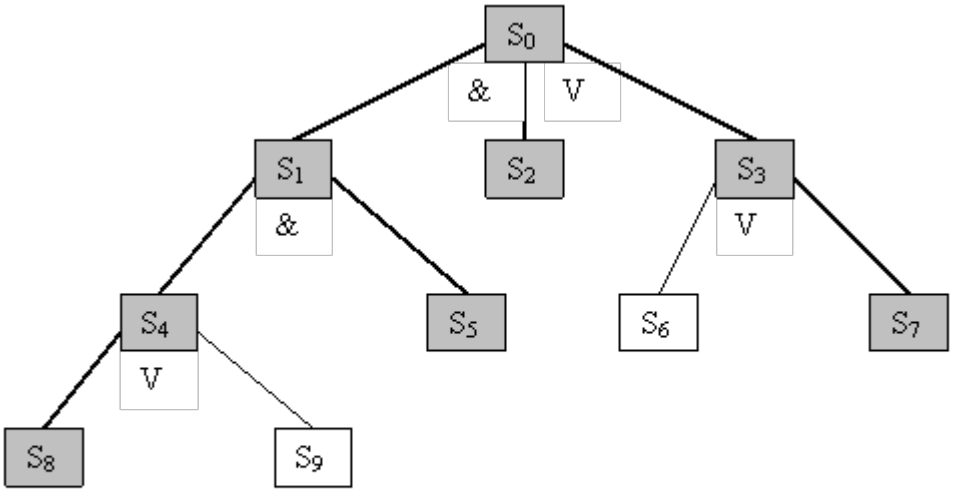

Рис. 4.4. Пример ДК-дерева. Затененными вершинами и жирными ребрами выделен один из 27 вариантов решающего подграфа

Смешивание дизъюнктивных и конъюнктивных вершин на одном уровне создает неудобства в организации поиска решения, поэтому часто в таких случаях в ДК-дерево вводят фиктивные ребра и вершины. Фиктивность таких вершин состоит в том, что для их выполнения не требуется решения каких-либо задач. По сути, они являют собой лишь логические операторы (например, в ДК-дерево на рис. 4.4 целесообразно ввести между вершиной S_0 и вершинами S_1 и S_2 фиктивную вершину S_{10}, роль которой сводится к констатации факта решения задач S_1 и S_2 и передаче сообщения об этом вершине S_0). Обобщением этого подхода являются различные методы структурирования процессов решения задач, широко применяемые в программировании, но они не годятся для поиска в больших пространствах.

Поиск способом "генерация-проверка". Если дерево поиска явно не может быть задано, что обычно бывает при бесконечном или просто слишком большом пространстве поиска, поиск организуется путем создания генератора возможных решений и распознавателя степени их пригодности.

В идеальном случае генератор должен быть *полным* (т.е. порождать все варианты структур, могущие быть решением) и *неизбыточным* (т.е. не должен порождать вариантов структур, не могущих быть решением).

Обеспечить полноту генерации в принципе несложно (например, используя полный перебор, что просто в реализации, но неэффективно в применении). Наилучшим же случаем генератора, обладающего свойством полноты, является тот, который генерирует все те и только те варианты структур, что удовлетворяют условиям задачи.

Неизбыточность, как правило, обеспечить не удается (поскольку, вообще говоря, избыточными можно считать все варианты, кроме искомых). Поэтому, чем больше требований и ограничений удается учесть в генераторе вариантов структур, тем меньше избыточность и соответственно тем меньше оказывается пространство поиска и меньше работы возлагается на распознаватель искомых решений (реализующий "целевую функцию", если говорить на языке методов оптимизации).

Применение этого способа поиска решений сложных задач весьма широко распространено не только в системах искусственного интеллекта, но и в таких областях, как исследование операций, методы оптимизации (например, метод ветвей и границ), теория формальных языков. Еще более широкое применение этот способ находит в самых различных сложных ситуациях, возникающих на практике. Типичным примером использования этого подхода являются различные системы конкурсного отбора (экономических, архитектурных, строительных, инженерных проектов), где роль генераторов вариантов исполняют участники конкурсов, а роль распознавателей - конкурсные комиссии.

Методы поиска в иерархии пространств

Если пространство поиска не только бесконечно или слишком велико, но и сложно организовано, метод "генерация-проверка" применить ко всему пространству поиска как единому целому часто невозможно. В таких случаях стремятся пространство поиска разбить тем или иным способом на ряд подпространств приемлемого размера и сложности, допускающих применение тех или иных методов поиска в одном пространстве. Наглядным примером такого поиска может служить задача прокладки маршрута движения из одного пункта в другой, при решении которой сначала определяется последовательность промежуточных пунктов, маршруты между которыми затем уточняются с помощью карт более мелкого масштаба.

Поиск в факторизуемых пространствах. Пространство поиска называют *факторизуемым*, если оно разбивается на непересекающиеся подпространства (области поиска, классы) частичными (неполными) решениями. Примерами действующих систем, в которых реализован этот способ поиска, являются системы Heuristic DENDRAL и Meta-DENDRAL, предназначенные для определения структуры химических молекул по масс-спектрограммам. Необходимость в создании этих систем обусловлена тем, что человеку очень трудно представить себе по масс-спектру структуру молекулы, включающей от 100 до 300 элементов.

Основным процессом в системе DENDRAL выступал процесс "порождение-проверка", но практика показала, что необходима также планирующая программа для выработки ограничений, учитываемых порождающей и проверяющей программами. В результате получился цикл "планирование-порождение-проверка". Система DENDRAL не моделирует процесс мышления химика, но дополняет его методы, производя тщательный поиск в пространстве возможных молекулярных структур.

Во время фазы планирования DENDRAL по масс-спектрограмме выдает списки нужных и запрещенных оснований (GOODLIS и BADLIST - "хороший список" и "плохой список"), классифицирующие вещество с помощью знаний по масс-спектрографии, закодированных в форме продукций.

Многие из правил-продукций планировщика весьма эффективно сужают область поиска, не только ограничивая поиск типом соединения, но и количественными характеристиками. Например, знание о том, что спектр содержит 8C, 16H, 1O, сокращает число возможных структур с 698 до 3, так как в список BADLIST будут занесены все структуры, кроме содержащих "этил-кетон-3".

Генератор структур в DENDRAL работает в составе системы, но может использоваться и автономно в диалоге с химиком, предоставляя ему информацию и возможность задавать дополнительные *ограничения* трех типов: *графические* (исключить заданного типа структуры); *синтаксические* (исключить неправдоподобные по валентности структуры); *семантические* (учесть данные, полученные в других проверках).

Возможности этой системы были проверены путем использования порожденных ею правил для предсказания масс-спектров новых молекул. Система не только повторно открыла уже известные правила масс-спектрометрии для двух классов молекул, но и неизвестные для трех родственных семейств структур.

Возможности факторизации пространства поиска в данном случае обусловлены спецификой предметной области, поскольку тот или иной набор признаков, характеризующих масс-спектр молекулы, резко сужает возможный состав соответствующего множества топологических структур молекулы.

Ассоциативный и каузальный поиск решения. Основные замечания в адрес систем MYCIN и PROSPECTOR состояли в следующем:

а) отсутствие прямого моделирования стратегий рассуждений, используемых экспертами, что снижает доверие к заключениям;

б) отсутствие использования знаний, заложенных в причинно-следственных (каузальных) отношениях, что снижает достоверность заключений.

Иначе говоря, ставился вопрос о построении причинно-следственной модели объекта экспертизы (в данном случае - заболевания) и о воспроизведении на ее основе человеческого способа рассуждений, осуществляемых специалистом. Без осознаваемых пользователем картины заболевания и причинно-следственных связей между рассматриваемыми явлениями ему сложно понять и, значит, поверить в даваемые ему системой рекомендации.

В некоторых системах эти вопросы были частично решены введением метаправил, формирующих для пользователя как бы обобщенную картину рассуждений. В системах INTERNIST и CASNET были предприняты попытки решить эти проблемы прямым способом.

Система INTERNIST разработана в Питтсбургском университете (США). Прототип создан в 1974 г. и с тех пор совершенствуется и используется. Создана и новая версия этой системы (CADUCEUS). Одно из замечательных свойств этой системы состоит в том, что процесс постановки диагноза явно приближается к модели человеческого мыслительного процесса, позволяя тем самым установить взаимосвязь между конкретными болезнями (причинами) и их проявлениями. INTERNIST решает только задачу постановки диагноза, но не лечения. Процесс постановки диагноза включает две стадии:

а) ограничение пространства диагностирования путем выбора правдоподобных гипотез о возможных заболеваниях ("дифференциальная модель диагностирования");

б) применение некоторой стратегии (с учетом набора правдоподобных гипотез) для окончательного решения задачи диагностирования, наиболее полно отвечающего симптомам.

Выполнение этих заданий позволяет резко сузить область рассуждений и выдвижения гипотез, сбора новых данных и определения стратегии идентификации заболевания.

Знания о заболеваниях в INTERNIST представлены деревом заболеваний (его фрагмент показан на рис. 4.5). Уровни дерева заболеваний связаны между собой отношением *тип-разновидность*.

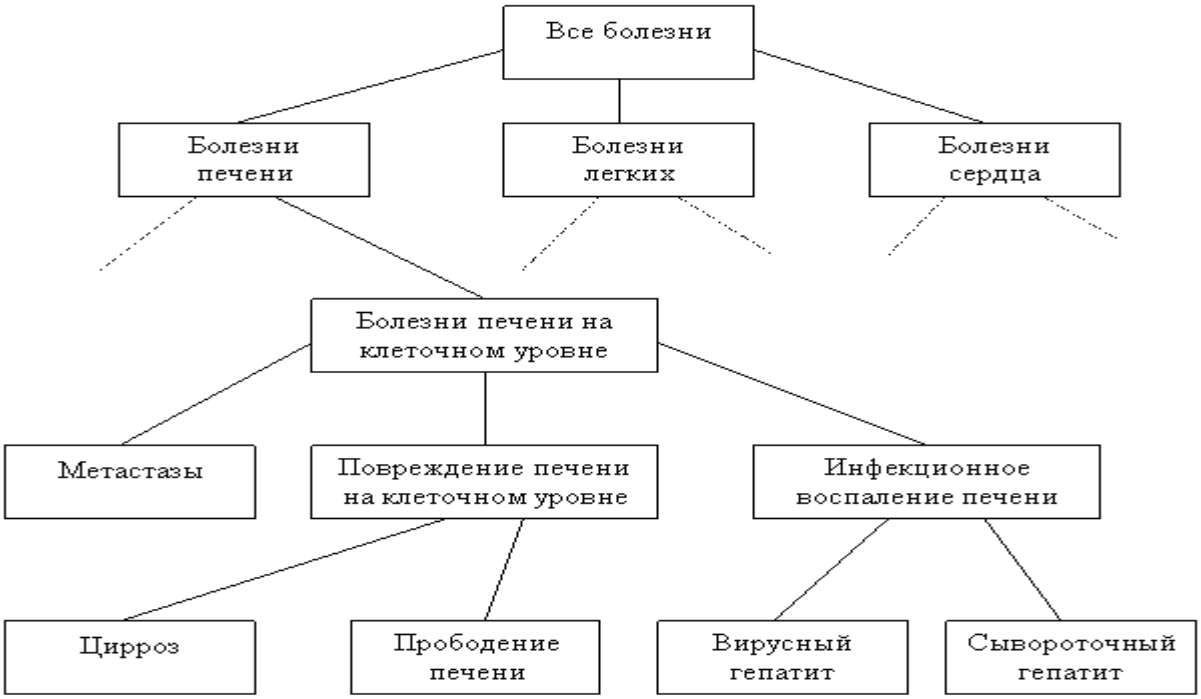

Рис.4.5. Фрагмент "дерева заболеваний" системы INTERNIST

Все заболевания в дереве соотносятся с их симптомами с помощью отношений *вызывает* и *показывает*. Мощность этих отношений оценивается частотой, с какой у данной болезни проявляется данный симптом. Каждому симптому ставится в соответствие две характеристики: *класс* и *значимость*. *Класс* - это мера стоимости (риск и затраты для пациента) проверки правильности симптома. *Значимость* - это мера важности (информативности) данного симптома для постановки диагноза. Эти отношения пронизывают всё дерево заболеваний.

В результате ввода данных о симптомах порождаются *модели заболевания*, используемые как гипотезы. Модель порождается добавлением четырех списков:

а) симптомы наблюдаемые, но не относящиеся к заболеванию;

б) симптомы не зарегистрированные, но которые должны быть;

в) симптомы наблюдаемые и относящиеся к заболеванию;

г) симптомы ожидаемые, но не зарегистрированные.

Каждая модель оценивается по *значимости* и планируется дальнейшая постановка диагноза с учетом *класса* симптомов.

Система CASNET (Causal ASsociational NETwork) разработана в первой половине 70-х годов в Рутгерском университете (США) для исследования стратегий диагностирования на основе психологических и функциональных моделей заболеваний. В качестве предметной области при разработке этой системы была выбрана глаукома из-за ее локальности и относительной структурной простоты.

Заболевание здесь рассматривается как процесс, состоящий из переходов от одного патофизиологического состояния к другому. Диагноз определяется путем идентификации взаимосвязи картины каузальных маршрутов, характерных для пациента, и категорий заболеваний. Преимущества такого описания болезни заключаются в том, что каузальную модель можно использовать, во-первых, для моделирования развития заболевания во времени и, во-вторых, для прогноза течения болезни как при наличии лечения, так и без него.

Знания в CASNET представляются четырьмя "слоями", три из которых описывают собственно болезнь, а четвертый - схемы ее лечения. Центральное место в модели болезни занимает описание патофизиологических состояний в форме семантической сети. Узлами сети представлены физиологические состояния, а дугами - причинные связи и переходы между состояниями заболевания. Маршрут от начального до конечного состояния отражает полный цикл течения болезни (по нарастанию). Ниже слоя описания патофизиологических состояний находится слой наблюдений, связанный со слоем состояний ассоциативными связями. Здесь узлами сети являются симптомы и наблюдения, по отдельности или совокупно ассоциирующиеся с патофизиологическими состояниями. Выше слоя описания патофизиологических состояний находится слой категорий заболеваний, связанный с патофизиологическими состояниями классификационными связями. Через категории заболеваний патофизиологические состояния связываются со схемами лечения, способ реализации которых уточняется по данным из слоя наблюдений. Причинным, ассоциативным и классификационным связям приписываются весовые коэффициенты.

Процесс постановки диагноза в CASNET сводится к исследованию возможных для пациента маршрутов движения от начальных патофизиологических состояний к заключительным. Сеанс начинается с постановки вопросов врачу-клиницисту о симптомах пациента, которые используются для присвоения патофизиологическим состояниям значений: *подтверждено*, *опровергнуто* или *неизвестно* (т.е. +1,-1,0). При этом используются показатели доверия при каузальных дугах и весовые показатели, связанные с ограничениями наблюдений. Значение патофизиологического состояния принимается, если степень доверия переходит за порог положительного или отрицательного, а в интервале порогов считается *неизвестно*.

Далее осуществляется интерпретация хода заболевания в слое патофизиологических состояний. Наиболее вероятные причины болезни задаются как "стартовые состояния" в сети патофизиологических состояний, из которых прокладываются маршруты, не содержащие ни одного опровергнутого состояния и проходящие через наибольшее число подтвержденных состояний (т.е. маршруты с наибольшей степенью подтверждения).

Идентифицировав один или несколько каузальных маршрутов, отражающих текущее состояние пациента, обращаются в слой категорий заболеваний и через них - к схемам лечения. Для каждого пациента используются наиболее вероятные стартовые состояния для более достоверной идентификации категории.

Таким образом, система INTERNIST является примером поиска решения по байесовскому принципу - путем выдвижения гипотез на основе введенных данных о пациенте с последующей их оценкой с использованием дополнительных симптомов. Заболевание здесь рассматривается в виде статичной категории и диагноз представляется как отнесение пациента к одной или нескольким категориям. Хотя такой подход и позволяет эффективно идентифицировать заболевания, он не дает представления о развитии заболевания и, значит, не дает убедительного объяснения выдаваемым заключениям.

Примененный в системе CASNET подход имеет ряд своих преимуществ:

а) представление процесса заболевания в явном виде позволяет отслеживать состояния пациента в процессе заболевания и, значит, анализировать реакции на лечение;

б) каузальные маршруты, завершающиеся на подтвержденных состояниях, дают объяснение диагноза, а неподтвержденные состояния указывают на возможное развитие болезни;

в) каузальные маршруты не только делают объяснения системы понятными для клинициста, но и повышают достоверность идентификации, ибо динамика намного информативнее статики.

Классификационный подход здесь применяется не только как способ идентификации, но и как средство обобщения для перехода на категориальный уровень знания, что говорит о необходимости и возможности применения не только правил, но и метаправил.

В обоих системах базы знаний созданы в результате глубокого изучения и учета специфики предметной области, несмотря на применение весьма общих концептуальных принципов искусственного интеллекта.

Общее правило, видимо, заключается в том, что эффективность решения задач достигается либо за счет учета специфики самих задач в методах поиска решения, либо специфики пространства поиска при использовании универсальных методов.

Представление и обработка нечетких знаний

До сих пор мы не принимали во внимание тот факт, что в реальных условиях знания, которыми располагает человек, всегда в какой-то степени неполны, приближенны, ненадежны. Так, в медицине всегда остаются сомнения в диагнозе заболевания, но отсутствует возможность ждать абсолютно точных свидетельств. В геологии тоже оценка месторождений из-за большой стоимости полномасштабного их изучения всегда имеется лишь приближенная. Тем не менее людям на основе таких знаний все же удается делать достаточно обоснованные выводы и принимать разумные решения. Следовательно, чтобы интеллектуальные системы были действительно полезны, они должны быть способны учитывать неполную определенность знаний и успешно действовать в таких условиях.

Неопределенность (не-фактор) может иметь различную природу. Наиболее распространенный тип недостаточной определенности знаний обусловлен объективными причинами: действием случайных и неучтенных обстоятельств, неточностью измерительных приборов, ограниченными способностями органов чувств человека, отсутствием возможности получения необходимых свидетельств. В таких случаях люди в оценках и рассуждениях прибегают к использованию вероятностей, допусков и шансов (например шансов победить на выборах). Другой тип неопределенности обусловлен субъективными причинами: нечеткостью содержания используемых человеком понятий (например "толпа"), неоднозначностью смысла слов и высказываний (например "ключ" или знаменитое "казнить нельзя помиловать"). Неоднозначность смысла слов и высказываний часто удается устранить, приняв во внимание контекст, в котором они употребляются, но это тоже получается не всегда или не полностью.

Таким образом, неполная определенность и нечеткость имеющихся знаний - скорее типичная картина при анализе и оценке положения вещей, при построении выводов и рекомендаций, чем исключение. В процессе исследований по искусственному интеллекту для решения этой проблемы выработано несколько подходов.

Самым первым, пожалуй, можно считать использование эвристик в решении задач, в которых достаточно отдаленный прогноз развития событий невозможен (как, например, в шахматной игре). Но самое серьезное внимание этой проблеме стали уделять при создании экспертных систем, и первым здесь был применен вероятностный подход (PROSPECTOR), поскольку теория вероятностей и математическая статистика в тот период были уже достаточно развиты и весьма популярны. Однако проблемы, возникшие на этом пути, заставили обратиться к разработке особых подходов к учету неопределенности в знаниях непосредственно для экспертных систем (коэффициенты уверенности в системах MYCIN и EMYCIN). В дальнейшем исследования в этой области привели к разработке особой (нечеткой) логики, основы которой были заложены Лотфи Заде.

В решении рассматриваемой проблемы применительно к экспертным системам, построенным на основе правил (систем продукций), выделяются четыре основных вопроса:

а) как количественно выразить достоверность, надежность посылок?

б) как выразить степень поддержки заключения конкретной посылкой?

в) как учесть совместное влияние нескольких посылок на заключение?

г) как строить цепочки умозаключений в условиях неопределенности?

На языке продукций эти вопросы приобретают следующий смысл. Будем обозначать $ct(A)$ степень уверенности в A (от англ. certainty - уверенность).

Тогда первый вопрос заключается в том, как количественно выразить степень уверенности *ct*(*A*) в истинности посылки (свидетельства) *A*.

Второй вопрос связан с тем, что истинность посылки *A* в продукции $A \to C$ может не всегда влечь за собой истинность заключения *C* (так высокая температура вызывает лишь определенное подозрение на заболевание гриппом, но не гарантирует правильности диагноза "грипп"). Степень поддержки заключения *C* посылкой *A* в продукции $A \to C$ обозначим через $ct(A \to C)$.

Третий вопрос обусловлен тем, что одно и то же заключение *C* может в различной степени поддерживаться несколькими посылками (например, заключение *C* может поддерживаться посылкой *A* посредством продукции $A \to C$ с уверенностью $ct(A \to C)$ и посылкой *B* посредством продукции $B \to C$ с уверенностью $ct(B \to C)$). В этом случае возникает необходимость учета степени совместной поддержки заключения несколькими посылками.

Последний вопрос вызван необходимостью оценки степени достоверности вывода, полученного посредством цепочки умозаключений (например, вывода *C*, полученного из посылки *A* применением последовательности продукций $A \to B$, $B \to C$, обеспечивающих степени поддержки соответственно $ct(A \to B)$ и $ct(B \to C)$).

3.1. Подход на основе условных вероятностей [4,17,18]

Рассматриваемый здесь подход к построению логического вывода на основе условных вероятностей называют байесовским. Реверенд Байес был английским священником, жившим в XVIII веке, который все свое время отдавал изучению статистики. Байесовский подход не является единственным подходом к построению выводов на основе использования вероятностей, но он представляется удобным в условиях, когда решение приходится принимать на основе части свидетельств и уточнять по мере поступления новых данных.

В сущности, Байес исходит из того, что любому предположению (например, что пациент болен гриппом) может быть приписана некая ненулевая априорная (от лат. a priori - из предшествующего) вероятность того, что оно истинно, чтобы затем путем привлечения новых свидетельств получить апостериорную (от лат. a posteriori - из последующего) вероятность истинности этого предположения. Если выдвинутая гипотеза действительно верна, новые свидетельства должны способствовать увеличению этой вероятности, в противном же случае должны ее уменьшать.

Примем для дальнейших рассуждений следующие обозначения:

$P(H)$ - априорная вероятность истинности гипотезы *H* (от англ. Hypothesis - гипотеза);

$P(H:E)$ - апостериорная вероятность истинности гипотезы *H* при условии, что получено свидетельство *E* (от англ. Evidence - свидетельство);

$P(E:H)$ - вероятность получения свидетельства *E* при условии, что гипотеза *H* верна;

$P(E:неH)$ - вероятность получения свидетельства *E* при условии, что гипотеза *H* неверна.

По определению условных вероятностей имеем

$$P(H:E) = \frac{P(H\ и\ E)}{P(E)} \quad \text{и} \quad P(E:H) = \frac{P(E\ и\ H)}{P(H)}.$$

Учитывая, что $P(H\ и\ E) = P(E\ и\ H)$, получаем теорему Байеса

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

Так как $P(E) = P(E:H) * P(H) + P(E:неH) * P(неH)$ и $P(неH) = 1 - P(H)$, получаем формулу, позволяющую уточнять вероятность истинности проверяемой гипотезы H с учетом полученного свидетельства *E*

$$P(H:E) = \frac{P(E:H)P(H)}{P(E:H)P(H) + P(E:неH)(1 - P(H))}.$$

Таким образом (продолжая рассматривать гипотезу *H* о заболевании пациента гриппом), начав с грубого представления о вероятности заболевания гриппом $P(H)$ и вероятности $P(E)$

свидетельства *E* (например высокой температуры), мы получили более точное представление о вероятности $P(H:E)$ заболевания гриппом при наличии высокой температуры.

Здесь обнаруживаются достоинства байесовского метода. Первоначальная (априорная) оценка вероятности истинности гипотезы $P(H)$ могла быть весьма приближенной, но она позволила путем учета свидетельства *E* получить более точную оценку $P(H:E)$, которую можно теперь использовать в качестве обновленного значения $P(H)$ для нового уточнения с привлечением нового свидетельства. Иначе говоря, процесс уточнения вероятности $P(H)$ можно повторять снова и снова с привлечением все новых и новых свидетельств, каждый раз обращаясь к одной и той же формуле. В конечном счете, если свидетельств окажется достаточно, можно получить окончательный вывод об истинности (если окажется, что $P(H)$ близка к 1) или ложности (если окажется, что $P(H)$ близка к 0) гипотезы *H*.

Шансы и вероятности связаны между собой следующей формулой:

$$O(H) = \frac{P(H)}{1 - P(H)}.$$

В некоторых странах использование шансов более распространено, чем использование вероятностей. Кроме того, использование шансов вместо вероятностей может быть более удобным с точки зрения вычислений.

Переходя к шансам в рассмотренных нами формулах, получим

$$O(H:E) = \frac{P(E:H)}{P(E:\text{не}H)} O(H).$$

Если же перейти к логарифмам величин, а в базе знаний хранить логарифмы отношений P(E:H)/P(E:неH), то все вычисления сводятся просто к суммированию, поскольку

$$\ln|O(H:E)| = \ln|P(E:H)/P(E:\text{не}H)| + \ln|O(H)|.$$

Против использования шансов есть несколько возражений, главное из которых состоит в том, что крайние значения шансов равны "плюс" и "минус" бесконечности, тогда как для вероятностей - это 0 и 1. Поэтому шансы использовать удобно в тех случаях, когда ни одна из гипотез не может быть ни заведомо достоверной, ни заведомо невозможной.

Как принцип байесовский подход выглядит прекрасно, но есть и несколько проблем, связанных с его применением.

Первое замечание касается возможности вычисления величины $P(E)$. Эту величину легко определить, если есть возможность вычислить $P(E:\text{не}H)$, а это не всегда можно сделать (например, можно подсчитать вероятность $P(E:H)$ наличия температуры у пациента при наличии гриппа, но как определить вероятность $P(E:\text{не}H)$ наличия температуры у пациента при отсутствии гриппа?).

Одна из возможностей обойти это затруднение состоит в переходе к полной группе событий, однако это не спасает положение, если состав полной группы событий неизвестен. Можно пользоваться и грубыми оценками, если сохраняется точность диагноза. Кроме того, если $P(H)$ уточняется в ходе работы, то $P(E)$ можно тоже уточнять.

Второе замечание касается используемого в этом подходе предположения о независимости свидетельств. С теоретической точки зрения это замечание очень серьезно, но поскольку в конце процесса диагноза нас интересуют не столько точные значения вероятностей (это больше беспокоит статистиков), сколько соотношения вероятностей, то при одинаковом порядке ошибочности оценок вероятностей гипотез для практики более важной оказывается правильность общей картины, создаваемой экспертной системой.

3.2. Подход с использованием коэффициентов уверенности [4,17,18]

Приведенные в предыдущем разделе замечания по поводу байесовского подхода - лишь часть затруднений, возникающих при использовании вероятностей. Привлекательный, на первый взгляд, метод в реальных условиях сталкивается с фактом нарастающего объема трудно разрешимых проблем, что побудило создателей экспертных систем искать иные подходы.

Коэффициенты уверенности Шортлиффа. В принципе можно создать множество разных схем учета неопределенностей и схем рассуждений на их основе. Наиболее развитой и успешной оказалась схема, реализованная Шортлиффом в MYCIN, представляющая собой основанную на здравом смысле модификацию байесовского метода, позволяющую достаточно просто решить поставленные во вступлении к данному разделу четыре основных вопроса:

а) как количественно выразить достоверность, надежность посылок?

б) как выразить степень поддержки заключения конкретной посылкой?

в) как учесть совместное влияние нескольких посылок на заключение?

г) как строить цепочки умозаключений в условиях неопределенности?

Коэффициенты уверенности для правила с одной посылкой. В данном случае речь идет о вычислении коэффициентов уверенности для правил вида

$E \rightarrow C$ ("если E то C").

Если иметь возможность присваивать коэффициент уверенности как посылке, так и импликации, то их можно использвать для оценки степени определенности заключения, выводимого по данному правилу. Шортлифф применяет в данном случае коэффициенты уверенности подобно вероятностям: коэффициент уверенности $ct(E)$ в посылке подобен $p(E)$; коэффициент уверенности $ct(E \rightarrow C)$ в импликации подобен $p(C{:}E)$. Для определения коэффициента уверенности в заключении Шортлифф использует схему: ct(*посылка*) * ct(*импликация*) = ct(*заключение*).

Пример: "**если** вы обучались в ТПУ, **то** вы прекрасный специалист". Неопределенность здесь имеет место как в посылке (обучался еще не значит, что обучился), так и в импликации (не всякий, кто даже обучился, становится прекрасным специалистом).

Логические комбинации посылок в одном правиле. Посылкой в правиле считается все, что находится между **если** и **то**. В MYCIN избран принцип: делать все правила простыми. Тогда простейшими логическими комбинациями посылок являются их конъюнкия или дизъюнкция, т.е. правила вида $(E_1 \,\&\, E_2) \rightarrow C$ или $(E_1 \vee E_2) \rightarrow C$.

Коэффициент уверенности конъюнкции посылок в MYCIN принято оценивать по наименее надежному свидетельству

$$ct(E_1 \,\&\, E_2) = \min\{ct(E_1), ct(E_2)\}.$$

Соответственно, коэффициент уверенности дизъюнкции посылок в MYCIN принято оценивать по наиболее надежному свидетельству:

$$ct(E_1 \vee E_2) = \max\{ct(E_1), ct(E_2)\}.$$

Впрочем, дизъюнкции стараются разбивать на отдельные правила, т.е. вместо правила $(E_1 \vee E_2) \rightarrow C$ создается пара правил $E_1 \rightarrow C$ и $E_2 \rightarrow C$, так как это позволяет лучше видеть роль каждой посылки в формировании заключения. Но если эксперт полагает, что оценка по наиболее надежному свидетельству лучше отражает суть дела, то возможно использование и правила с логической комбинацией посылок в форме дизъюнкции.

Поддержка одного заключения множеством правил. В этой ситуации тоже могут быть предложены различные способы учета совместного влияния свидетельств на заключение. В MYCIN применена схема, подобная схеме вычисления суммарной вероятности нескольких независимых событий.

Пусть имеются правила $E_1 \rightarrow C$ и $E_2 \rightarrow C$, первое из которых обеспечивает поддержку заключения C с уверенностью ct_1, а второе - с уверенностью ct_2. По логике вещей, если обе посылки E_1 и E_2 верны, эти два правила совместно должны обеспечивать заключению C большую поддержку, чем каждое из них в отдельности. В MYCIN это достигается вычислением степени совместной поддержки по следующей формуле:

$$ct = ct_1 + ct_2 - ct_1 * ct_2.$$

При наличии более двух правил, поддерживающих одно и то же заключение, их совместное влияние может быть учтено последовательным применением этой схемы для объединения суммарной поддержки уже учтенных правил с поддержкой очередного, еще не учтенного правила.

Наглядно суть этой схемы можно представить с помощью диаграмм Венна, применяемых в учебниках по теории вероятностей. Диаграмма Венна (рис. 3.1) представляет собой квадрат размером 1 ×1 (площадью, равной 1, символизирущей сумму вероятностей полной группы независимых событий), в рамках которого овалами соответствующей площади представлены вероятности изображаемых событий. При этом площади пересечений овалов соответствуют вероятностям соответствующих совместных событий.

Согласно рассматриваемой схеме расчета степени совместной поддержки заключения несколькими правилами, степень поддержки в случае, показанном на рис. 3.1, а, равна общей площади, занимаемой овалами, помеченными ct_1 и ct_2, а в случае, приведенном на рис.3.1, б, - равна общей площади, занимаемой овалами, помеченными ct_1, ct_2 и ct_3.

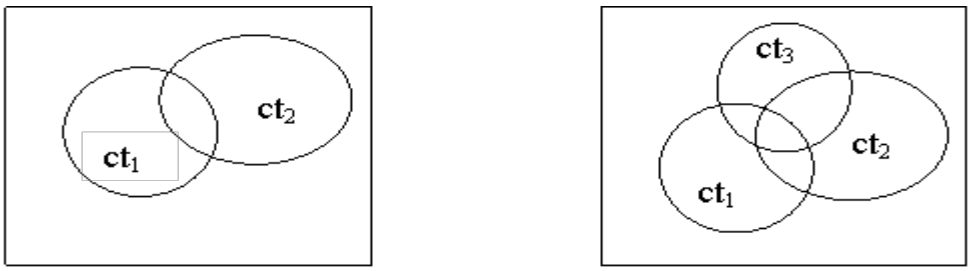

Рис. 3.1. Диаграммы Венна для двух (а) и трех (б) событий

Учитывая, что эти площади равны площади квадрата (которая равна 1) за вычетом ее части, не входящей ни в один из овалов, тот же результат можно получить и с помощью другой схемы расчета.

Пусть ct_i - степень поддержки i-м правилом заключения C, i = 1,2,…,n. Пусть также $\overline{ct_i} = 1 - ct_i$. Тогда степень совместной поддержки заключения C всеми n правилами можно вычислить по формуле $ct = 1 - \overline{ct_1} \cdot \overline{ct_2} \cdots \overline{ct_n}$, согласно которой величина ct равна площади квадрата (т.е. равна1) за вычетом ее части $\overline{ct_1} \cdot \overline{ct_2} \cdots \overline{ct_n}$, не входящей ни в один из овалов.

Таким образом, использованная Шортлиффом схема объединения поддержки одного заключения множеством правил позволяет учитывать коэффициенты уверенности поддерживающих правил в произвольном порядке и объединять их по мере поступления свидетельств.

Вместе с тем важно помнить, что эти принципы учета коэффициентов уверенности исходят из предположения о независимости свидетельств, а также о том, что правомерность такого способа комбинирования не имеет иного обоснования, кроме того, что он прост, соответствует здравому смылу и общему правильному поведению людей, если не относиться к нему излишне доверчиво.

Биполярные схемы для коэффициентов уверенности. Коэффициенты уверенности, примененные в MYCIN, представляют собой грубое подобие вероятностей. В усовершенствованной системе EMYCIN для выражения степени определенности использован интервал [-1,+1] в следующем смысле:

+1 - полное доверие посылке или заключению;

0 - отсутствие знаний о посылке или заключении;

-1 - полное недоверие посылке или заключению.

Промежуточные значения выражают степень доверия или недоверия к ситуации. Все описанные для однополярных коэффициентов процедуры здесь тоже имеют место, но при вычислении max и min учитываются знаки при величинах коэффициентов (например, что +0,1 > -0,2). Биполярность коэффициентов повлияла и на вид используемых формул.

Так, для вычисления коэффициента уверенности отрицания посылки достаточно лишь поменять знак коэффициента, т.е. ct(неE) = -ct(E).

Процедура расчета степени поддержки заключения несколькими правилами тоже претерпела соответствующие изменения:

а) если оба коэффициента положительны, то

$$ct = ct_1 + ct_2 - ct_1 * ct_2;$$

б) если оба коэффициента отрицательны, то

$$ct = ct_1 + ct_2 + ct_1 * ct_2;$$

в) если один из коэффициентов положителен, а другой отрицателен, то

$$ct = \frac{ct_1 + ct_2}{1 - \min(|ct_1|, |ct_2|)},$$

при этом, если один из коэффициентов равен +1, а другой -1, то *ct* = 0.

Работа с биполярными коэффициентами может привести к нереальным результатам, если правила сформулированы неточно. В частности, ошибка возникает, если не учитывается, что правила бывают *обратимыми* (применимыми при любых значениях коэффициентов уверенности в посылке) и *необратимыми* (применимыми лишь при положительных значениях коэффициента уверенности в посылке).

Например, правило "Если у вас грипп, то вызовите врача" необратимо (нереверсивно), поскольку замена посылки и заключения на противоположные превращает его в неверное правило "Если у вас не грипп, то не вызывайте врача". Напротив, правило "Если у вас высокая температура, то примите аспирин" обратимо (реверсивно), так как его обращение приводит к верному правилу "Если у вас нет высокой температуры, то не принимайте аспирин".

Многоступенчатые рассуждения и сети вывода. До сих пор речь шла о ситуациях, когда заключение отделялось от посылки одним шагом рассуждений. Более типична ситуация, когда вывод от посылок отделен рядом промежуточных шагов рассуждений.

Процесс многошаговых рассуждений с применением биполярных коэффициентов продемонстрируем на конкретном примере. Представьте себе, что вы заболели (простуда, вирусная инфекция или грипп). Что предпринять? Сеть вывода, призванная помочь вам в этом случае, приведена на рис. 3.2.

Сеть такого типа, как на рис. 3.2, представляет собой графическое изображение системы продукций, оперирующих с коэффициентами уверенности. Приведем несколько примеров записи продукций по рис. 3.2, предоставив читателю выписать остальные в качестве упражнения:

"**если** у вас насморк **и** мышечные боли **и нет** лихорадки, **то** *с уверенностью 0.7* можно заключить, что это простуда";

"**если** вам меньше 8 лет **или** больше 60 лет, **то** *с уверенностью 0.7* можно заключить, что у вас уязвимый возраст";

"**если** у вас грипп **и не** уязвимый возраст, **то** *с уверенностью 0.4* можно заключить, что вам следует принять аспирин и лечь в постель";

"**если** у вас острый фарингит, **то** *с уверенностью 1.0* можно заключить, что вам следует вызвать врача".

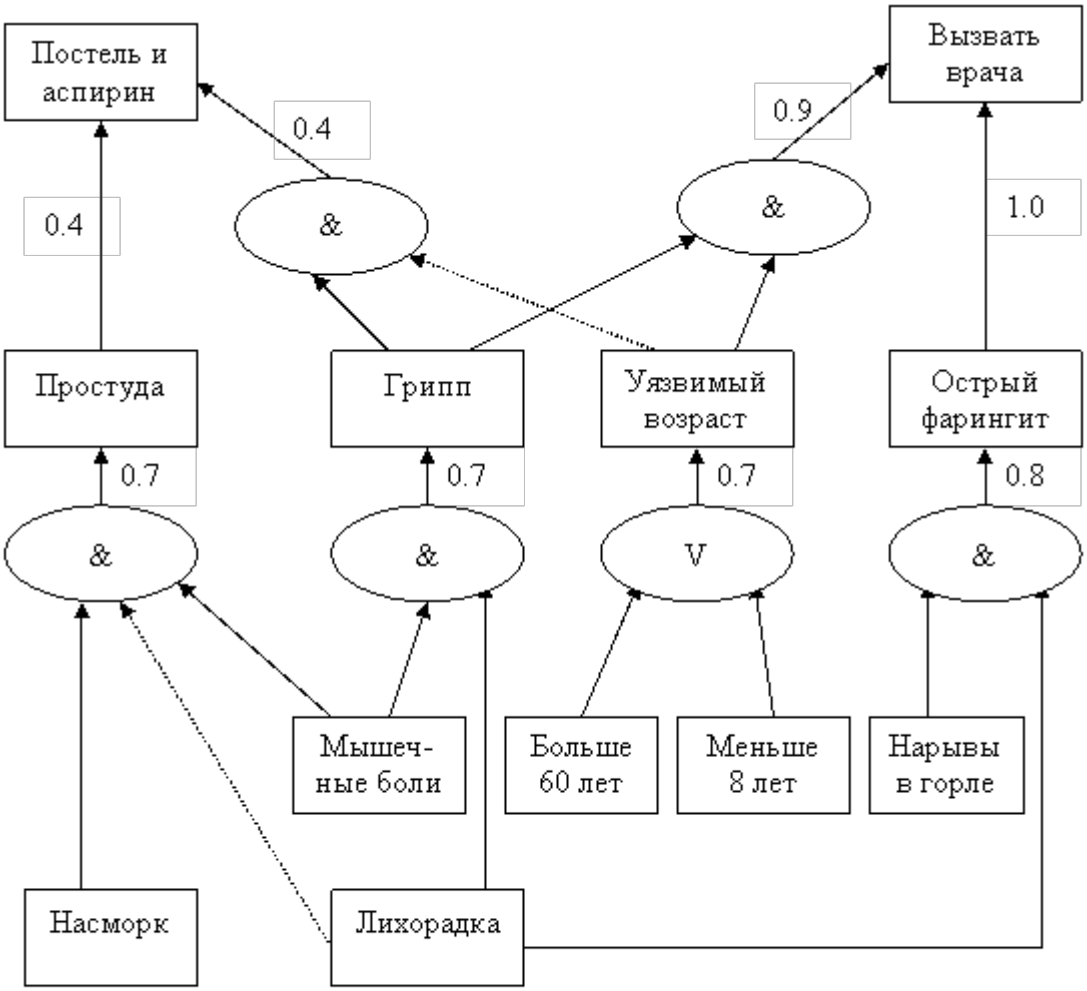

Рис. 3.2. Сеть многоступенчатого вывода (сплошными стрелками указаны требования наличия фактора, а пунктирными - отсутствия)

Чтобы в полном объеме продемонстрировать вычисления с биполярными коэффициентами, воспользуемся абстрактной сетью вывода, изображенной на рис. 3.3.

На рис. 3.3 сплошными стрелками указаны условия без отрицания, а пунктирными - с отрицанием; жирными стрелками отмечены обратимые (реверсивные) правила, а стрелками обычной толщины - необратимые (нереверсивные); числа в прямоугольниках жирным шрифтом - это коэффициенты уверенности в свидетельствах (**E**vidence), а обычным шрифтом - это вычисленные по правилам коэффициенты уверенности в соответствующих заключениях (**C**onclusion); надписи min и max обращают внимание на то, что операции конъюнкция (&) и дизъюнкция ($\vee$) в нечеткой логике означают выбор соответственно наименьшего и наибольшего элемента.

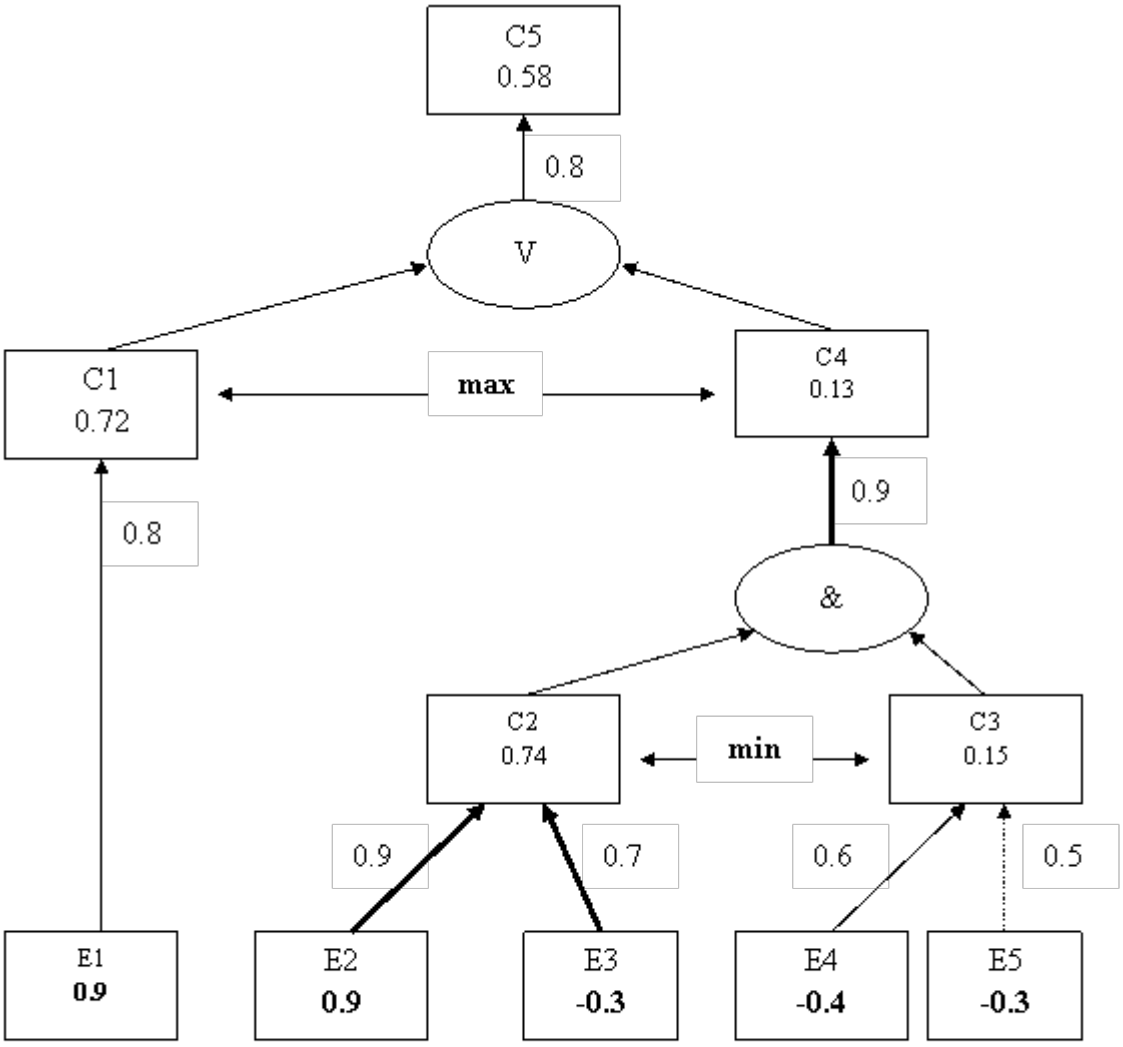

Рис. 3.3. Сеть многоступенчатого вывода, **демонстрирующая вычисления с применением биполярных коэффициентов**

Процесс вычислений идет снизу (от свидетельств E1, E2, E3, E4, E5) вверх (к заключению C5).

Коэффициент увереннности в C1 определяется через простую необратимую импликацию, но коэффициент уверенности в E1 положителен и правило можно применять:

$ct(C1) = 0{,}9 \cdot 0{,}8 = 0{,}72.$

Заключение C2 поддерживают два обратимых правила, дающих коэффициенты уверенности с разным знаком, поэтому их объединение осуществляется по формуле для этого случая:

$ct_1(C2) = 0{,}9 \cdot 0{,}9 = 0{,}81;$

$ct_2(C2) = -0{,}3 \cdot 0{,}7 = -0{,}21;$

$ct(C2) = (0{,}81 - 0{,}21) / (1 - 0{,}21) = 0{,}74.$

Для заключения C3 левое правило применять нельзя, так как оно необратимое, а коэффициент уверенности в посылке отрицателен. Правое правило имеет коэффициент уверенности в посылке, равный -0,3, который (благодаря отрицанию) превращается в 0,3. В итоге получаем

$ct(C3) = 0{,}3 \cdot 0{,}5 = 0{,}15.$

Заключение C4 поддерживается конъюнкцией посылок, поэтому из них выбирается посылка с наименьшим коэффициентом уверенности

$ct(C2\&C3) = \min\{ct(C2); ct(C3)\} = \min\{0{,}15;0{,}74\} = 0{,}15.$

Следовательно, $ct(C4) = 0{,}15 \cdot 0{,}9 = 0{,}13.$

Заключение C5 поддерживается дизъюнкцией посылок, поэтому из них выбирается посылка с наибольшим коэффициентом уверенности:

$ct(C1 \vee C4) = \max\{ct(C1); ct(C4)\} = \max\{0{,}72;0{,}13\} = 0{,}72.$

Следовательно, $ct(C5) = 0{,}72 \cdot 0{,}8 = 0{,}58.$

Принцип последовательного учета свидетельств работает успешно в том случае, когда применима так называемая монотонная логика. Но часто встречаются ситуации, когда она не имеет места. Это те случаи, когда какое-либо очередное свидетельство опровергает выводы, сделанные на основе предшествующих свидетельств. Для таких ситуаций разрабатываются специальные типы логик. Другого вида трудность возникает вследствие "незамкнутости мира": птицы летают, но не все; можно перечислить все предметы в комнате, но нельзя перечислить то, что вне ее. В таких случаях принимается "гипотеза закрытого мира": все, что вне его, то - ложь.

Нечеткая логика Заде [4,7]

Нечетким множеством $\tilde{A}$, по Лотфи Заде, называется множество, определенное на произвольном непустом множестве **X** как множество пар вида

$$\tilde{A} = \{\mu_A(x)/x\}, \text{ где } x \in X, \ \mu_A(x) \in [0,1].$$

Множество X называется *базовым множеством*, или *базовой шкалой* (если множество X линейно упорядочено). Функция $\mu_A(x) : X \to [0,1]$ называется *функцией принадлежности* множества $\tilde{A}$. Величина $\mu_A(x)$ для каждого конкретного $x \in X$ называется *степенью принадлежности* элемента x нечеткому множеству $\tilde{A}$. Принято, что в нечеткое множество $\tilde{A}$ не входят элементы $x \in X$, имеющие $\mu_A(x) = 0$. Подмножество $A \subset X$, содержащее все те элементы $x \in X$, для которых $\mu_A(x) > 0$, называется *носителем* нечеткого множества $\tilde{A}$.

Пример. В рейтинговой системе оценки знаний по каждой дисциплине задается некоторая базовая шкала баллов (обычно используется шкала [0,1000]), на которой задаются интервалы, дающие студенту право на получение оценки "неудовлетворительно", "удовлетворительно", "хорошо", "отлично". Фактически эти интервалы выступают в качестве носителей нечетких множеств "неудовлетворительно", "удовлетворительно", "хорошо", "отлично", так как за разные баллы из одного и того же интервала преподаватель ставит одну и ту же оценку с разной степенью уверенности. На рис.3.4 приведены графики функций принадлежности оценок, характеризующие степень уверенности некоторого преподавателя в оценке знаний студента по дисциплине, в зависимости от величины набранного им по рейтингу балла.

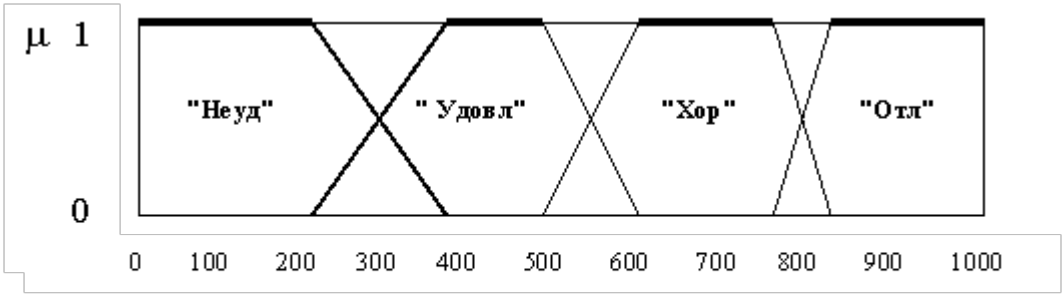

Рис. 3.4. Пример функций принадлежности оценок, заданных на базовой шкале [0,1000] баллов, набранных по рейтингу

Приведенный пример показывает, что функция принадлежности нечеткого множества (понятия) формируется субъективно и может иметь для одного и того же понятия различный вид у разных субъектов и даже у одного и того же субъекта при различных обстоятельствах и настроениях.

Нечеткие высказывания. Высказывание $\tilde{A}$ называется нечетким высказыванием, если допускается, что $\tilde{A}$ может быть одновременно истинным и ложным (в отличие от аристотелевской логики, где такая возможность исключается). Любое оценочное суждение, основанное на неполных или недостоверных данных, является нечетким и сопровождается обычно выражением степени уверенности (или сомнения) в его истинности. Например, утверждение "Наверное, завтра похолодает".

Мера истинности нечеткого высказывания $\tilde{A}$ определяется функцией принадлежности $\mu_A(x)$, $x \in X$, заданной на множестве X = {"ложь", "истина"}.

При таком определении нечеткого высказывания несомненно истинное высказывание характеризуется функцией принадлежности μ_A("истина") = 1 (или μ_A("ложь") = 0). Соответственно несомненно ложное высказывание будет характеризоваться функцией принадлежности μ_A("истина") = 0 (или функцией μ_A("ложь") = 1). Нечеткие высказывания, характеризующиеся равной степенью уверенности и сомнения (т.е. когда μ_A("истина") = 0.5 и μ_A("ложь") = 0.5), называют *нечетко индифферентными.*

В дальнейшем, во избежание путаницы, будем говорить лишь о мере истинности нечетких высказываний, если не оговаривается иное толкование. Кроме того, для упрощения записи, будем обозначать, как это принято в обычной ("четкой" логике), меру истинности нечеткого высказывания тем же симоволом, что и само высказывание (например, вместо μ_A("истина") = 0.8 будем писать $\tilde{A}$ = 0.8).

Логические операции над нечеткими высказываниями. Нечеткие высказывания могут быть простыми и составными. Составные высказывания образуются из простых с помощью логических операций, часто называемых в логике также логическими связками из-за их роли в предложениях естественного языка. Так в обычной речи часто употребляются слова *не, и, или*, и словосочетания *если, …то…*; *тогда и только тогда*; *равносильно*, соответствующие основным логическим операциям математической логики.

В отличие от традиционной математической логики в нечеткой логике этим операциям придается специфический смысл. Причем, в зависимости от области применения, этот смысл может быть различным. Например, при изучении случайных явлений целесообразно степени уверенности рассматривать как вероятности и тогда логические операции над нечеткими высказываниями приобретают смысл известных операций над вероятностями случайных событий.

Здесь будет рассматриваться интерпретация логических операций над нечеткими высказываниями, предложенная основоположником нечеткой логики Лотфи Заде (Lotfi Zadeh) и применяемая преимущественно в тех случаях, когда нечеткость высказываний обусловлена неполнотой информации о предмете суждения. Речь пойдет о так называемой *минимаксной логике* Заде.

Отрицанием нечеткого высказывания $\tilde{A}$ называется нечеткое высказывание $\neg \tilde{A}$, степень истинности которого определяется выражением $\neg \tilde{A}$= 1 - $\tilde{A}$. Отсюда следует, что степень ложности $\neg \tilde{A}$ равна степни истинности $\tilde{A}$.

Конъюнкцией нечетких высказываний $\tilde{A}$ и $\tilde{B}$ называется нечеткое высказывание $\tilde{A}\&\tilde{B}$, степнь истинности которого определяется выражением $\tilde{A}\&\tilde{B} = min(\tilde{A}, \tilde{B})$, т.е. есть степень истинности нечеткого высказывания $\tilde{A}\&\tilde{B}$ определяется наименее истинным высказыванием.

Дизъюнкцией нечетких высказываний $\tilde{A}$ и $\tilde{B}$ называется высказывание $(\tilde{A} \vee \tilde{B})$, степень истинности которого определяется выражением $(\tilde{A} \vee \tilde{B})$=max($\tilde{A}$, $\tilde{B}$). То есть степень истинности нечеткого высказывания $(\tilde{A} \vee \tilde{B})$ определяется наиболее истинным высказыванием.

Импликацией нечетких высказываний $\tilde{A}$ и $\tilde{B}$ называется нечеткое высказывание $\tilde{A} \to \tilde{B}$, степень истинности которого определяется выражением $\tilde{A} \to \tilde{B}$ =max(1- $\tilde{A}$, $\tilde{B}$).

Данное выше определение импликации основано на логической равносильности формулы $\tilde{A} \to \tilde{B}$ и формулы $\neg \tilde{A} \vee \tilde{B}$.

Эквиваленцией (эквивалентностью) нечетких высказываний $\tilde{A}$ и $\tilde{B}$ называется нечеткое высказывание $\tilde{A} \leftrightarrow \tilde{B}$, степень истинности которого определяется выражением $\tilde{A} \leftrightarrow B$=min(max(1- $\tilde{A}$, $\tilde{B}$),max($\tilde{A}$,1- $\tilde{B}$)). Данное определение эквиваленции основано на равносильности формулы $\tilde{A} \leftrightarrow \tilde{B}$ формуле $(\tilde{A} \to \tilde{B})\ \&\ (\tilde{B} \to \tilde{A})$.

Нечеткие высказывания $\tilde{A}$ и $\tilde{B}$ называются *нечетко близкими*, если $\tilde{A} \approx \tilde{B} \geq 0.5$ (т.е.степень эквивалентности высказываний не ниже 0.5), *нечетко индифферентными*, если $\tilde{A} \approx \tilde{B} = 0.5$, и *нечетко неблизкими*, если $\tilde{A} \approx \tilde{B} < 0.5$.

В составных высказываниях порядок выполнения введенных логических операций определяется скобками, а при отсутствии скобок - в следующем порядке: $\neg$, &, $\vee$, $\rightarrow$, $\leftrightarrow$.

Пример. Вычислите степень истинности составного нечеткого высказывания $\tilde{D}$ при условии, что входящие в него простые нечеткие высказывания имеют значения степеней истинности $\tilde{A}$ =0.7, $\tilde{B}$=0.4, $\tilde{C}$=0.9, а формула $\tilde{D}$=($\tilde{A}$&$\neg$ $\tilde{B}$) $\vee$ $\neg$ $\tilde{A}$& $\tilde{B}$) $\rightarrow$ $\neg$($\tilde{A}$& $\tilde{C}$). Если вы правильно используете определения логических операций над нечеткими высказываниями и будете следовать принятому порядку их применения при отсутствии скобок, то вы получите $\tilde{D}$=0.4.

Введенные выше определения, как уже было сказано, представляют собой так называемую минимаксную интерпретацию логических операций над нечеткими высказываниями, предложенную Лотфи Заде. Существуют и другие интерпретации.

В математической логике логические операции $\neg$, &, $\vee$ в совокупности составляют функционально полную систему логических операций. Это значит, что любое высказывание может быть описано логической формулой, составленной из простых высказываний с использованием конечного числа только этих логических операций. Аналогичным свойством по отношению к нечетким высказываниям обладает этот набор логических операций в интерпретации Заде.

Нечеткие логические формулы и их свойства. Нечеткое высказывание, степень истинности которого может принимать произвольное значение из интервала [0,1], Заде называет *нечеткой логической переменной*. В определении понятия нечеткой логической формулы логические переменные и их значения (константы из интервала [0,1]) считаются простейшими нечеткими логическими формулами, а само понятие нечеткой логической формулы вводится индуктивно.

Нечеткой логической формулой называется:

а) нечеткая логическая переменная или константа из интервала [0,1];

б) всякое выражение, построенное из нечетких логических формул применением любого конечного числа логических операций (связок);

в) нечеткими логическими формулами считаются те и только те выражения, которые построены согласно пунктам а) и б).

Рассматривавшиеся ранее составные нечеткие высказывания являются нечеткими логическим формулами, если входящие в них простые нечеткие высказывания рассматривать как нечеткие логические переменные.

Важнейшим фактором в осуществлении преобразований логических формул является равносильность логических формул. В нечеткой логике возможности осуществления равносильных преобразований расширяются за счет того, что для таких преобразований здесь достаточно лишь наличия необходимой степени равносильности нечетких логических формул.

Понятие равносильности нечетких логических формул $\tilde{A}(\tilde{x}_1, \tilde{x}_2,..., \tilde{x}_n)$ и $\tilde{B}(\tilde{x}_1, \tilde{x}_2,..., \tilde{x}_n)$, определенных на наборах значений одних и тех же нечетких логических переменных $\tilde{x}_1, \tilde{x}_2,..., \tilde{x}_n$, вводится как обобщение равносильности четких логических формул через определение степени их равносильности.

Степень равносильности $\mu(\tilde{A}, \tilde{B})$ двух формул $\tilde{A}(\tilde{x}_1, \tilde{x}_2,..., \tilde{x}_n)$ и $\tilde{B}(\tilde{x}_1, \tilde{x}_2,..., \tilde{x}_n)$ определяется выражением

$$\mu(\tilde{A}, \tilde{B}) = \underset{(\tilde{x}_1,..., \tilde{x}_n)}{\&} (\tilde{A}(\tilde{x}_1,..., \tilde{x}_n) \leftrightarrow \tilde{B}(\tilde{x}_1,..., \tilde{x}_n))$$

.

Формулы $\tilde{A}$ и $\tilde{B}$ называют: *нечетко равносильными*, если $\mu(\tilde{A}, \tilde{B}) \geq 0.5$ (пишут $\tilde{A} \approx \tilde{B}$); *взаимно нечетко индифферентными*, если $\mu(\tilde{A}, \tilde{B}) = 0.5$ (пишут $\tilde{A} \approx \tilde{B}$); *нечетко неравносильными*, если $\mu(\tilde{A}, \tilde{B}) < 0.5$ (пишут $\tilde{A} \not\approx \tilde{B}$).

Понятие равносильности четких логических формул, как уже упоминалось, является частным случаем нечеткой равносильности нечетких логических формул. Благодаря тому, что для нечеткой равносильности формул $\tilde{A}$ и $\tilde{B}$ достаточно, чтобы $\square(\tilde{A}, \tilde{B}) \square 0.5$, нечетко равносильными могут быть такие формулы $\tilde{A}$ и $\tilde{B}$, которые в четком понимании не являются равносильными (эквивалентными). Такими, например, являются формулы $\tilde{A}(\tilde{x}_1, \tilde{x}_2) = (\neg\tilde{x}_1 \to \tilde{x}_2)$ и $\tilde{B}(\tilde{x}_1, \tilde{x}_2) = (\tilde{x}_1 \& \neg\tilde{x}_2)$, степень равносильности которых при $\tilde{x}_1 \in \{0{,}8;\ 0{,}6;\ 0{,}7\}$ и $\tilde{x}_2 \in \{0{,}3;\ 0{,}4\}$ равна 0,6.

Хотя нечеткая логика Заде является наиболее теоретически обоснованной и имеет множество практических применений, при ее использовании возникает немало трудностей из-за отсутствия общепринятого языка для выражения уверенности в нечетких суждениях.

Тема 5 Нейронные сети. Основные понятия об естесвенных и искуственных нейронных сетях и нейронах. Классификация нейронных сетей. Программная и аппаратная реализация нейронных сетей.

Прецептроны. Однослойные прецептроны. Многослойные прецептроны. Обучение прецептронов.

Обучающиеся и самообучающиеся системы. Обучение нейронной сети. Обучения с учителем и без учителя. Алгоритм обратного распространения ошибки.

Модели нейронных сетей. Сети встречного распространения. Модель Хопфилда. Модель Кохоннена.

Использование нечетких знаний в моделях нейронных сетей. **Основные понятия теории нейронных сетей**

Теория искусственных нейронных сетей (НС), положившая начало исследованиям по искусственному интеллекту, в последние десятилетия бурно развивается. Актуальность этих исследований подтверждается множеством различных практических применений НС (распознавание речи и зрительных образов, адаптивное управление, прогнозирование, организация ассоциативной памяти и многие другие приложения). Специфика задач вынуждает создавать разнообразные специализированные НС, имеющие различную структуру и функционирующие по различным алгоритмам. Реализация НС может быть аппаратной или программной. В данном случае речь будет идти о программно реализуемых НС. Несмотря на множество различий, НС имеют и общие черты, характеризующие их как искусственные нейронные сети. Искусственная НС состоит из искусственных нейронов и соединений между ними.

Нейроны. Прообразом искусственного нейрона является *биологический нейрон*. Человеческий мозг - это сложная биологическая сеть, состоящая из миллионов связанных между собой клеток, называемых нейронами. Точный механизм работы человеческого мозга неизвестен, но мы знаем о нем достаточно для того, чтобы имитировать некоторые из его способностей, таких как обучение, распознавание образов, обобщение.

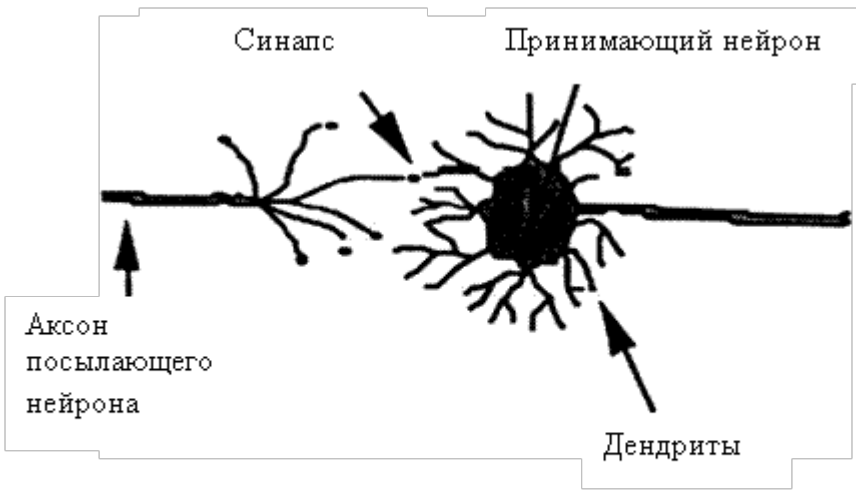

Рис. 6.1. Основные части биологического нейрона

Нейрон мозга имеет четыре основные части (рис. 6.1): тело, входные каналы, выходной канал и соединительные точки между нейронами, называемые синапсами.

Нейрон получает сигналы от многих других нейронов на синапсы, где происходят некоторые процессы, прежде чем сигналы будут посланы в тело нейрона. Синапсы "взвешивают" входные сигналы так, что каждый из сигналов оказывает различное действие на нейрон. Синапс может поднять или понизить уровень сигнала так, что он окажет более сильное или более слабое действие на нейрон. Действие синапса на сигнал может стать причиной "включения" (возбуждения) или "выключения" (торможения) нейрона. Сильно возбужденный нейрон посылает выходной сигнал, а заторможенный - нет. Работа тела нейрона состоит в том, чтобы суммировать все входные сигналы и решить, достаточно ли полного сигнала, чтобы послать выходной сигнал.

Один нейрон может обнаружить и послать сигнал лишь об одной простой вещи. Более сложные явления распознаются группами взаимосвязанных нейронов. Обучение проявляется в мозге в виде изменений в синапсах. Существуют разные теории о том, как это происходит, но общая точка зрения состоит в том, что "обучение" нейрона является функцией поступающих на него сигналов.

Искусственные нейроны (рис. 6.2) имитируют работу нейронов мозга. Искусственные нейроны также называют узлами, единицами или ячейками.

Каждый искусственный нейрон получает на свои входы ("дендриты") сигналы с выходов ("аксонов") многих других нейронов и может находиться, по аналогии с нервными клетками головного мозга, в возбужденном или заторможенном состоянии. Текущее состояние *s* искусственного нейрона определяется как взвешенная сумма $s = \sum_{i=1}^{n} x_i \cdot w_i$ его входов x_i, i=1,…,n.

Роль "синапсов" в искусственном нейроне выполняют весовые коэффициенты w_i, i=1,…,n. Весовой коэффициент характеризует силу синаптической связи между выходом посылающего сигнал нейрона и входом принимающего этот сигнал нейрона.

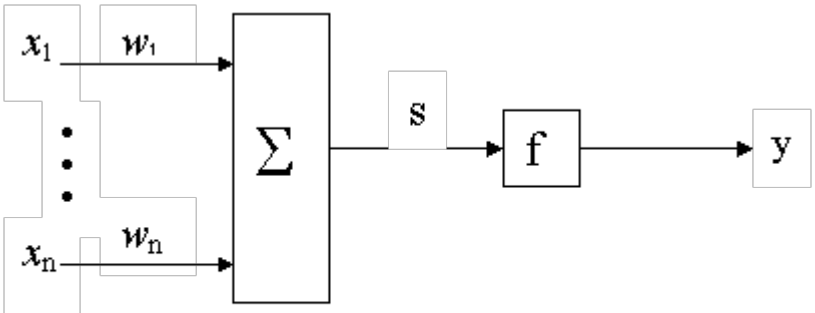

Рис. 6.2. Искусственный нейрон

Выход *y* нейрона есть функция *y = f(s)* его состояния, называемая выходной или передаточной функцией. Передаточная функция *f* может иметь различный вид, как показано на рис. 6.3.

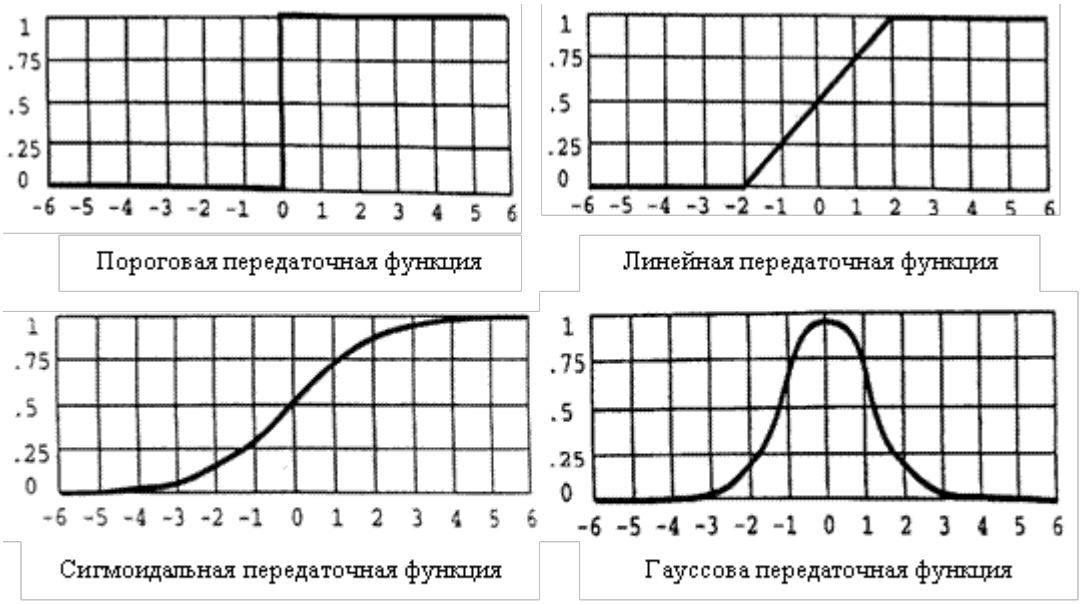

Рис. 6.3. Главные типы передаточных функций

Пороговая передаточная функция может принимать только два значения: пока значение *s* ниже порога, выход нейрона *y* всегда имеет низкое значение (равен 0); пока значение *s* выше порога, выход нейрона всегда имеет высокое значение (равен 1). При значении *s*, равном порогу, выход нейрона *y* совершает скачок от низкого значения к высокому (если *s* увеличивается) или от высокого к низкому (если *s* уменьшается). Пороговая передаточная функция использовалась в некоторых ранних исследованиях нейронных сетей (Маккаллок и Питтс, 1943) .

Линейная пороговая передаточная функция - это функция, выход которой *y* равен константе, умноженной на вход *s*, пока вход находится в некотором заданном интервале. Ниже этого интервала функция имеет постоянное низкое значение (равна 0), а выше этого интервала - постоянное высокое значение (равна 1). Центром является значение *s*, при котором *y*=0.5. Прирост значения этой функции в пределах интервала равен наклону линейной части кривой. Линейная пороговая передаточная функция позволяет строить нейронные сети, обладающие более интересным поведением, чем сети, построенные с помощью пороговой передаточной функции.

Сигмоидальная передаточная функция, известна также как S-образная фунция, полулинейная, функция с насыщением или логистическая функция. В общем виде логистическая кривая описывается выражением $y = k/(1+e^{a+bs})$. В качестве передаточной функции искусственного нейрона обычно используется логистическая функция вида $y = 1/(1+e^{-bs})$. При уменьшении *b* сигмоид становится более пологим, в пределе при *b*=0 вырождаясь в горизонтальную линию на уровне *y*=0.5, при увеличении *b* сигмоид приближается по внешнему виду к пороговой функции единичного скачка с порогом в точке *s*=0. Из выражения для сигмоида очевидно, что выходное значение нейрона лежит в диапазоне [0,1]. Центр сигмоида - это значение его входа *s*, при котором значение выхода y=0.5. Прирост значения выхода сигмоида прямо пропорционален производной функции в центральной точке. Следует отметить, что сигмоидная функция дифференцируема на всей оси абсцисс, что используется в некоторых алгоритмах обучения (одно из ценных свойств сигмоидной функции - простое выражение для ее производной $f'(s) = b \cdot f(s) \cdot (1 - f(s))$, применение которого будет рассмотрено в дальнейшем). Кроме того она обладает свойством усиливать слабые сигналы лучше, чем большие, и предотвращает насыщение от больших сигналов, так как они соответствуют областям аргументов, где сигмоид имеет пологий наклон.

Гауссова передаточная функция, известная также как колоколообразная, является наиболее необычной передаточной функцией в нейронных сетях. При использовании этой функции нейрон формирует отклик на вход избирательно. Центром служит значение входа *s*, при котором выход *y*=1. Прирост пропорционален гауссову отклонению. Полезность этой передаточной функции находится в стадии исследований.

Слои нейронов. Нейронная сеть состоит из соединенных между собой нейронов. Выбор способа соединения нейронов между собой - один из наиболее важных вопросов при построении нейронной сети. Нейроны могут размещаться в трех типах мест (рис. 6.4): во входном слое, во внутренних (скрытых) слоях, в выходном слое.

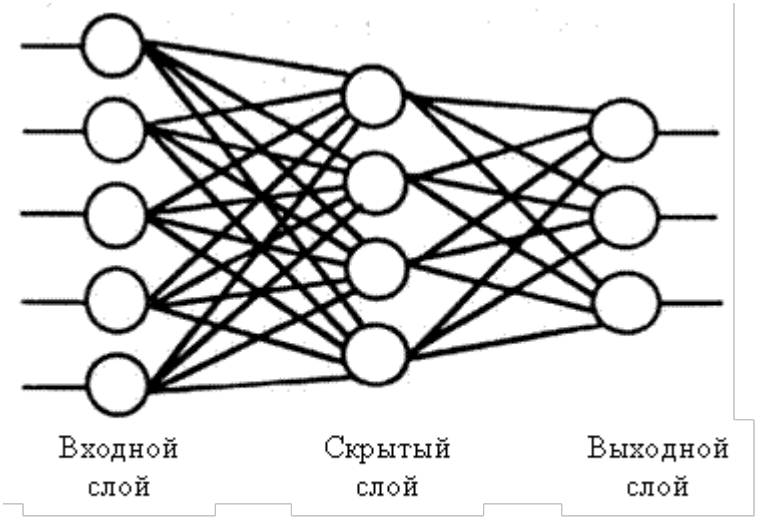

Рис. 6.4. Слои нейронной сети

Входные нейроны получают данные из внешнего мира. Нейроны входного слоя посылают сигналы нейронам внутренних слоев. Внутренними нейронами являются все нейроны между входным и выходным слоями. Мы не можем наблюдать входы и выходы этих нейронов, так как они соединены только с другими нейронами.

Главная часть знаний нейронной сети накапливается во внутренних нейронах и соединениях между ними. Выходные нейроны посылают информацию во внешний мир о том, как нейронная сеть отвечает на входные данные.

Соединения нейронов. Соединение - это линия коммуникации, идущая от посылающего нейрона к получающему. Возможны два типа соединений: возбуждающие и тормозящие. Тормозящие соединения стремятся предотвратить возбуждение нейрона. Возбуждающие соединения стремятся вызвать возбуждение нейрона.

Нейронная сеть может содержать нейрон, оказывающий тормозящее действие на все остальные нейроны того же слоя. Такое действие называют боковым торможением. Иногда боковое торможение столь сильно, что лишь один нейрон в слое, обычно в выходном слое, может быть активирован в каждый момент времени. Этот эффект минимизации числа активных нейронов представляет собой одну из форм конкуренции, ведущую к специализации функций нейронов в слое.

То, как нейроны соединены между собой, оказывает огромное влияние на работу сети. Иногда выходы нейронов одного слоя соединяются со входами нейронов предшествующего или того же слоя. Такие соединения называют обратной связью. Чаще используют модели с обратной связью, в которых каждый нейрон соединен с каждым. На рис. 6.5 показана сеть с обратной связью и специализацией.

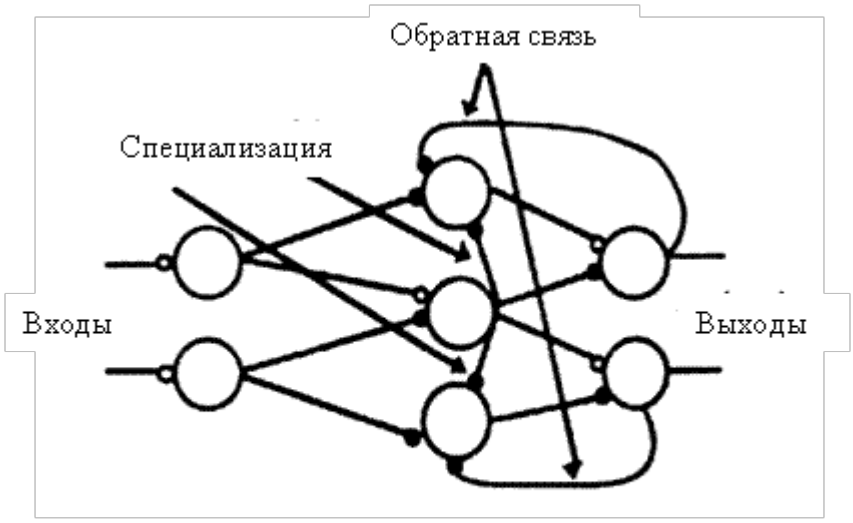

Рис. 6.5. Типы соединений нейронных сетей

Современные искусственные нейронные сети имеют очень мало соединений в сравнении с числом соединений в мозге. Но большинство проблем сегодня могут быть решены искусственными нейронными сетями, содержащими не более, чем 500 нейронов и 30000 соединений. Возможно, что мозг тоже содержит много меньшего размера сети для решения частей проблем и использует сети более высокого уровня, чтобы связывать части решений в целое.

Структуры нейронных сетей. Многообразие задач, решаемых нейронными сетями, вынуждает разрабатывать нейронные сети, имеющие различную структуру и функционирующие по различным алгоритмам.

Теоретически число слоев в нейронной сети и число нейронов в каждом слое может быть любым, но фактически оно ограничено ресурсами компьютера или микросхемы, на которых обычно реализуется нейронная сеть.

Выбор структуры нейронной сети осуществляется с учетом особенностей и сложности задачи. Для решения некоторых типов задач уже найдены хорошо проверенные конфигурации, описанные, например, в [Artificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1992].

Если же задача не может быть решена нейронной сетью ни одной из известных конфигураций, разработчику приходится создавать сеть новой конфигурации, учитывая, что возможности сети возрастают с увеличением числа нейронов, числа слоев и соединений между нейронами сети. Однако следует учитывать, что увеличение сложности сети делает более сложным ее обучение, а введение обратных связей ставит вопрос о динамической устойчивости сети. Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Тем не менее. проблема синтеза нейронной сети столь сильно зависит от решаемой задачи, что дать общие подробные рекомендации затруднительно и оптимальный вариант обычно получают на основе интуиции и испытаний.

Методы обучения нейронных сетей. Качество работы нейронной сети зависит от правильности настройки величин синаптических связей. Поэтому, задавшись структурой нейронной сети, отвечающей решаемой задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые синаптические связи могут быть постоянными). Этот этап называется обучением нейронной сети. На этапе обучения, кроме качества подбора весов соединений, важную роль играет время обучения. Обычно эти два параметра связаны обратной зависимостью и их приходится выбирать на основе компромисса.

Обучение нейронной сети может вестись с учителем или без него. В первом случае сети предъявляются значения как входных, так и соответствующих им выходных сигналов, по которым сеть настраивает веса своих синаптических связей. Во втором случае выходы нейронной сети формируются самостоятельно, а веса изменяются по алгоритму, учитывающему только входные и производные от них сигналы. Процесс обучения состоит в изменении силы сигналов, передаваемых через синапсы.

Существует много различных правил обучения. Наиболее известные из них это правило Хебба, Дельта-правило, и правило обратного распространения ошибки (Back Propagation Rule).

Правило Хебба применяется при обучении без учителя. Правило Хебба усиливает соединения между одновременно возбужденными нейронами, изменяя веса соединений по формуле:

$$\Delta w_{ji} = \alpha \cdot y_j \cdot s_i$$

где Δw_{ji} - изменение веса соединения от нейрона *j* к нейрону *i*, y_j - выход нейрона *j*, s_i - активация нейрона *i*, α - темп обучения. Темп обучения определяет, как велики изменения в сети или как быстро происходит адаптация. Если темп обучения слишком велик, соединения изменяются слишком сильно и сеть перетренируется. В этом случае она может обучаться слишком долго или обучаться плохо.

Дельта-правило применяется при обучении с учителем. Дельта-правило изменяет веса соединений так, чтобы уменьшить различие между реальным выходным образом и желаемым выходным образом. С этой целью изменение весов вычисляется по формуле:

$$\Delta w_{ji} = \alpha \cdot (T_i(t) - y_j(t) \cdot s_i(t)),$$

где Δw_{ji} - изменение веса соединения от нейрона *j* к нейрону *i*, T_i - правильный ответ (выход нейрона *i*), y_j - выход нейрона j, s_i - активация нейрона *i*, α - темп обучения. Дельта-правило было разработано Бернардом Видроу и Тедом Хоффом в Стэнфордском университете в 1960 году. Это лучшее правило для обучения сетей из линейных нейронов. Оно позволяет обучать любым ассоциациям при условии, что все входы линейно независимы. Другие правила обучения, вроде правила Хебба, требуют, чтобы входы были также ортогональны. Правило обратного распространения ошибки является обобщением Дельта-правила.

Модели нейронных сетей

Развитие теории нейронных сетей. Исследования нейронных сетей дали первые интересные результаты, когда Уоррен Маккаллок (Warren McCulloch) и Уолтер Питтс (Walter Pitts) показали, что сеть, построенная из двузначных (binary valued) нейронов способна выполнять простые логические вычисления. В 1949 году Дональд Хебб в своей книге Организация поведения (Organization of Behavior) предложил правдоподобный механизм обучения. Большинство современных правил обучения сетей происходят от правила Хебба или его модификаций. Правило Хебба очень простое: каждый раз, когда два нейрона возбуждаются одновременно, соединение между ними должно быть усилено.

В 1950-х годах доминирующей фигурой в исследованиях нейронных сетей был психолог Фрэнк Розенблатт (Frank Rosenblatt). Он изобрел класс нейронных сетей, названных Перцептроны (Perceptrons). Перцептрон был создан как модель биологической сенсорной системы, в которой использовались несколько слоев пороговых нейронов и вариант правила Хебба для обучения. В 1960-х Марвин Мински (Marvin Minsky) и Сеймур Пейперт (Seymour Papert) показали, что большой класс важных проблем не может быть решен Перцептронами. Полный анализ возможностей Перцептронов они опубликовали в 1969 году в книге Перцептроны.

В конце 1960-х и в начале 1970-х немногие исследователи продолжали работать по нейронным сетям. Наиболее известные из них Стефен Гроссберг (Stephen Grossberg), Джеффри Хайтон (Geoffery Hiton), Тейво Кохонен (Teuvo Kohonen), Кунихико Фукушима (Kunihiko Fukushima), Дж.А. Андерсон (J.A. Anderson). Андерсон и Кохонен (1972) разработали «линейный ассоциатор» (linear associator). Стефен Гроссберг одним из первых проанализировал некоторые свойства конкурентного обучения (1976). Дж.А. Андерсон при участии Джеймса Макклиланда (James McClelland) и Давида Румельхарта (David Rumelhart) работали над идеей параллельно распределенных вычислений.

В 1982 году Джон Хопфилд, профессор химии и биологии в Калифорнийском технологическом институте (California Institute of Technology), опубликовал статью, в которой показал, что нейронные сети способны восстанавливать нечеткий входной образ, и подкрепил это теоретически. Две ключевые идеи присутствовали в этой и последующих его статьях: введение в систему обратных связей и функции глобальной вычислительной энергии, характеризующей состояние системы. Со времени опубликования первой статьи Хопфилда разработано несколько новых моделей нейронных сетей и правил обучения, демонстрирующих такие замечательные способности, как распознавание образов, терпимость к искажениям, предсказание.

Классификация нейронных сетей. Рисунок 6.6 демонстрирует два принципа построения нейронной сети: некоторые сети содержат обратные связи, а некоторые - нет. Если выход каждого нейрона не зависит от выходов последующих нейронов, - это сеть без обратных связей, в такой сети отсутствуют циклы.

Нейронные сети могут быть также классифицированы по нейронным передаточным функциям, обычно на линейные и нелинейные модели. Выход линейного нейрона может быть представлен с использованием линейной алгебры, так как значение выхода прямо зависит от суммы значений его входов. В нелинейной модели выход нейрона является нелинейной функцией суммы его входов и может иметь сложную зависимость от этой суммы. Большинство простых нелинейных моделей представляют собой сети из пороговых нейронов. Значение выхода такого

нейрона равно 1, если взвешенная сумма его входов превышает заданный порог, в противном случае оно равно 0.

<table>
<tr><th colspan="6">НЕЙРОННЫЕ СЕТИ</th></tr>
<tr><th colspan="3">с обратными связями</th><th colspan="3">без обратных связей</th></tr>
<tr><th colspan="2">создаваемые</th><th>тренируемые</th><th>линейные</th><th colspan="2">нелинейные</th></tr>
<tr><th>BAM</th><th>Hopfield</th><td rowspan="2">Adaptive
Resonance</td><td rowspan="2">Adaline
Perceptron
Linear
associator</td><th>без учителя</th><th>с учителем</th></tr>
<tr><td></td><td>C.A.M
T.S.P</td><td>Kohonen
Counterpropag.
Neocognitron
(1980)</td><td>Backpropag.
Neocognitron
(1983)</td></tr>
</table>

Рис. 6.6. Простая классификация НС

Нейронные сети могут быть классифицированы далее по способам обучения на обучение с учителем или без учителя. Сети с учителем сравнивают свой выход с правильным ответом в течение тренировки - подобно тому, как дети учат таблицу умножения. Сети без учителя при обучении используют механизм стимул-реакция - подобно тому, как люди учат первый свой язык.

Нейронные сети с обратной связью. В нейронной сети с обратной связью выходные сигналы нейронов возвращаются на входы нейронов того же или предшествующих слоев (рис.6.7).

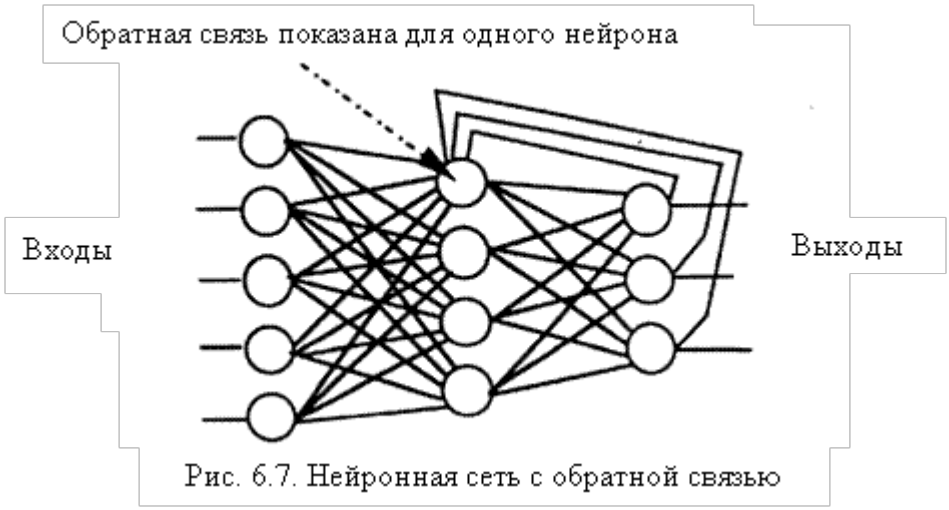

Рис. 6.7. Нейронная сеть с обратной связью

Нейронная сеть с обратной связью (feedback network) не то же самое, что обратное распространение ошибки (back propagation). Обратное распространение ошибки - это не сеть, а метод тренировки сетей без обратных связей. Нейронные сети с обратной связью (feedback networks) не используют обратное распространение ошибки (back propagation) для тренировки.

Модели с обратной связью бывают конструируемые или тренируемые. В конструируемой модели матрица весов создается. Модель Хопфилда и двунаправленная ассоциативная память (Bidirectorial Associative Memories - BAM) являются двумя хорошо известными конструируемыми моделями нейронных сетей с обратной связью. Так как всегда присутствует некоторый элемент случайности, нейронные сети с обратной связью выдают для одних и тех же входов не всегда те же самые, но обычно весьма близкие решения.

В модели Хопфилда (рис.6.8) выход каждого нейрона соединен со входом каждого другого нейрона. Поэтому процесс реакция-стимул-реакция между нейронами происходит до тех пор, пока сеть не "успокоится" в некотором фиксированном состоянии, называемом устойчивым состоянием.

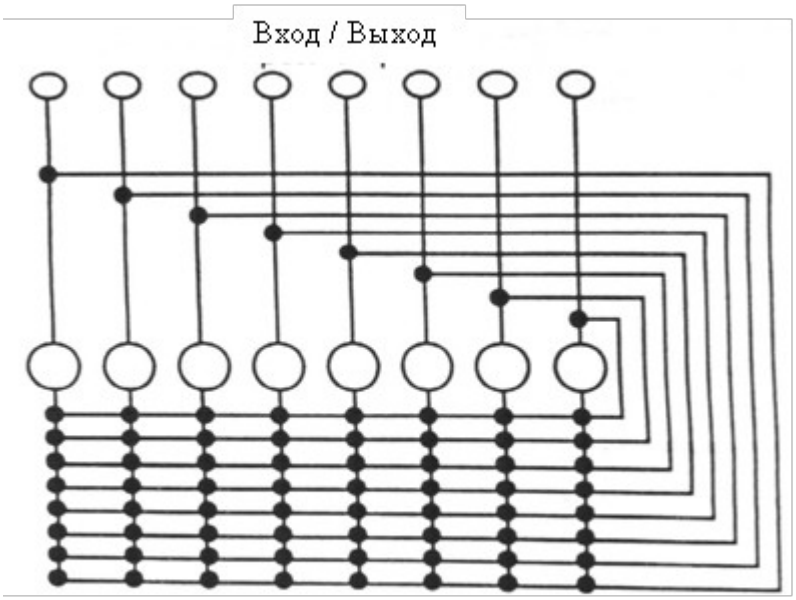

Рис. 6.8. Сеть Хопфилда

Тренировка сети с обратной связью много сложнее, так как адаптация влияет на сигналы как тогда, когда они движутся в прямом направлении, так и тогда, когда они возвращаются на входы предшествующих нейронов. Сложную процедуру адаптивной тренировки нейронных сетей с обратной связью (Adaptive Resonance Theory - ART) разработали Стефен Гроссберг и Гэйл Карпентер (Gail Carpenter).

Вычислительная энергия есть математическая функция, определяющая устойчивые состояния сети и пути, ведущие в них. Если набор значений входов сети линейно упорядочить и представить векотором чисел, то функция вычислительной энергии может быть изображена как кривая поверхность. Устойчивые состояния (минимумы энергии) окажутся во впадинах этой поверхности.

Нейронные сети без обратных связей. Это вторая важная категория нейронных сетей. В нейронных сетях без обратной связи (рис. 6.9) сигналы идут лишь в одном направлении, в них нет циклов.

Ранние модели нейронных сетей были линейными сетями без обратной связи. В 1972 Дж.А. Андерсон и Тейво Кохонен независимо предложили одну и ту же модель для ассоциативной памяти (линейный ассоциатор). Сегодня большинство нейронных сетей используют нелинейные модели нейронных сетей без обратной связи.

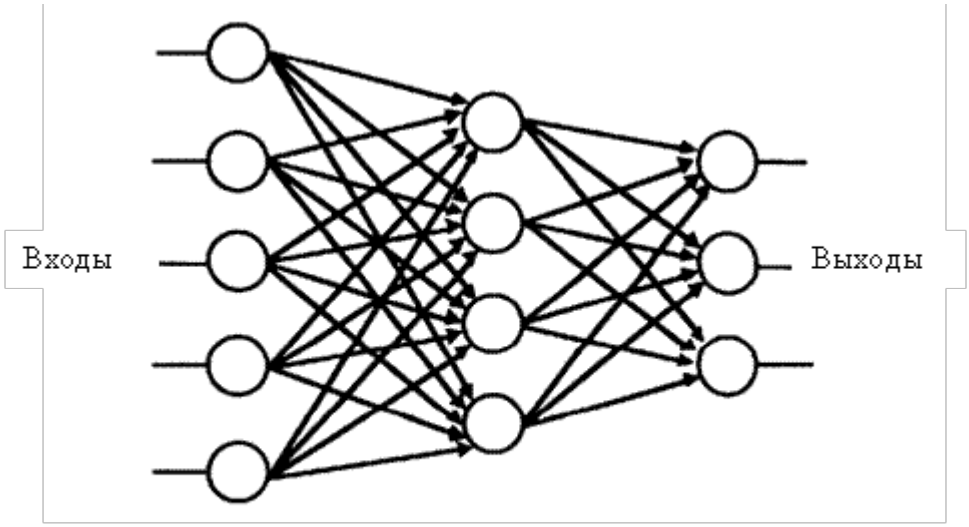

Рис. 6.9. Нейронная сеть без обратных связей

Можно математически строго показать, что любая сеть с обратной связью имеет эквивалентную ей сеть без обратной связи, решающую ту же задачу. Сеть без обратной связи имеет несколько меньший объем памяти, но работает лучше, чем сеть с обратной связью. Работают сети без обратной связи быстрее, поскольку нуждаются лишь в одном проходе сигнала через систему для получения решения. Сети же с обратной связью должны циклически повторять проходы до тех пор, пока не прекратятся изменения выходов нейронов. Обычно это требует от 3 до 1000 циклов.

Методы обучения нейронных сетей

Обучение сетей без обратной связи может быть с учителем и без учителя. При обучении с учителем сеть имеет возможность в ходе тренировки сравнивать с правильными ответами свои реакции на входные воздействия, чтобы корректировать их, в то время как сеть без учителя такой возможности не имеет. Обучение с учителем и без учителя представляют собой исключающие друг друга методы.

При ***обучении с учителем*** сигнал ошибки подается обратно через сеть, изменяя веса соединений в процессе движения таким образом, чтобы та же самая ошибка не случалась снова. Обучение с учителем представляет собой простейшую форму адаптации. Оно требует наличия априорного знания о том, каким должен быть результат: выходные нейроны должны знать, что точно соответствует входным сигналам. Для однослойной сети это легко выполнить путем индивидуального контроля каждого нейрона. В многослойной сети режим обучения с учителем реализовать сложнее, так как трудно корректировать веса внутренних слоев.

Метод обратного распространения ошибки (back propagation), использующий обобщенное Дельта-правило, сегодня наиболее популярный метод обучения многослойных нейронных сетей без обратной связи.

При ***обучении без учителя*** "тренер" для сети не предусмотрен. Сеть сама организует себя так, чтобы прийти к собственной классификации входов. Самоорганизация, демонстрируемая нейронными сетями без учителя, может быть также основана на использовании конкуренции, кооперации или их обоих. В схемах *конкурентного обучения* нейроны распределены по классам. Нейроны каждого класса соединены с каждым нейроном того же класса тормозящими связями, а с нейронами других классов возбуждающими связями. Нейроны каждого класса соревнуются между собой за право распознавать некоторую особенность во входе. Нейрон, который более чутко реагирует на эту особенность, чем другие нейроны в классе, "выигрывает" в конкуренции. В конечном счете различные классы приобретают способность представлять различные аспекты входа.

Часто конкуренция и кооперация бывают представлены в одной и той же сети. Кооперация характерна для сетей, которые могут достигать глобальной организации путем локальных взаимодействий. Необходимы параллельные, рекурсивные и нелинейные взаимодействия, чтобы обеспечить кооперацию. Например, в 1976 Д. Марр и Т. Поггио (D. Marr and T. Poggio) применили кооперативную сеть для решения проблемы совмещения образов, как это происходит у человека с образами, возникающими в левом и в правом глазах.

Обратное распространение ошибки *(Back Propagation)* представляет собой схему обучения с учителем и позволяет тренировать многослойные сети без обратной связи. Сеть обучается путем корректировки весов соединений с учетом ошибки на выходе сети. Цель тренировки сети состоит в достижении минимума ошибки *E* за счет надлежащей настройки весов соединений.

Ошибку реакции сети (на выходе сети) на данный образ можно вычислить по формуле:

$$E = \frac{1}{2}\sum_i (t_i - y_i)^2$$

, где

t_i - правильный выход нейрона i;

y_i - реальный (действительный) выход нейрона i.

Простейший метод для поиска минимума ошибки *E* известен как метод градиентного (наискорейшего) спуска. Он предусматривает всегда движение (небольшими шагами) в направлении наибольшего уменьшения ошибки.

Если изменение веса w_{ji} обозначить как Δw_{ji}, то по методу градиентного спуска это изменение должно быть равно:

$$\Delta w_{ji} = \alpha \cdot (-\partial E_i / \partial w_{ji}),$$

где α - константа, задающая темп обучения (величину шага спуска).

Таким образом, для определения величины изменения весов требуется найти производную $\partial E_i / \partial w_{ji}$. Согласно известным правилам дифференцирования мы имеем:

$$\partial E_i / \partial w_{ji} = (\partial E_i / \partial y_i) \cdot (\partial y_i / \partial s_i) \cdot (\partial s_i / \partial w_{ji}).$$

Тогда, с учетом вида формул для E_i, y_i, s_i, получим:

$$\partial E_i / \partial y_i = \frac{-2 \cdot (t_i - y_i)}{2} = -(t_i - y_i);$$

$$\partial y_i / \partial s_i = \partial f(s_i) / \partial s_i = f'(s_i);$$

$$\partial s_i / \partial w_{ji} = \partial(\sum_i x_j w_{ji}) / \partial w_{ji} = x_j.$$

Таким образом, производная $\partial E_i / \partial w_{ji}$ имеет следующий вид:

$$\partial E_i / \partial w_{ji} = -(t_i - y_i) \cdot f'(s_i) \cdot x_j.$$

Обозначим $\delta_i = (t_i - y_i) \cdot f'(s_i)$. Величина δ_i называется локальной ошибкой. Тогда правило изменения весов соединений нейронов выходного слоя сети согласно методу градиентного спуска приобретает вид:

$$\Delta w_{ji} = \alpha \cdot \delta_i \cdot x_j.$$

Таким образом, если нейрон *i* находится в выходном слое, веса его соединений с нейронами *j* предыдущего слоя мы можем вычислить немедленно.

Если нейрон *i* находится не в выходном слое, локальная ошибка для этого нейрона вычисляется через сумму взвешенных локальных ошибок δ_k нейронов k следующего слоя:

$$\delta_i = (\sum_k \delta_k w_{ik}) \cdot f'(s_i).$$

Этот прием позволяет, вычислив локальные ошибки нейронов выходного слоя, вычислить затем локальные ошибки нейронов предшествующего слоя и так повторять до тех пор, пока не будут вычислены локальные ошибки нейронов входного слоя. При этом правило корректировки весов соединений применяется одно и то же.

Зейновски (Sejnowski) и Розенберг (Rosenberg) в их NetTalk использовали правило, в котором, кроме параметра темп тренировки α, используется параметр μ, известный как "сглаживающий фактор". $\Delta w^n_{ji} = \alpha \cdot ((1-\mu) \cdot \delta_i \cdot x_j + \mu \cdot \Delta w^{n-1}_{ji})$. Использование "фактора сглаживания" улучшает сходимость процесса тренировки. Если μ=0, алгоритм тоже сходится, но делает это несколько дольше.

Проектирование нейронных сетей

Чтобы создать нейронную сеть, не требуется понимать её внутреннюю работу, надо лишь знать ответы на следующие основные вопросы: что вы хотите узнать от нейронной сети, какую информацию следует предоставить нейронной сети, как тренировать нейронную сеть, что вы будете делать с тренированной нейронной сетью?

Процесс проектирования. Чтобы создать нейронную сеть, надо точно решить, что она должна делать: предсказывать, обобщать или распознавать. Требуется также выбрать информацию, на основе которой нейронная сеть будет это делать. Нейронная сеть обучается путем создания ассоциаций между входами и выходами. Поэтому думать следует только о том, какие входные данные нейронная сеть может использовать для создания нужной ассоциации с желаемым выходом.

Другая важная часть процесса проектирования - собирание примеров для каждого из известных правильных ответов. Примеры должны быть организованы как факты. Факт - это набор входов в паре с правильным выходом (выходами). Факт можно мыслить как карточку, одна сторона которой содержит входную информацию, а другая известный правильный ответ, который нейронная сеть должна заучить в ходе тренировки. Данных может оказаться слишком много, хотя это бывает редко. Хорошее правило состоит в том, что число фактов не должно превышать более чем в десять раз число соединений. Случайные факты должны быть исключены из тренировочного набора.

Данные могут представлять собой символы, картины или числа. Однако нейронные сети могут понимать только числа. Поэтому данные следует преобразовать в числовые значения и нормализовать применительно к используемой передаточной функции, обычно от 0 до 1 или от -1 до +1.

Построение сети. Прежде всего следует уточнить число входных и выходных нейронов. Число входных и выходных нейронов равно, соответственно, числу входных предметов и числу правильных ответов. Далее следует уточнить число внутренних слоев и внутренних нейронов создаваемой сети. Не существует формул для выбора числа внутренних нейронов. Одно из правил рекомендует использовать среднее от числа входных и выходных нейронов. Другое правило предлагает использовать меньшее из этих двух чисел. Третье правило предлагает тренировать с малым числом внутренних нейронов и добавлять внутренние нейроны после завершения тренировки. Опасность от наличия лишних нейронов в том, что сеть может запомнить тренирующие факты, а не обобщить их.

Построение сети может включать также уточнение типа передаточной функции и диапазона ее значений, темпа обучения и сглаживающего фактора. Темп обучения определяет величину коррекции при неверном выходе сети в процессе тренировки. Сглаживающий фактор определяет, как влияет предыдущая коррекция на новую коррекцию. Большинство пакетов коммерческого назначения позволяют определять эти параметры.

Тренировка сети включает повторяющееся предъявление нейронной сети набора фактов. Нейронная сеть берет каждый вход, вычисляет выход, проверяет, насколько этот выход соответствует факту, и делает коррекцию весов соединений так, чтобы выход стал правильным. Этот процесс повторяется для каждого факта по очереди до тех пор, пока обученность сети станет достаточной. Для оценки достигнутого качества выхода сети для каждого из тренирующих фактов, нужен критерий. Большинство пакетов коммерческого назначения позволяют определять такой критерий. Тренировать сеть до тех пор, пока каждый выход не станет безупречным, не требуется. Напротив, обычно сеть тренируют до тех пор, пока не будет достигнута необходимая точность (обычно 90%).

Сеть может распознать факт правильно в одной итерации тренировки и ошибочно в другой. Когда сеть совершает ошибку в ходе тренировки, она вносит компенсирующие ошибку изменения. По этой причине тренировка должна быть итеративным процессом, в котором факты последовательно предъявляются сети снова и снова до тех пор, пока все факты станут восприниматься сетью правильно с необходимой точностью. Иногда в ходе тренировки сеть достигает наилучшего обобщения, когда еще не все факты воспринимаются корректно. Продолжение тренировки в этом случае непродуктивно. Поэтому сеть полезно тестировать в процессе тренировки, чтобы найти лучшую по тесту сеть. Полезно также проверять факты, которые сеть находит трудными для обучения. Эти факты могут противоречить другим фактам набора или не иметь отношения к проблеме.

Тестирование сети по сути то же самое, что и тренировка, за исключением того, что сети предъявляются факты, которые она не видела до этого, и не делается коррекция, когда сеть совершает ошибку. Если результат тестирования окажется хорош, сеть можно использовать. В противном случае нужно больше данных или лучшие данные, чтобы сеть перепроектировать.

Применение сети состоит в предъявлении ей рабочих входных данных и сборе полезных результатов. Применение тренированной сети происходит значительно быстрее, чем тренировка, так как выход не оценивается и коррекция не делается. Тренированную нейронную сеть можно вызвать в другую программу или в экспертную систему, можно также для увеличения скорости работы реализовать её в виде чипа. Тренированная сеть (матрица весов соединений) рассматривается как интеллектуальное свойство и может быть защищена от копирования.

Что могут и что не могут делать нейронные сети. Нейронные сети больше известны способностью к распознаванию образов. Если нужно что-либо распознавать или классифицировать, нейронная сеть это сделает быстрее и правильнее, чем человек. Нейронные сети непригодны там, где требуется точность. Если точность не так важна, как скорость, нейронная сеть может служить подходящим инструментом. Если известны вещи, вовлеченные в процесс, но нет формального описания процесса, нейронная сеть может найти отношение между этими вещами. Если анализируемый процесс изменится, достаточно лишь собрать новые факты и выполнить тренировку сети.

Рекомендуемая литература:

1 Балдин К.В., Уткин В.Б, Информатика. Учебник для вузов.- М.: Проект, 2003. С.304
2 Романов А. Информационные интеллектуальные системы в экономике: учебник для ВУЗов. – М., Высшая школа, 2005. –511с.:ил.
3 Экспертные системы: принципы работы и примеры. //под ред. Р. Форсайта. –М.: Радио и связь, 1987. –223 с.
4 Д. Поспелов . "Справочник по ИИ том-2".
5 Люгер, Джордж, Ф. Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е издание. : Пер. с англ. - М.: Издательский дом "Вильяме", 2003. - 864 с.-С. 777-840.
6 Болотова Л.С., Комаров М.А., Смольянинов А.А. Системы искусственного интеллекта. Теоретические основы СИИ и формальные модели представления знаний: Учеб. пособие - М.: МИРЭА, 1998. - 108 с.

7 Искуственный интеллект. Кн.1 Системы общения и экспертные системы/ Под ред. Э.В.Попова. – М.: Радио и связь,2000