

Министерство образования и науки Российской Федерации
Южно-Уральский государственный университет
Кафедра информатики

004.8(07)
П512

Г.А. Поллак

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Учебное пособие

Челябинск
Издательский центр ЮУрГУ
2011

УДК 004.8(075.8)
П512

Одобрено
учебно-методической комиссией
факультета экономики и управления

Рецензенты: д.п.н. Е.А. Мартынова, к.т.н. И.А. Прохорова

Поллак, Г.А.

П512 Интеллектуальные информационные системы: учебное пособие / Г.А. Поллак – Челябинск: Издательский центр ЮУрГУ, 2011. – 141 с.

Основано на материалах лекций по дисциплине «Интеллектуальные информационные системы».

Рассмотрены особенности интеллектуальных информационных систем, структура и классификация экспертных систем, основные модели представления знаний, а также методы поиска решений.

Описаны базовые методы приобретения знаний, а также технология машинного обучения на примере нейронных сетей.

Каждый раздел снабжен выводами, вопросами для самоконтроля, имеется глоссарий.

Для студентов учреждений высшего профессионального образования всех форм обучения специальности «Информационные системы (экономика)».

УДК 004.8(075.8)

© Издательский центр ЮУрГУ, 2011

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
1. ОСОБЕННОСТИ И ПРИЗНАКИ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ	
1.1. Определение интеллектуальных информационных систем	7
1.2. Классификация интеллектуальных информационных систем.....	9
1.3. Связь интеллектуальных информационных систем с искусственным интеллектом	14
1.4. Выводы.....	16
1.5. Вопросы для самоконтроля.....	16
2. СИСТЕМЫ, ОСНОВАННЫЕ НА ЗНАНИЯХ	
2.1. Определение экспертных систем	17
2.2. Классы экспертных систем	21
2.3. Технология разработки экспертных систем.....	24
2.4. Выводы.....	30
2.5. Вопросы для самоконтроля.....	31
3. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ	
3.1. Знания и их свойства	32
3.2. Классификация моделей представления знаний	35
3.3. Логическая модель представления знаний.....	36
3.4. Продукционная модель представления знаний	41
3.5. Семантические сети	44
3.6. Фреймовые модели представления знаний	48
3.7. Выводы.....	55
3.8. Вопросы для самоконтроля.....	56
4. МЕТОДЫ ПОИСКА РЕШЕНИЯ	
4.1. Логический вывод в продукционных системах.....	57
4.1.1. Поиск решения в одном пространстве	58
4.1.2. Стратегии неинформированного поиска.....	61
4.1.3. Стратегии эвристического поиска	65
4.1.4. Поиск решения в иерархии пространств	68
4.2. Вывод в семантических сетях.....	70
4.3. Поиск решения во фреймовых системах.....	71
4.4. Выводы.....	74
4.5. Вопросы для самоконтроля.....	75

5. ОСОБЕННОСТИ РЕШЕНИЯ ЗАДАЧ В ПРОДУКЦИОННЫХ СИСТЕМАХ

5.1. Конфигурация системы продукций и цикл работы интерпретатора....	76
5.2. Пример решения задачи на основе цели	78
5.3. Системы с доской объявлений	81
5.4. Экспертные системы, основанные на прецедентах.....	83
5.5. Динамические экспертные системы	84
5.6. Выводы.....	86
5.7. Вопросы для самоконтроля.....	87

6. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА НЕОПРЕДЕЛЕННОСТИ В ПРОДУКЦИОННЫХ СИСТЕМАХ

6.1. Источники неопределенности	88
6.2. Неточный вывод на основе фактора уверенности.....	89
6.3. Рассуждения, основанные на нечетких множествах	92
6.4. Байесовские рассуждения	96
6.5. Байесовские сети доверия	99
6.6. Диаграммы влияния.....	101
6.7. Выводы.....	103
6.8. Вопросы для самопроверки	103

7. НЕЙРОННЫЕ СЕТИ..... 105

7.1. Формальная модель нейрона	106
7.2. Типы функций активации	108
7.3. Представление нейронных сетей с помощью графов	110
7.4. Архитектура нейронных сетей	113
7.5. Обучение нейронной сети.....	115
7.5.1. Парадигма обучения с учителем	115
7.5.2. Парадигмы обучения без учителя	116
7.5.3. Алгоритмы обучения.....	117
7.6. Выводы.....	123
7.7. Вопросы для самоконтроля.....	124

ГЛОССАРИЙ..... 126

БИБЛИОГРАФИЧЕСКИЙ СПИСОК..... 140

ВВЕДЕНИЕ

Создание компьютерной техники стало не только шагом на пути повышения эффективности вычислений, но и вызвало к жизни новые технологии обработки информации в различных областях.

Интеллектуальные информационные системы (ИИС) являются естественным результатом развития обычных информационных систем. Системы применяются для решения неструктурированных или слабоструктурированных задач, опираясь на заложенные в них знания предметной области. Таким образом, применение интеллектуальных систем позволяет автоматизировать не только процессы подготовки информации для принятия решений, но и сложные процессы выработки решений.

Целью изучения курса «Интеллектуальные информационные системы» является формирование у студентов теоретической и практической подготовки в области разработки и внедрения профессионально-ориентированных информационных систем с учетом современных и перспективных технологий и методов.

Материал данного пособия основан на курсе лекций, прочитанных автором для студентов, обучающихся по специальности «Прикладная информатика (экономика)».

Пособие состоит из семи глав.

В *первой главе* определены основные понятия ИИС, рассмотрена классификация ИИС, а также виды задач, решаемых ИИС.

Во *второй главе* дано общее определение систем, основанных на знаниях, приведена структура статических и динамических экспертных систем (ЭС), приведена их классификация. В этой же главе рассмотрены вопросы проектирования экспертных систем.

Третья глава полностью посвящена вопросам представления знаний в интеллектуальных системах. Рассмотрены вопросы отличия знаний от данных, классификация знаний, достаточно подробно рассмотрены основные модели представления знаний: продукционные правила, семантические сети, фреймовое представление знаний и модели, основанные на исчислении предикатов первого порядка.

В *главе четыре* описаны методы и стратегии поиска решений в системах, использующих различные модели представления знаний.

Большинство экспертных систем используют продукционную модель знаний. Поэтому в *пятой главе* обсуждаются особенности работы интерпретатора продукционной системы, а также кратко охарактеризованы системы с доской объявлений и системы, основанные на прецедентах. Дополнительно на примере

системы G2 рассматриваются требования, предъявляемые к экспертным системам динамического типа.

Для получения решения в системах, основанных на нечетких и неполных знаниях, применяются различные методы. В *шестой главе* описываются некоторые известные методы нечеткого вывода: байесовские вероятностное рассуждение и его расширения, теория уверенности, нечеткая логика.

И, наконец, *седьмая глава* посвящена нейронным сетям. Кратко описана формальная модель нейрона, архитектура и алгоритмы обучения нейронных сетей.

Каждая глава пособия содержит выводы, а также вопросы для самопроверки. Описание многих теоретических вопросов снабжено поясняющими примерами.

1. ОСОБЕННОСТИ И ПРИЗНАКИ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1. Определение интеллектуальных информационных систем

Информационная система – это набор информационных технологий, направленных на поддержку жизненного цикла информации и включающего три основных процесса: обработку данных, управление информацией и управление знаниями.

Одним из методов формализации знаний является алгоритмический или процедурный подход. Необходимым условием решения задач в рамках процедурного подхода является наличие четкого алгоритма, что возможно только для формализованных задач, поэтому знания выражаются в виде жесткой последовательности действий, которые оформляются в виде компьютерной программы.

Общие недостатки традиционных информационных систем заключаются в слабой адаптации к изменениям в предметной области и информационным потребностям пользователей.

Существует класс задач, называемых неформализованными. Их особенности [11]:

- алгоритм решения задачи неизвестен или его нельзя использовать из-за ограниченности ресурсов компьютера;
- задачу нельзя определить в числовой форме;
- цели задачи нельзя выразить в терминах точно определенной целевой функции.

Попытки решения неформализованных задач, а также попытки устранить недостатки процедурного подхода привели к формированию *инженерии знаний*. Системы, основанные на знаниях, называются *интеллектуальными информационными системами*.

Согласно определению Д.А. Поспелова [12], интеллектуальная система может:

1. Накапливать знания об окружающем мире, инициировать процессы получения новых знаний.
2. Пополнять знания с помощью логического вывода, отражающего закономерности в окружающем мире, получать обобщенные знания на основе более частных знаний и логически планировать свою деятельность.
3. Общаться с человеком на языке, максимально приближенном к естественному человеческому языку.
4. Формировать объяснение собственной деятельности.

5. Оказывать пользователю помощь за счет тех знаний, которые хранятся в памяти, и тех логических средств рассуждений, которые присущи системе.

Интеллектуальные информационные системы решают так называемые интеллектуальные задачи. Каковы их признаки? Принято считать, что интеллектуальной задачей является задача отыскания ранее неизвестного алгоритма решения практической или теоретической проблемы, универсального на множестве свойств этой проблеме исходных данных. После того, как алгоритм найден, процесс решения задач можно поручить любому исполнителю (человеку или автомату). Задачи, связанные с отысканием алгоритма решения других задач, будем называть *интеллектуальными*.

Существует множество определений интеллектуальных информационных систем. Мы будем использовать следующее определение [15].

Определение

Интеллектуальная информационная система (ИИС) – это компьютерная модель интеллектуальных возможностей человека в целенаправленном поиске, анализе и синтезе текущей информации об окружающей действительности для получения о ней новых знаний и решения на этой основе различных жизненно важных задач.

Приведем обобщенный перечень классов задач, которые уже сейчас решаются с помощью интеллектуальных информационных систем:

- разработка средств, обеспечивающих эффективное общение человека с машиной (автоматом) на естественном языке в форме речевого обмена и (или) обмена текстами;
- автоматический перевод текстовой и речевой информации с одних языков на другие, в том числе – с естественных языков на внутримашинные и обратно;
- автоматический концептуальный анализ, поиск и интерпретация данных и знаний;
- разработка алгоритмов и методов поддержки принятия решений по целесообразному управлению (в том числе – в реальном масштабе времени) различными системами, объектами и процессами в различных отраслях хозяйства с учетом неопределенности в реализации факторов внешней среды и связанных с этим рисков;
- разработка алгоритмов и методов мониторинга и диагностики состояния систем, объектов и процессов;
- автоматическое проектирование систем и устройств с оптимальными на множестве ситуаций свойствами;
- разработка алгоритмов логических выводов и доказательства теорем;
- разработка поведенческих алгоритмов в условиях неопределенности и риска;
- автоматическое распознавание образов различной природы;
- создание автоматически обучающихся систем.

1.2. Классификация интеллектуальных информационных систем

Интеллектуальные информационные системы (ИИС) характеризуются наличием следующих признаков.

- Имеют развитые коммуникативные способности.
- Умеют решать сложные плохо формализуемые задачи.
- Способны к самообучению.
- Обладают адаптивностью.

Коммуникативные способности ИИС характеризуют способ взаимодействия (интерфейс) конечного пользователя с системой, в частности, возможность формулирования произвольного запроса в диалоге с ИИС на языке, максимально приближенного к естественному языку.

Сложные плохо формализуемые задачи требуют построения оригинального алгоритма решения в зависимости от конкретной ситуации, для которой могут быть характерны неопределенность и динамичность исходных знаний и данных.

Способность к обучению проявляется в возможности автоматического извлечения знаний для решения задач из накопленного опыта конкретных ситуаций.

Адаптивность – это способность к развитию системы в соответствии с объективными изменениями модели предметной области.

В различных ИИС перечисленные признаки интеллектуальности развиты в неодинаковой степени и редко, когда все четыре признака реализуются одновременно. Условно каждому признаку интеллектуальности соответствует свой класс ИИС (рис. 1.1):

- Системы с интеллектуальным интерфейсом.
- Экспертные системы.
- Самообучающиеся системы.
- Адаптивные системы.

Мы приведем краткую характеристику основных типов интеллектуальных информационных систем.

Системы с интеллектуальным интерфейсом

Естественно-языковый интерфейс используется для:

- доступа к интеллектуальным базам данных;
- контекстного поиска документальной текстовой информации;
- голосового ввода команд в системах управления;
- машинного перевода с иностранных языков.

Гипертекстовые системы предназначены для реализации поиска по ключевым словам в базах текстовой информации. Интеллектуальные гипертекстовые системы отличаются возможностью сложной семантической организации ключевых слов, которая отражает различные смысловые отношения терминов. Возможен поиск мультимедийной информации, включающей помимо текстовой и цифровой информации графические, аудио и видео-образы.

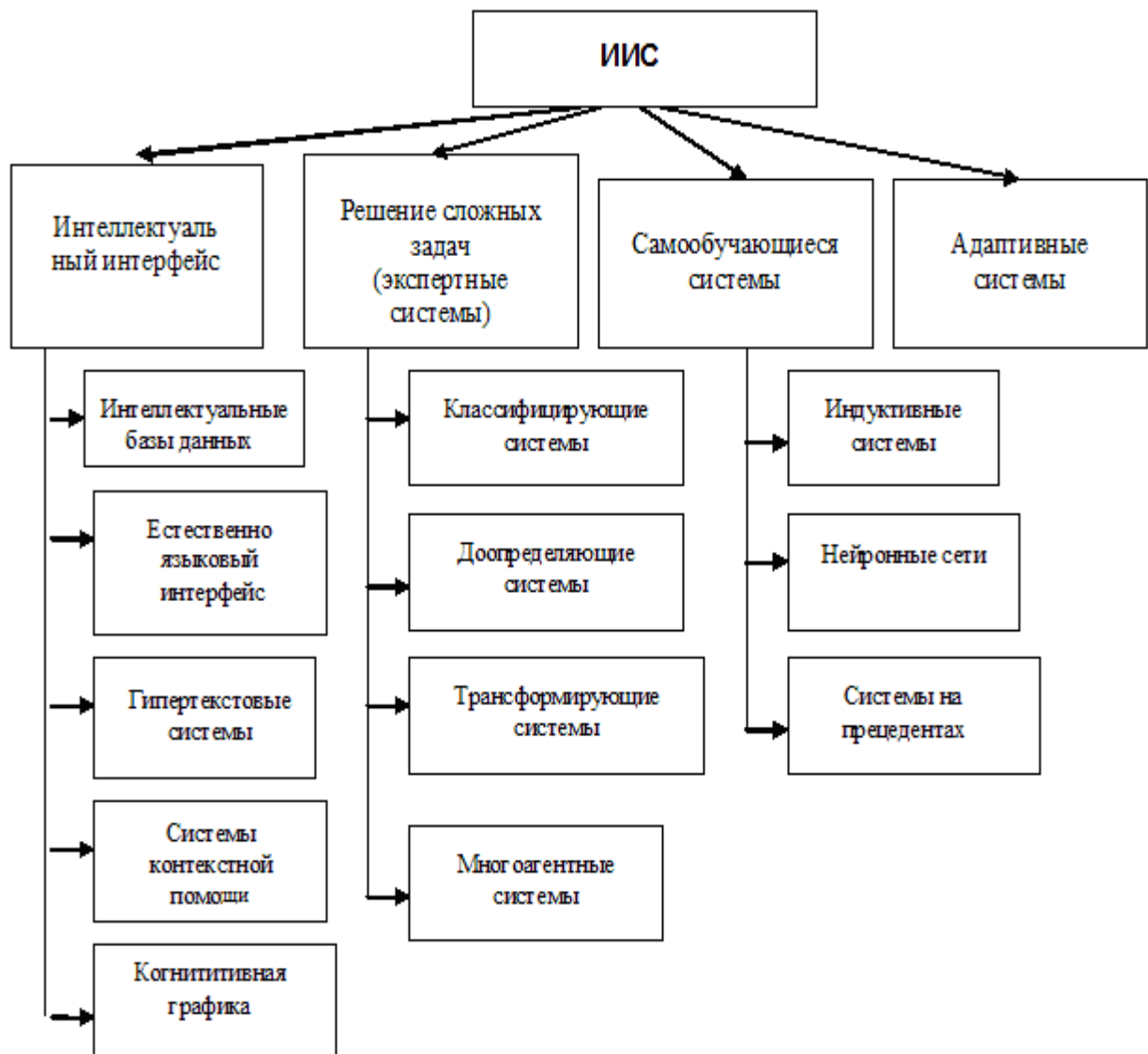


Рис. 1.1. Классификация интеллектуальных систем

Системы контекстной помощи можно рассматривать как частный случай интеллектуальных гипертекстовых и естественно-языковых систем. В системах контекстной помощи пользователь описывает проблему, а система с помощью дополнительного диалога ее конкретизирует и сама выполняет поиск относящихся к ситуации рекомендаций. Такие системы относятся к классу систем распространения знаний (Knowledge Publishing) и создаются как приложение к системам документации.

Системы когнитивной графики позволяют осуществлять интерфейс пользователя с ИИС с помощью графических образов, которые генерируются в соответствии с происходящими событиями. Такие системы используются в мониторинге и управлении оперативными процессами. Они также нашли широкое применение в обучающих и тренажерных системах, где графические

образы моделируют ситуации, в которых необходимо принимать решения и выполнять определенные действия.

Экспертные системы (ЭС)

Назначение экспертных систем состоит в решении задач на основе накапливаемой базы знаний, отражающей опыт работы экспертов в некоторой проблемной области. Достоинство применения экспертных систем заключается в возможности принятия решений в уникальных ситуациях, для которых алгоритм заранее неизвестен и формируется по исходным данным в виде цепочки рассуждений (правил принятия решений) из базы знаний. Решение предполагается осуществлять в условиях неполноты, недостоверности, многозначности исходной информации и качественных оценок процессов.

Самообучающиеся системы

В основе самообучающихся систем лежат методы автоматической классификации примеров ситуаций реальной практики (обучения на примерах). В результате обучения системы автоматически формируются знания, используемые при решении задач классификации и прогнозирования.

Существуют несколько классов самообучающихся систем: индуктивные системы, дедуктивные системы, нейронные сети, системы, основанные на прецедентах. В нашем курсе мы подробно изучим нейронные сети, а сейчас дадим краткую характеристику каждого представителя этих систем.

Индуктивные системы. При индуктивной обработке информации на основании практического опыта и потоков данных создаются общие шаблоны и правила. Обобщение примеров происходит по принципу от частного к общему и сводится к выявлению подмножеств примеров, относящихся к одним и тем же подклассам, и определению для них значимых признаков.

Процесс классификации примеров осуществляется следующим образом.

1. Выбирается признак классификации.
2. По значению выбранного признака множество примеров разбивается на подмножества.
3. Выполняется проверка, принадлежит ли каждое образовавшееся подмножество примеров одному классу.
4. Если какое-то подмножество примеров принадлежит одному подклассу, то процесс классификации заканчивается.
5. Для подмножеств примеров с несовпадающим значением классобразующего признака процесс классификации продолжается, начиная с пункта 1.

Процесс классификации представляется в виде дерева решений. В нем промежуточные узлы содержат значения признаков последовательной классификации, а в конечных узлах – значения признака принадлежности определенному классу. Пример построения дерева решений на основе фрагмента таблицы примеров (табл. 1.1) показан на рис.1.2.

Анализ новой ситуации сводится к выбору ветви дерева, которая полностью определяет ситуацию. Поиск решения осуществляется в результате последовательной проверки признаков классификации.

Таблица 1.1

Классообразующий признак	Признаки классификации			
Цена	Спрос	Конкуренция	Издержки	Качество
Низкая	Низкий	Невысокая	Низкие	Низкое
Высокая	Низкий	Невысокая	Большие	Высокое
Высокая	Высокий	Невысокая	Большие	Низкое
Высокая	Высокий	Невысокая	Низкие	Высокое
Высокая	Высокий	Невысокая	Низкие	Низкое
Высокая	Высокий	Невысокая	Большие	Высокое

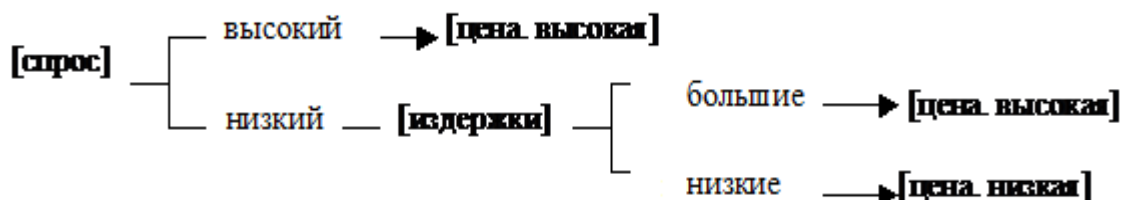


Рис. 1.2. Фрагмент дерева решений

Каждая ветвь дерева соответствует одному правилу решения, например, такому:

Если Спрос = «низкий» и Издержки = «низкие»
 То Цена = «низкая»

При дедуктивной обработке информации для определения конкретных фактов используются общие правила. Обучение на основе подоби́я представляет собой индуктивный процесс, а доказательство теорем – дедуктивный, поскольку опирается на известные аксиомы и уже доказанные теоремы.

Нейронные сети являются технологией обработки информации, вдохновленной изучением мозга и нервной системы человека.

Нейронные сети позволяют решать задачи прогнозирования, классификации, поиска оптимальных вариантов, и совершенно незаменимы в тех случаях, когда в обычных условиях решение задачи основано на интуиции или опыте, а не на строгом (в математическом смысле) ее описании

Более подробно о нейронных сетях мы поговорим позднее.

Базы знаний систем, основанных на прецедентах (Case-based reasoning) содержат описания разнообразных ситуаций. Поиск решения проблемы сводится к поиску по аналогии (абдуктивному выводу от частного к частному) подходящей ситуации и включает следующие шаги.

1. Получение подробной информации о текущей проблеме.
2. Сопоставление полученной информации со значением признаков прецедентов из базы знаний.

3. Выбор прецедента из базы знаний, наиболее близкого к рассматриваемой проблеме.

4. В случае необходимости выполняется адаптация выбранного прецедента к текущей проблеме.

5. Проверка корректности каждого полученного решения.

6. Занесение информации о полученном решении в базу знаний.

В отличие от индуктивных систем допускается нечеткий поиск с получением множества допустимых альтернатив, каждая из которых оценивается некоторым коэффициентом уверенности. Обучение системы сводится к запоминанию каждой новой обработанной ситуации с принятыми решениями в базе прецедентов.

Эти системы применяются как системы распространения знаний с расширенными возможностями или как системы контекстной помощи.

Адаптивные системы

Ядром *адаптивных систем* является постоянно развиваемая модель проблемной области, поддерживаемая в специальной базе знаний – репозитории, на основе которого осуществляется генерация или конфигурация программного обеспечения. Таким образом, проектирование и адаптация системы сводится, прежде всего, к построению модели проблемной области и ее своевременной корректировке.

При проектировании адаптивной системы обычно используется два подхода: **оригинальное** или типовое **проектирование**. *Первый* подход предполагает разработку ИИС с «чистого листа» в соответствии с требованиями экономического объекта. Он реализуется с использованием систем автоматизированного проектирования информационных систем или CASE-технологий. Согласно этой технологии каждый раз при изменении проблемной области выполняется генерация программного обеспечения. При *втором* подходе выполняется адаптация типовых разработок к особенностям экономического объекта. Он предполагает использование систем компонентного (сборочного) проектирования.

Оба подхода ориентируются на тщательное изучение экономического объекта и его моделирование [15]. Отличие между ними заключается в следующем: при использовании CASE-технологии на основе репозитория при возникновении изменения выполняется генерация программного обеспечения; при использовании сборочной технологии – конфигурация программного обеспечения и только в редких случаях выполняется их переработка с использованием CASE-средств.

1.3. Связь интеллектуальных информационных систем с искусственным интеллектом

Итак, мы кратко охарактеризовали различные классы интеллектуальных информационных систем. Эти системы строятся на принципах, разработанных в области искусственного интеллекта (ИИ).

Определение

ИИ – научная дисциплина, основной задачей которой является разработка парадигм или алгоритмов, обеспечивающих компьютерное решение когнитивных задач, свойственных человеческому мозгу.

Следует заметить, что это определение искусственного интеллекта не является единственно возможным.

С самого начала исследований в области моделирования процесса мышления (конец 40-х годов) выделились два до недавнего времени практически независимых направления: логическое и нейрокибернетическое.

Первое (традиционное) направление связано с моделированием мыслительной деятельности человека при решении конкретных задач в определенной проблемной области (экспертные системы; системы, основанные на знаниях). Исследования, использующие это направление, базируются на двух гипотезах: гипотезе символических систем и гипотезе поиска [11].

Первая гипотеза в упрощенном виде утверждает, что знания можно представить с помощью символьных структур. Согласно второй гипотезе символические системы решают задачи с помощью поиска, т.е. они генерируют потенциальные решения и постепенно их модифицируют, пока решения не будут удовлетворять заданным условиям.

В дальнейшем это направление вылилось в направление информатики «инженерия знаний», занимающееся созданием так называемых «систем, основанных на знаниях» (Knowledge Based Systems).

Таким образом, первое направление рассматривает результат интеллектуальной деятельности человека, изучает его структуру (выделяя различные проявления интеллектуальной деятельности — решение задач, доказательство теорем, игры) и стремится воспроизвести этот продукт средствами ЭВМ. Успехи этого направления ИИ тесно связаны с развитием ЭВМ и искусством программирования. Это направление ИИ часто называют *машинным интеллектом*.

Второе направление – нейрокибернетическое – основано на построении самоорганизующихся систем, состоящих из множества элементов, функционально подобных нейронам головного мозга. Это направление началось с концепции формального нейрона Мак-Каллока–Питтса и исследований Розенблатта [17] с различными моделями персептрона – системы, обучающейся распознаванию образов. Развитие этого направления, называемого *искусственным разумом*, тесно связано с успехами наук о человеке. Характерным в данном случае является стремление к воспроизведению более

широкого, чем в машинном интеллекте, спектра проявлений разумной деятельности человека.

Оба основных направления ИИ связаны с моделированием: в первом случае – с имитационным моделированием, а во втором – со структурным. В упрощенном виде структура основных направлений ИИ, изображена на рис. 1.3.

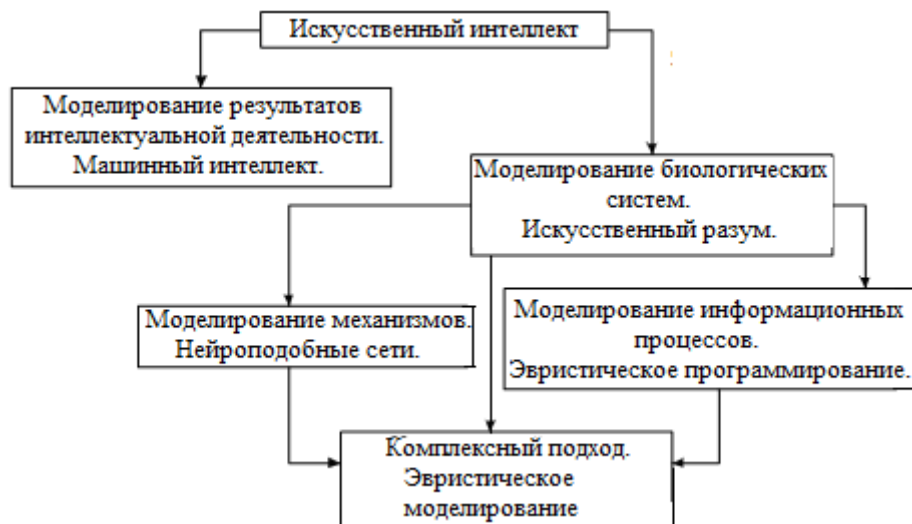


Рис. 1.3. Структура основных направлений работ в области ИИ

Достоинствами логического направления являются:

1. Возможность относительно легкого понимания работы системы.
2. Достижимость однозначности поведения системы в одинаковых ситуациях.

Недостатками этого подхода являются:

1. Трудность и неестественность реализации нечетких знаков (образов).
2. Трудность (или даже невозможность) реализации адекватного поведения в условиях неопределенности (недостаточности знаний, зашумленности данных, не точно поставленной цели и т.п.).
3. Трудность и неэффективность распараллеливания процесса решения задач.

Достоинства нейрокибернетического направления – это отсутствие недостатков, свойственных логическому направлению, а недостатки – отсутствие его достоинств. Кроме того, в нейрокибернетическом направлении привлекает возможность получить систему, настраивающуюся на сколь угодно сложное поведение и адекватное решаемой задаче. Причем ее сложность зависит только от количественных факторов модели нейронной сети.

Еще одним достоинством в случае аппаратной реализации нейронной сети является ее живучесть, т.е. способность сохранять приемлемую эффективность решения задачи при выходе из строя элементов сети. Это свойство нейронных сетей достигается за счет избыточности. В случае программной реализации структурная избыточность нейронных сетей позволяет им успешно работать в условиях неполной или зашумленной информации.

1.4. Выводы

1. Информационные системы, откликаясь на запросы пользователей, развивались в направлении усложнения класса решаемых задач. В настоящее время широкое применение находят интеллектуальные информационные системы.

2. Интеллектуальные системы применяют знания, используемые человеком при решении сложных плохо формализуемых или не формализуемых задач.

3. В интеллектуальных информационных системах признаки интеллектуальности развиты в неодинаковой степени. Соответственно различают системы с интеллектуальным интерфейсом, экспертные системы, моделирующие рассуждения человека-эксперта при решении задач, самообучающиеся системы и адаптивные системы.

4. В основу интеллектуальных информационных систем положены принципы, разрабатываемые в искусственном интеллекте. Согласно этим принципам интеллектуальные задачи можно решать, манипулируя символами с помощью механизмов поиска решений.

5. В настоящее время развивается новое направление в построении интеллектуальных систем. В таких системах моделируются биологические нейроны, составляющие основу человеческого мозга.

1.5. Вопросы для самоконтроля

1. Дайте определение интеллектуальных информационных систем. Укажите их отличия от традиционного программного обеспечения.

2. Назовите признаки интеллектуальности информационных систем.

3. Охарактеризуйте системы с интеллектуальным интерфейсом.

4. Какие задачи решаются с применением экспертных систем? Дайте характеристику этих задач.

5. Укажите основные классы самообучающихся систем.

6. Опишите кратко принцип работы индуктивных систем.

7. Известно, что существуют системы, основанные на прецедентах. К какому классу информационных систем они относятся? Охарактеризуйте принцип их работы. Укажите их главное отличие от индуктивных систем.

8. Какие задачи решаются с помощью адаптивных систем? Чем этот класс задач интересен? В чем состоят трудности при проектировании систем этого класса?

9. Разработки в области интеллектуальных информационных систем основаны на принципах, высказанных в области искусственного интеллекта. Что изучает эта наука? Кратко укажите основные принципы, лежащие в основе интеллектуальных систем.

10. Опишите основные направления развития искусственного интеллекта. Какие классы систем, по вашему мнению, развиваются в рамках этих направлений?

2. СИСТЕМЫ, ОСНОВАННЫЕ НА ЗНАНИЯХ

2.1. Определение экспертных систем

К системам, полностью основанным на знаниях, относятся два класса систем: экспертные системы (ЭС) и интеллектуальные пакеты прикладных программ (ИППП). Структура такого пакета приведена на рис. 2.1 [16].

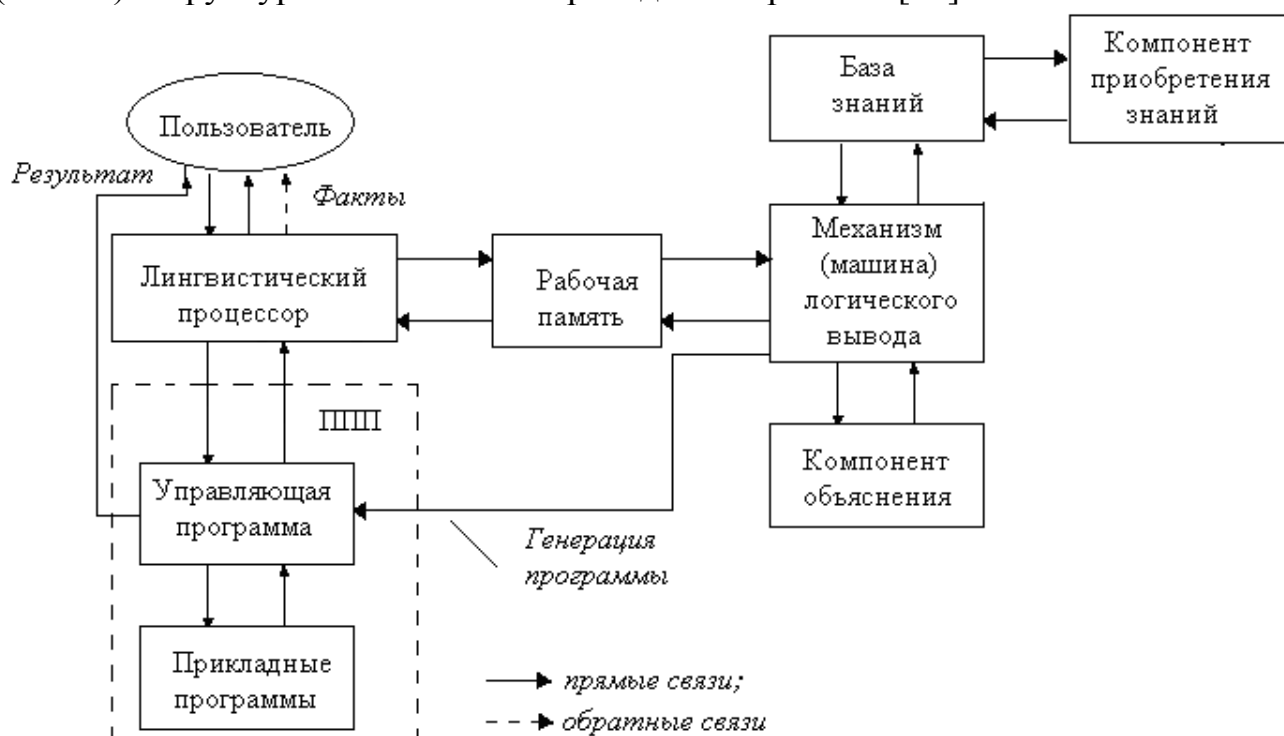


Рис. 2.1. Структура ИППП

ИППП дают возможность конечному пользователю решать прикладные задачи по их описаниям и исходным данным без программирования – генерация («сборка») программы «под задачу» осуществляется автоматически механизмом логического вывода. База знаний в ИППП может строиться по любому из известных эвристических методов (часто используются семантические сети и фреймы), чтобы настраиваемая машиной логического вывода программа была эффективна для решения поставленной задачи.

Однако наибольшее распространение получили экспертные системы (ЭС) различных типов. Огромный интерес к ЭС обусловлен тремя основными обстоятельствами.

1. ЭС ориентированы на решение широкого круга задач в ранее неформализуемых областях, которые считались малодоступными для использования ЭВМ.

2. ЭС предназначены для решения задач в диалоговом режиме со специалистами, от которых не требуется знания программирования – это резко расширяет сферу использования вычислительной техники, которая в данном случае выступает как инструмент поддержки памяти специалиста и усиления его способностей к логическому выводу.

3. Специалист, использующий ЭС для решения своих задач, может достигать, а иногда и превосходить по результатам возможности экспертов в данной области знаний, что позволяет резко повысить квалификацию рядовых специалистов за счет аккумуляции знаний в ЭС, в том числе знаний экспертов высшей квалификации.

Определение

Под *экспертной системой* понимается программная система, выполняющая действия, аналогичные тем, которые выполняет эксперт в некоторой прикладной предметной области, делая определенные заключения в ходе выдачи советов и рекомендаций.

Экспертные системы предназначены для решения так называемых неформализованных задач. Ранее мы уже дали понятие неформализованных задач, а также определили их свойства.

Экспертная система отличается от прочих прикладных программ наличием следующих признаков.

- Моделирует *механизм мышления человека*, умеющего решать задачи определенной проблемной области. Основное внимание уделяется воспроизведению компьютерными средствами методики решения проблемы, которая применяется экспертом.

- Система, помимо выполнения вычислительных операций, формирует определенные *соображения и выводы*, основываясь на тех знаниях, которыми она располагает.

- При решении задач основными являются *эвристические и приближенные методы*, которые, в отличие от алгоритмических, не всегда гарантируют успех. Эвристика – это знание, приобретенное человеком по мере накопления практического опыта решения проблем. Такие методы являются приближительными в том смысле, что, во-первых, они не требуют исчерпывающей исходной информации, и, во-вторых, существует определенная степень уверенности (или неуверенности) в том, что предлагаемое решение является верным.

В основу экспертных систем положены три принципа.

1. Возможность ЭС по решению практических задач обусловлена в первую очередь объемом базы знаний и возможностью ее пополнения.

2. Знания, позволяющие ЭС получить качественные и эффективные решения, являются в основном эвристическими, экспериментальными. Причина этого заключается в том, что решаемые задачи являются неформализованными или слабо формализованными. Необходимо также подчеркнуть, что знания

экспертов имеют индивидуальный характер, т.е. свойственны конкретному человеку.

3. Учитывая неформализованность решаемых задач и эвристический, личностный характер используемых знаний, пользователь (эксперт) должен иметь возможность непосредственно взаимодействовать с экспертной системы в виде диалога.

Архитектура экспертной системы вытекает из принципов, сформулированных выше. В соответствии с первыми двумя принципами ЭС включает в себя два компонента: *решатель* (процедуру вывода) и динамически изменяемую *базу знаний*.

Третий принцип предъявляет к системе следующие требования.

1. ЭС должна быть способна вести диалог о решаемой задаче на языке, удобном пользователю (эксперту), и, в частности, приобретать в ходе диалога новые знания.

2. ЭС должна быть способна при решении задачи следовать линии рассуждения, понятной пользователю (эксперту).

3. ЭС должна быть способна объяснять ход своего рассуждения, что необходимо как при использовании, так и при совершенствовании системы.

Первое требование реализуется диалоговым компонентом ЭС и компонентом приобретения знаний, а для выполнения второго и третьего требований в ЭС вводится объяснительный компонент. Кроме того, второе требование накладывает ограничения на способ решения задачи: ход рассуждения в процессе решения должен быть понятен пользователю (эксперту).

Структура *статической* ЭС приведена на рис. 2.2.

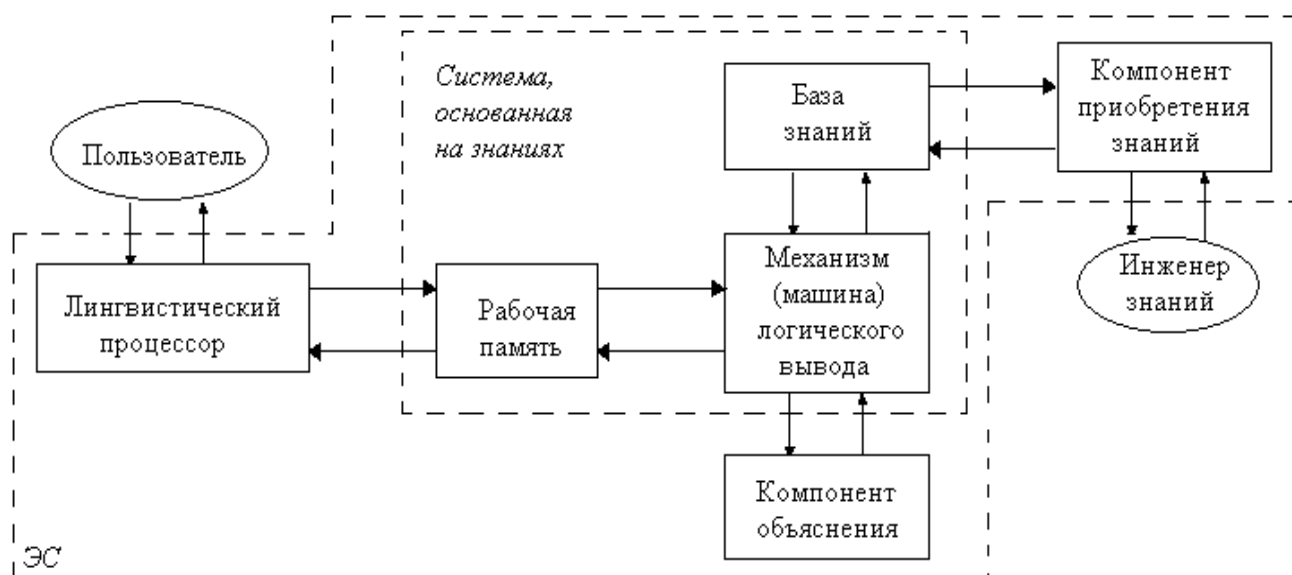


Рис. 2.2. Структура статической экспертной системы

База знаний – это совокупность единиц знаний, которые представляют собой формализованное с помощью некоторого метода представления знаний отражение объектов проблемной области и их взаимосвязей, а также действий, выполняемых над объектами.

Лингвистический процессор предназначен для обеспечения комфортного взаимодействия между конечным пользователем и ЭС. В нем реализуются процедуры морфологического, синтаксического и семантического контроля поступающих в систему запросов и приведение их к виду, «понятному» ЭВМ. При выдаче ответной информации осуществляется обратная операция – заключение «переводится» на ограниченный естественный язык, понятный конечному пользователю.

Механизм (машина) логического вывода – часть ЭС, реализующая анализ поступающей в ЭС и имеющейся в ней информации и формирование (вывод) на ее основе новых заключений (суждений) в ответ на запрос к системе. Синонимы: *решатель, дедуктивная машина, блок логического вывода*.

Рабочая память – часть ЭС, предназначенная для информационного обеспечения работы механизма логического вывода, прежде всего в части хранения и обработки поступивших фактов (суждений) и промежуточных результатов логического вывода.

Компонент приобретения знаний предназначен для обеспечения работы инженера знаний по поддержанию модели знаний, адекватной реальной предметной области (генерации базы знаний, ее тестирования, пополнения новыми знаниями и т.п.).

Наличие *компонента объяснений*, обеспечивающего по запросу пользователя выдачу информации о ходе и результате логического вывода, принципиально отличает ЭС от всех других программных систем. Он поясняет, *как* система получила решение задачи (или *почему* получила такое решение) и *какие* знания она при этом использовала, что облегчает эксперту тестирование системы и повышает доверие пользователя к полученному результату. Ответ на вопрос «как» – это трассировка всего процесса получения решения с указанием использованных фрагментов БЗ, т.е. всех шагов цепи умозаключений. Ответ на вопрос «почему» – ссылка на умозаключение, непосредственно предшествовавшее полученному решению, т.е. отход на один шаг назад.

Структуру, приведенную на рис. 2.2, называют *структурой статической ЭС*. ЭС данного типа используют в тех приложениях, где можно не учитывать изменения окружающего мира, происходящие за время решения задачи. Они нашли применение в широком классе приложений.

В *динамических экспертных системах* знания меняются во времени. Кроме того эти системы должны вырабатывать решения в реальном времени и моделировать различные состояния окружающего мира. Поэтому они снабжаются *блоками моделирования внешнего мира и сопряжения с внешним миром* (рис. 2.3).

Подсистема сопряжения с внешним миром осуществляет связи с внешним миром через систему датчиков и контроллеров. Такие компоненты как база знаний и машина вывода существенно изменяются, чтобы отразить временную логику событий, происходящих в реальном мире.

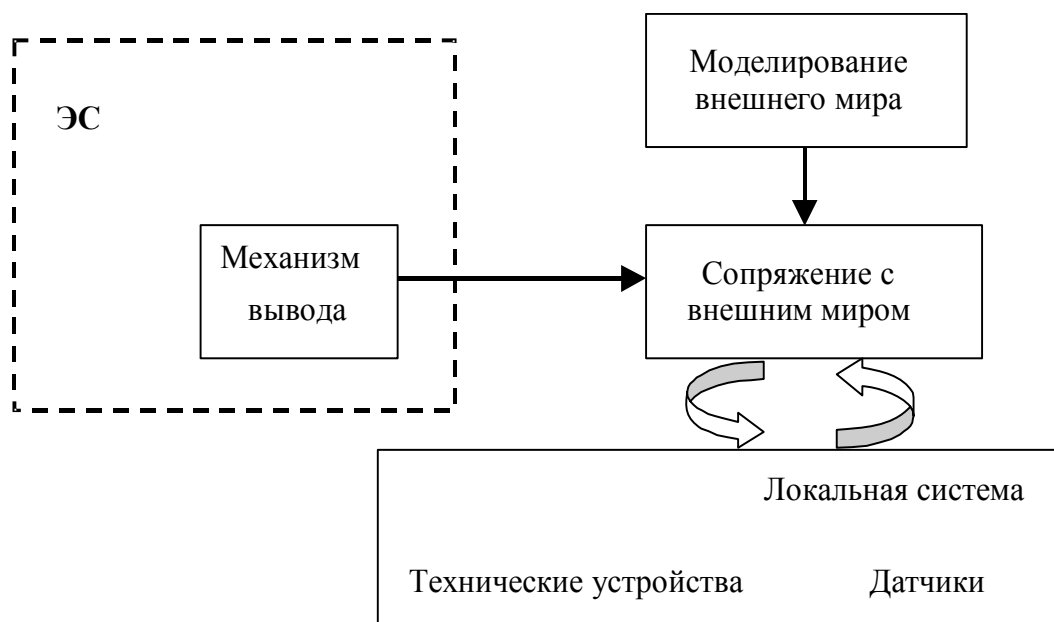


Рис. 2.3. Архитектура динамической экспертной системы

Подсистема моделирования внешнего мира на основании показаний датчиков и знаний, хранящихся в базе знаний, моделирует поведение системы в реальном масштабе времени и пытается предсказать поведение системы в будущем.

Следует учесть, что реальная ЭС может иметь более сложную структуру, однако блоки, изображенные на рис. 2.2 и рис. 2.3, непременно присутствуют в любой экспертной системе, поскольку представляют собой стандарт структуры современной ЭС.

2.2. Классы экспертных систем

Существуют различные основания классификации экспертных систем. Мы приведем некоторые из них (рис. 2.4).

По способу формирования решения экспертные системы разделяются на два класса: *аналитические* и *синтетические*. В аналитических системах предполагается выполнять выбор решений из множества известных альтернатив (определение характеристик объектов). В синтетических системах генерируются неизвестные решения (формируются объекты).

По способу учета временного фактора экспертные системы могут быть *статическими* и *динамическими*. Статические системы решают задачи при неизменяемых в процессе решения данных и знаниях, динамические системы допускают такие изменения. Статические системы осуществляют монотонное непрерывное решение задачи от ввода данных до конечного результата, динамические системы предусматривают возможность пересмотра в процессе решения полученных ранее результатов.

По видам используемых знаний и данных экспертные системы классифицируются на системы с *детерминированными* (четко определенными) знаниями и *неопределенными* знаниями.

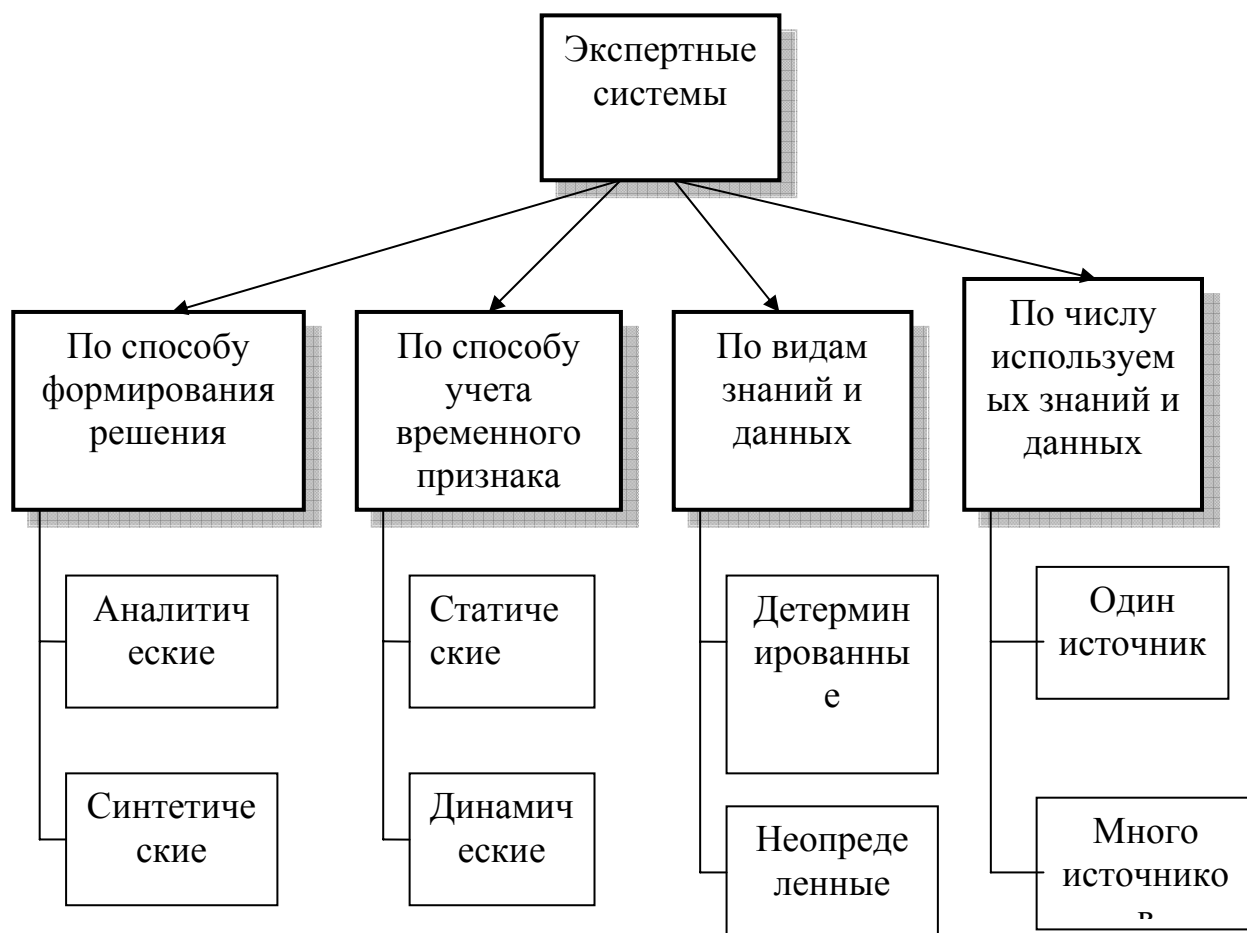


Рис. 2.4. Классификация экспертных систем

По числу используемых источников знаний экспертные системы могут быть построены с использованием *одного* или *множества* источников знаний. Источники знаний могут быть альтернативными или дополняющими друг друга.

В соответствии с перечисленными признаками классификации, как правило, выделяются следующие четыре основных класса ЭС (табл. 2.1).

Таблица 2.1

	Анализ	Синтез	
Детерминированность знаний	Классифицирующие	Трансформирующие	Один источник знаний
Неопределенность знаний	Доопределяющие	Многоагентные	Множество источников знаний
	Статика	Динамика	

Классифицирующие экспертные системы

К классифицирующим аналитическим задачам, прежде всего, относятся задачи распознавания различных ситуаций, когда по набору заданных факторов выявляется ситуация, в зависимости от которой выбирается определенная последовательность действий. В качестве основного метода формирования

решений в этих системах используется метод логического дедуктивного вывода от общего к частному, когда путем подстановки исходных данных в некоторую совокупность взаимосвязанных общих утверждений получается частное заключение.

Доопределяющие экспертные системы

Более сложный тип аналитических задач представляют задачи, которые решаются на основе неопределенных исходных данных и применяемых знаний. В качестве методов работы с неопределенностями могут использоваться байесовский вероятностный подход, коэффициенты уверенности, нечеткая логика.

Для аналитических задач классифицирующего и доопределяющего типов характерны следующие проблемные области:

- *Интерпретация данных* – выбор решения из фиксированного множества альтернатив на базе введенной информации о текущей ситуации. Типичным примером является ЭС анализа финансового состояния предприятия.
- *Диагностика* – выявление причин неисправности объекта. Под неисправностью понимается любое отклонение от нормы. Такая трактовка позволяет с единых теоретических позиций рассматривать и неисправности в технических системах, и заболевания живых организмов, и всевозможные природные аномалии.
- *Коррекция* – диагностика, дополненная возможностью оценки и рекомендации действий по исправлению отклонений от нормального состояния рассматриваемой ситуации.

Трансформирующие экспертные системы

В отличие от аналитических статических ЭС, синтезирующие динамические ЭС предполагают повторяющееся преобразование знаний в процессе решения задач, что связано с характером результата, который нельзя заранее предопределить, а также с динамичностью самой проблемной областью.

Многоагентные системы

Для многоагентных динамических систем характерна интеграция в базе знаний нескольких разнородных источников знаний, обменивающихся между собой получаемыми результатами на динамической основе.

Для многоагентных систем характерны следующие особенности.

1. Проведение альтернативных рассуждений на основе использования различных источников знаний с механизмом устранения противоречий.
2. Распределенное решение проблем, которые разбиваются на параллельно решаемые подпроблемы, соответствующие самостоятельным источникам знаний.
3. Применение множества стратегий работы механизма вывода заключений в зависимости от типа решаемой проблемы.

Для синтезирующих динамических экспертных систем наиболее применимы следующие проблемные области:

- *Мониторинг* – слежение за текущей ситуацией с возможной последующей корректировкой.
- *Проектирование* – определение конфигурации объектов с точки зрения достижения заданных критериев эффективности и ограничений.
- *Прогнозирование* – предсказание последствий некоторых событий или явлений на основании анализа имеющихся данных.
- *Планирование* – выбор последовательности действий пользователя по достижению поставленной цели.

2.3. Технология разработки экспертных систем

Коллектив разработчиков ЭС

Под коллективом разработчиков экспертных систем будем понимать группу специалистов, ответственных за создание ЭС.

В состав коллектива разработчиков входят, по крайней мере, четыре человека – пользователь, эксперт, программист и инженер по знаниям.

Руководителем группы при отсутствии специального менеджера является *инженер по знаниям*. *Пользователь* – это обычно заказчик системы. Необходимо, чтобы пользователь имел некоторый базовый уровень квалификации, который позволит ему правильно истолковать рекомендации ЭС.

Эксперт – чрезвычайно важная фигура в группе коллектива разработчиков. Его подготовка определяет уровень компетенции базы знаний. Помимо высокого профессионализма в выбранной предметной области, желательно знакомство эксперта с основными понятиями теории искусственного интеллекта и экспертными системами для того, чтобы этап извлечения знаний прошел успешно.

Программист должен быть обязательно знаком с основными структурами представления знаний и механизмами вывода, состоянием отечественного и мирового рынка программных продуктов для разработки ЭС.

Самые высокие требования предъявляются к квалификации инженера по знаниям, ибо от его компетенции зависит успешность разработки ЭС. Инженерия знаний принадлежит к такому виду профессиональной деятельности, для которых природные качества личности имеют характер абсолютного показателя или противопоказания к таким занятиям. По различным оценкам [4] это одна из самых малочисленных, высокооплачиваемых и дефицитных в мире специальностей.

Инженер по знаниям «задает тон» в общении с экспертом, он ведет диалог, и от него, в конечном счете, зависит его продуктивность.

В профессиональной области инженер по знаниям должен быть знаком с психологией и способами представления понятий в памяти человека, с различными способами активизации мышления; должен иметь широкую общенаучную подготовку. Для моделирования знаний ему необходимо знать

методологию представления знаний, системный анализ, теорию познания, кластерный и факторный анализ; изучить аппарат формальной логики и современные языки представления знаний.

Выбор подходящей проблемы

Этап выбора подходящей проблемы включает в себя [5]:

- определение проблемной области и задачи;
- нахождение эксперта, желающего сотрудничать при решении проблемы, и назначение коллектива разработчиков;
- определение предварительного подхода к решению проблемы;
- анализ расходов и прибылей от разработки;
- подготовку подробного плана разработки.

Правильный выбор проблемы представляет самую критическую часть разработки в целом. При выборе области применения следует учитывать, что если знание, необходимое для решения задач, постоянное, четко сформулировано и связано с вычислительной обработкой, то обычные алгоритмические программы, вероятно, будут самым целесообразным способом решения проблем в этой области. Но если результативность задачи зависит от знания, которое является субъективным, изменяющимся, символьным или вытекающим частично из соображений здравого смысла, тогда можно обоснованно разрабатывать экспертную систему.

Обычно экспертные системы разрабатываются путем получения специфических знаний от эксперта. В процессе разработки и последующего расширения системы инженер по знаниям и эксперт обычно работают вместе. Поэтому на предварительном этапе необходимо оценить возможность творческой совместной работы над проектом.

После того, как инженер по знаниям убедился, что:

- данную задачу можно решить с помощью экспертной системы;
- экспертную систему можно создать предлагаемыми на рынке средствами;
- имеется подходящий эксперт;
- предложенные критерии производительности являются разумными;
- затраты и срок окупаемости приемлемы для заказчика,

он составляет план разработки. План определяет шаги процесса разработки и необходимые затраты, а также ожидаемые результаты.

Этапы разработки ЭС

В ходе работ по созданию ЭС сложилась определенная технология их разработки, включающая шесть этапов (рис. 2.5).

Приведем краткую характеристику этих этапов.

На этапе *идентификации* проблемы уточняется задача, планируется ход разработки прототипа экспертной системы, определяются:

- необходимые ресурсы;
- источники знаний (книги, дополнительные эксперты, методики);

- имеющиеся аналогичные экспертные системы;
- цели создания экспертной системы.

Задача этапа *идентификации* – знакомство и обучение членов коллектива разработчиков, а также создание неформальной формулировки проблемы.

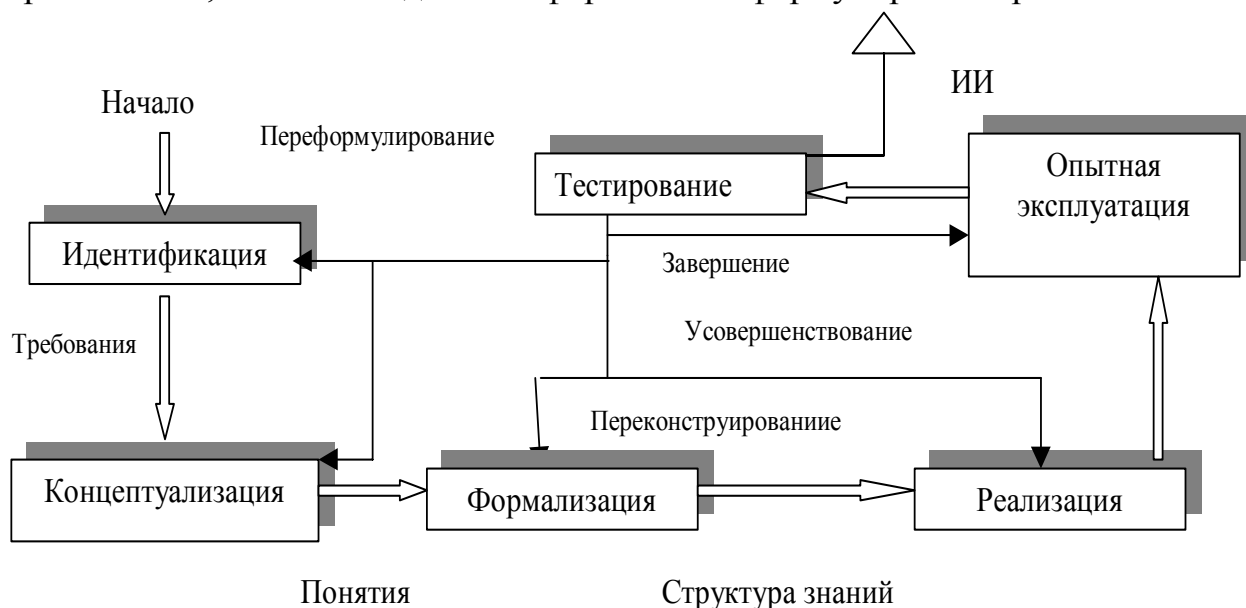


Рис. 2.5. Этапы создания экспертной системы

Второй этап технологии разработки экспертных систем называется этапом *структурирования* (*концептуализации*) знаний. Разделение стадий извлечения и структурирования знаний является весьма условным, поскольку хороший инженер по знаниям, уже извлекая знания, начинает работу по структурированию знаний.

На этапе построения концептуальной модели создается целостное и системное описание используемых знаний, отражающее сущность функционирования предметной области. На этом этапе выявляется структура полученных знаний о предметной области, т.е. определяется список основных понятий и атрибутов, характеризующих предметную область; выявляются отношения между понятиями, структура входной и выходной информации и т.п.

Задача этапа *концептуализации* – разработка неформального описания знаний о предметной области, которые отражают основные концепции и взаимосвязи между понятиями предметной области.

От качества построения концептуальной модели проблемной области во многом зависит, насколько часто в дальнейшем по мере развития проекта будет выполняться перепроектирование базы знаний. Хорошая концептуальная модель может только уточняться (детализироваться или упрощаться), но не перестраиваться.

Результат концептуализации статической проблемной области обычно отображают графически на объектном и функциональном уровнях моделирования.

Объектная модель описывает структуру предметной области как совокупности взаимосвязанных объектов.

Функциональная модель отражает действия и преобразования над объектами.

Объектная модель отражает фактуальное знание о составе объектов, их свойствах и связях. Элементарной единицей структурного знания является факт, описывающий одно свойство или одну связь объекта, который представляется в виде триплета *предикат (объект, значение)*.

Если предикат определяет название свойства объекта, то в качестве значения выступает конкретное значение этого свойства, например, *профессия («Иванов», «Экономист»)*.

Если предикат определяет название связи объекта, то значению соответствует объект, с которым связан первый объект, например, *работает («Иванов», «Бухгалтерия»)*.

В качестве важнейших типизированных видов отношений рассматриваются следующие отношения:

- «целое» – «часть» (агрегация);
- «причина» – «следствие»;
- «цель» – «средство»;
- «функция» – «аргумент»;
- «ассоциация» и др.

Функциональная модель описывает преобразования фактов, зависимости между ними, показывающие как одни факты образуются из других. Функциональная зависимость между фактами выражается в следующем виде:

$$A_1 \wedge A_2 \wedge \dots \wedge A_n \Rightarrow B.$$

Эта запись означает, что факт *B* имеет место только в том случае, если имеет место конъюнкция фактов или их отрицаний A_1, A_2, \dots, A_n , например:

сбыт (товар, «слабый») И прибыль (товар, «ничтожная») И потребители (товар, «любители нового») И число_конкурентов (товар, «небольшое») \Rightarrow жизненный_цикл (товар, «выведение на рынок»).

Функциональную зависимость фактов можно трактовать как отражение следующих отношений между фактами:

- «причина» – «следствие»;
- «средство» – «цель»;
- «аргумент» – «функция»;
- «ситуация» – «действие».

Функциональная модель строится путем последовательной декомпозиции цели на подцели. Каждой цели (подцели) соответствует некоторая задача (подзадача), которая не может быть решена, пока не будут достигнуты ее нижестоящие подцели (решены подзадачи). Таким образом, функциональная модель отражает в обобщенной форме процесс решения характерных для нее задач.

Обычно функциональные зависимости фактов представляются графически в виде деревьев целей или графов «И-ИЛИ».

После того как составлена объектная и функциональная модели предметной области можно переходить к этапу *формализации знаний* о предметной области.

На этапе формализации выбирается язык представления знаний и осуществляется проектирование логической структуры базы знаний. *Задача* этапа – *разработать базу знаний* на языке, который, с одной стороны, соответствует структуре знаний, а с другой – позволяет реализовать прототип системы.

На этапе *реализации* строится прототип экспертной системы. *Задача* этапа – разработка программного обеспечения комплекса, демонстрирующего жизнеспособность подхода в целом.

На этапе реализации происходит физическое заполнение базы знаний и настройка всех программных механизмов в рамках выбранного *инструментального* средства, а при необходимости и программирование специализированных модулей программного инструмента.

Под *инструментальными* средствами разработки экспертных систем понимается совокупность технических и программных средств.

При создании ЭС используются такие программные средства, как символьные языки логического программирования, языки инженерии знаний, программное обеспечение на базе проблемно- и процедурно-ориентированных языков программирования, инструментальные комплексы для разработки интеллектуальных систем, оболочки систем.

Как и любую программу, ЭС можно написать на машинно-ориентированном языке или на языке высокого уровня. Зачем же нужны специализированные средства? Практика показывает, что процесс программирования ЭС на специализированных средствах занимает в 2–3 раза меньше времени, чем на универсальных. Однако следует понимать, что ЭС, реализованные на базе специализированных средств, работают менее эффективно, чем реализованные на базе универсальных средств.

Поэтому прототипы ЭС строят с использованием специализированных средств, а промышленная система «переписывается» на традиционных программных средствах.

Использование *языков представления знаний*, таких как язык логического программирования PROLOG, язык функционального программирования LISP, язык объектно-ориентированного программирования SmallTalk, язык продукционных правил OSP5 и других повышает гибкость разрабатываемой системы и одновременно увеличивает трудоемкость разработки.

Инструментальные комплексы состоят из набора программных средств, необходимых для выполнения базовых функций по созданию интеллектуальных систем. Примером является CLIPS, содержащий среду для разработки ЭС и выполняемый модуль.

В настоящее время часто используются *оболочки* ЭС, т.е. программные средства, комплексно реализующие функции, как специальных языков, так и пакетов программ автоматизации проектирования. Оболочки содержат средства извлечения знаний, их тестирования, редактирования и накопления.

Более развиты оболочки для продукционной модели представления знаний. В состав оболочек включают СУБД, электронные таблицы, пакеты деловой графики. Примером таких оболочек являются MultiLogic, Crystal, EXSYS.

Технические средства построения экспертных систем разделяется на три категории:

- специализированные (Лисп-машины в США, Пролог-машины в Японии, объектно-ориентированные в Европе);
- Лисп-сопроцессоры;
- традиционные технические средства, такие как персональные электронные машины, рабочие станции.

На этапе *тестирования* оценивается и проверяется работа прототипа экспертной системы с целью приведения в соответствие с реальными запросами пользователей. Выделяются три аспекта тестирования.

1. Тестирование исходных данных с целью проверки набора фактов, лежащих в основе ЭС.

2. Логическое тестирование базы знаний, которое заключается в выявлении полноты и непротиворечивости правил. Возможные ошибки носят формальный характер, поэтому этот этап можно автоматизировать.

3. Концептуальное тестирование проводится для проверки общей структуры системы и учета в ней всех аспектов решаемой задачи. На этом этапе следует привлекать пользователя ЭС.

На этапе *опытной* эксплуатации осуществляется решение реальных задач, выполняется оценка стоимости системы и ее эффективность. По результатам эксплуатации может потребоваться модификация системы. Можно выделить следующие виды модификации системы.

- *Усовершенствование* прототипа осуществляется в процессе циклического прохождения через этапы реализации и тестирования с целью отладки правил и процедуры вывода.
- *Переконструирование* представления знаний выполняется после выполнения тестирования и требует возврата к этапу формализации.

Если возникшие проблемы очень серьезные, то после неудачи на этапе тестирования может потребоваться возврат к этапам концептуализации и идентификации. В этом случае ЭС строится заново.

Стадии существования экспертных систем

Выделяют пять стадий существования экспертных систем [5]. Первоначально строится *демонстрационный* прототип ЭС. Это усеченная версия экспертной системы, спроектированная для проверки правильности фактов, связей и стратегий рассуждений эксперта.

Объем прототипа – несколько десятков правил. Он должен удовлетворять двум противоречивым требованиям: с одной стороны он должен решать типичные задачи данного приложения, с другой – трудоемкость его создания должна быть минимальной.

Цель создания демонстрационного прототипа – продемонстрировать применимость методов инженерии знаний для данного приложения. В случае успеха эксперт с помощью инженера по знаниям расширяет знания прототипа о предметной области. При неудаче может потребоваться разработка нового прототипа.

Исследовательский прототип решает большинство задач, но неустойчив в работе и не полностью проверен (содержит несколько сотен правил и понятий).

Действующий прототип надежно решает все задачи на реальных примерах, но для сложной системы требует много времени и памяти.

Промышленная система обеспечивает высокое качество решений при минимизации требуемого времени и памяти; переписывается с использованием эффективных средств представления знаний.

Коммерческая система – это промышленная система, пригодная к продаже, т.е. хорошо документированная и снабжена сервисом.

2.4. Выводы

1. Экспертная система (ЭС) – это ИИС, предназначенная для решения слабо формализуемых задач на основе накапливаемого в базе знаний опыта работы экспертов в проблемной области.

2. Типичная архитектура ЭС включает в свой состав следующие компоненты. База знаний (БЗ) – это хранилище единиц знаний, описывающих атрибуты и действия, связанные с объектами проблемной области, а также возможные при этом неопределенности. Это центральный компонент ЭС, который определяет ценность ЭС и с которым связаны основные затраты на разработку.

Механизм вывода – это обобщенная процедура поиска решения задачи, которая на основе БЗ и в соответствии с информационной потребностью пользователя строит цепочку рассуждений, приводящую к конкретному результату.

Механизм приобретения знаний – это процедура накопления знаний в базе знаний, включающая ввод, контроль полноты и непротиворечивости единиц знаний и, возможно, автоматический вывод новых единиц знаний из вводимой информации.

Механизм объяснения – это процедура, выполняющая обоснование полученного результата.

Интеллектуальный интерфейс – это процедура, выполняющая интерпретацию запроса пользователя к базе знаний и формирующая ответ в удобной для него форме.

3. Назначение ЭС – консультирование и обучение неопытных пользователей, помощь экспертам в решении задач, советы экспертам по вопросам из смежных областей.

4. Коллектив разработчиком ЭС обычно включает в себя инженера по знаниям, эксперта, программиста и пользователя. Ключевую роль играет инженер по знаниям.

4. Технология создания экспертных систем включает в себя этапы: идентификация, концептуализация, формализация, реализация, тестирование и опытная эксплуатация. На каждом этапе решаются свои задачи.

5. Создание экспертных систем – это процесс доработки демонстрационного прототипа до коммерческого продукта.

2.5. Вопросы для самоконтроля

1. Укажите отличия экспертных систем от обычного программного обеспечения.

2. Дайте определение экспертной системы. Что в этом определении является основополагающим?

3. Охарактеризуйте принципы, положенные в основу экспертных систем. Какие требования эти принципы предъявляют к экспертным системам?

4. Дайте характеристику основным модулям статической и динамической экспертных систем.

5. Укажите основные отличия аналитических экспертных систем от синтетических.

6. Какие задачи решают классифицирующие экспертные системы?

7. Какие черты присущи доопределяющим экспертным системам?

8. Дайте характеристику коллективу разработчиков экспертных систем.

9. Вы изучали психологию. Какими чертами личности с точки зрения психологии должен обладать инженер по знаниям?

10. Изучение каких дисциплин по вашей специальности может способствовать вашей плодотворной работе в области инженерии знаний?

11. Определите цели и задачи этапа идентификации предметной области.

12. В чем заключается концепция быстрого прототипа?

13. Опишите цели и задачи этапа концептуализации при построении экспертной системы

14. Какие свойства предметной области отражает объектная модель? Как она строится?

15. Какие связи между объектами предметной области отображает функциональная модель?

16. Перечислите этапы проектирования экспертных систем. Назовите цели, стоящие перед разработчиками системы на каждом этапе.

17. Назовите инструментальные средства, используемые при проектировании экспертных систем.

21. Для чего используются оболочки экспертных систем? Что может входить в их состав?

22. Можно ли при проектировании экспертных систем использовать язык C++? Ответ обоснуйте.

3. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

3.1. Знания и их свойства

При работе с интеллектуальными информационными системами традиционно возникает вопрос о том, как соотносятся понятия «данные» и «знания».

Попытаемся ответить на вопрос, что же такое знания и чем они отличаются от обычных данных. Возьмем за рабочую основу такие определения [4].

Определение

Данные – это информация, полученная в результате наблюдений или измерений отдельных свойств (атрибутов), характеризующих объекты, процессы и явления предметной области.

Иначе, данные – это конкретные факты.

Знания основаны на данных, полученных эмпирическим путем. Они представляют собой результат опыта и мыслительной деятельности человека, направлены на обобщение этого опыта, полученного в результате практической деятельности.

Определение

Знания – это связи и закономерности предметной области (принципы, модели, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в данной области.

Часто используется такое определение знаний.

Определение

Знания – это хорошо структурированные данные, или данные о данных, или метаданные.

Существует множество способов определять понятия. Один из широко применяемых способов основан на идее интенционала [5].

Определение

Интенционал понятия – это определение его через соотнесение с понятием более высокого уровня абстракции с указанием специфических свойств.

Интенционалы формируют знания об объектах. Другой способ определяет понятие через соотнесение с понятиями более низкого уровня абстракции или перечисление фактов, относящихся к определяемому понятию.

Определение

Экстенсионал определяет понятия через данные.

На рис. 3.1. интенсиналом понятия «Собака» является определение ее свойства «Домашнее животное, которое лает», экстенсионалом этого понятия является перечисление пород.

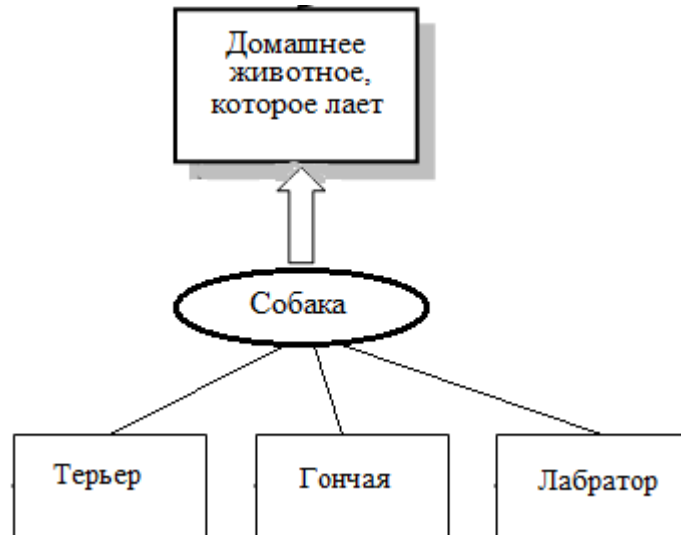


Рис. 3.1. Интенсинал и экстенсионал понятия «Собака»

Для хранения данных используются базы данных (для них характерны большой объем и относительная небольшая удельная стоимость информации). Для хранения знаний – базы знаний (исключительно дорогие информационные массивы). База знаний – основа любой интеллектуальной системы.

Какие же свойства превращают данные в знания? Перечислим и кратко охарактеризуем основные свойства знаний (часть из них присуща и данным).

1. Внутренняя интерпретация. Существенным отличием знаний от данных является их интерпретируемость, Если для интерпретации данных необходимы соответствующие программы и сами по себе они не несут содержательной информации, то знания всегда содержательны.

Это свойство предполагает, что в ЭВМ хранятся не только собственно данные, но и данные о данных, что позволяет содержательно их интерпретировать [16].

Предприятие	Место нахождения	Что выпускает
Завод им. Хруничева	Москва	Космическую технику
НПО «Энергия»	Королев	Космическую технику
НПО «Комета»	Москва	Конструкторскую документацию

Имея такую информацию, можно ответить на вопросы типа: «Где находится завод им. Хруничева?», или «Какие предприятия выпускают космическую технику?». При этом в первой строке находятся данные о метаданных, а в остальных – сами данные.

2. Структурированность. Информационные единицы должны обладать гибкой структурой. Структура знаний задается с помощью отношений, существующих между единицами знаний. Различают следующие типы отношений:

- отношения структуризации, которые задают иерархию информационных единиц. Например, в отношении иерархии находятся объекты факультет – курс – учебная группа – студент;
- функциональные отношения, несущие процедурную информацию, позволяющую находить одни информационные единицы через другие;
- каузальные отношения, задающие причинно-следственные связи;
- семантические отношения, которые соответствуют всем остальным отношениям.

3. Ситуативные связи. Это свойство характеризует ситуационную близость информационных единиц. Ее можно назвать отношением релевантности, которое дает возможность выделять типовые ситуации (например, покупка, банкротство, производство). Это свойство позволяет находить в базе знаний знания, близкие по смыслу.

4. Активность. При работе с ЭВМ данные пассивны, команды активны и все процессы, протекающие в ЭВМ, инициируются командами. В ИИС выполнение действий должно инициироваться текущим состоянием базы знаний.

Резкой границы между данными и знаниями нет, и с развитием средств информатики отличия знаний от данных сглаживаются (рис. 3.2.)



Рис. 3.2. Особенности знаний

Перечисленные особенности определяют грань, за которой данные превращаются в знания, а базы данных в базы знаний.

Следует отметить, что в системах, основанных на знаниях, знания отделены от алгоритмов, могут быть выведены в явном виде, отредактированы, дополнены без вмешательства в исходный код самой программы.

5. Шкалирование. Предполагает введение отношений между различными информационными единицами (т.е. их измерение в какой-либо шкале – порядковой, классификационной, метрической и т.п.) и упорядочение информационных единиц путем изменения интенсивности отношений и свойств.

Пример

«Группа ЭиУ-425 занимает первое место на курсе по успеваемости».

3.2. Классификация моделей представления знаний

Для использования знаний в информационных системах, их надо представить в виде определенной модели. При проектировании модели представления знаний необходимо учитывать два требования:

- однородность представления;
- простота понимания.

Выполнение этих требований позволяет упростить механизм логического вывода и процессы приобретения знаний и управления ими.

Модели представления знаний можно условно разделить на *декларативные* и *процедурные*.

Исторически первыми были процедурные знания, т.е. знания, «растворенные» в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы. Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся, и все большая часть знаний сосредоточивалась в структурах данных, т.е. увеличивалась роль декларативных знаний.

Модели представления знаний, кроме того, делятся на *логические* и *эвристические* модели. В основе логических моделей лежит понятие формальной системы (теории). Как правило, используется исчисление предикатов первого порядка, дополненное рядом эвристических стратегий. Эти методы являются системами дедуктивного типа, т.е. в них используется модель получения вывода из заданной системы посылок с помощью фиксированной системы правил вывода. Развитием предикатных систем являются системы индуктивного типа, в которых правила вывода порождаются системой на основе обработки конечного числа обучающих примеров.

В отличие от формальных моделей эвристические модели имеют набор средств, передающих специфические особенности проблемной области. Поэтому они превосходят логические как по возможности адекватно представлять предметную область, так и по эффективности использования правил вывода. К эвристическим моделям относят сетевые, фреймовые, продукционные и объектно-ориентированные модели.

Знания можно также разделить на *поверхностные* и *глубинные*. Глубинные знания – абстракции, аналогии, схемы, отображающие структуру и природу

процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объекта. Поверхностные – знания о видимых взаимосвязях между отдельными событиями и фактами предметной области.

В свою очередь все множество моделей можно разделить на две большие группы: *модульные* и *сетевые*.

Модульные модели оперируют отдельными (не связанными) элементами знаний, будь то правила или аксиомы предметной области.

Сетевые модели предоставляют возможность связывать фрагменты знаний через отношения в семантические сети или сети фреймов.

Современные экспертные системы работают, в основном, с поверхностными, профессиональными, явными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные и неявные структуры знаний и работать с ними.

Классификация наиболее популярных моделей представления знаний показана на рис. 3.3.

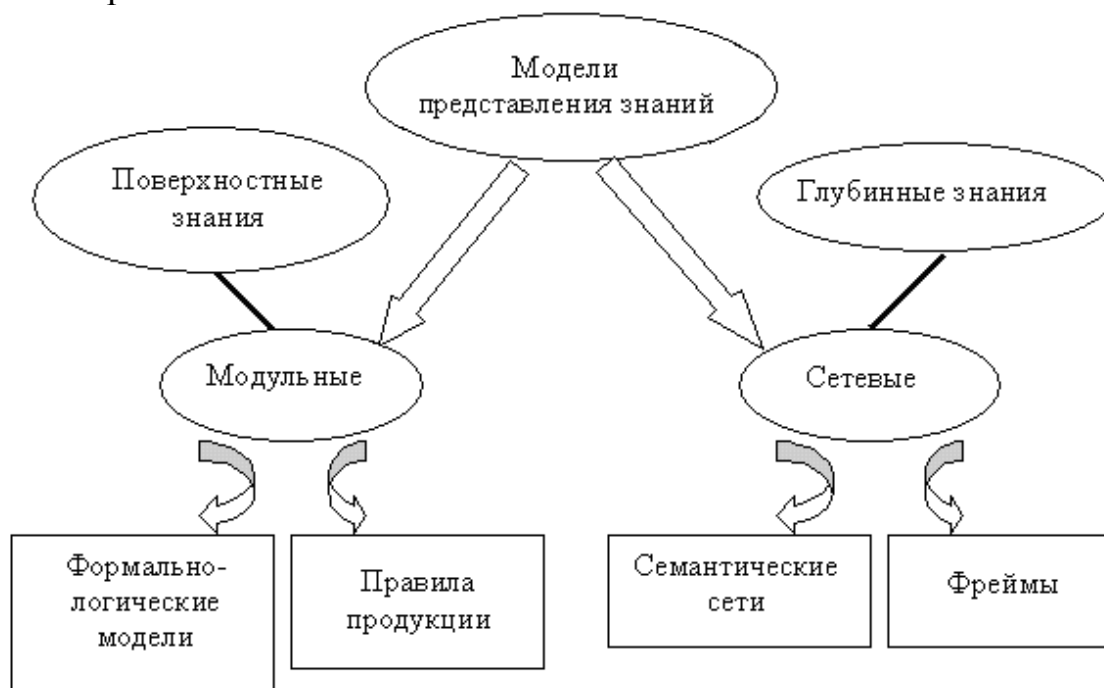


Рис. 3.3. Классификация моделей представления знаний

3.3. Логическая модель представления знаний

Логика предикатов первого порядка

Определение

В основе логических моделей лежит понятие формальной теории, которая задается кортежем: $S = \langle T, P, A, F \rangle$, здесь T – конечное множество базовых элементов (алфавит) системы; P – множество синтаксических правил, позволяющих из T строить синтаксически правильные выражения (формулы); A – множество априорно истинных формул (аксиомы); F – семантические правила

или правила вывода, позволяющие получить формулы, которые считаются истинными в данной теории.

Наиболее распространенной формальной системой, используемой для представления знаний, является исчисление предикатов первого порядка. Алфавит исчисления предикатов первого порядка состоит из следующего набора символов:

- знаков пунктуации $\{ (,) \}$;
- пропозициональных связок $\{ \rightarrow, \wedge, \neg, \vee \}$;
- знаков кванторов $\{ \exists, \forall \}$;
- символов переменных $x_k, k = 1, 2, \dots$;
- n -местных функциональных символов $f_k^n, k \geq 1, n \geq 0, (f_k^0 - \text{константная буква})$;
- n -местных предикатных символов $p_k^n, k \geq 1, n \geq 1$.

Из символов алфавита строят выражения, среди которых выделяют *термы*, *элементарные формулы* и *правильно построенные формулы* (или просто *формулы*).

Определения

Всякий символ переменной или константной буквы – *терм*. Если t_1, \dots, t_n – терм ($n \geq 1$), то $f_k^n(t_1, \dots, t_n)$ – терм.

Если p_k^n – предикатный символ, а t_1, \dots, t_n – терм, то $p_k^n(t_1, \dots, t_n)$ – *элементарная формула (атом)*. Атом – это правильно построенная формула.

Если A, B – это правильно построенные формулы или просто *формулы*, то $\neg A, A \wedge B, A \vee B, A \rightarrow B$ – есть правильно построенные формулы.

Если A – формула, а x – переменная в A , то конструкции $\exists x(A)$ и $\forall x(A)$ – формулы. Выражение является правильно построенной формулой, только если оно получено с соблюдением приведенных выше правил.

Для придания формуле содержания ее *интерпретируют* как утверждение, касающееся рассматриваемой предметной области. Пусть D – *область интерпретации*. Под интерпретацией понимают всякую систему, состоящую из непустого множества D и какого-либо соответствия, относящего каждой предикатной букве p_k^n некоторое n -местное отношение в D ; каждой функциональной букве f_k^n некоторую n -местную функцию, отображающую $D^n \rightarrow D$; каждой константной букве f_k^0 некоторый элемент из D . При заданной интерпретации всякой элементарной формуле приписывается значение «истинно» или «ложно».

Приписывание значения элементарной формуле $p_k^n(t_1, \dots, t_n)$ осуществляется по следующему правилу: если термы предикатной буквы соответствуют элементам из D , удовлетворяющим отношению, определяемому данной интерпретацией, то значением элементарной формулы будет истина, в противном случае – ложь. Значение неэлементарной формулы вычисляется рекуррентно, исходя из таблиц истинности. Очевидно, что значения формул могут быть истинными или ложными в зависимости от выбранной интерпретации.

Основной задачей, решаемой в рамках исчисления предикатов, является выяснение истинности или ложности заданной формулы на некоторой области интерпретации. Формулы, являющиеся истинными при любой интерпретации, являются *общезначимыми*, а формулы, ложные при любой интерпретации – *невыполнимыми*.

Справедлива следующая основополагающая *теорема дедукции*: «Пусть даны формулы B_1, \dots, B_n и формула A . Формула A является *логическим следствием* B_1, \dots, B_n тогда и только тогда, когда формула $B_1 \wedge B_2 \wedge \dots \wedge B_n \supset A$ общезначима». A логически следует из B_1, \dots, B_n тогда и только тогда, когда всякая интерпретация I , удовлетворяющая $B_1 \wedge \dots \wedge B_n$ удовлетворяет и A . Формулы B_1, \dots, B_n называются *посылками*, A – *заключением* логического следования и обозначают $B_1, \dots, B_n \models A$.

Итак, при использовании данного представления необходимо доказывать общезначимость формулы $B_1 \wedge B_2 \wedge \dots \wedge B_n \supset A$ или невыполнимость формулы $B_1 \wedge \dots \wedge B_n \wedge \neg A$.

Для исчисления предикатов первого порядка не существует общего метода установления общезначимости любых формул, т.е. исчисление предикатов первого порядка неразрешимо. Но если некоторая формула общезначима, то существует процедура для проверки этого факта. Наиболее известными применяемыми методами являются методы *резольюции* и *обратный метод*.

Приведем пример записи некоторого факта в виде формулы исчисления предикатов:

ДАТЬ(МИХАИЛ, ВЛАДИМИРУ, КНИГУ);

$(\exists x)(\text{ЭЛЕМЕНТ}(x, \text{СОБЫТИЕ-ДАТЬ}) \wedge \text{ИСТОЧНИК}(x, \text{МИХАИЛ})$
 $\wedge (\text{АДРЕСАТ}(x, \text{ВЛАДИМИР}) \wedge \text{ОБЪЕКТ}(x, \text{КНИГА})).$

Здесь описаны два способа записи одного факта: «Михаил дал книгу Владимиру».

Поиск решения при использовании исчисления предикатов первого порядка рассматривается как доказательство теоремы.

Рассмотрим представление знаний с использованием исчисления предикатов первого порядка на простом примере.

Пример

Никакой торговец поддержанными автомобилями не покупает автомобиль для своей семьи. Некоторые люди, которые покупают авто для своих семей, абсолютно нечестные люди. Доказать, что некоторые абсолютно нечестные люди не являются торговцами поддержанных авто.

Доказательство. Введем следующие обозначения:

предикат $P(x)$ означает что x – торговец поддержанными авто;

предикат $Q(x)$ – x купил поддержанное авто для своей семьи;

предикат $R(x)$ – x абсолютно нечестен.

Тогда имеются формулы:

$B1: \quad \{ \forall x \} (P(x) \rightarrow \neg Q(x))$

$B2: \quad \{\exists x\} (Q(x) \wedge R(x))$

Доказать, что $A: \quad \{\exists x\} (R(x) \wedge \neg P(x))$ является истинной формулой.

Предположим, что $B1$ и $B2$ истинны в интерпретации I с областью D . Так как $B2$ истинна в I , то $\{\exists x\} \in D$ такой, что $x = a$ и формула $Q(a) \wedge R(a)$ истинна в I . Следовательно, $Q(a)$ истинна в I , т.е. $\neg Q(a)$ ложна в I . Формулу $B1$ можно записать так: $\{\forall x\} (\neg P(x) \vee \neg Q(x))$. Так как $B1$ истинна в I , а $\neg Q(x)$ ложна в I , то $\neg P(a)$ должна быть истинна в I . Отсюда следует, что формула $R(a) \wedge \neg P(a)$ является истинной в I . Таким образом, формула $\{\exists x\} (R(x) \wedge \neg P(x))$, истинна в I . Следовательно, формула A является логическим следствием формул $B1$ и $B2$.

Основным достоинством использования исчисления предикатов в качестве модели представления знаний является наличие единообразной формальной процедуры доказательства теорем. Однако высокая степень единообразия влечет за собой и основной недостаток данного подхода – сложность использования при доказательстве эвристик, отражающих специфику конкретной предметной области. К другим недостатком формальных систем следует отнести их монотонность, отсутствие средств для структурирования используемых элементов и недопустимость противоречий.

Логика предикатов первого порядка используется как основа для конструирования более сложных и удобных логических методов представления знаний. В этом качестве она используется в модальных и псевдофизических логиках.

Модальные логики

Первой попыткой расширить возможности логики предикатов первого порядка явилось появление множества *модальных* логик, в которых вводились различные кванторы (модальности) и аксиомы, отражающие тот или иной аспект реального мира. Модальные понятия, или модальности, служат для оценки высказываний, данных с той или иной точки зрения. Они конкретизируют качественный характер связи, установленной в высказывании.

Наиболее известны модальные логики «возможного–необходимого» (алетическая логика), деонтическая логика (модальности «разрешено–обязательно»), эпистемическая логика (логика «знания–веры»), временная модальная логика (модальности «всегда–никогда», «часто–иногда»).

Для интерпретации модальных логик возможностей предикатов, имеющих всего два значения (двузначной семантики), было недостаточно. Поэтому, сначала появилась трехзначная логика Лукасевича (L–логика), где логические переменные могут принимать значения 0, 1, 2, а затем семантика возможных миров (четырёхзначная логика).

Нечеткая логика

Для представления нечетких понятий и оперирования с ними американский ученый Лотфи Заде в 60-х годах создал теорию нечетких множеств, а затем – нечеткую логику, базирующуюся на ней. В основе теории нечетких множеств

лежит интерпретация факта принадлежности элемента a множеству A как факта, который может быть истинным или ложным с некоторой оценкой истинности $\mu_A(a)$, пробегающей значения от 0 до 1. Оценка истинности называется *функцией принадлежности* элемента a множеству A .

Функция принадлежности интерпретируется как мера истинности, уверенности или достоверности и отражает нечеткость знаний.

Предположим, существуют следующие высказывания:

«Иванов – хороший человек» с $\mu = 0,8$,

«Политик – хороший человек» с $\mu = 0,3$.

Тогда конъюнкция этих двух высказываний (имеющая смысл как уточнение мнения об Иванове, когда стало известно, что он – политик) определяется функцией принадлежности $\mu = 0,3$, а дизъюнкция – $\mu = 0,8$.

В теории нечетких множеств функция принадлежности может интерпретироваться как субъективное представление об истинности высказываний или объективная нечеткость знаний (информации). В первом случае описание нечетких высказываний является как бы снимком состояния некоторой интеллектуальной системы, обученной на примерах взаимодействия с внешней средой или заполненной субъективными знаниями экспертов. Во втором случае нечеткость является следствием каких-либо помех при поступлении информации в систему и интерпретации ее в виде знаний. В обоих случаях функцию принадлежности можно интерпретировать как вероятностную меру истинности и применять теорию вероятности к ее обработке и анализу.

Можно развить логику нечетких высказываний до логики нечетких предикатов, которая обычно рассматривается в рамках псевдофизических логик.

Псевдофизические логики

Наиболее используемыми псевдофизическими логиками являются пространственная, временная или каузальная (причинно-следственная) логика.

На рис. 3.4 показана структура составляющих пространственной логики.



Рис. 3.4. Структура пространственной логики

Логика взаимного расположения объектов, расстояний и направлений делится на метрическую и топологическую логики. В отличие от метрической логики топологическая логика не связана с метрической шкалой.

Метрические шкалы подразделяются на *экзоцентрические* и *эндоцентрические*, относительные и абсолютные. Экзоцентрические шкалы имеют началом координат точку, связанную с самой интеллектуальной системой. Примером такой шкалы является шкала для описания лингвистической переменной «Расстояние до объекта» в логике расстояний. Ее символическими значениями могут быть следующие: «совсем рядом», «рядом», «очень-очень близко», «не очень близко», «не близко», «недалеко» и т.п.

Эндоцентрическая шкала имеет началом координат точку вне системы. Примером такой шкалы является шкала для описания лингвистической переменной «Расстояние между двумя объектами» в той же логике расстояний. Относительные шкалы имеют изменяемую точку отсчета (начало координат), а абсолютные – неизменную (т.е. явно не заданную).

Логика направлений оперирует с понятиями «справа», «слева», «вперед», «сзади» и т.п.

В логике взаимного расположения объектов описываются следующие базовые отношения: унарные – «иметь горизонтальное положение», «иметь вертикальное положение», бинарные – «находиться внутри», «находиться вне», «находиться на поверхности», «быть там же, где..», «находиться в окрестности», «быть частью», «иметь точку опоры на..», «иметь точку подвеса на..» и т.п., n -арное отношение – «быть между».

Из базовых отношений с помощью логических связок строятся производные отношения, такие как «не соприкасаться» (отрицание «соприкасаться»), «быть вместе..» (следствие от «находиться там же..»), «висеть» (конъюнкция «иметь вертикальное положение» и «висеть на...»), «стоять» (конъюнкция «иметь вертикальное положение» и «иметь точку опоры на...») и т.п.

3.4. Продукционная модель представления знаний

Определение

Под продукцией понимается выражение вида

$$(i): P; A \Rightarrow B; Q,$$

здесь i – номер продукции, с помощью которого данная продукция выделяется из множества продукций. В качестве номера (имени) может выступать некоторая лексема, отражающая суть продукции. $A \Rightarrow B$ – ядро продукции. В ядре продукции « A » и « B » являются высказываниями, к которым применимы операции отрицания, конъюнкции и дизъюнкции. Высказывание « A » – это посылка (условие, антецедент) правила. Высказывание « B » – заключение (действие, консеквент) правила. Символ \Rightarrow трактуется как знак логического следования « B » из истинного « A ». (Если « A » не истинно, то о « B » ничего сказать нельзя). P – условие применимости ядра продукции (предусловие правила). Q –

постусловие, которое описывает действия и процедуры, выполняемые после реализации ядра.

Ядро правила продукции интерпретируется по-разному. Обычное прочтение ядра выглядит так: «Если A , то B » или «Если A , то B иначе C ».

Пример

Правило 1: ЕСЛИ Образование = «Высшее» И
Возраст = «Молодой» И
Коммуникабельность = «Высокая»
ТО Шансы найти работу = «Высокие».

При срабатывании этого правила в базу данных интеллектуальной системы добавляется факт, означающий, что шансы найти работу высоки. Понятия «Образование», «Возраст», «Коммуникабельность» служат для задания условия (в данном случае, конъюнкции), при котором срабатывает правило.

Факты хранятся в базе данных продукционной системы в форме («объект», «значение») или («объект», «атрибут», «значение»).

Объект	Атрибут	Значение
Дом	Цвет	Белый
Пациент	Температура	Нормальная

Могут использоваться и другие структуры для хранения фактов.

При интерпретации (выполнении) правила в ходе проверки условия система проверяет факты, уже находящиеся в базе данных, и, если соответствующего факта нет, обращается за ним к источнику данных (пользователю, базе данных и т.д.) с вопросом (или запросом).

Кроме правил в продукционных базах знаний могут использоваться метаправила для управления логическим выводом.

Пример

Так может выглядеть метаправило для гипотетической базы знаний:

ЕСЛИ Экономика = «Развивается»
ТО Увеличить приоритет правила 1

Ядра продукции можно классифицировать различными способами. Прежде всего, различают *детерминированные* и *недетерминированные* ядра. В детерминированных ядрах при выполнении левой части правила, правая часть выполняется обязательно. В недетерминированных правилах правая часть правила может не выполняться. В этом случае интерпретация ядра может выглядеть так: «Если A , то возможно B ». Возможность определяется некоторыми оценками реализации ядра. Например, если задана вероятность выполнения B при актуализации A , то правило продукции может быть таким: «Если A , то возможно B с вероятностью P ». Оценка может быть и лингвистической, например, «Если A , то с большой долей уверенности B » или «Если A , то B с коэффициентом уверенности α ».

Детерминированные продукции могут быть однозначными и альтернативными. Во втором случае в правой части ядра продукции указывается альтернативные возможности выбора.

Пример

Правила продукции при анализе финансового состояния предприятия [15].

1. ЕСЛИ коэффициент ликвидности = «удовлетворительно»

И ликвидность баланса = «удовлетворительно»

ТО платежеспособность = «удовлетворительно»

2. ЕСЛИ коэффициент ликвидности = «неудовлетворительно»

И ликвидность баланса = «удовлетворительно»

ТО платежеспособность = «удовлетворительно»

3. ЕСЛИ коэффициент ликвидности = «удовлетворительно»

И ликвидность баланса = «неудовлетворительно»

ТО платежеспособность = «удовлетворительно»

4. ЕСЛИ коэффициент ликвидности = «неудовлетворительно»

И ликвидность баланса = «неудовлетворительно»

ТО платежеспособность = «неудовлетворительно»

Можно указать следующие достоинства представления знаний в виде продукционных правил.

1. Универсальность метода представления знаний. Это позволяет создавать проблемно-ориентированные экспертные системы.

2. Модульная организации знаний, что обеспечивает независимость правил.

3. Использование предусловий применения правил позволяет использовать наиболее рациональную стратегию вывода, существенно сокращая перебор правил.

Продукционные модели имеют, по крайней мере, два серьезных недостатка. При большом числе продукции становится сложной проверка непротиворечивости системы продукции. Поэтому при добавлении новых продукции много времени тратится на проверку. Системе присуща недетерминированность (неоднозначность выбора выполняемой продукции из активизируемых продукции), поэтому возникают трудности при проверке корректности работы системы.

Продукционное представление знаний соединяет в себе свойства как декларативного, так и процедурного представлений. Эта модель знаний в настоящее время наиболее часто применяется в промышленных экспертных системах. Она привлекает разработчиков своей наглядностью, высокой модульностью, легкостью внесения изменений и дополнений, а также простотой механизма логического вывода.

3.5. Семантические сети

Эвристические модели представления знаний в отличие от логических моделей предоставляют более широкие возможности для описания сложных структур. Это достигается выделением и включением в модель в явной форме всех отношений, которые образуют информационную структуру, а также семантики этих отношений.

Термин семантическая означает «смысловая», а сама семантика – это наука, определяющая смысл знаков.

Определение

Семантическая сеть – это ориентированный граф, в котором вершины соответствуют объектам и понятиям, а дуги отражают семантические отношения между вершинами.

Семантическую сеть можно построить для любой предметной области и для самых разнообразных объектов и отношений.

В узлах семантических сетей могут помещаться три типа объектов: понятия, события, свойства. *Понятия* (обычно это существительные) – сведения об абстрактных или физических объектах проблемной области. *События* (обычно это глаголы) – действия, происходящие в реальном мире. Для них указываются тип действия и роль, которую играют объекты в этом действии. *Свойства* (прилагательные, наречия, определения) используются для уточнения понятий, событий или других свойств. Применительно к понятиям свойства описывают их характеристики (цвет, качество и т.д.). Применительно к событиям: место, время и т.д.

Пример

На рис. 3.5 представлена семантическая сеть для предложения «студент Иванов добросовестно изучает теоретические вопросы перед сдачей экзамена».

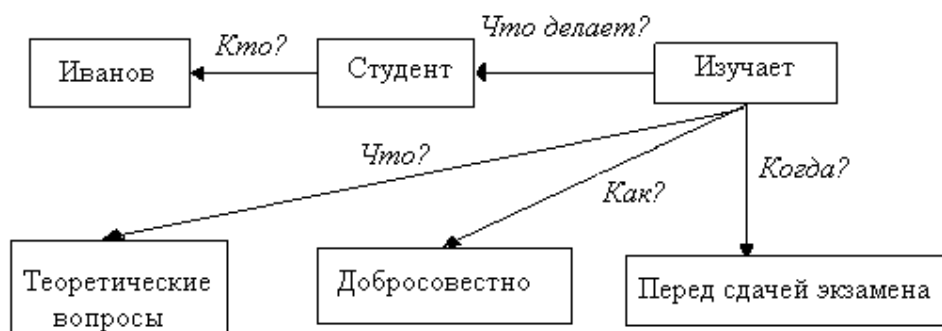


Рис. 3.5. Пример семантической сети

Семантические отношения условно делятся на четыре класса: лингвистические, логические, теоретико-множественные и квантифицированные. К наиболее распространенным лингвистическим отношениям относятся падежные и атрибутивные отношения.

Все связи понятий, событий и свойств с действиями называют *падежами*, или падежными отношениями. Основные падежи приведены в таблице 3.1.

Таблица 3.1.

Падеж	Отношение, определяющее связь действия с:
агент	– предметом, являющимся инициатором действия
объект	– предметом, подвергающимся действию
источник	– размещением предмета перед действием
приемник	– размещением предмета после действия
время	– моментом выполнения действия
место	– местом проведения действия
цель	– действием другого события

Атрибутивные отношения – это отношения между объектом и свойством, например, цвет, размер, форма, модификация и т.д. На рис. 3.6 приведен пример семантической сети с использованием атрибутивных отношений.

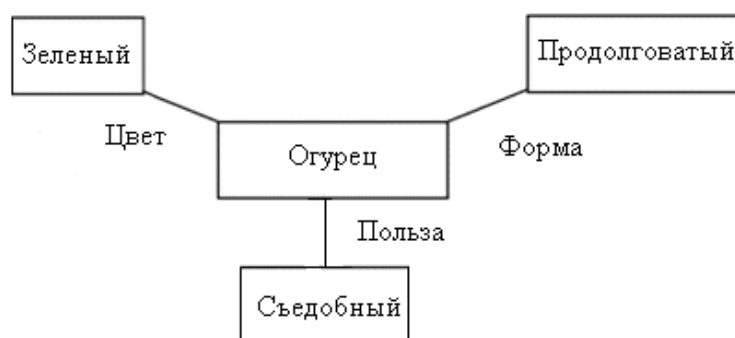


Рис. 3.6. Пример атрибутивных отношений

Логические отношения – это операции, используемые в исчислении высказываний: дизъюнкция, конъюнкция, импликация, отрицание.

Теоретико-множественные отношения – это отношение подмножества SUB и супермножества SUP, элемент множества, отношение части и целого, отношение множества и элемента (is-a, part-of). Этот класс отношений не обладает высокой степенью интеллектуальности, но позволяет строить иерархические структуры. В таких структурах все свойства SUB-понятий автоматически присваиваются SUP-понятиям. Этот тип отношений иллюстрирует рис. 3.7.

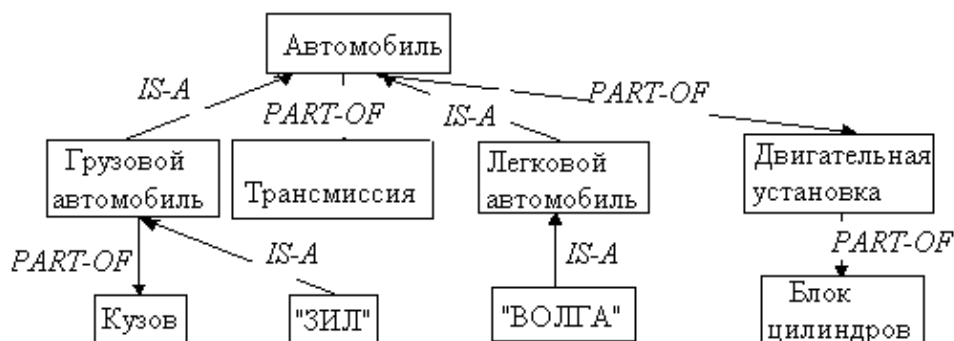


Рис. 3.7. Теоретико-множественные отношения

Квантифицированные отношения используются для представления знаний декларативного типа. Кванторы – это логические операторы, переводящие одну высказывательную форму в другую и позволяющие указывать объем тех значений предметных переменных, для которых данная высказывательная форма истинна.

Основой для определения любого понятия является множество его отношений с другими понятиями. Обязательными отношениями являются:

- класс, которому принадлежит данное понятие;
- свойства, выделяющие понятие из всех понятий данного класса;
- примеры (экземпляры) данного понятия.

При представлении событий предварительно выделяются простые отношения, которые характеризуют основные компоненты события. В первую очередь из события выделяется действие, которое определяется глаголом. Затем определяются объекты, которые действуют, и объекты, над которыми выполняются действия.

Сети различного вида получаются в зависимости от того, какие ограничения накладываются на вершины и дуги. Если вершины сети не обладают внутренней структурой, то такие сети называют *простыми* сетями. Если вершины обладают внутренней структурой, то такие сети называют *иерархическими*. Одно из основных отличий иерархической сети от простой сети состоит в возможности разделять сеть на подсети и устанавливать отношения не только между вершинами, но и между подсетями. *Динамические* семантические сети – это сети с событиями.

Можно ввести несколько классификаций семантических сетей, например, по количеству типов отношений:

- однородные (с единственным типом отношений);
- неоднородные (с различными типами отношений).

По типам отношений:

- бинарные (в которых отношения связывают два объекта);
- n-арные (в которых есть специальные отношения, связывающие более двух понятий).

Рассмотрим формальное определение семантической сети.

Определение

Дано конечное множество символов: $A = \{A_1, \dots, A_n\}$, называемых атрибутами. Схемой или *интенционалом* некоторого отношения R_i в атрибутивном формате называют набор пар $INT(R_i) = \{\dots A_j, Dom(A_j) \dots\}$, где R_i – имя отношения; $A_j \in A$, $j = 1, \dots, n$ – атрибуты отношения R_i , n_i – целое положительное число, его арность; $Dom(A_j)$ – множество значений атрибута A_j отношения R_i , т.е. домен A_j .

Определение

Домен – это совокупность значений некоторой информационной единицы, помещенная в БД и маркируемая своим атрибутом. Объединение всех доменов

это – набор объектов, на которых задаются отношения R_i ; m – число различных отношений.

Определение

Экстенсионалом отношения R_i называют множество $EXT(R_i) = \{...F_k..., I, k=1, ..., p_i\}$, где p_i – кардинальность множества $EXT(R_i)$; F_k – факты отношения R_i , записываемые в виде:

$$F_k = (R_i... A_j \in V_{ijk}, Dom(A_j)...),$$

V_{ijk} – значение j -го атрибута k -го факта экстенсионала отношения R_i . Последовательность из двух элементов вида «атрибут-значение» называют атрибутивной парой. Порядок записи атрибутивных пар и фактов произвольный. Все факты и атрибутивные пары внутри каждого факта попарно различны. Тогда семантическая связь – это совокупность

$$\{... \langle INT(R_i), EXT(R_i) \rangle \}... \text{ (для } i = 1, ..., m),$$

записываемая в виде ассоциативной структуры данных.

В семантической сети используются различные типы отношений, но свойство ассоциативности, т.е. группировки информации вокруг фактов, атрибутов и/или объектов, является для нее характерным. Экстенсиональные семантические сети соответствуют базе данных, интенциональные – базе знаний. Их объединение образует систему представления данных и знаний в интеллектуальной информационной системе.

Пример

В качестве примера рассмотрим представление знаний, содержащееся в высказывании «Поставщик N отгрузил товар со склада М автотранспортом [1]. На рис. 3.8 представлена интенциональная, а на рис. 3.9 – экстенциональная семантическая сеть. Факты обозначены овалом, а понятия и объекты прямоугольником.



Рис. 3.8. Интенциональная семантическая сеть

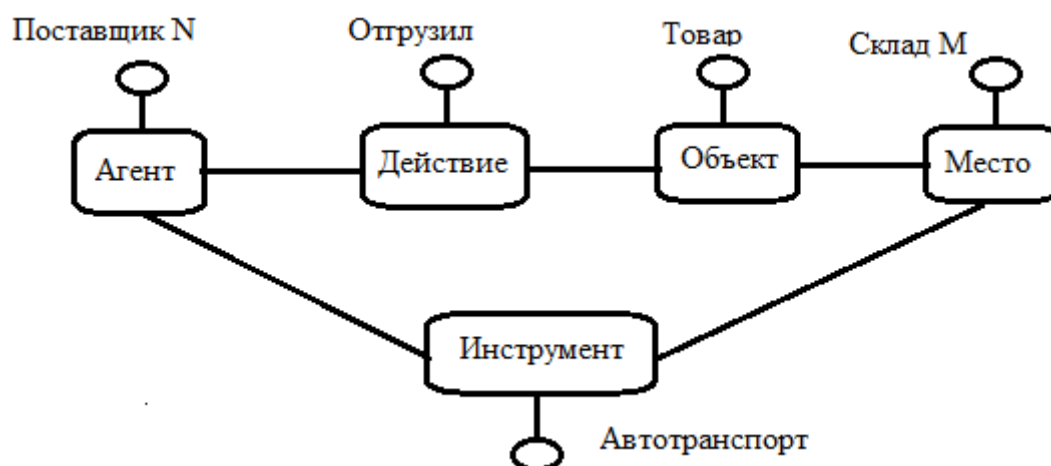


Рис. 3.9. Экстенциональная семантическая сеть

Данная модель представления знаний была предложена американским психологом Куиллианом. Основным ее преимуществом является то, что она более других соответствует современным представлениям об организации долговременной памяти человека. Теория и технология семантических сетей позднее получила развитие в виде диаграмм «сущность–связь», диаграммы потоков данных в рамках CASE-технологии проектирования баз данных.

Достоинством семантических сетей является их универсальность, достигаемая за счет выбора соответствующего применения набора отношений. В принципе с помощью семантической сети можно описать сколь угодно сложную ситуацию, факт или предметную область.

Недостатки представления знаний в виде семантических сетей вытекают из их достоинств:

- имеют произвольную структуру, что усложняет их построение и модификацию;
- разнообразие типов вершин и связей между ними требует разнообразия процедур их обработки.

3.6. Фреймовые модели представления знаний

Для представления и описания стереотипных объектов, событий или ситуаций введены понятия «фрейма». Это одна из наиболее психологически обоснованных и практически ценных моделей, которая была предложена Марвином Мински. В 1974 г. вышла книга «Фреймы для представления знаний», в которой были обобщены исследования как в области психологии, так и компьютерном моделировании психики.

В психологии известно понятие абстрактного образа. Например, слово «комната» порождает образ комнаты. Это – «жилое помещение с четырьмя стенами, полом и потолком, окнами и дверью, определенной площади». Из этого описания ничего нельзя убрать, но в нем есть незаполненные значения некоторых атрибутов – например, площадь комнаты, количество окон, высота потолка и др. Незаполненные значения атрибутов называются *слотами*. Они

заполняются в процессе активизации фрейма в соответствии с определенными условиями.

В теории фреймов образ комнаты называется *фреймом комнаты*. Фреймом также называется и формализованная модель для отображения образа.

Можно сказать, что фрейм – это *абстрактный образ* для представления некоего стереотипа восприятия.

Модель фрейма является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о мире. В моделях используются:

- фреймы-структуры, применяемые для обозначения объектов и понятий (заем, залог, вексель);
- фреймы-роли (менеджер, кассир, клиент);
- фреймы-сценарии (банкротство, собрание акционеров, празднование именин);
- фреймы-ситуации (тревога, авария, рабочий режим) и др.

Определение

Формально фрейм можно представить в виде следующей конструкции:

$$F = [<r_1, v_1>, <r_2, v_2>, \dots, <r_n, v_n>],$$

где F – имя фрейма; r_i – имя слота; v_i – значение слота.

Слотом называется элемент данных для фиксации знаний об объекте, которому отведен данный фрейм.

Имя слота является уникальным для фрейма, которому он принадлежит;

Существуют два способа организации фреймовых знаний: целевые структуры и сценарии.

Целевые структуры – это ролевые фреймы, в которых в качестве имен слотов выступают вопросительные слова, ответы на которые являются значениями слотов. Например, ролевой фрейм деловая поездка = [$<\text{кто}, x>$, $<\text{куда}, y>$, $<\text{когда}, z>$, $<\text{цель}, w>$, $<\text{с кем}, t>$, $<\text{вид транспорта}, r>$]. Среди ролей выделяются *главные роли*, заполнение которых обязательно.

Большое значение имеют стереотипные знания, которые описывают знания о стандартных ситуациях реального мира. Такие знания называют понятием. Система понятий рассматривается как неформальное знание о проблемной области. В противоположность этому фрейм – формализованное знание о стереотипной ситуации. Для планирования действий в таких ситуациях используются фреймы–сценарии.

Сценарий – это формализованное описание стандартной последовательности взаимосвязанных фактов, определяющих типичную ситуацию в проблемной области. Это может быть последовательность действий или процедур, описывающих способы достижения цели. Сценарий представляется некоторой сетью, вершины которой соответствуют фактам, а дугам – связи, описывающие отношений специального типа.

Сценарий должен содержать:

1. Цель, которой должны удовлетворять все действия сценария.

2. Предварительные условия, которые должны быть удовлетворены перед применением сценария.

3. Заключительные условия, характеризующие ситуацию после завершения применения сценария.

В качестве примера рассмотрим сценарий посещения ресторана [7].

Пример

Цель: поесть без самостоятельного приготовления пищи.

Предварительные условия: голоден, есть деньги, ресторан работает.

Состояние после завершения: сыт, денег стало меньше.

Действие первое

1. Войти в ресторан.
2. Сдать верхнюю одежду в гардероб, если она есть.
3. Найти место самостоятельно, либо с помощью метрдотеля.

Действие второе

1. Просмотреть меню.
2. Сделать заказ.
3. Поесть.

Действие третье

1. Получить чек.
 2. Заплатить официанту или кассиру.
 3. Покинуть ресторан.
-

При реализации фреймового представления знаний различают *фреймы-образцы* или *прототипы*, хранящиеся в базе знаний, и *фреймы-экземпляры*, которые создаются для отображения реальных фактических ситуаций на основе поступающих данных. Фрейм-прототип представляет собой шаблон, в котором позиции, соответствующие значениям слотов, не заполнены.

Пример

Фрейм-прототип:

Имя фрейма: руководитель

Является: экономист

Имя: _____

Возраст: _____

Адрес: _____

Слот «является» служит для построения иерархии фреймов. Его значение указывает, что фреймом более высокого уровня служит фрейм «экономист». Следовательно, фрейм «руководитель» наследует все свойства фрейма «экономист».

На месте значений слотов *фрейма-прототипа* могут быть указаны условия заполнения слотов.

Пример

Фрейм-прототип с условиями заполнения слотов:

Имя фрейма: руководитель

Является: экономист
Имя: агрегат (фамилия, имя)
Возраст: агрегат (годы)
Адрес: адрес.

Значением слота может быть что угодно: числа или математические зависимости, тексты на естественном языке, программы, ссылки на другие слоты данного фрейма или слоты других фреймов. В качестве значения могут выступать слоты более низкого уровня. Это позволяет организовать иерархическую подчиненность фреймов.

Пример

Во *фрейме-экземпляре* значения слотов заполнены.

Имя фрейма: VIP1

Является: экономист

Имя: Петров И.И.

Возраст: 40

Адрес: адрес_1 (указатель на экземпляр фрейма).

Существует несколько способов получения слотом значений во фрейме-экземпляре:

- по умолчанию от фрейма-образца (*Default*–значение);
- через наследование свойств от вышележащего фрейма;
- по формуле, указанной в слоте;
- явно из диалога с пользователем;
- из базы данных.

С каждым слотом можно связать процедуру, которая выполняется автоматически при изменении значений слотов. Это процедуры следующих типов.

- *IF ADDED* – запускается при подстановке в слот значения;
- *IF NEEDED* – вызывается, когда требуется значение слота, но он пуст;
- *IF REMOVED* – выполняется при удалении информации из слота.

Процедуры, запускающие операции по добавлению, вычислению или передаче значений слотов при обращении к слотам, называются *демонами*. *Процедуры-слуги* определяют поведение всего объекта в некотором состоянии или при некоторых внешних условиях и запускаются по запросу.

Еще один пример фрейма-прототипа, описывающего земельный участок в виде многоугольника [10].

Пример

Имя фрейма: Земельный участок

Количество сторон: (4)

Длины сторон:

Размеры углов:

Площадь: *IF_NEEDED*: Вычисление площади

Цена:

Здесь слот «*Количество сторон*» имеет значение «по умолчанию», равное 4, т.к. подавляющее большинство земельных участков имеет форму четырехугольника.

К слоту «*Площадь*» присоединены процедуры-демоны вычисляющие площадь и цену, соответственно, запускаемые при запросе слота (событие «*IF_NEEDED*») и добавлении значения слота «*Площадь*» (событие «*IF_ADDED*»).

Указатель наследования, присоединенный к слоту, показывает, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты с теми же именами во фрейме нижнего уровня. Типичные указатели наследования следующие:

- *S* – слот наследуется с теми же значениями данных.
- *U* – слот наследуется, но данные в каждом фрейме могут принимать любое значение.
- *I* – независимый, слот не наследуется.

На практике наибольшее распространение модель, основанная на фреймах, получила в объектно-ориентированном программировании. Современное понятие объекта достаточно точно соответствует понятию фрейма. Основными свойствами фреймов, используемыми в современных языках, являются: инкапсуляция, наследование и полиморфизм объектов.

Инкапсуляция – единство данных и методов в рамках объекта.

Наследование – способность объекта пользоваться методами и данными, определенными в рамках одного из его предков.

Полиморфизм – способность объекта в разные моменты времени вести себя по-разному, то как объект своего типа, то как объект типа любого из предков.

Достоинства фреймового представления знаний:

1. Фреймы – единственная глубинная модель, хорошо обоснованная теоретически. Ее практическая реализация – построение микромиров – позволяет сосредоточиться на свойствах и поведении участников микромира, предоставив им самим организовывать взаимодействие на основе правил, привычек, сценариев.

2. Фреймы обеспечивают требования структурированности и связности знаний. Это достигается за счет свойств наследования значений и вложенности фреймов.

3. Фрейм реализуется как независимый модуль, что облегчает модификацию системы фреймов.

Для реализации фреймового представления знаний используются специальные языки, например, язык FRL (Frame Representation Language). Приведем пример записи фрейма на этом языке.

Пример

Фрейм СТОЛ может быть записан в виде трех слотов: слот НАЗНАЧЕНИЕ (purpose), слот ТИП (type) и слот ЦВЕТ (colour) следующим образом:

(frame СТОЛ

(purpose (value (размещение предметов для деятельности рук)))

(type (value (письменный)))

(colour (value (коричневый))))

Во *фрейме* СТОЛ представлены только **ДЕКЛАРАТИВНЫЕ** средства для описания объекта, и это *фрейм-образец*. Для построения *фрейма-экземпляра*, отображающего фактическую ситуацию, нужны данные, поступающие при запуске процедур, связанных с фреймом.

Слот *is-a* или *АКО* (*A Kind Of*) определяет иерархию *фреймов* в сети *фреймов*. Такая связь обеспечивает наследование свойств. Слот *is-a* указывает на *фрейм* более высокого уровня, откуда неявно наследуются свойства аналогичных слотов.

Приведем еще один пример (рис. 3.9).

Пример

Рассмотрим фрагмент описания из «мира блоков» в виде *фреймов* [13].

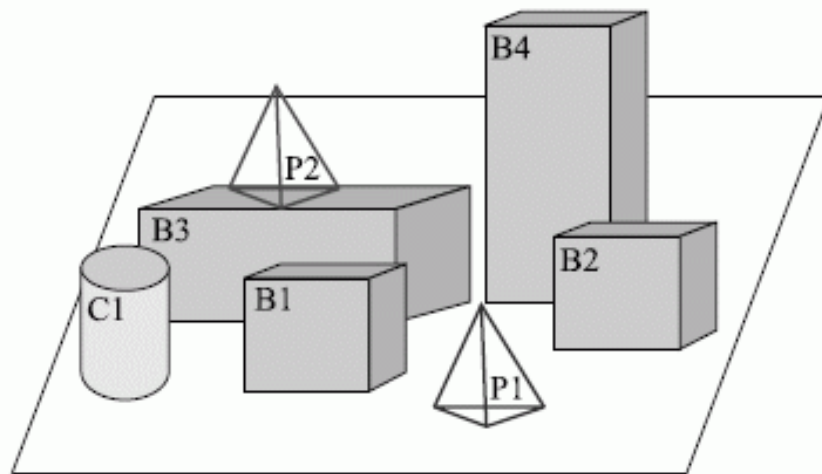


Рис. 3.9. Мир блоков

(frame (name (Cube)) // имя фрейма

(isa (Block World))

(length (NULL))

(width (IF-DEFAULT (use length)))

(height (IF-DEFAULT (use length))))

По умолчанию значение ширины (width) и высоты (height) объекта Cube равно его длине (length).

(frame (name (B1)) // имя объекта B1

```

(isa (Cube))
(color (red))          // цвет объекта красный
(length (80)))         // длина объекта 80
(frame (name (B2))     // имя объекта B2
(isa (Cube))
(color (green))        // цвет объекта зеленый
(length (65))          // длина объекта 65

```

```

(who_put (value (NULL))
(IF_NEEDED (askuser))))

```

Слот *isa* указывает на то, что объекты *B1* и *B2* являются подтипом объекта *Cube* и наследуют его свойства, а именно, *length = width = height*. Процедура-демон *IF_NEEDED* запускается автоматически, если понадобится узнать, кто поставил *B2* на стол. Полученный ответ будет подставлен в значение слота *who_put*. Аналогично работают процедуры-демоны *IF-ADDED* и *IF-REMOVED*.

Допустим, роботу дается приказ «Возьми желтый предмет, который поддерживает пирамиду». На языке представления знаний вопрос записывается так:

```

(object ? X
(color (yellow))
(hold ? Y
(type (pyramid))))

```

Пусть, например, желтый цвет имеет кубик *B3*. Программа сопоставления с образцом находит в базе знаний описание объектов:

```

(frame (name (B3))
(type (block))
(color (yellow))
(size (20 20 20))
(coordinate (20 50 0))
(hold (P2)))
(frame (name (P2))
(type (pyramid))
...)

```

Ответ получен $X = B3$, $Y = P2$ и роботу выдается команда *take(object=B3)*.

Таков общий механизм представления знаний в виде *фреймов*.

3.7. Выводы

1. Данные – это информация, полученная в результате наблюдений или измерений отдельных свойств (атрибутов), характеризующих объекты, процессы и явления предметной области. Знания – это связи и закономерности предметной области (принципы, модели, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в данной области.

2. Свойства знаний, отличающих их от данных: внутренняя интерпретируемость; структурированность; связность; наличие семантической метрики; внутренняя активность.

3. Метод представления (модель) знаний – совокупность средств структурирования и обработки единиц знаний. Декларативное представление знаний основано на использовании универсального множества процедур, обрабатывающих факты любого типа, и на множестве специфических фактов, описывающих проблемную область. Процедурное представление исходит из того, что интеллектуальная деятельность связана с действиями по обработке фактов.

4. Модульные модели представления знаний отражают поверхностные знания о предметной области.

5. В основе формально–логических моделей лежит исчисление предикатов первого порядка. Область определения предиката задается либо перечислением фактов, либо в виде правил. Основной задачей, решаемой в рамках исчисления предикатов, является выяснение истинности или ложности заданной формулы на некоторой области интерпретации. Исчисление предикатов первого порядка в промышленных системах не используется, так как предъявляет очень высокие требования и ограничения к предметной области.

6. Кроме логики предикатов в интеллектуальных системах используются модальные и псевдофизические логики, которые расширяют возможности классической логики и позволяют учесть особенности предметной области.

7. В интеллектуальных системах, основанных на правилах, знания о решении задач представляют в виде правил «если..., то...». Факты – это значения переменных, операции над фактами – это правила. Этот подход, являясь одним из старейших методов представления знаний о предметной области, наиболее широко применяется в коммерческих экспертных системах.

8. Семантическая сеть отражает как объектное, так и операционное знание в виде двухместных или n-местных предикатов. Предикаты – это разнообразные отношения между объектами предметной области.

9. Фреймовая модель – это семантическая сеть с N-арными отношениями и присоединенными процедурами. Используются механизмы наследования свойств по иерархии классов объектов и вызова процедур в зависимости от происходящих событий.

3.8. Вопросы для самоконтроля

1. Укажите отличия знаний от данных.
2. Как понимать свойство внутренней интерпретируемости знаний? Приведите пример. Обладают ли этим свойством данные?
3. Что понимается под структурированностью знаний? Приведите примеры для каждого типа отношений, определяющих структурированность знаний.
4. Семантическая метрика позволяет выделять типовые ситуации и находить знания, близкие по смыслу. Приведите пример, и поясните, как вы понимаете это свойство знаний.
5. Укажите отличия поверхностных знаний от глубинных.
6. Укажите отличия декларативного представления знаний от процедурного способа представления.
7. Укажите основные отличия сетевых моделей знаний от модульных.
8. Дайте определение логической модели представления знаний.
9. Поясните, в чем особенность интерпретируемых знаний.
10. Определите достоинства и недостатки использования для представления знаний логики предикатов первого порядка.
11. Дайте определение модальных логик.
12. Перечислите типы псевдофизических логик. Для каких задач они используются?
13. Перечислите достоинства и особенности продукционных моделей знаний.
14. Дайте сравнительную характеристику логической и продукционной моделей знаний.
15. Дайте определение семантической сети.
16. Что называется экстенсионалом и интенсионалом отношения? Как они определяются?
17. Какие типы объектов применяются в семантических сетях? Охарактеризуйте их.
18. Перечислите типы отношений, используемых в семантических сетях.
19. Можно ли в семантической сети отобразить действия, совершаемые объектами? Если да, то как это происходит?
20. Для чего используются падежные отношения?
21. Чем иерархические семантические сети отличаются от простых сетей?
22. Сформулируйте основные особенности семантических сетей.
23. Сформулируйте основные особенности фреймовой модели представления знаний. Укажите ее отличия от модели знаний в виде семантической сети.
24. Какие типы фреймов применяются для представления знаний?
25. Что называется слотом? Для чего он используется и как получает свое значение?
26. Какие типы данных можно хранить в слоте? Чем определяется тип данных, подлежащий хранению в слоте?
27. Какие типы процедур используются во фреймах при означивании слотов?

4. МЕТОДЫ ПОИСКА РЕШЕНИЯ

4.1. Логический вывод в продукционных системах

Механизм логического вывода – неотъемлемая часть системы, основанной на знаниях, реализующий функции вывода умозаключений на основе информации из базы знаний и рабочей памяти.

Как следует из определения, для работы механизма логического вывода необходима как «долговременная» информация, содержащаяся в базе знаний в выбранном формализме, так и оперативная информация, поступающая в рабочую память после обработки в лингвистическом процессоре запроса пользователя.

Перед рассмотрением конкретных механизмов поиска решения подчеркнем несколько важных обстоятельств:

- единого механизма поиска решения для произвольных систем, основанных на знаниях, не существует;
- механизм поиска решения полностью определяется моделью представления знаний, принятой в данной системе;
- существующие механизмы поиска решения не являются строго фиксированными для каждого типа систем, основанных на знаниях.

Методы решения задач, используемые в системах, основанных на знаниях, зависят от особенностей проблемной области и от требований, предъявляемых к их решению

Особенности проблемной области с точки зрения методов решения можно характеризовать следующими параметрами:

- размер пространства, в котором ищется решение;
- степень изменяемости объекта во времени и пространстве;
- полнота модели, описывающей область;
- определенность данных о задаче.

Требования пользователя к решению характеризуются количеством решений и способом их получения.

Существующие методы решения задач можно классифицировать следующим образом.

1. Методы поиска в одном пространстве – это методы, предназначенные для использования в малых статических областях, использующие точные и полные данные.

2. Методы поиска в иерархических пространствах – это методы, предназначенные для работы в больших статических областях.

3. Методы поиска при неточных и неполных данных.

4. Методы поиска в динамических областях.
5. Методы поиска, использующие несколько моделей.

4.1.1. Поиск решения в одном пространстве

Представление задачи в пространстве состояний

Решение многих задач в интеллектуальных системах можно определить как проблему поиска решения, где искомое решение – это цель поиска, а множество возможных путей достижения цели представляет собой пространство поиска (или пространство состояний). Поиск решений в пространстве состояний состоит в определении последовательности операторов, которые преобразуют начальное состояние в конечное.

Определим представление задачи в пространстве состояний формально.

Определение

Пространство состояний представляется кортежем (S, F, T) со следующими обозначениями: S – множество начальных состояний; F – множество операторов, отображающих одни состояния в другие; T – множество целевых состояний.

Решить задачу – значит определить такую последовательность операторов, которая преобразует начальные состояния в конечные. Процесс решения можно представить в виде графа $G = (X, Y)$, где $X = \{x_0, x_1, \dots\}$ – множество вершин графа, каждая из которых отождествляется с одним из состояний, Y – множество, содержащее пары вершин (x_i, x_j) , $x_i, x_j \in X$. Наличие пары (x_i, x_j) свидетельствует о существовании некоторого оператора $f \in F$, преобразующего состояние, соответствующее вершине x_i , в состояние x_j .

Итак, граф G задает пространство состояний. Построение пространства осуществляется с помощью следующего процесса. Берется некая вершина из $x_0 \in X$, к ней применяются все возможные операторы, порождающие все дочерние вершины. Порождение всех дочерних вершин для некоторой вершины x_i называется *процессом раскрытия вершин*. Целевая вершина не раскрывается. Процесс построения пространства состояний заканчивается, когда все нераскрытые вершины являются целевыми или терминальными (т.е. вершинами, к которым нельзя применить никаких операторов).

Для иллюстрации поиска в пространстве состояний рассмотрим следующую задачу.

Пример

Пусть перед интеллектуальным роботом поставлена задача по переупорядочиванию кубиков, поставленных друг на друга (рис. 4.1).

Условия задачи.

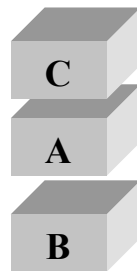
1. На каждом шаге можно переставлять только один кубик.
2. Кубик можно взять только тогда, когда его верхняя грань свободна.
3. Кубик можно поставить либо на стол, либо на другой кубик.

Задачу можно представить как задачу выбора среди множества возможных альтернатив.

В исходной ситуации альтернатива одна – поставить кубик *C* на стол. После этого имеется три альтернативы:

1. Поставить кубик *A* на стол;
2. Поставить кубик *A* на кубик *C*;
3. Поставить кубик *C* на кубик *A*.

Исходное состояние



Конечное состояние

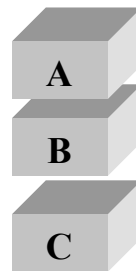


Рис. 4.1. Исходное и конечное состояния кубиков

Частичное пространство состояний этой задачи показано на рис. 4.2.

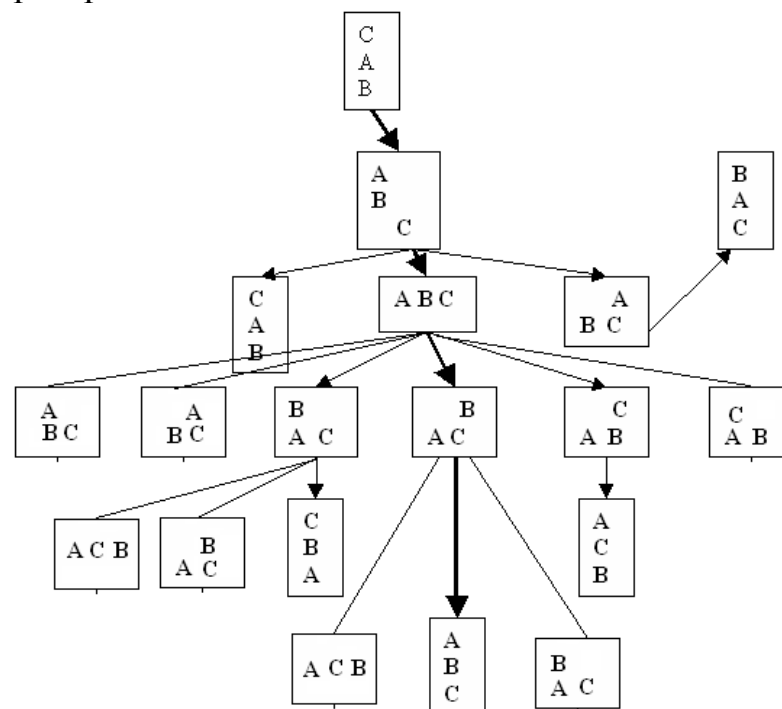


Рис. 4.2. Частичное пространство состояний решения задачи

Путь, ведущий к решению задачи, на рис. 4.2 выделен. Состояниями в пространстве являются все возможные конфигурации положений кубиков *A*, *B* и *C*, которые могут возникнуть в процессе перестановки кубиков. Дуги определяются допустимыми перестановками кубиков, целевое состояние определяется условием задачи.

Представление задачи в пространстве задач

При планировании в пространстве задач ситуация несколько иная. Пространство образуется в результате введения на множестве задач отношения типа: «часть – целое», «задача – подзадача», «общий случай – частный случай» и т. п. Другими словами, пространство задач отражает декомпозицию задач на подзадачи (цели на подцели). Поиск решения задачи методом редукции можно представить в виде И/ИЛИ графа. На рис. 4.3 [15] приведен пример И/ИЛИ графа.



Рис. 4.3. Пример И/ИЛИ графа

Чтобы представить различные отношения, на графе И/ИЛИ различают узлы «И» и «ИЛИ». Вершины, связанные операцией И, соединяются на графе дугой, вершины ИЛИ дугой не соединяются. Узлы И означают декомпозицию задачи. Узел ИЛИ определяет точку выбора между альтернативными путями решения задачи или стратегиями. Нахождение пути к цели вдоль любой из ветвей является достаточным условием для решения общей задачи.

Каждой вершине ставится в соответствие описание некоторой задачи. Вершина типа И со своими дочерними вершинами интерпретируется так: решение задачи сводится к решению всех ее подзадач, соответствующих дочерним вершинам. Вершины типа ИЛИ вместе со своими дочерними вершинами интерпретируется так: решение задачи сводится к решению любой из ее подзадач, соответствующих дочерним вершинам.

Например, задачу S_0 (рис. 4.4) можно решить либо путем решения задачи S_3 либо путем решения S_1 и S_2 .

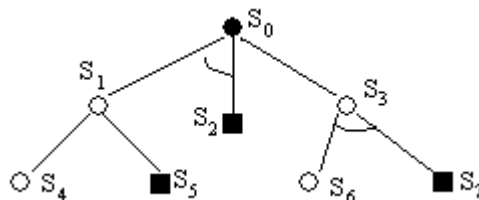


Рис. 4.4. Граф для решения задачи S_0

На рис. 4.4 тип вершины S_0 определен не однозначно. В связи с тем, что каждая вершина графа должна относиться только к одному типу, вводим дополнительную вершину R_1 (рис. 4.5).

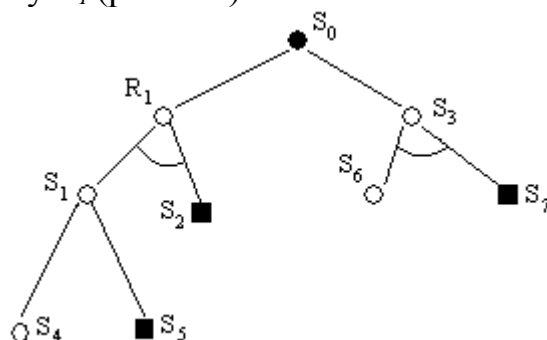


Рис. 4.5. Преобразованный граф

Цель поиска – показать, что начальная вершина разрешима, т.е. для этой вершины существует решающий граф.

Разрешимая вершина в И/ИЛИ графе определяется рекурсивно: 1) конечные вершины разрешимы, т.к. их решение известно по исходному предположению; 2) вершина ИЛИ разрешима тогда и только тогда, когда разрешима по крайней мере одна из ее дочерних вершин; 3) вершина И разрешима тогда и только тогда, когда разрешима каждая из ее дочерних вершин. Итак, *решающий граф* определяется как подграф, состоящий из разрешимых вершин, и показывает, что начальная вершина разрешима.

4.1.2. Стратегии неинформированного поиска

Термин неинформированный поиск (или слепой поиск) означает, что в данных стратегиях не используется дополнительная информация, кроме той, которая представлена в постановке задачи. Все, на что они способны, – вырабатывать преемников и отличать целевое состояние от нецелевого. Все стратегии поиска различаются тем, в каком порядке происходит развертывание узлов.

Поиск на графе, как в пространстве состояний, так и в пространстве подзадач, можно вести в двух направлениях: от исходных данных задачи к цели и в обратном направлении от цели к исходным данным.

При *поиске на основе данных* (прямая цепочка рассуждений) процесс решения задачи начинается с анализа условия (антецедента) правила.

При *поиске на основе цели* (обратная цепочка рассуждений) процесс решения задачи начинается с анализа заключения (консеквента) правила.

Таким образом, *поиск на основе данных* начинается с условий задачи и выполняется путем применения правил для получения новых фактов, ведущих к цели. *Поиск от цели* начинается с обращения к цели и продолжается путем определения правил, которые могут привести к цели, и построения цепочки подцелей, ведущей к исходным данным задачи.

Возможен и двунаправленный поиск решения, в основе которого лежит идея о том, что можно одновременно проводить поиск в прямом направлении от

начального состояния, и в обратном направлении, от цели. Завершается такой поиск тогда, когда два процесса поиска встречаются.

Процесс *поиска от цели* рекомендован в следующих случаях.

1. Цель поиска (или гипотеза) явно присутствует в постановке задачи или ее можно легко сформулировать. Многие диагностические системы рассматривают возможные диагнозы, систематически подтверждая или отвергая некоторые из них способом поиска от цели.

2. Имеется большое число правил, которые на основе полученных фактов позволяют производить возрастающее число заключений или целей. Своевременный отбор целей позволяет отсеять множество возможных ветвей, что делает процесс поиска в пространстве состояний более эффективным.

3. Исходные данные не приводятся в задаче, но подразумевается, что они известны решателю. В этом случае поиск от цели может служить руководством для правильной постановки задачи. Например, в программе медицинской диагностики имеются всевозможные диагностические тесты. Доктор выбирает из них только те, которые позволяют подтвердить или опровергнуть конкретную гипотезу о состоянии пациента.

Таким образом, при поиске от цели очередное выбранное правило исключает неперспективные ветви поиска.

Поиск на *основе данных* применим в следующих случаях.

1. Все или большинство исходных данных задано в постановке задачи. Задача интерпретации состоит в выборе этих данных и их представлении в виде, подходящем для использования в интерпретирующих системах более высокого уровня. На стратегии поиска от данных основаны системы анализа данных.

2. Существует большое число потенциальных целей, но всего лишь несколько способов применения фактов и представления информации о конкретном примере задачи.

3. Сформировать цели и гипотезы трудно

При поиске на основе данных знания и ограничения, заложенные в исходной постановке задачи, используются для нахождения пути к решению.

При решении задач путем поиска в пространстве состояний решатель должен переходить от вершины к вершине на графе и рассматривать различные пути до тех пор, пока не достигнет цели. Методом систематической проверки различных путей *является поиск с возвратами*.

Алгоритм запускается из начального состояния и следует по некоторому пути до тех пор, пока не достигнет цели либо не упрется в тупик. Если цель достигнута, поиск завершается, в качестве решения задачи возвращается путь к цели. Если же поиск привел в тупиковую ситуацию, то алгоритм возвращается в ближайшую из пройденных вершин и исследует все ее вершины-братья, а затем запускается по одной из ветвей, ведущих от вершины-брата. Алгоритм работает до тех пор, пока не достигнет цели, либо не исследует все пространство поиска [10].

В алгоритме используется три списка, позволяющих запоминать путь от узла к узлу на графе.

SL (State List) – список исследованных вершин рассматриваемого пути. Если цель найдена, то SL содержит список вершин на пути к решению.

NSL (New State List) – список новых состояний, он содержит вершины, подлежащие рассмотрению, т.е. список вершин, потомки которых еще не были порождены и рассмотрены.

DE (Dead List) – список тупиков, т.е. список вершин, потомки которых уже были рассмотрены, но не привели к цели. Если на очередном шаге встречается вершина, помещенная в этот список, то она исключается из рассмотрения.

При описании поиска с возвратами необходимо учитывать возможность повторного появления состояний. Чтобы избежать их повторного рассмотрения, каждая вновь порожденная вершина проверяется на вхождение в один из трех списков. Вершина, попавшая в один из списков, исключается из рассмотрения.

Function backtrack;

begin

SL := [Start]; NSL := [Start]; DE := []; CS := Start;

% инициализация

while NSL <> () do % пока существуют неисследованные состояния

begin

if CS = goal (или соответствует описанию цели)

return SL % при нахождении пути вернуть список состояний пути

if CS не имеет потомков (исключая узлы, входящие в DE, SL, NSL)

then begin

while SL не пуст и CS = первый элемент из списка SL do

begin

добавить CS в DE % внести состояние в список тупиков

удалить первый элемент из SL;

удалить первый элемент из NSL;

CS := первый элемент NSL;

end

else begin

поместить потомок CS (кроме узлов, уже содержащихся в DE, SL или NSL) в NSL;

CS := первый элемент NSL;

добавить CS в SL

end

end

return FAIL

End.

Определив направление поиска (от данных или от цели), алгоритм поиска должен определить порядок исследования вершин дерева или графа. Существует два возможных варианта последовательности обхода узлов графа: *поиск в глубину* и *поиск в ширину*.

При *поиске в глубину* всегда раскрывается та вершина, которая имеет наибольшую глубину. Из вершин, расположенных на одинаковой глубине,

выбор вершины для раскрытия определяется произвольно. Для сдерживания возможности исследования по бесконечному пути вводится ограничение на глубину. Вершины, находящиеся на граничной глубине, не раскрываются (рис. 4.6).

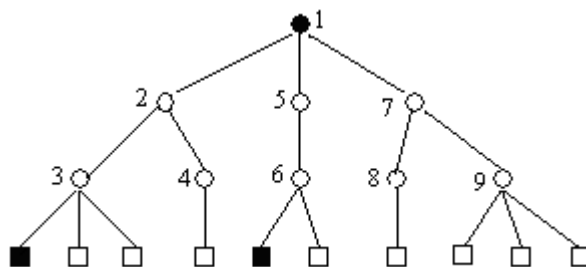


Рис. 4.6. Граф, демонстрирующий работу алгоритма поиска в глубину

Поиск в глубину завершается в следующих случаях:

1) при достижении целевой вершины; 2) при достижении терминальной вершины; 3) при построении вершины, глубина которой превышает граничную глубину.

Недостатком поиска в глубину является то, что в нем может быть сделан неправильный выбор и переход в тупиковую ситуацию, связанную с прохождением вниз по очень длинному пути, притом, что другой вариант мог бы привести к решению, находящемуся недалеко от корня дерева поиска.

Поиск в ширину – это простая стратегия, в которой вначале разворачивается корневой узел, затем – все преемники корневого узла и т.д. таким образом, при этом методе поиска исследуется пространство поиска по уровням, один за другим. Если узлов на данном уровне поиска нет, алгоритм переходит к следующему уровню (рис. 4.7).

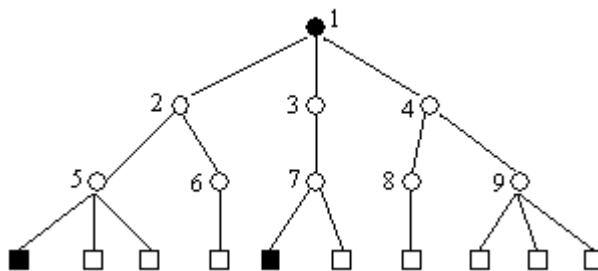


Рис. 4.7. Граф, демонстрирующий работу алгоритма при поиске в ширину

Поиск в ширину требует значительного времени для выполнения, кроме того, он использует значительный объем памяти для хранения всех развернутых узлов.

Для оптимизации метода используется алгоритм поиска по критерию стоимости, который обеспечивает развертывание узла n с наименьшей стоимостью пути. Если стоимости всех этапов равны, такой поиск идентичен поиску в ширину.

Приведем модификации алгоритма с возвратами для реализации стратегии поиска в ширину и глубину.

Алгоритм поиска в ширину.


```

function breadth_first_search;
begin
    open := [Start];                                % инициализация
    closed := [];
    while open <> [] do                                % есть состояния
        удалить крайнее слева состояние из open, скажем X;
        if X – цель then return Success                % цель найдена
        else begin
            сгенерировать потомок X;
            поместить X в список closed;
            исключить потомок X, если он уже в списке
            open или closed;
            поместить остальные потомки в правый конец списка open;
        end
    return Fail
end.

```

Алгоритм поиска в глубину.

```

function depth_first_search;
begin
    open := [Start];                                % инициализация
    closed := [];
    while open <> [] do                                % есть состояния
        удалить крайнее слева состояние из open, скажем X;
        if X – цель then return Success                % цель найдена
        else begin
            сгенерировать потомок X;
            поместить X в список closed;
            исключить потомок X, если он уже в списке
            open или closed;
            поместить остальные потомки в левый конец списка open;
        end
    return Fail
end.

```

Проанализируйте работу рассмотренных алгоритмов на тех примерах, которые есть в данной главе.

4.1.3. Стратегии эвристического поиска

Методы поиска в ширину и в глубину называются методами *слепого поиска*, так как в них порядок раскрытия вершин предопределен и не зависит от расположения цели. При увеличении пространства поиска методы требуют больших ресурсов. Эвристические методы используют информацию о предметной области. Это позволяет рассмотреть не все пространство поиска, а только те пути, которые с наибольшей вероятностью приводят к цели.

Эвристические стратегии поиска используют различные функции оценки $f(n)$. Рассмотрим некоторые из них.

В алгоритме поиска «по первому наилучшему совпадению» для дальнейшего развертывания выбирается узел с наименьшей оценкой, поскольку такая оценка измеряет расстояние до цели. Так как точное значение $f(n)$ неизвестно, то используется эвристическая функция оценки $h(n)$:

$h(n)$ = оценка стоимости наименее дорогостоящего пути от узла n до целевого узла.

Для целевого узла $h(n) = 0$.

Существует несколько разновидностей таких алгоритмов.

Алгоритм жадного поиска по первому наилучшему совпадению

При *жадном поиске по первому наилучшему совпадению* предпринимается попытка развертывания узла, который рассматривается как ближайший к цели на том основании, что он с наибольшей вероятностью должен быстро привести к решению. Таким образом, в этом алгоритме оценка узлов производится с использованием только эвристической функции $f(n) = h(n)$.

Жадный поиск по первому наилучшему совпадению напоминает поиск в глубину, т.к. алгоритм на пути к цели развертывает только один узел, не рассматривая альтернативные пути. Алгоритм имеет те же недостатки, что и алгоритм поиска в глубину: он не является оптимальным, к тому же он — не полный (поскольку может бесконечно следовать по одному пути, не используя другие варианты поиска).

Алгоритм A^*

Наиболее широко известная разновидность поиска по первому наилучшему совпадению называется *поиском A^** .

Алгоритм для каждого узла использует оценочную функцию вида:

$$f(n) = g(n) + h(n).$$

Мы обозначили $g(n)$ – стоимость достижения данного узла; $h(n)$ – нижняя оценка действительного расстояния от узла n до цели. Поскольку функция $g(n)$ позволяет определить стоимость пути от начального узла до узла n , а функция $h(n)$ определяет оценку стоимости наименее дорогостоящего пути от узла n к цели, то справедлива следующая формула:

$f(n)$ = оценка стоимости наименее дорогостоящего пути решения, проходящего через узел n .

Таким образом, при попытке найти наименее дорогостоящее решение вначале проверяется узел с наименьшим значением $f(n)$.

Если эвристическая функция $h(n)$ удовлетворяет некоторым условиям, то поиск A^* становится и полным, и оптимальным

Функция $h(n)$ должна быть допустимой эвристической оценкой, то есть не должна переоценивать расстояния к целевой вершине. Например, для задачи маршрутизации $h(n)$ может представлять собой расстояние до цели по прямой

линии, так как это физически наименьшее возможное расстояние между двумя точками.

Алгоритм был впервые описан в 1968 году Питером Хартом, Нильсом Нильсоном и Бертраном Рафаэлем. В их работе он упоминается как «алгоритм А». Но так как он вычисляет лучший маршрут для заданной эвристики, он был назван алгоритмом A^* .

В отличие от «жадного» алгоритма алгоритм A^* при выборе очередной раскрываемой вершины учитывает, помимо прочего, весь пройденный до неё путь (составляющая $g(x)$ — это стоимость пути от начальной вершины, а не от предыдущей, как в жадном алгоритме).

Как и алгоритм поиска в ширину, A^* является *полным* в том смысле, что он всегда находит решение, если таковое существует.

Временная сложность и потребляемые ресурсы памяти алгоритма A^* зависят от эвристики. В худшем случае, число вершин, исследуемых алгоритмом, растёт экспоненциально по сравнению с длиной оптимального пути и алгоритму приходится запоминать экспоненциальное количество узлов.

Существует несколько вариаций алгоритма, таких как алгоритм A^* с итеративным углублением (iterative deeping A^* , IDA*), A^* с ограничением памяти (memory-bounded A^* , МА*), упрощённый МА* (simplified МА*, SMA*) и рекурсивный поиск по первому наилучшему совпадению (recursive best-first search, RBFS).

Рассмотренные алгоритмы используются для систематического исследования пространства поиска.

При решении задач, которые относятся к задачам планирования (проектирование интегральных схем, автоматическое программирование, управление портфелем акций, составление расписания), интерес представляет лишь окончательный результат. В этом случае применяются алгоритмы другого класса.

Алгоритм «восхождение к вершине»

В этом алгоритме используется оценочная функция, которая оценивает близость исходного состояния к целевому. В алгоритме постоянно происходит перемещение в направлении возрастания значения оценочной функции. Работа заканчивается после достижения «пика», в котором ни одно из соседних состояний не имеет более высокого значения.

Иллюстрация возможного ландшафта процесса поиска глобального максимума с использованием алгоритма, приведена на рис. 4.8 [10].

Применение алгоритма наталкивается на ряд трудностей.

1. Первая и наиболее важная заключается в трудности формулирования оценочной функции, которая адекватно отражает степень близости текущего состояния к целевому состоянию.

2. В процессе поиска возможен выход на «плато», когда ни один из возможных вариантов продолжения поиска не влечет за собой увеличение оценочной функции.

3. Оценочная функция может иметь локальные максимумы, достигнув которые мы не сможем улучшить значение оценочной функции.

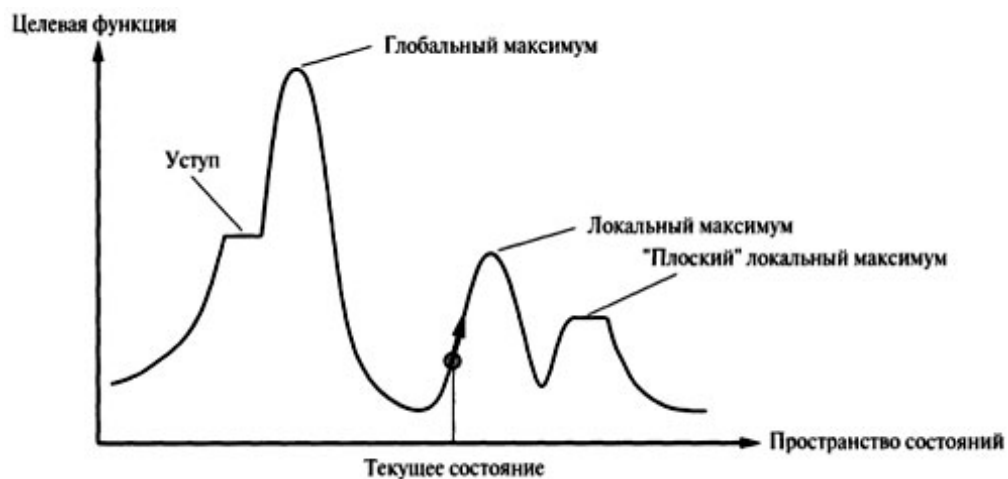


Рис. 4.8. Иллюстрация пространства поиска при использовании алгоритма «восхождение к вершине»

Поиск с восхождением к вершине иногда называют локальным поиском, поскольку в процессе его выполнения происходит захват самого хорошего соседнего состояния без предварительных рассуждений о том, куда следует отправиться дальше.

Модификации алгоритма

Например, поиск с восхождением к вершине и перезапуском случайным образом руководствуется рекомендацией: «если первая попытка оказалась неудачной, пробуйте снова и снова». В алгоритме поиска с восхождением к вершине с выбором первого варианта реализуется стохастический поиск с восхождением к вершине путем формирования преемников случайным образом до тех пор, пока не будет сформирован преемник, лучший по сравнению с текущим состоянием.

Простой алгоритм с восхождением к вершине всегда способен зайти в тупик, достигнув локального максимума, поэтому он является неполным. В отличие от этого алгоритм с чисто случайным выбором вершины-преемника является полным, но чрезвычайно неэффективным. Поэтому представляется разумной попытка скомбинировать каким-то образом восхождение к вершине со случайным блужданием, что позволит обеспечить и эффективность, и полноту. Примером такого алгоритма является алгоритм с эмуляцией отжига.

Поскольку метод отжига использует процесс генерации случайных чисел, поиск решения с использованием данного алгоритма может занять значительное время. В некоторых случаях алгоритм вообще не находит решения, или выбирает не самое оптимальное.

4.1.4. Поиск решения в иерархии пространств

Методы поиска в одном пространстве не позволяют решать сложные задачи, так как с увеличением размера пространства время поиска растет

экспоненциально. Поэтому вводится иерархия пространств (конкретное, абстрактное и метaprостранство), каждое пространство факторизуется. Пространство называется *факторизованным*, если оно разбивается на непересекающиеся подпространства частичными решениями. Для поиска в таких пространствах можно применить метод «*иерархическая генерация – проверка*».

Применение метода ограничено тем, что для ряда областей не удастся по частичному решению сделать вывод о его непригодности (например, задачи планирования и конструирования). В этих случаях применяются методы, использующие идею *абстрактного* пространства. Методы различаются предположениями о природе этих пространств. Абстракция должна подчеркнуть особенности задачи, позволить разбить задачи на подзадачи. В простейшем случае используется фиксированная последовательность подзадач, с помощью которой можно решить любую исходную задачу.

Подобный метод реализован в экспертной системе *RI*. На основании заказа покупателя на конфигурацию компьютера *VAX* система *RI* определяет, не содержит ли заказ несовместимых компонентов, выявляет недостающие компоненты и строит диаграммы, отображающие взаимосвязи компонентов *VAX*. Система *RI* разбивает задачу на 6 подзадач. Порядок их выполнения зависит от комбинации заказанных компонентов и способа их взаимосвязи. Каждой подзадаче соответствует свой набор правил, т.е. каждая подзадача решается в своем пространстве.

В ряде приложений не удастся все решаемые задачи свести к фиксированному набору подзадач. Для решения подобных задач используется метод «*нисходящего уточнения*». Для упрощения процесса решения задачи в сложном пространстве целесообразно получить обобщенное (следовательно, более простое) пространство и найти в нем решение. При таком подходе процесс решения задачи можно представить как нисходящее движение в иерархии пространств от абстрактного к конкретному пространству, в котором получается окончательное решение. Формирование абстрактного пространства получается путем игнорирования части описаний менее абстрактного пространства.

Метод применим, когда существует фиксированная упорядоченность понятий области и фиксированный частичный порядок между подзадачами. Основным недостаток метода заключается в том, что в одинаковых ситуациях при решении любой задачи принимаются одни и те же решения.

В случае если решаемые задачи взаимосвязаны и для решения одной из них нужна информация, полученная при решении другой подзадачи, и подзадачи нельзя упорядочить по времени решения, используется принцип «*наименьших свершений*». Его идея заключается в следующем: решение подзадачи, для которой недостаточно информации, приостанавливается. Решается другая подзадача, решение приостановленной подзадачи возобновляется, когда информации для ее решения будет достаточно.

Принцип наименьших свершений может приводить к тупиковым ситуациям, для разрешения которых используются эвристические методы.

Этот принцип используется в экспертной системе MOLGEN, предназначенной для планирования экспериментов в генной инженерии. В системе MOLGEN взаимодействие между подзадачами представлено в виде ограничений. Решение принимается тогда, когда ограничения определяют узкий набор альтернатив. В противном случае решение приостанавливается и осуществляется переход к другой подзадаче. Связь между подзадачами происходит через ограничения, т.е. ограничения, выставленные одной подзадачей, могут существенно сузить набор альтернатив другой подзадачи.

Если нельзя сделать выбор между альтернативными решениями, то система переключается на эвристическую стратегию и делает предположение. Во многих случаях угадывание позволяет решить задачу. В противном случае делается новая попытка угадывания.

Данный метод трудно применить, когда существует много возможностей продолжения решения, но нет достаточных оснований для выбора одного из них на очередном шаге поиска решения.

4.2. Вывод в семантических сетях

Особенность семантической сети заключается в целостности системы, выполненной на ее основе, не позволяющей разделить базу знаний и механизм вывода. Обычно интерпретация семантической сети определяется с помощью использующих ее процедур [1].

Механизм логического вывода в семантических сетях использует два принципа: наследования свойств и сопоставления по совпадению.

Первый принцип, в свою очередь, базируется на учете важнейших связей, отражаемых в семантической сети. К таким связям относятся:

- связь «есть», «является» *IS-A*;
- связи «имеет часть», «является частью» *HAS-PART*, *PART-OF*.

Последовательно переходя от одного узла к другому по направлению соответствующих связей, можно выявить новую информацию, характеризующую тот или иной узел.

На рис. 4.9 показан фрагмент некоторой семантической сети и обозначена так называемая ветвь наследования свойств. Из этого фрагмента можно вывести заключения типа «у Ивана есть голова», «мужчина имеет голову» и т.п.



Рис. 4.9. Часть семантической сети с наследованием свойств

В принципе сопоставления по совпадению вопрос к системе представляется в виде фрагмента семантической сети с использованием тех же названий

сущностей (узлов) и связей, что и в основной сети, и реализации процедуры «наложения» вопроса на сеть и поиска такого его положения, которое соответствует ответу на вопрос. На рис. 4.10 помимо уже известной связи «есть» представлено отношение владения (связь «владеет»).

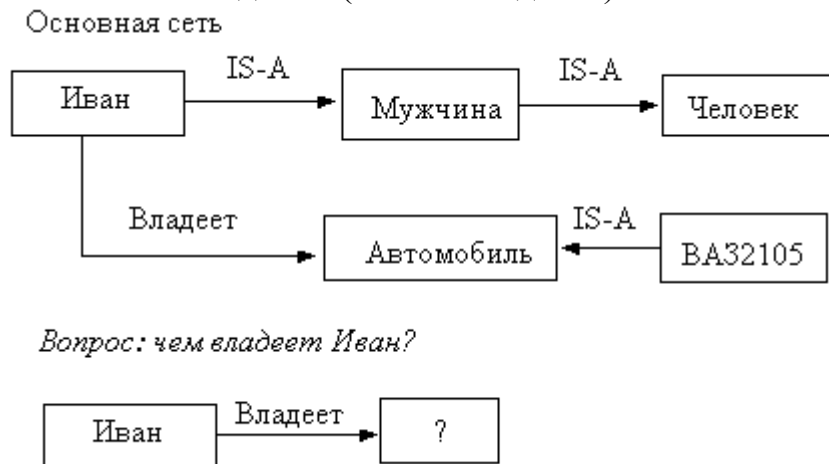


Рис. 4.10. Часть семантической сети, дополненная свойством «владеет»

Вопрос: «Чем владеет Иван?» – формализуется с помощью узла «Иван» и отношения «владеет». Далее в простейшем случае осуществляется перебор узлов сети, имеющих имя «Иван» (если они имеются), и поиск такого из них, который имеет связь «владеет». Затем может быть задействован принцип наследования свойств. Ответами на поставленный в примере вопрос будут суждения «Иван владеет автомобилем» и «Иван владеет (автомобилем) BA32105». Понятно, что в практике использования ЭС такого типа приходится реализовывать значительно более сложную процедуру поиска, включающую элементы семантического анализа.

4.3. Поиск решения во фреймовых системах

Фреймовая модель знаний имеет сложную иерархическую структуру, отражающую реальные объекты (понятия) и отношения (связи) некоторой предметной области.

В рамках фреймового подхода предполагается, что знания в системе представляются в виде отдельных кластеров знаний, или подструктур, содержащих сведения о стереотипах (т.е. о некоторых общих характеристиках данного класса объектов или ситуаций) [1]. Согласно данному представлению понимание ситуации для системы означает поиск в перечне накопленных структур такой, которая наилучшим способом описывала бы рассматриваемую ситуацию. При этом слоты заполняются некоторой информацией, и заполненный фрейм проверяется на адекватность данной ситуации. В случае несовпадения ищется новый фрейм и процесс продолжается.

Можно выделить три процесса, происходящих во фреймовых системах.

1. Создание экземпляра фрейма. Для создания экземпляра фрейма необходимо найти подходящий фрейм и заполнить его слоты информацией, описывающей специфику ситуации.

2. Активация фреймов. Если фрейм считается подходящим, осуществляется его активация глобальным процессом.

3. Организация вывода, заключающаяся в последовательном поиске и активации в сети фреймов до нахождения наиболее соответствующего фрейма и построения на его основе экземпляра фрейма.

Таким образом, механизм логического вывода основан на обмене значениями между одноименными слотами различных фреймов и выполнении присоединенных процедур «*IF-ADDED*», «*IF-NEEDED*», «*IF-REMOVED*».

Для реализации поиска решений предлагается три подхода.

1. Модель знаний реализуется в виде совокупности фреймов-прототипов. Организация вывода возлагается на пользователя. С этой целью используются универсальные языки функционального программирования, например, язык FRL (Frame Relation Language); KRL (Knowledge Relation Language); ЛИСП, расширенный аппаратом присоединенных процедур.

2. Для систем фреймов определенной конструкции вводится единый вычислительный процесс, в основе которого лежит выбор фрейма, управляющего дальнейшими вычислениями.

3. Выделяются узкие подклассы фреймов, для которых разрабатываются унифицированные алгоритмы поиска решений.

Пример

Рассмотрим конкретный пример, иллюстрирующий работу ЭС с фреймовым представлением знаний, которая используется в подразделении, организующим научно-исследовательскую работу в некотором учреждении [16]. На рис. 4.11 представлена иерархия справочной информации об отчете по научно-исследовательской работе.

Рис. 4.12 содержит структуры понятий «Отчет по научно-исследовательской работе» и «Этапный отчет по научно-исследовательской работе».

Рис. 4.13 содержит структуру понятия «Этапный отчет по научно-исследовательской работе «Эталон» со значениями некоторых слотов, заданных по умолчанию.

Фреймовая система функционирует следующим образом. Пусть в ЭС поступил запрос от уполномоченного пользователя: «Необходима информация о ходе выполнения научно-исследовательской работы «Эталон». Информация проходит через лингвистический процессор, анализируется и в виде значения «Эталон» вносится в слот *Шифр* узла «Этапный отчет по научно-исследовательской работе «Эталон». Далее начинают работать присоединенные процедуры.

1. Выполняется процедура *IF-ADDED*, связанная со слотом *Шифр*, поскольку в слот было введено некоторое значение. Эта процедура осуществляет поиск сведений о руководителе научно-исследовательской работы

«Эталон» и вписывает полученное имя в слот *Автор* узла «Этапный отчет по научно-исследовательской работе «Эталон».

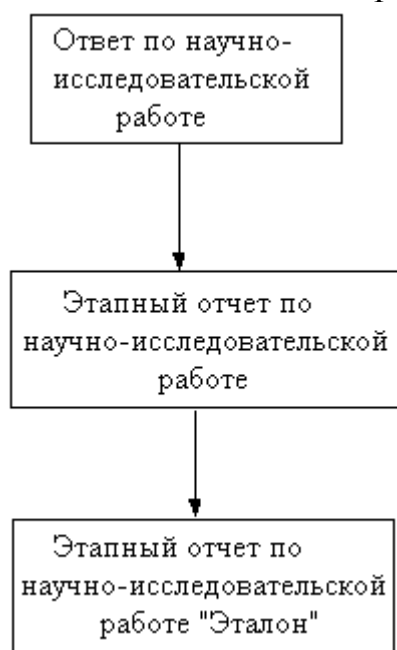


Рис. 4.11. Иерархия понятия «Отчет по научно-исследовательской работе»

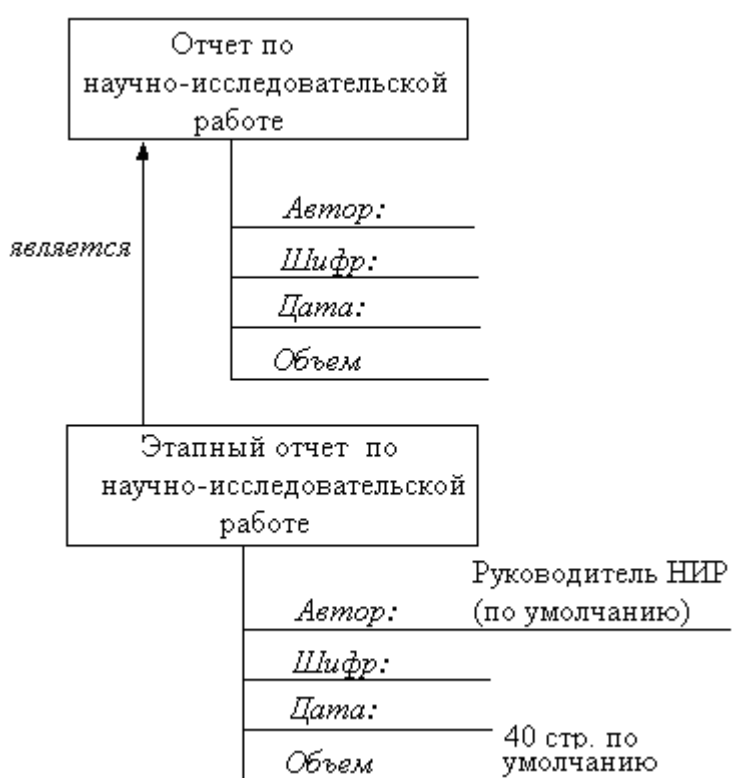


Рис. 4.12. Структура понятий «Отчет по научно-исследовательской работе» и «Этапный отчет по научно-исследовательской работе»

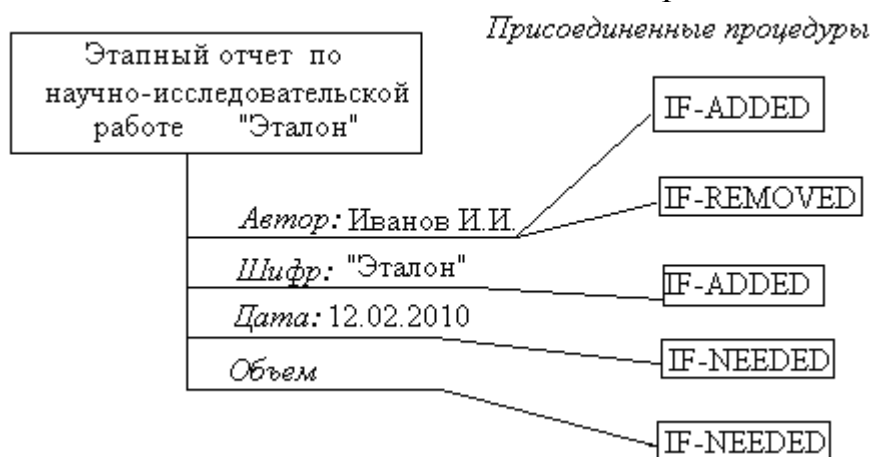


Рис. 4.13. Структура понятия «Этапный отчет по научно-исследовательской работе» со значениями слотов

2. Выполняется процедура IF-ADDED, связанная со слотом *Автор*, , так как в слот было вписано значение. Эта процедура начинает составлять сообщение, чтобы отправить его Иванову И.И., но обнаруживает, что отсутствует значение слота *Дата*.

3. Процедура «IF–ADDED, просматривая слот *Дата* и найдя его пустым, активизирует процедуру IF–NEEDED, связанную с этим слотом. Процедура найдет текущую дату, используя календарь ЭС, выберет ближайшую к ней (но большую) и впишет ее в слот *Дата*.

4. Процедура ID–ADDED, связанная со слотом *Автор*, найдет, что отсутствует еще одно значение, необходимое для формирования выходного сообщения, а именно значение слота *Объем*. Данный слот не имеет присоединенных процедур, поэтому приходится брать значение по умолчанию из одноименного слота общей концепции «Этапного отчета по научно-исследовательской работе».

Теперь ЭС может сформировать выходное сообщение типа: «Этапный отчет по научно-исследовательской работе «Эталон» должен быть представлен И.И. Ивановым к 12.02.2011 г. Предполагаемый объем отчета 40 стр.»

Если в какой-либо момент значение слота *Автор* будет удалено, то сработает процедура «IF–REMOVED» и система автоматически отправит Иванову И.И. сообщение о том, что отчет не требуется.

4.4. Выводы

1. Решение задачи в ЭС с использованием продукционной модели знаний осуществляется путем поиска решения в символической системе. Поиск решения в одном пространстве выполняется на графе. Граф редукции задач фиксирует зависимость целевой задачи от комбинации составляющих ее подзадач. Граф пространства состояний фиксирует изменение состояния каждой подзадачи в процессе поиска решения основной задачи.

2. Поиск по первому наилучшему совпадению использует эвристическую оценочную функцию $h(n)$, которая оценивает стоимость достижения решения из узла n .

3. При «жадном» поиске по первому наилучшему совпадению развертываются узлы с минимальным значением $h(n)$. Этот поиск не оптимален, но часто является эффективным.

4. При поиске A^* развертываются узлы с минимальным значением $f(n) = g(n) + h(n)$.

5. Производительность алгоритмов эвристического поиска зависит от качества эвристической функции.

6. Алгоритмы RBFS и SMA* представляют собой надежные, оптимальные алгоритмы поиска, в которых используются ограниченные объемы памяти.

7. Такие методы локального поиска, как поиск с восхождением к вершине, действуют на основе формулировок полного состояния и предусматривают хранение в памяти лишь небольшого количества узлов. Разработано также несколько стохастических алгоритмов, включая алгоритм отжига, которые возвращают оптимальные решения при наличии подходящего графика постепенного уменьшения величины случайного разброса.

8. В семантических сетях используются методы поиска решения по образцу, который сводится к изоморфному вложению подграфа запроса в семантическую сеть.

9. Главным недостатком фреймового представления знаний является отсутствие единого механизма поиска решений.

4.5. Вопросы для самоконтроля

1. Как вы понимаете термин «пространство поиска»? Что представляет собой пространство поиска при игре в шахматы?

2. Как вы понимаете термин «пространство решений? Что представляет собой пространство решения при игре в шахматы.

3. Дайте сравнительную характеристику алгоритмов поиска в ширину и в глубину при решении задач в пространстве состояний. Какой алгоритм находит кратчайший путь на графе? Какой из алгоритмов является разрешимым?

4. Охарактеризуйте методы эвристического поиска, применяемые при выводе решения в одномерном пространстве состояний.

5. Дайте определение разрешимой вершины в И/ИЛИ графе.

6. Как формируется оценочная функция в методе «восхождение в гору»? Назовите трудности, которые встречаются при реализации этого метода.

7. Сравните метод формирования оценочной функции в методе «восхождение в гору» с методом формирования оценочной функции в алгоритме A^* . В чем достоинства алгоритма A^* ?

8. Охарактеризуйте алгоритмы поиска решения в иерархии пространств.

9. Когда применяются алгоритмы стохастического поиска? Дайте им сравнительную характеристику.

10. Как используется свойство наследования при выводе решения во фреймовой модели представления знаний?

11. Какие подходы предлагаются для получения решения во фреймовой модели представления знаний?

5. ОСОБЕННОСТИ РЕШЕНИЯ ЗАДАЧ В ПРОДУКЦИОННЫХ СИСТЕМАХ

5.1. Конфигурация системы продукций и цикл работы интерпретатора

Несмотря на все недостатки, наибольшее распространение на практике получила продукционная модель знаний. Общим для систем продукций является наличие трех элементов.

1. Набор правил базы знаний.
2. Рабочая память, где хранятся предпосылки, касающиеся отдельных задач, а также результаты выводов, получившиеся на основе этих предпосылок (динамическая база данных).
3. Механизм логического вывода, использующий правила базы знаний в соответствии с содержимым рабочей памяти.

Конфигурацию систем продукций упрощенно можно представить в следующем виде (рис. 5.1.)

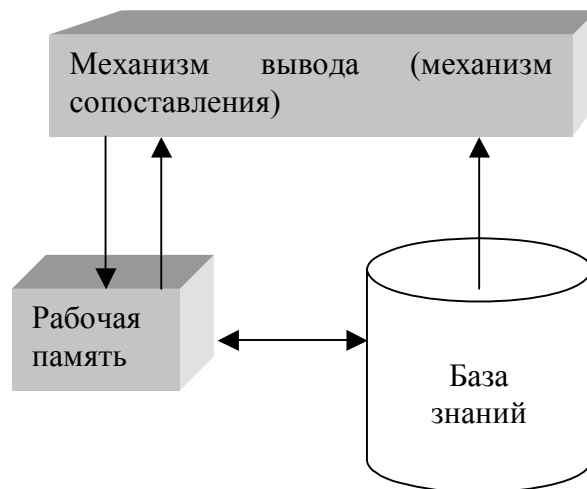


Рис. 5.1. Конфигурация системы продукций

Механизм вывода (машина вывода, интерпретатор) выполняет две функции:

1. Просматривает существующие факты в рабочей памяти и правила в базе знаний и добавляет (при необходимости) в рабочую память новые факты.
2. Определяет порядок просмотра и применения правил.

Следовательно, машина вывода включает в себя два компонента: один реализует собственно вывод, другой управляет этим процессом.

Действие машины вывода основано на применении правила *modus ponens*.

Определение

Если известно, что истинно утверждение A и существует правило вида «Если A То B », то утверждение B тоже истинно.

Машина вывода должна работать даже при недостатке информации. Решение может быть неточным, однако система не должна останавливаться из-за отсутствия информации.

Управляющий компонент машины вывода определяет порядок применения правил и выполняет 4 функции.

1. *Сопоставление* – образец правила (его левая часть) сопоставляется с имеющимися фактами.

2. *Выбор* – если в конкретной ситуации можно применить несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию (разрешение конфликта).

3. *Срабатывание* – если образец правила при сопоставлении совпадает с каким-либо фактом из рабочей памяти, то правило выполняется.

4. *Действие* – рабочая память подвергается изменению путем добавления в нее заключения выполненного правила. Если в правой части правила содержится указание на какое-либо действие, то оно выполняется.

Интерпретатор работает циклически. Цикл работы интерпретатора схематически изображен на рис. 5.2.

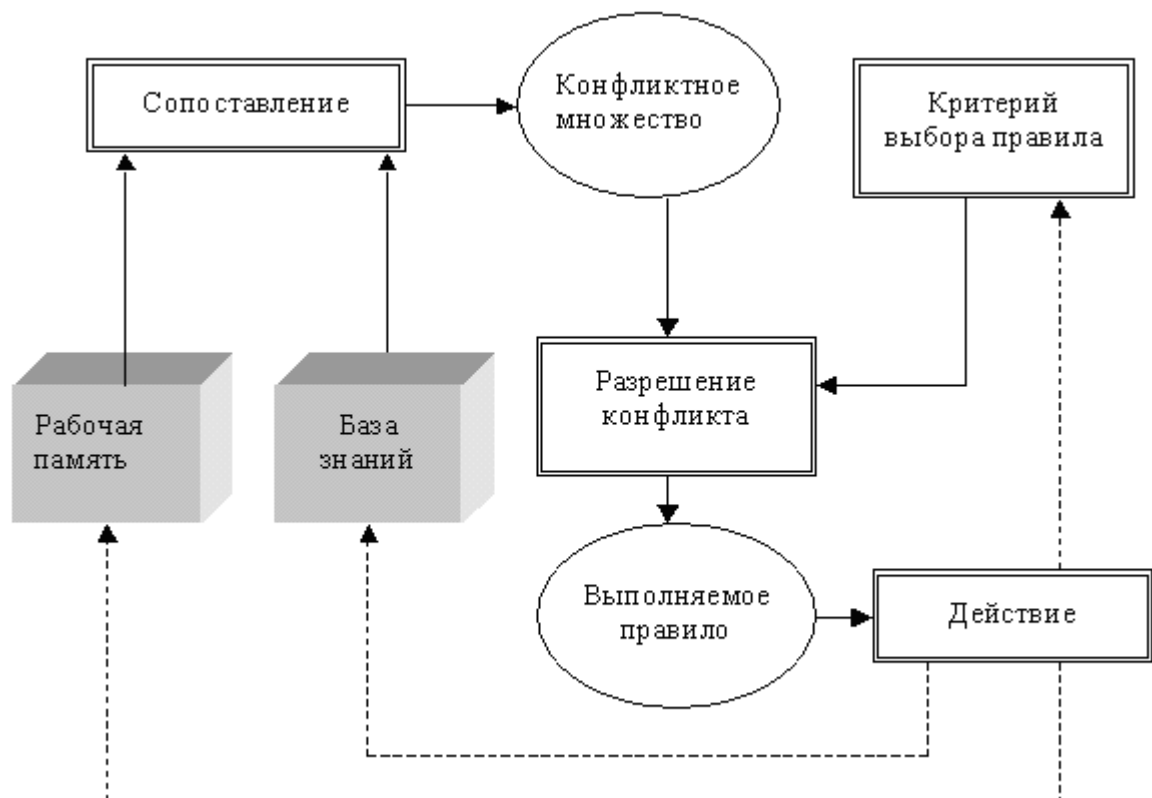


Рис. 5.2. Цикл работы интерпретатора

В каждом цикле просматриваются все правила, чтобы выявить те, послышки которых совпадают с известными на данный момент фактами из рабочей

памяти. Если отобрано несколько правил, то их совокупность называется *конфликтным множеством*. Для разрешения конфликта интерпретатор имеет критерий, с помощью которого он отбирает одно правило, которое выполняется. Заключение правила заносится в рабочую память. В одном цикле может выполниться только одно правило. От выбранного метода поиска, т.е. стратегии вывода, зависит порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления. Процедуры, реализующие поиск, в большинстве коммерческих систем обычно «зашиты» в механизм вывода.

При разработке стратегии управления выводом важны:

1. Начальная точка в пространстве состояний. От выбора этой точки зависит метод осуществления поиска – в прямом или обратном направлении.
2. Метод и стратегия перебора – в глубину, в ширину, по подзадачам или иначе.

При *обратном* порядке вывода вначале выдвигается некоторая гипотеза, а затем механизм вывода, переходя к фактам, пытается найти те, которые подтверждают гипотезу. Если она оказывается правильной, то выбирается следующая гипотеза, детализирующая первую и являющаяся по отношению к ней подцелью. Далее отыскиваются факты, подтверждающие истинность подчиненной гипотезы. Вывод такого типа называется *управляемым* целями. Обратный поиск применяется в тех случаях, когда цели известны и их сравнительно немного.

В системах с *прямым* выводом отыскивается заключение, которое следует из этих фактов. Если такое заключение удастся найти, то оно заносится в рабочую память. Прямой вывод называется выводом, управляемым данными.

Существуют системы, в которых вывод основывается на сочетании обратного и ограниченного прямого выводов. Такой комбинированный вывод называется *циклическим*.

5.2. Пример решения задачи на основе цели

Многие предметные области больше соответствуют прямому поиску. Например, при решении проблемы интерпретации большая часть информации представляет собой исходные данные, при этом трудно сформулировать гипотезы или цель. Это приводит к прямому процессу рассуждения, при котором факты помещаются в рабочую память, и система ищет для них интерпретацию.

В экспертной системе на основе цели в рабочую память помещается целевое выражение. Система сопоставляет заключения правил с целевым выражением и помещает их предпосылки в рабочую память. Это соответствует декомпозиции проблемы на более простые подцели. На следующей итерации работы продукционной системы процесс продолжается, эти предпосылки становятся новыми подцелями, которые сопоставляются с заключениями правил. Система работает до тех пор, пока все подцели в рабочей памяти не станут истинными,

подтверждая гипотезу. Таким образом, обратный поиск в экспертной системе приближенно соответствует процессу проверки гипотезы решения проблем человеком.

В экспертной системе, чтобы достичь цели, следует запросить информацию у пользователя. В некоторых экспертных системах разработчики определяют, какие подцели могут достигаться путем запроса к пользователю. Другие системы обращаются к пользователю, если не могут вывести истинность подцели на основе правил из базы знаний.

Пример

В качестве примера решения задач на основе цели с запросами к пользователю рассмотрим небольшую экспертную систему диагностики автомобиля [10]. Она приводится как пример, демонстрирующий образование цепочек правил, интеграцию новых правил и использование возможностей объяснения.

- Правило 1: ЕСЛИ топливо поступает в двигатель
И двигатель вращается,
ТО проблема в свечах зажигания.
- Правило 2: ЕСЛИ двигатель не вращается
И фары не горят,
ТО проблема в аккумуляторе или проводке.
- Правило 3: ЕСЛИ двигатель не вращается
И фары горят,
ТО проблема в стартере.
- Правило 4: ЕСЛИ в баке есть топливо
И топливо поступает в карбюратор,
ТО топливо поступает в двигатель.

Для работы с этой базой знаний цель верхнего уровня *проблема в X* помещается в рабочую память, как показано на рис. 5.3. Здесь *X* – переменная, которая может сопоставляться с любой фразой, например, *проблема в аккумуляторе или проводке*.

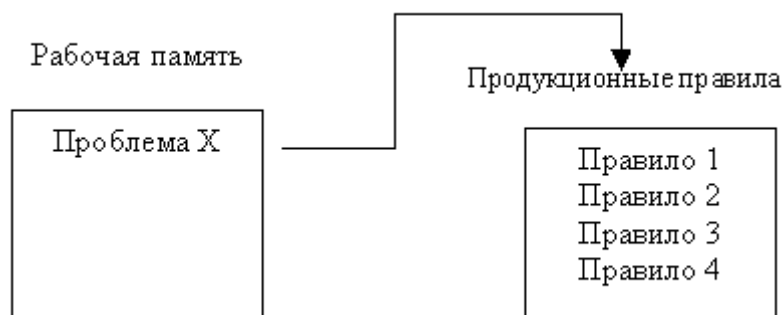


Рис. 5.3. Начало работы продукционной системы диагностики автомобиля

В процессе решения задачи переменная *X* будет связана с некоторым значением. С выражением в рабочей памяти сопоставляются три правила: правило 1, правило 2 и правило 3. Если мы разрешаем конфликты, предпочитая

правила с наименьшим номером, будет активизироваться правило 1. При этом X связывается со значением *свечи зажигания*, и предпосылки правила 1 помещаются в рабочую память (рис. 5.4). Таким образом, система выбирает для исследования гипотезу о неисправности свечей зажигания. Это можно рассматривать как выбор системой ветви ИЛИ в графе И/ИЛИ.

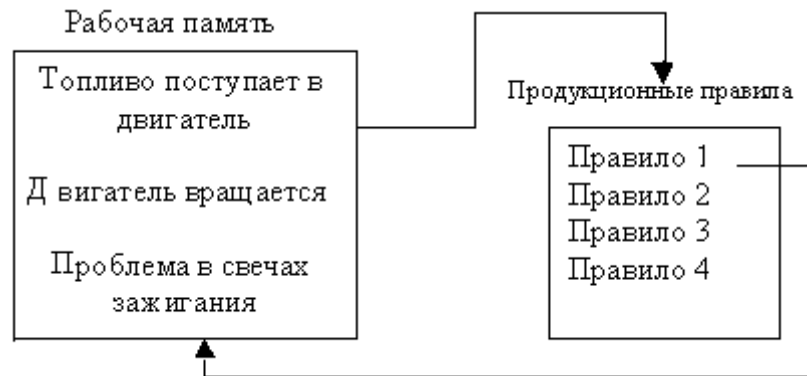


Рис. 5.4. Продукционная система после активизации правила 1

Теперь в рабочей памяти существуют три элемента, которые не соответствуют ни одному заключению в наборе правил. В подобной ситуации экспертная система будет запрашивать пользователя. Согласно правилу 4 топливо поступает в двигатель, если в баке есть топливо и топливо поступает в карбюратор. Если пользователь подтвердит истинность этих подцелей, то экспертная система определит, что в двигатель поступает топливо. Для определения истинности второй посылки первого правила нужно задать вопрос: «Двигатель вращается?». При положительном ответе система успешно определит, что автомобиль не заводится из-за неисправностей в свечах зажигания.

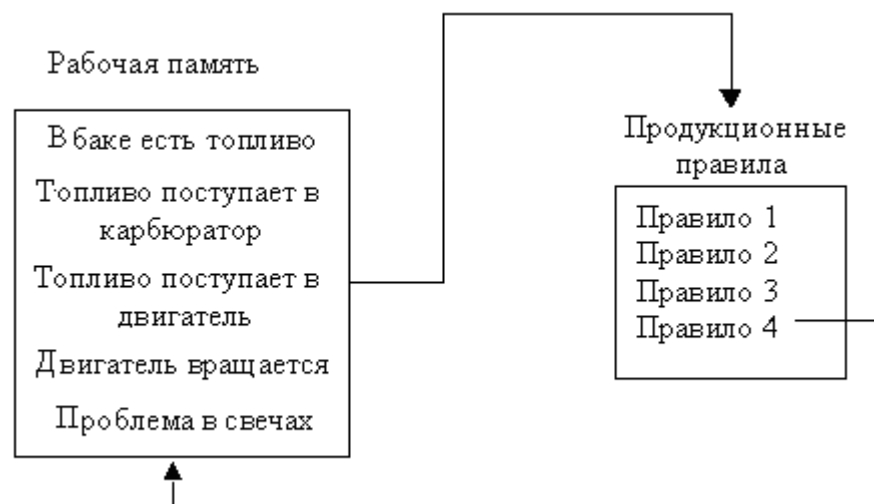


Рис. 5.5. Продукционная система после активизации правила 4

При поиске этого решения система исследовала крайнюю слева вервь графа И/ИЛИ, показанного на рис. 5.6.

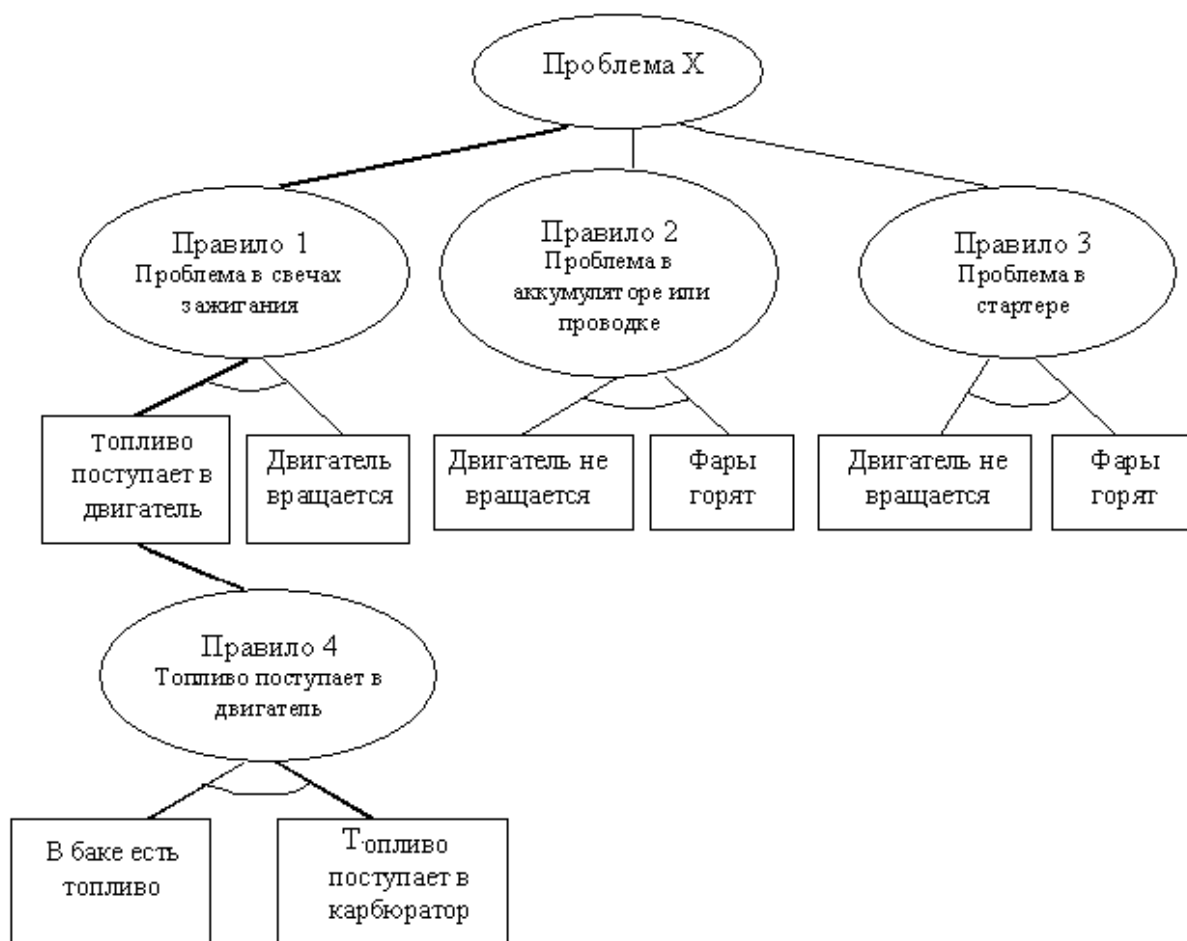


Рис. 5.6. Граф И/ИЛИ для решения проблем с двигателем

Конечно, это очень простой пример. И не только потому, что база знаний об автомобиле ограничена, но также потому, что в этом примере не учитывается ряд важных аспектов. Правила написаны на русском, а не на формальном языке. При обнаружении решения, реальная система сообщит пользователю свой диагноз (наша система просто останавливается). Необходимо также поддерживать достаточно длинную историю рассуждений, чтобы в случае необходимости возвратиться назад. Если вывод о неисправности свечей зажигания окажется неудачным, нужно иметь возможность возвратиться на второй уровень и проверить правило 2. Заметим, что информация об упорядочении подцелей в рабочей памяти на рис. 5.5 и на графе, представленном на рис. 5.6, явно не выражена. Однако, несмотря на простоту, этот пример подчеркивает важность поиска на основе продукционной системы и его представления графом И/ИЛИ в экспертных системах, основанных на правилах.

5.3. Системы с доской объявлений

Системы с *доской объявлений* (black board systems) имеют иерархически организованную базу знаний и могут использовать как прямой, так и обратный метод поиска решения. Их архитектура подходит для решения задач

конструирования, для которых характерно большое факторизованное многомерное пространство решений.

Системы применяются для интерпретации данных, анализа и синтеза многокомпонентных структур.

С точки зрения организации процесса вычислений, можно выделить следующие компоненты систем с доской объявлений.

1. Знания о проблемной области разделены между независимыми источниками знаний, которые работают под управлением планировщика.

2. Решение формируется в некоторой глобально доступной структуре, которую называют *доской объявлений*.

3. Доска объявлений разделена на несколько уровней описания, каждый уровень соответствует определенной степени детализации.

4. Источники знаний формируют объекты на доске объявлений с помощью планировщика. Обычно записи активизации источников знаний помещаются в специальный список выбора, откуда их извлекает планировщик.

5. Источники знаний общаются друг с другом только через доску объявлений и не могут непосредственно передавать друг другу данные.

В реальной экспертной системе доска объявлений представляет собой *ассоциативную память* для источников знаний. Типичным представителем систем с доской объявлений является система RGB.

Разделы доски объявления в системе RGB называются пространствами, они разделены на размерности, для доступа к ним используется атрибут индексации. Элементы структуры данных доски объявлений расположены в этих пространствах и имеют свои атрибуты индексации, которые позволяют реализовать какой-либо из методов доступа к ним (рис. 5.7).

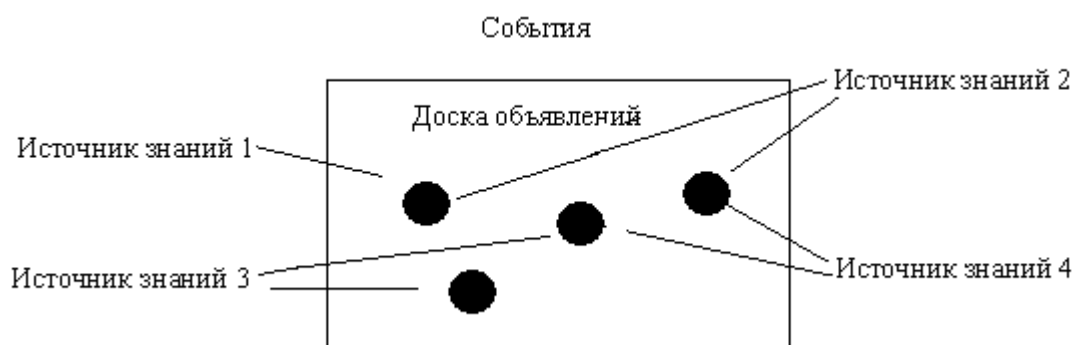


Рис. 5.7. Доска объявлений

Размерности, используемые для структуризации пространств, могут быть упорядоченными или перечисляемыми. Индексы могут быть скалярными, т.е. состоять из единственного элемента данных, или составными. Скалярные индексы упорядоченных размерностей – это числа из некоторого диапазона, а составные индексы – это либо наборы меток, либо некоторая упорядоченная последовательность.

В каждом цикле поиска решений отыскивается элемент в определенном пространстве доски объявлений. Процесс извлечения элемента состоит из четырех этапов:

1. *Первичное извлечение.* Выбирается множество всех элементов данного пространства, которые «претендуют» на сопоставление с шаблоном.

2. *Предварительная фильтрация.* К элементам в отобранном множестве применяются какие-либо процедуры предварительного отбора, позволяющие отсеять некоторые элементы.

3. *Фильтрация по шаблону.* Каждый элемент отобранного множества сравнивается с шаблоном. Проверяется, удовлетворяет ли он заданным в шаблоне ограничениям.

4. *Постфильтрация.* К каждому отобранному элементу применяются дополнительные процедуры проверки.

Схема обработки отображает процесс в общих чертах. Следует отметить, что уровень сложности таких систем на порядок выше, чем в экспертных системах с другой архитектурой, использующих в чистом виде представление знаний в виде правил и фактов.

5.4. Экспертные системы, основанные на прецедентах

В большинстве экспертных систем используется большое количество правил, специфичных для данной предметной области, а поиск вывода основан на пошаговом процессе.

Существуют, однако, задачи, которые не вписываются в эту парадигму.

При решении многих задач человек стремится распознать ситуацию и найти для нее аналог, а не заново формировать решение.

При решении задач подобного типа с помощью компьютера создается некоторая «библиотека» ситуаций, которые имеют отношение к решаемой проблеме. Такая библиотека индексируется для быстрого поиска ситуаций, аналогичной текущей. Кроме того, используется механизм адаптации ранее принятого решения к новой ситуации.

Описанный подход называется рассуждениями, основанными на прецедентах.

Прецеденты содержат знания о ситуации, «вставленные» в контекст. Контекст описывает состояние внешнего мира, в котором это знание применяется.

Прецедент должен представлять решение проблемы в определенном контексте и описывать состояние мира, которое получится, если будет принято решение.

Прецедент обычно описывается в виде фрейма, в котором структурирована информация о проблеме, решение и контекст. Описание прецедента может быть сопоставлено с данными или описанием цели. Однако поиск описания прецедента требует использования сложного механизма индексирования.

Архитектура экспертной системы, основанной на прецедентах, показана на рис.5.8.

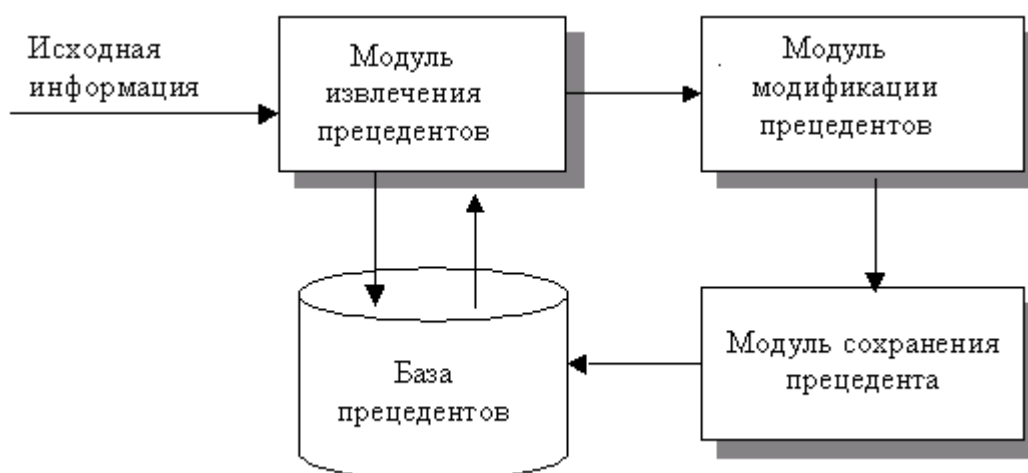


Рис. 5.8. Архитектура системы, основанной на прецедентах

Модуль извлечения прецедентов отыскивает в базе прецедентов ситуацию, наиболее подходящую к текущей. Модуль модификации прецедентов копирует и переименовывает выбранный прецедент и пытается скорректировать его в соответствии с полученной целевой спецификацией. После выполнения коррекции новый прецедент записывается модулем сохранения в базу прецедентов.

5.5. Динамические экспертные системы

Среди специализированных систем, основанных на знаниях, наиболее значимы экспертные системы реального времени, или динамические экспертные системы. На их долю приходится 70% этого рынка.

Использование подобных средств позволило создать стратегически значимые приложения в таких областях, как управление непрерывными производственными процессами в химии, фармакологии, производстве цемента, продуктов питания и т.п., аэрокосмические исследования, транспортировка и переработка нефти и газа, управление атомными и тепловыми электростанциями, финансовые операции, связь и многие другие.

Требования, предъявляемые к системам, работающим в реальном времени, следующие.

1. Представление изменяющихся во времени данные, поступающие от внешних источников, обеспечение хранения и анализ изменяющихся данных.
2. Выполнение временных рассуждений о нескольких различных асинхронных процессах одновременно (т.е. планировать в соответствии с приоритетами обработку поступивших в систему процессов).
3. Обеспечение механизма рассуждения при ограниченных ресурсах (время, память).

4. Обеспечение «предсказуемости» поведения системы, т.е. гарантию того, что каждая задача будет запущена и завершена в строгом соответствии с временными ограничениями.

5. Моделирование «окружающего мира», рассматриваемого в данном приложении, обеспечение создания различных его состояний.

6. Протоколирование своих действий и действий персонала, обеспечение восстановления после сбоя.

7. Обеспечение наполнения базы знаний для приложений реальной степени сложности с минимальными затратами времени и труда.

8. Обеспечение настройки системы на решаемые задачи (проблемная/предметная ориентированность).

9. Обеспечение создания и поддержки пользовательских интерфейсов для различных категорий пользователей.

10. Обеспечение уровня защиты информации (по категориям пользователей) и предотвращение несанкционированного доступа.

Мы познакомимся с динамическими экспертными системами на основе инструментального средства G2 (www.gensym.com), которое применяется для построения динамических экспертных систем.

G2 – это среда для разработки интеллектуальных приложений. Она применяется при решении задач диагностики, оптимального управления сложными технологическими процессами, для сетевого администрирования, администрирования потребления энергии и мониторинга окружающей среды. Система G2 позволяет создавать гибридные системы, использующие нейронные сети, нечеткую логику и генетические алгоритмы. На рис. 5.9 показан интерфейс системы при управлении динамическими объектами.

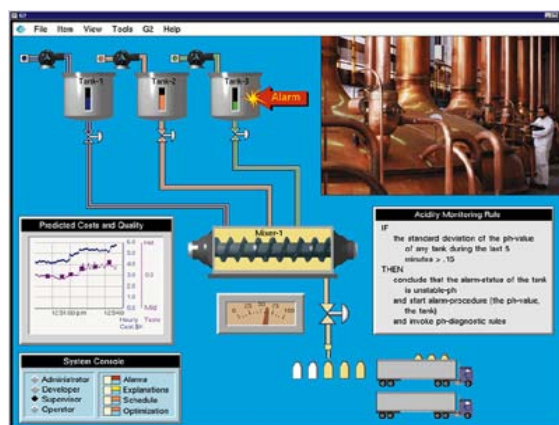


Рис. 5.9. Интерфейс G2 при управлении динамическими объектами

Для представления знаний эксперта о проблемной области используются правила. Все знания в G2 хранятся в двух типах файлов: базы знаний и библиотеки знаний. В файлах первого типа хранятся знания о приложениях: определения всех объектов, объекты, правила, процедуры и т.п. В файлах библиотек хранятся общие знания, которые могут быть использованы более чем в одном приложении, например, определение стандартных объектов. Файлы баз знаний могут преобразоваться в библиотеки знаний и обратно.

Знания структурируются: предусмотрены иерархия классов, иерархия модулей, иерархия рабочих пространств.

Для машины вывода G2 характерен богатый набор способов возбуждения правил. Предусмотрено девять случаев:

1. Данные, входящие в антецедент правила изменились (прямой вывод).
2. Правило определяет значение переменной, которое требуется другому правилу или процедуре (обратный вывод).
3. Каждые n секунд, где n – число, определенное для данного правила (сканирование).
4. Явное или неявное возбуждение другим правилом – путем применения действий фокусирования и возбуждения.
5. Каждый раз при запуске приложения.
6. Входящей в антецедент переменной присвоено значение, независимо от того, изменилось оно или нет.
7. Определенный объект на экране перемещен пользователем или другим правилом.
8. Определенное отношение между объектами установлено или уничтожено.
9. Переменная не получила значения в результате обращения к своему источнику данных.

Если первые два способа достаточно распространены и в статических системах, а третий хорошо известен как механизм запуска процедур-демонов, то остальные являются уникальной особенностью системы G2

Подсистема моделирования G2 – достаточно автономная, но важная часть системы. На различных этапах жизненного цикла прикладной системы она служит достижению различных целей. Во время разработки подсистема моделирования используется вместо объектов реального мира для имитации показаний датчиков.

На этапе эксплуатации прикладной системы процедуры моделирования выполняются параллельно функциям мониторинга и управления процессом, что обеспечивает следующие возможности:

- верификация показаний датчиков во время исполнения приложения;
- подстановка модельных значений переменных при невозможности получения реальных значений (выход из строя датчика или длительное время реакции на запрос).

Как видим, играя роль самостоятельного агента знаний, подсистема моделирования повышает жизнеспособность и надежность приложений. Для описания внешнего мира подсистема моделирования использует уравнения трех видов: алгебраические, разностные и дифференциальные (первого порядка).

5.6. Выводы

1. Экспертные системы, использующие продукционную модель знаний, находят наиболее широкое практическое применение.

2. В продукционных системах обязательно наличие кроме базы знаний, рабочей памяти, которая хранит факты, и механизма вывода (интерпретатора).

3. Интерпретатор работает итерационно и на каждом шаге вывода просматривает базу знаний полностью, формируя конфликтное множество и разрешая конфликты.

4. Системы с доской объявлений имеют несколько независимых источников знаний. Взаимодействие между источниками происходит в глобальной памяти. Эти системы применяются, в основном, для решения задач анализа и синтеза многокомпонентных структур.

5. Системы, основанные на прецедентах, имеют библиотеку прецедентов, в которой хранятся знания различных ситуаций и способы разрешения ситуаций. Знания о ситуации контекстно-зависимые.

6. Для управления динамическими системами нужны экспертные системы, работающие в реальном времени. Представителем таких систем является инструментальное средство проектирования G2.

5.7. Вопросы для самоконтроля

1. Поясните конфигурацию системы продукций. Для каких целей используется рабочая память?

2. Поясните основные функции машины вывода.

3. Для чего используется управляющий механизм машины вывода?

4. При поиске решения в экспертных системах, использующих продукционную модель знаний, возникают конфликтные ситуации. В связи с этим поясните термины «конфликтующее множество» и «разрешение конфликтов».

5. Поясните отличия методов поиска, основанных на прямой и обратной цепочках рассуждений.

6. Что такое источник знаний в системе с доской объявлений? Как связываются между собой источники знаний в системах с доской объявлений.

7. Сравните использование разных вариантов архитектуры экспертных систем – продукционную систему и систему с доской объявлений. При каких условиях можно считать оправданным усложнение системы, использующей модель с доской объявлений?

8. В чем состоит основное отличие методики формирования суждений на основе прецедентов от логического вывода на основе правил?

9. Как соотносятся прецеденты и фреймы? Как можно применить концепцию фреймов по отношению к прецедентам?

10. Чем динамические экспертные системы отличаются от статических экспертных систем?

11. Поясните, как представляются знания при использовании системы G2.

6. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА НЕОПРЕДЕЛЕННОСТИ В ПРОДУКЦИОННЫХ СИСТЕМАХ

6.1. Источники неопределенности

При решении проблем мы часто сталкиваемся с множеством источников неопределенности используемой информации. В большинстве случаев их можно разделить на две категории: недостаточно полное знание предметной области и недостаточная информация о конкретной ситуации.

Теория предметной области (т.е. наши знания об этой области) может быть неясной или неполной: в ней могут использоваться недостаточно четко сформулированные концепции или недостаточно полно изученные явления.

Неопределенность знаний приводит к тому, что используемые правила решения не всегда приводят к корректным результатам. Располагая неполным знанием, мы не можем уверенно предсказать, какой эффект даст то или иное действие. Например, терапия, использующая новые препараты, довольно часто дает совершенно неожиданные результаты. И, наконец, даже когда мы имеем достаточно полную теорию предметной области, эксперт может предпочесть использовать не точные, а эвристические методы решения. Так, методика устранения неисправности в электронном блоке путем замены подозрительных узлов оказывается значительно более эффективной, чем анализ цепей в поиске вышедшей из строя детали.

Но помимо неточных знаний, неопределенность может быть внесена и *неточными* или *ненадежными данными* о конкретной ситуации. Любой датчик имеет ограниченную разрешающую способность и не стопроцентную надежность. При составлении отчета могут быть допущены ошибки или в них могут попасть недостоверные данные. Помимо всего прочего, существует еще фактор времени. Не всегда есть возможность быстро получить необходимые данные, когда ситуация требует принятия срочного решения.

Таким образом, эксперты могут пользоваться неточными методами по двум главным причинам:

- точных методов не существует;
- точные методы существуют, но не могут быть применены из-за отсутствия необходимого объема данных или невозможности их накопления по соображениям стоимости, риска или из-за отсутствия времени на сбор необходимой информации.

Как учитывается неопределенность в экспертных системах? Рассмотрим простую ситуацию. Пусть используется правило

Если А, то В

и предположим, что никакие другие правила и посылки не имеют отношения к рассматриваемой ситуации. Где же возникает неопределенность? В экспертных системах она может быть двух типов:

- неопределенность в *истинности* самой *посылки* (например, если степень уверенности в том, что A истинно составляет 90%, то какие значения примет B ?);
- неопределенность *самого* правила (например, мы можем сказать, что в большинстве случаев, но не всегда, если есть A , то есть также и B).

Еще более сложная ситуация возникает в случае, если правило имеет вид:

Если A и B , то C

где мы можем с некоторой степенью быть уверены как в истинности каждой из посылок (A , B), а тем более их совместного проявления, так и в истинности самого вывода.

Итак, источники неопределенности можно разделить на два вида: недостаточно полное знание предметной области и недостаточная информация о конкретной ситуации. Существуют четыре важные проблемы, которые возникают при проектировании и создании экспертных систем с неопределенными знаниями:

- как количественно выразить степень определенности при установлении истинности (или ложности) некоторой части правил?
- как выразить степень поддержки заключения конкретной посылкой?
- как использовать совместно две (или более посылки), независимо влияющие на заключение?
- как быть в ситуации, когда нужно получить цепочку вывода для подтверждения заключения в условиях неопределенности?

Мы рассмотрим наиболее известные методы нечеткого вывода: байесовские вероятностное рассуждение и его расширения, теорию уверенности, нечеткую логику.

6.2. Неточный вывод на основе фактора уверенности

Стандартные статистические методы основываются на допущении, что неопределенность является вероятностью того, что событие (или факт) истинно или ложно. В теории уверенности неопределенность представлена как степень уверенности. Теория уверенности использует коэффициенты уверенности. Коэффициент уверенности CF выражают доверие событию (факту или гипотезе), основанное на свидетельстве (или оценке эксперта). Существует несколько методов использования коэффициентов уверенности.

Стэнфордская теория факторов уверенности основывается на ряде наблюдений [10]. Во-первых, в традиционной теории вероятностей сумма вероятностей отношения и его отрицания должна равняться единице. Однако нередки ситуации, при которых человек-эксперт, оценив вероятность (достоверность) некоторого отношения значением, например, 0,7, полагает, что отношение истинно, и не учитывает, что оно может быть ложным. Еще одно предположение, подкрепляющее теорию фактора уверенности, состоит в том,

что знание самих правил намного важнее, чем знание алгебры для вычисления их достоверности. Мера уверенности (или доверия) – это неформальная оценка, которую человек–эксперт добавляет к заключению. Могут использоваться такие оценки как, например: «вероятно, это так», «почти наверняка, это так» или «совершенно невероятно».

Стэнфордская теория фактора уверенности определяет понятия доверия и недоверия и предлагает правила для объединения свидетельств при выводе заключения. Первым допущением является разделение меры доверия и недоверия («за» и «против») для каждого отношения.

Пусть $MB(H|E)$ – мера уверенности в гипотезе H при заданном свидетельстве E .

Обозначим через $MD(H|E)$ меру недоверности гипотезы H при заданном свидетельстве E . Тогда:

$$0 < MB(H|E) < 1, \text{ если } MD(H|E) = 0,$$

или

$$0 < MD(H|E) < 1, \text{ если } MB(H|E) = 0.$$

Эти две меры накладывают ограничения друг на друга, так как заданной считается часть свидетельств в пользу данной гипотезы, либо против нее. Если связь между мерами доверия и недоверия установлена, то можно вычислить фактор уверенности, характеризующий данную гипотезу:

$$CF(H|E) = MB(H|E) - MD(H|E).$$

С приближением фактора уверенности CF к 1 усиливается доверие к гипотезе, а с приближением CF к -1 – ее отрицание. Близость значения CF к 0 означает, что свидетельств в пользу гипотезы и против нее слишком мало, либо эти свидетельства сбалансированы.

Когда эксперты формируют базу правил, они сопоставляют с каждым правилом определенное значение фактора уверенности CF . Фактор CF отражает их уверенность в надежности правила.

Коэффициенты уверенности можно использовать при объединении различных оценок экспертов различными способами. Наиболее приемлемым способом их объединения в системах, основанных на правилах, является подход, использованный в системе EMYCIN.

При этом подходе мы различаем два случая.

1. Объединение нескольких коэффициентов уверенности в одном правиле.

Предпосылка каждого правила состоит из ряда фактов, связанных операциями конъюнкции и дизъюнкции. При использовании продукционного правила учитываются факторы доверия, связанные с каждым условием предпосылки. Их сочетание определяет меру доверия всей предпосылки следующим образом.

Для предпосылок $P1$ и $P2$

$$CF(P1 \text{ и } P2) = \text{MIN}(CF(P1), CF(P2))$$

и

$$CF(P1 \text{ или } P2) = \text{MAX}(CF(P1), CF(P2)).$$

Для получения фактора уверенности в заключении правила объединенный фактор уверенности в предпосылках CF , полученный с помощью приведенных правил, умножается на CF самого правила.

Пример

Рассмотрим, например, следующее правило базы знаний

$$(P1 \text{ и } P2) \text{ или } P3 \rightarrow R1(0,7) \text{ и } R2(0,3),$$

где $P1$, $P2$, и $P3$ – предпосылки, а $R1$ и $R2$ – заключения правила с фактами доверия CF , равным соответственно 0,7 и 0,3 соответственно. Эти числа добавляются к правилу при его разработке и представляют уверенность эксперта в выводе, если все предпосылки известны с полной определенностью.

Меру доверия к предпосылкам правила вычисляем следующим образом:

$$CF(P1(0,6) \text{ и } P2(0,4)) = \min(0,6; 0,4) = 0,4,$$

$$CF(0,4 \text{ или } P3(0,2)) = \max(0,4; 0,2) = 0,4.$$

Значение CF для заключения правила $R1$ равно 0,7, так что коэффициент уверенности правила равен $CF = 0,7 \cdot 0,4 = 0,28$. Значение CF для заключения правила $R2$ равно 0,3, так что коэффициент уверенности этого правила равен $CF = 0,3 \cdot 0,4 = 0,12$.

2. Объединение двух и более правил.

Требуется определить еще одну метрику. Как объединить несколько значений CF , если имеются альтернативные правила? В этом случае меры уверенности при объединении независимых свидетельств перемножаются. Многократно используя это правило, можно объединять результаты любого количества правил, используемых для определения результата R . Если $CF(R1)$ представляет фактор уверенности в результате R , а ранее не использованное правило приводит к результату R (снова) со значением $CF(R2)$, то новое значение CF результата вычисляется следующим образом [10]:

$CF(R1) + CF(R2) - (CF(R1) \cdot CF(R2))$, если $CF(R1)$ и $CF(R2)$ положительны,

$CF(R1) + CF(R2) + (CF(R1) \cdot CF(R2))$, если $CF(R1)$ и $CF(R2)$ отрицательны и

$$\frac{CF(R1) + CF(R2)}{1 - \min(|CF(R1)|, |CF(R2)|)}, \text{ если оба коэффициента имеют разные знаки.}$$

Кроме легкости вычислений эти уравнения имеют другие полезные свойства. Во-первых, значение фактора CF , вычисленное согласно этому правилу, всегда будет лежать между 1 и -1. Во-вторых, в результате объединения противоположные значения CF сокращаются, что тоже является положительным моментом. И, наконец, комбинированная мера CF является монотонно возрастающей (убывающей) функцией.

Итак, мера уверенности стэнфордской алгебры описывает человеческую (субъективную) оценку причинной вероятностной меры. Этот подход позволяет специалисту по знаниям описать все эти взаимосвязи одним фактором CF доверия к правилу, т.е. в виде

$$\text{ЕСЛИ } A \text{ И } B \text{ И } C, \text{ ТО } D (CF).$$

Эта простая алгебра отражает способ мышления человека-эксперта.

Теорию фактора уверенности можно подвергнуть критике как необоснованную. Несмотря на то, что она определяется в рамках формальной алгебры, значение меры доверия не так строго обосновано, как в теории вероятностей. Однако она обеспечивает компромисс, позволяющий экспертной системе объединять свидетельства по мере решения задачи. Эти меры являются эвристическими в том смысле, что уверенность эксперта в результатах является неполной, эвристической и неформальной.

Пример

Пусть имеем следующий фрагмент базы знаний.

Правило 1.

ЕСЛИ X получит кредиты (0,8)
И X запустит в производство новую продукцию (0,75),
ТО X будет нашим поставщиком (0,5).

Правило 2.

ЕСЛИ X покупает эту продукцию у А (0,4)
ИЛИ X покупает эту продукцию у В (0,6)
ТО X будет нашим поставщиком (0,7).

Определим коэффициент уверенности заключения в том, что «X будет нашим поставщиком».

$CF(\text{условий первого правила}) = \text{MIN}(0,8; 0,75) = 0,75.$

$CF(\text{условий второго правила}) = \text{MAX}(0,4; 0,6) = 0,6.$

$CF(\text{заключения первого правила}) = 0,75 \cdot 0,5 = 0,38.$

$CF(\text{заключения второго правила}) = 0,6 \cdot 0,7 = 0,42.$

$CF(\text{заключения, исходя из обоих правил}) = 0,38 + 0,42 - 0,38 \cdot 0,42 = 0,64.$

6.3. Рассуждения, основанные на нечетких множествах

Традиционная формальная логика основывается на двух предположениях. Первое связано с установлением принадлежности – для любого элемента и множества, принадлежащего некоторому универсуму, элемент является либо членом множества, либо членом дополнения множества. Второе предположение основано на законе исключения третьего, утверждающего, что элемент не может одновременно принадлежать множеству и его дополнению. Оба этих предположения нарушаются в теории нечетких множеств Лотфи Заде. С точки зрения нечетких множеств множества и законы рассуждений в рамках традиционной логики называют четкими.

Главное утверждение Л. Заде заключается в том, что теория вероятностей является хорошим инструментом для измерения случайности информации, но не подходит для измерения смысла информации. В самом деле, путаница, возникающая при использовании слов и фраз естественного языка, связана скорее с отсутствием ясности (определенности), чем со случайностью. Существует целый класс описаний, оперирующий качественными характеристиками объектов (много, мало, сильный, хороший, близкий и т.д.).

Эти характеристики обычно являются размытыми и их нельзя однозначно интерпретировать, однако они содержат важную информацию.

Для измерения меры неопределенности Заде предложил теорию возможностей, в то время как теория вероятностей позволяет определить меру случайности.

Аппарат нечеткой логики лежит в основе так называемых мягких вычислений (soft computing). Л. Заде ввел одно из главных понятий в нечеткой логике – понятие лингвистической переменной.

Лингвистическая переменная – это переменная, значение которой определяется набором вербальных характеристик некоторого свойства.

Значения лингвистической переменной определяются через так называемые нечеткие множества.

Нечеткое множество определяется через некоторую базовую шкалу и функцию принадлежности нечеткого множества – $\mu(x)$, $x \in F$, принимающую значения на интервале $[0, 1]$. Таким образом, нечеткое множество F – это совокупность пар вида $(x, \mu(x))$, где $x \in F$.

Функция принадлежности определяет субъективную степень уверенности эксперта в том, что данное конкретное значение базовой шкалы соответствует определяемому нечеткому множеству. Эту функцию не следует путать с вероятностью, носящей объективный характер и подчиняющейся другим математическим зависимостям.

В большинстве случаев функции принадлежности строятся на основе субъективных оценок по опросам экспертов. Так как эти оценки недостаточно адекватно характеризуют явление или объект, то для упрощения расчетов выбирают трапециевидную функцию принадлежности (рис. 6.1).

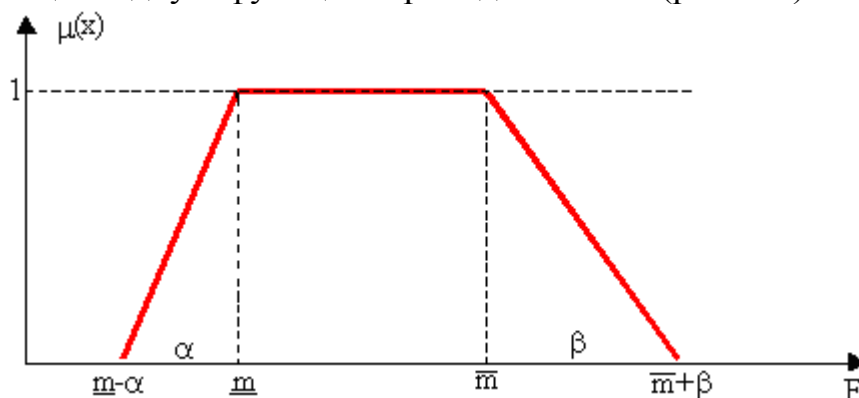


Рис. 6.1. Трапециевидная функция принадлежности

Тогда $\mu(x)$ характеризуется четверкой $(\underline{m}, \bar{m}, \alpha, \beta)$. В частном случае при $\underline{m} = \bar{m}$ функция принадлежности является треугольной.

Пример

Пусть в рамках составления проекта бюджета рассматриваются различные источники финансирования. Некоторые из них характеризуются неточностью оценки денежных сумм, другие отличаются малой надежностью. Кроме того, из

бюджета необходимо отдать долги, количество которых неточно, т.к. зависит от того, потребует ли кредитор все или только часть долга.

Источник *A*: финансирование обеспечивается, его сумма может изменяться от 40 до 100 млн., в зависимости от конъюнктуры, но с наибольшей вероятностью можно ожидать поступления в сумме от 50 до 70 млн. руб.

Источник *B*: источник надежен, разумно предполагать, что финансирование будет предоставлено в сумме 100–110 млн. руб.

Источник *C*: ненадежен, а если и даст, то не более 20 млн. руб.

Долг *D*: плата за кредиты – 50–100 млн. руб., но наиболее вероятна выплата 80 млн. руб.

Таким образом, имеем три источника поступлений и один источник расхода. Построим функции принадлежности для каждой из четырех нечетких переменных (рис. 6.2).

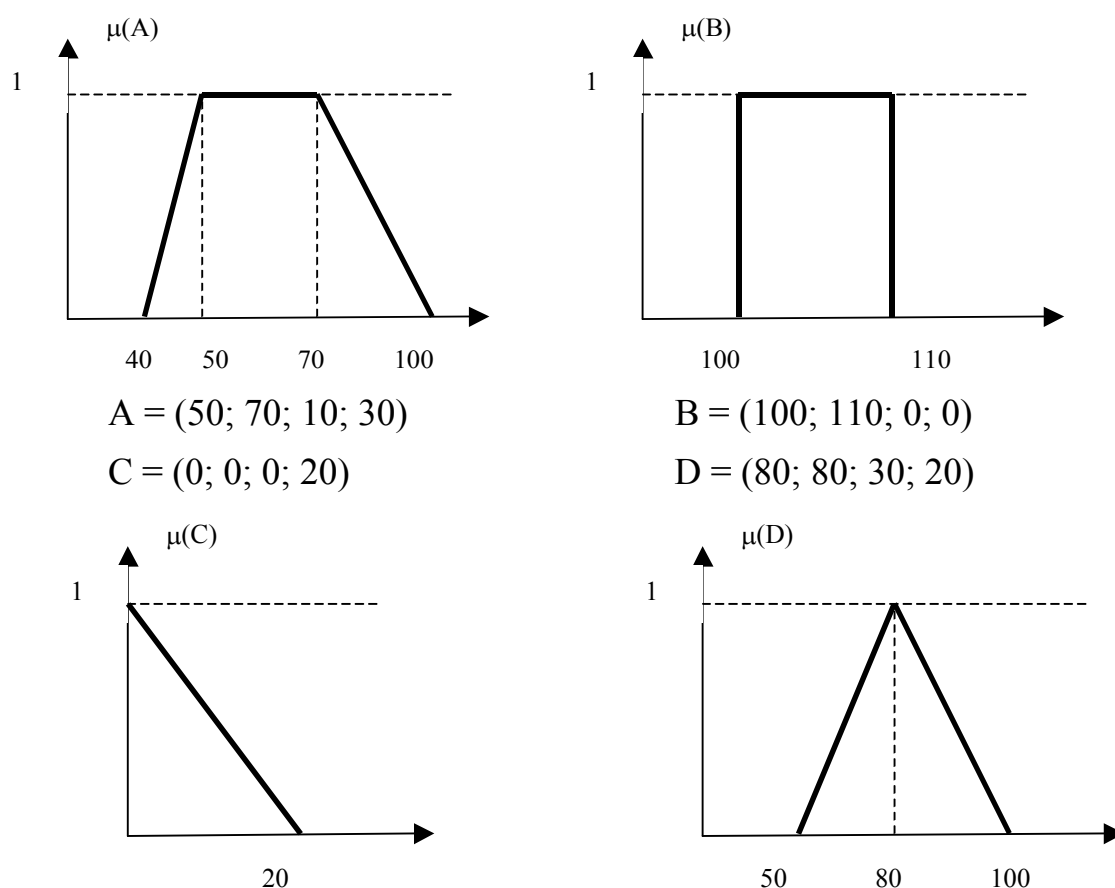


Рис. 6.2. Функции принадлежности составляющих оценки бюджета

После задания всех нечетких переменных встает задача определения суммы всего бюджета, которая также будет нечеткой величиной. Для этого надо уметь выполнять простейшие арифметические операции над нечеткими переменными.

Определение этих операций рассмотрим для случая двух нечетких переменных $A1$ и $A2$, которые заданы своими трапецевидными функциями принадлежности вида

$$A1 = (\underline{m1}, \overline{m1}, \alpha1, \beta1),$$

$$A2 = (\underline{m2}, \overline{m2}, \alpha2, \beta2).$$

Результатом операции также будет нечеткая переменная $A = (\underline{m}, \overline{m}, \alpha, \beta)$, которая также имеет трапецевидную функцию принадлежности, параметры которой определяются в зависимости от вида арифметической операции (табл. 6.1).

Таблица 6.1

Тип операции	Зависимости параметров функций принадлежности
$A = A1 (+) A2$	$\underline{m} = \underline{m1} + \underline{m2}, \overline{m} = \overline{m1} + \overline{m2}, \alpha = \alpha1 + \alpha2, \beta = \beta1 + \beta2$
$A = A1 (-) A2$	$\underline{m} = \underline{m1} - \underline{m2}, \overline{m} = \overline{m1} - \overline{m2}, \alpha = \alpha1 + \beta2, \beta = \beta1 + \alpha2$

Продолжение примера

Определим оценку бюджета без учета долгов:

$B = A (+) B (+) C = (50 + 100 + 0; 70 + 110 + 0; 10 + 0 + 0; 30 + 0 + 20) = (150; 180; 10; 50)$ (рис. 6.3, а).

Для получения полной оценки бюджета из полученного результата следует вычесть предполагаемые выплаты по кредитам (рис. 6.3, б):

$B1 = B (-) D = (150 - 80; 180 - 80; 10 + 20; 50 + 30) = (70; 100; 30; 80)$.

Таким образом, в бюджете может быть сумма от 40 до 180 млн. руб., но с наибольшей степенью уверенности можно говорить о суммах от 70 до 100 млн. руб.

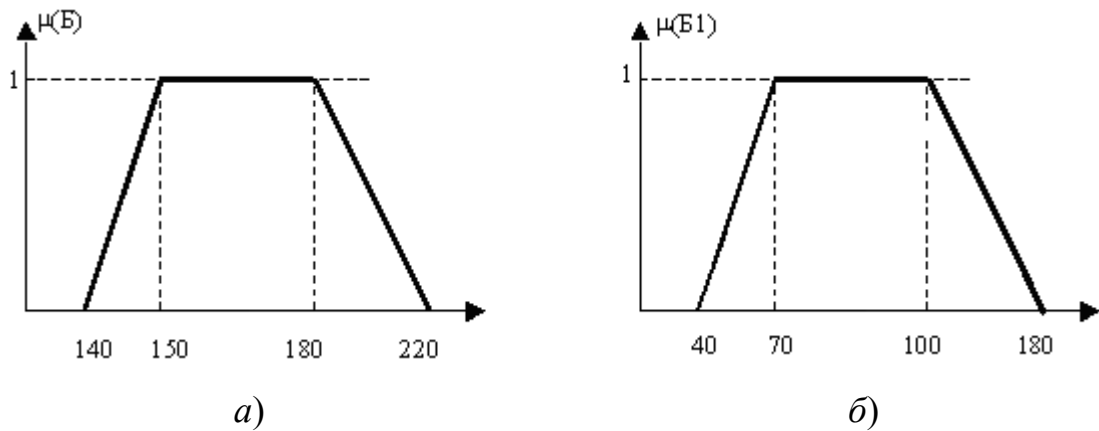


Рис. 6.3. Оценка бюджета

Мы привели один пример использования нечеткого вывода для решения финансовой задачи. Системы, основанные на нечеткой логике, успешно работают в различных отраслях. Одной из первых сфер, где они нашли применение, стала военная промышленность. Аппарат нечеткой логики позволил создавать высокоточные системы управления оружием. К числу отраслей, в которых использование нечеткой логики наиболее заметно, относятся:

- автомобильная промышленность (системы управления двигателями, трансмиссиями, антиблокировочные тормозные системы);
- аэрокосмическая промышленность (высокопроизводительные системы управления самолетами и космическими аппаратами);

- приборостроение и производство бытовой техники (стиральные машины, телевизоры, видеокамеры, фотоаппараты и др.);
- системы управления производством и транспортом;
- анализ и прогнозирование в сфере политики и экономики;
- финансы (системы управления портфелем ценных бумаг, системы анализа рисков);
- анализ данных (системы классификации, кластеризации и распознавания образов).

Применение нечеткой логики в этих отраслях позволяет повысить эффективность создаваемых систем, уменьшить время и затраты на их разработку.

6.4. Байесовские рассуждения

Рассмотрим случай, когда правила экспертной системы имеют вид

ЕСЛИ H является истинным,

ТО C наблюдается с вероятностью p

Это правило подразумевает, что если событие H происходит, то вероятность того, что событие C произойдет, равна p . Что будет, если событие C произошло, но мы не знаем, произошло ли событие H ? Иными словами, является ли H причиной события C ?

В контексте интеллектуальных систем:

C – событие, заключающееся в том, что данная гипотеза верна;

H – событие, заключающееся в том, что наступило определенное доказательство (свидетельство), которое может подтвердить правильность указанной гипотезы.

Согласно формуле Байеса, примененной к данной ситуации, имеем

$$p(H | C) = \frac{p(C | H) \times p(H)}{p(C | H) \times p(\overline{H}) + p(C | \overline{H}) \times p(\overline{H})}, \quad (6.1)$$

$p(H)$ – априорная (безусловная) вероятность того, что гипотеза H является истинной;

$p(C|H)$ – вероятность того, что гипотеза H является истинной, при условии, что наблюдается событие C ;

$p(\overline{H})$ – априорная (безусловная) вероятность того, что гипотеза H является ложной;

$p(C|\overline{H})$ – вероятность нахождения свидетельства C , даже когда гипотеза H ложна.

Уравнение (6.1) предполагает, что вероятность гипотезы H должна быть определена перед тем, как получены какие-либо свидетельства в ее пользу.

В интеллектуальных информационных системах эксперт определяет априорные вероятности для возможной гипотезы $p(H)$ и $p(\overline{H})$, а также условные вероятности для наблюдаемого свидетельства C , если гипотеза H истинна, то

$p(C|H)$, и если гипотеза H ложна, то $p(C|\bar{H})$. Вероятность $p(C|H)$ называется апостериорной вероятностью гипотезы H при наблюдаемом свидетельстве C .

Рассмотрим ситуацию, когда имеется простое свидетельство C , которое подтверждается многочисленными гипотезами H_1, H_2, \dots, H_m , или когда имеются многочисленные свидетельства C_1, C_2, \dots, C_n , которые подтверждают множество гипотез.

Мы можем обобщить уравнение 6.1 для обоих случаев, однако следует помнить, что гипотезы, как и свидетельства, должны быть взаимно исключающими. Для свидетельства C и многочисленных гипотез H_1, H_2, \dots, H_m следует:

$$p(H_i | C) = \frac{p(C | H_i) \times p(H_i)}{\sum_{k=1}^n (p(C | H_k) \times p(H_k))} \quad (6.2)$$

Для многочисленных свидетельств C_1, C_2, \dots, C_n и многочисленных гипотез следует H_1, H_2, \dots, H_m

$$p(H_i | C_1, C_2, \dots, C_n) = \frac{p(C_1, C_2, \dots, C_n | H_i) \times p(H_i)}{\sum_{k=1}^n (p(C_1, C_2, \dots, C_n | H_k) \times p(H_k))} \quad (6.3)$$

Применение уравнения 6.3 требует от эксперта получения условных вероятностей всех возможных комбинаций свидетельств для всех гипотез, что практически невыполнимо.

Поэтому в интеллектуальных системах допускается условная независимость между различными событиями.

Два события C_1 и C_2 являются условно независимыми, если их совместная вероятность при условии некоторой гипотезы H равна произведению условных вероятностей этих событий при условии H , т.е.

$$p(C_1 C_2 | H) = p(C_1 | H) \times p(C_2 | H).$$

Следовательно, вместо уравнения 6.3 можно записать уравнение 6.4

$$p(H_i | C_1, C_2, \dots, C_n) = \frac{p(C_1 | H_i) \times p(C_2 | H_i) \times \dots \times p(C_n | H_i) \times p(H_i)}{p(C_1 | H_k) \times p(C_2 | H_k) \times \dots \times p(C_n | H_k) \times p(H_k)} \quad (6.4)$$

Для того, чтобы пояснить, каким образом интеллектуальная система вычисляет все апостериорные вероятности и в итоге ранжирует потенциально истинную гипотезу, рассмотрим простой пример.

Пример

Предположим, что в некоторой базе знаний имеется три взаимно независимых гипотезы: H_1, H_2, H_3 , которые имеют априорные вероятности: $p(H_1), p(H_2), p(H_3)$, соответственно. Правила базы знаний содержит два условно независимых свидетельства, которые поддерживают исходные гипотезы в различной степени. Априорные и условные вероятности всех гипотез и свидетельств этого примера имеют следующие значения:

$\begin{matrix} i \\ p \end{matrix}$	1	2	3
$p(H_i)$	0,5	0,3	0,2
$p(C_1 H_i)$	0,4	0,8	0,3
$p(C_2 H_i)$	0,7	0,9	0,0

При этом исходные гипотезы характеризуют событие, связанное с определением надежности некоторой фирмы:

H_1 – «средняя надежность фирмы»;

H_2 – «высокая надежность фирмы»;

H_3 – «низкая надежность фирмы».

Событиями, являющимися условно независимыми свидетельствами, поддерживающими исходные гипотезы, являются: C_1 – «наличие прибыли у фирмы», C_2 – «своевременный расчет с бюджетом».

В процессе сбора фактов вероятности гипотез будут повышаться, если факты поддерживают их или уменьшаться, если опровергают их. Предположим, что мы имеет только одно свидетельство (то есть с вероятностью единица наступил факт C_1). Наблюдая C_1 , мы вычисляем априорные вероятности для гипотез согласно формуле Байеса для одного свидетельства.

Таким образом

$$p(H_1 | C_1) = \frac{0,4 \times 0,5}{0,4 \times 0,5 + 0,8 \times 0,3 + 0,3 \times 0,2} = 0,40$$

$$p(H_2 | C_1) = \frac{0,8 \times 0,3}{0,4 \times 0,5 + 0,8 \times 0,3 + 0,3 \times 0,2} = 0,48$$

$$p(H_3 | C_1) = \frac{0,3 \times 0,2}{0,4 \times 0,5 + 0,8 \times 0,3 + 0,3 \times 0,2} = 0,12$$

После того, как C_1 произошло, доверие к гипотезам H_1 и H_3 понизилось, в то время, как доверие к H_2 возросло. В тех случаях, когда имеются факты, подтверждающие как событие C_1 , так и событие C_2 , то апостериорные вероятности исходных гипотез также можно вычислить по правилу Байеса. Для условно независимых событий имеем:

$$p(H_3 | C_1 C_2) = \frac{0,3 \times 0,9 \times 0,2}{0,4 \times 0,7 \times 0,5 + 0,8 \times 0,9 \times 0,3 + 0,3 \times 0,0 \times 0,2} = 0,151,$$

$$p(H_2 | C_1 C_2) = \frac{0,8 \times 0,9 \times 0,3}{0,4 \times 0,7 \times 0,5 + 0,8 \times 0,9 \times 0,3 + 0,3 \times 0,0 \times 0,2} = 0,607,$$

$$p(H_1 | C_1 C_2) = \frac{0,4 \times 0,7 \times 0,5}{0,4 \times 0,7 \times 0,5 + 0,8 \times 0,9 \times 0,3 + 0,3 \times 0,0 \times 0,2} = 0,393.$$

Таким образом, мы видим, что наиболее вероятным событием является событие H_2 .

Однако реально, распространение вероятностей происходит поэтапно с суммированием отдельных свидетельств и их влияния на условную вероятность по мере поступления отдельных значений C .

Для использования теоремы Байеса существуют два главных требования. Первое состоит в том, что необходимо получить от экспертов все субъективные вероятности взаимосвязей наступления события с различными гипотезами. Второе, и иногда более трудное в определении требование заключается в том, что нужно вычислить все взаимосвязи между событием и гипотезами, т.е. $p(C|H_k)$. Хотя и предполагается условная независимость гипотез для уменьшения числа требуемых вероятностных оценок, все же число оценок остается большим. Поэтому теория субъективных вероятностей в чистом виде редко используется в экспертных системах.

Таким образом, несмотря на то, что байесовская теория вероятностей обеспечивает математическую основу для рассуждений в условиях неопределенности, сложность, возникающая при ее применении к реальным предметным областям, может оказаться недопустимой. На практике используются байесовские сети доверия, к обсуждению которых мы сейчас и приступим.

6.5. Байесовские сети доверия

Байесовские сети доверия позволяют ослабить многие ограничения байесовской модели. Сети доверия позволяют уменьшить таблицу условных вероятностей, содержащую вероятности всех возможных комбинаций событий и свидетельств. При их построении эксперт выбирает главные явления, которые заведомо связаны друг с другом, полагая остальные события условно независимыми.

Байесовские сети доверия применяются для моделирования ситуаций, в которых случайные события соединены причинно-следственными связями.

В рассуждении эксперта, использующего причинно-следственный вывод, неявно предполагается, что эти влияния являются направленными, т.е. реализация некоторого события вызывает другие события в сети. Кроме того, рассуждение причинного влияния не является циклическим, а раз так, воздействие не может вернуться назад, чтобы вызвать себя.

Можно выделить три типа связей между узлами (рис. 6.4) [10].

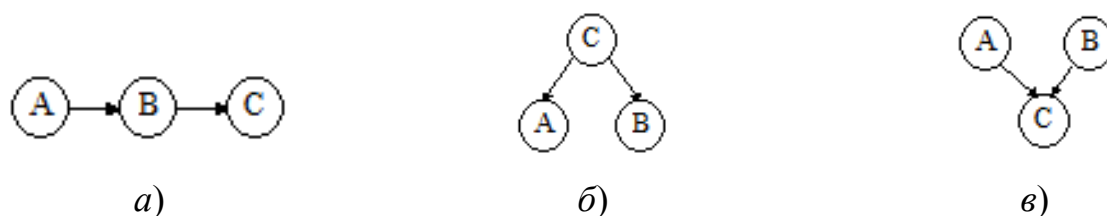


Рис. 6.4. Типы связей между узлами

Последовательная связь узлов показана на (рис. 6.4, а), в которой влияние от A к C распространяется до тех пор, пока существует B ; расходящаяся связь узлов

(рис. 6.4, б), где информация распространяется к наследникам C до тех пор, пока определено значение C ; сходящаяся связь (рис. 6.4. в) если ничего не известно о C то его родители независимы, в противном случае между родителями существует корреляция.

Покажем, как использование байесовской сети доверия упрощает вычисления условных вероятностей. По закону Байеса любое объединенное распределение вероятностей может рассматриваться как произведение условных вероятностей. На рис. 6.4, а условная вероятность для пути от A до B и от B до C вычисляется следующим образом

$$p(A, B, C) = p(A) \cdot p(B|A) p(C|A, B).$$

В байесовских сетях доверия условная вероятность переменной при заданных значениях вероятностей всех ее предшественников равна условной вероятности при заданных значениях лишь для родителей. В результате в приведенном выше уравнении $p(C|A, B)$ заменяется на $p(C|B)$, поскольку B является прямым потомком C , а A – нет. Объединенные вероятностные распределения для трех сетей, показанных на рис. 6.5, вычисляются следующим образом.

а) $p(A, B, C) = p(A) \cdot p(B|A) p(C|B).$

б) $p(C, A, B) = p(C) \cdot p(A|C) p(B|C).$

в) $p(A, B, C) = p(A) \cdot p(B) p(C|A, B).$

Определим теперь более точно основные понятия, используемые в байесовских сетях доверия.

Определение

Байесовские сети доверия – это направленный ациклический граф, обладающий следующими свойствами:

- каждая вершина представляет собой событие, описываемое случайной величиной, которая может иметь несколько состояний;
- все вершины, связанные с «родительскими» определяются таблицей условных вероятностей или функцией условных вероятностей;
- для вершин без «родителей» вероятности ее состояний являются безусловными (маргинальными).

Другими словами в байесовских сетях доверия вершины представляют собой случайные переменные, а дуги – вероятностные зависимости, которые определяются через таблицы условных вероятностей. Таблица условных вероятностей состояний каждой вершины содержит вероятности состояний этой вершины при условии состояний ее «родителей».

Процесс рассуждений (вывода) в сетях доверия заключается в пересчете вероятностей узлов при поступлении нового свидетельства. Процесс распространения вероятностей основывается на механизме пересчета, который мы подробно рассматривать не будем.

Одними из наиболее распространенных для современных персональных ЭВМ программных систем, реализующих теорию байесовских сетей доверия, являются:

- «MSBN» фирмы Microsoft;
- «Hugin» фирмы Hugin AIS, Дания.

В лабораторных работах мы познакомимся с системой Hugin, позволяющей создавать экспертные системы, основанные на теореме Байеса и использующие байесовские сети доверия.

При построении сети доверия мы полагали, что каждое из событий, связанное с вершиной, характеризуется конечным множеством состояний и вероятностями пребывания в каждом из них. Однако во многих случаях события могут принимать любые состояния из некоторого диапазона, т.е. они характеризуются непрерывной случайной величиной. Распределение вероятностей непрерывной случайной величины определяется функцией распределения вероятностей и плотностью распределения вероятностей. Однако во многих практических задачах оказывается трудно или даже невозможно полностью описать эти функции.

В то же время для решения многих задач достаточно знать лишь некоторые параметры, характеризующие случайную величину с той или иной точки зрения. Наиболее распространенными числовыми характеристиками непрерывных случайных величин являются математическое ожидание и дисперсия.

Математическое ожидание является величиной, вокруг которой группируются значения случайной переменной. Дисперсия характеризует отклонение значений случайной величины от математического ожидания, то есть является характеристикой рассеивания случайной величины. Чем меньше дисперсия, тем более тесно группируются отдельные значения случайной величины вблизи математического ожидания.

Однако в ряде случаев дисперсия оказывается также неудобной для практического применения, так как имеет размерность квадрата случайной величины. Поэтому в качестве характеристики рассеивания случайной величины часто используется среднеквадратичное отклонение (корень квадратный из значения дисперсии).

В лабораторной работе мы рассмотрим пример построения байесовской сети доверия с непрерывными и дискретными вершинами.

Здесь же мы обратим свое внимание на построение диаграмм влияния.

6.6. Диаграммы влияния

Диаграммы влияния используются для принятия решений. Фактически диаграммы влияния – это байесовские сети доверия, расширенные понятиями пользы и решения. Байесовские сети доверия содержат только один тип вершин, которые мы будем называть вершинами шансов. Эти вершины соответствуют состоянию случайных переменных. В диаграмме влияния используются, как минимум два типа вершин: вершины *решения* и вершины *пользы*.

Вершины решения, а точнее указания, содержащиеся в них, определяют порядок принятия решений во времени (рис. 6.6):

- стрелка от случайной переменной (вершины шансов) к переменной решения (вершине решения) указывает, что значение случайной переменной известно на момент принятия решения;
- стрелка от переменной решения к какой-либо другой переменной указывает порядок выполнения событий, связанных с узлами шансов во времени.

На рис.6.6 буквой V обозначена вершина шансов, буквой R – вершина решения, а буквой U – вершина пользы.

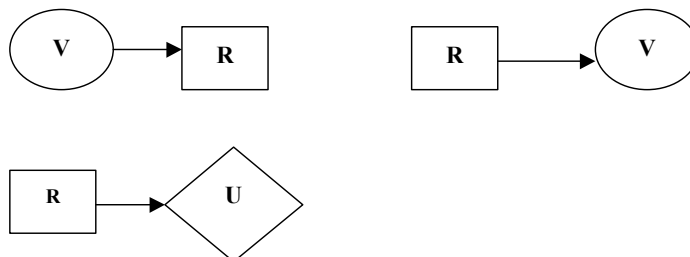


Рис. 6.6. Типовые узлы диаграммы влияния

При этом сеть должна оставаться ациклической и должен существовать путь, содержащий все вершины решения в сети.

В процессе принятия решения важно не просто найти решение, а найти решение, наилучшее в некотором смысле. С этой целью в диаграммах влияния с состоянием сети связываются «вершины пользы».

Каждая вершина пользы содержит функцию полезности, которая связывает каждую конфигурацию состояния ее родителей с полезностью. Вершины полезности не имеют наследников (а, следовательно, стрелка может быть направлена только к ним, рис. 6.7).



Рис. 6.7. Связи вершины полезности

Принимая решение, мы исходим из вероятности конфигурации сети. Поэтому можно вычислить ожидаемую полезность каждой альтернативы и выбрать альтернативу с наибольшей ожидаемой полезностью. Это принцип максимальной ожидаемой полезности.

Диаграмма влияния может содержать несколько вершин полезности. При этом общая функция полезности представляет собой сумму всех локальных функций полезности:

$$F(x_1, \dots, x_n) = \sum_{i=1}^k f_i(x_1, \dots, x_n)$$

Система должна вычислить полезность каждого решения в предположении, что все будущие решения будут сделаны оптимально, используя все имеющиеся свидетельства в момент каждого решения.

6.7. Выводы

1. Различают три аспекта неполноты информации: неточность, неопределенность, нечеткость. Неточные данные задаются в некотором интервале. Понятие неопределенности связано со степенью истинности некоторого утверждения. Нечеткость данных связана с заданием функции принадлежности элементов множества, при этом семантику функции принадлежности можно задать как распределение возможностей. Нечеткие, расплывчатые категории возникают там, где представления человека о процессах, и явлениях выражаются с помощью недостаточно определенных качественных оценок.

2. При неточности знаний используется теория факторов уверенности, которая позволяет указать для правила продукции степень уверенности эксперта в применимости правила в конкретной ситуации.

3. При неточности данных используются методы нечеткой логики, которые позволяют строить логико-лингвистические модели, использующие качественные представления, соответствующие способам принятия решений человеком.

4. При учете неопределенностей при проектировании экспертных систем используется субъективный или основанный на суждениях взгляд. Он заключается в том, что вероятностная мера рассматривается как степень доверия тому, как отдельный эксперт судит об истинности или ложности некоторого высказывания. При такой вероятностной интерпретации используется схема вывода, основанная на теореме Байеса.

5. Новым инструментом для решения перечисленных проблем являются сети доверия, в которых регулируются информационные потоки.

6. Для принятия решения используются диаграммы влияния, которые дополняют байесовские сети доверия вершинами принятия решений и вершинами полезности, с помощью которых оценивается принятое решение.

6.8. Вопросы для самопроверки

1. Какие классы неопределенностей можно выделить?
2. Какие современные теории описывают различные классы неопределенностей?
3. С чем связана неточность данных и неточность знаний?
4. В чем заключается теория факторов уверенности. Кратко охарактеризуйте условия ее применения. Как объединяются факторы уверенности правых частей правил?

5. Дайте определение функции принадлежности. В чем состоит ее смысл? Какой вид функций принадлежности применяется на практике? Укажите параметры функции принадлежности.

6. Укажите основные проблемы, возникающие при проектировании экспертных систем с неопределенными знаниями. Как эти проблемы решаются в рамках теории функций принадлежности?

7. Поясните, в каких случаях для описания неопределенностей применяется теорема Байеса.

8. Поясните отличия апостериорной вероятности от априорной. В тексте лекции мы рассмотрели пример маркетингового исследования. Укажите априорную и апостериорную вероятности в этом примере.

9. Почему при использовании теоремы Байеса требуется независимость событий?

10. Для чего применяются байесовские сети доверия? Какие типы вершин и связей в них используются?

11. Для чего применяются диаграммы влияния? Какие типы вершин и связей в них используются? Почему сеть должна быть ациклической? Что означает это требование для базы знаний экспертных систем?

12. Как вы думаете, для какого типа задач можно использовать байесовские сети доверия и диаграммы влияния. Приведите примеры.

7. НЕЙРОННЫЕ СЕТИ

Нейронные сети представляют собой новую и весьма перспективную вычислительную технологию, дающую новые подходы к исследованию динамических задач в различных областях.

Нейронные сети можно реализовать в виде быстрых аппаратных устройств, однако большинство практических приложений выполнено с использованием программного моделирования на обычных компьютерах. В большинстве случаев нейронные сети используются тогда, когда невозможно написать подходящую программу, или по причине того, что найденное нейронной сетью решение оказывается более совершенным.

Нейронные сети решают задачи следующего класса [8].

1. *Распознавание образов.* Задача состоит в отнесении входного набора данных, представляющего распознаваемый объект, к одному из заранее известных классов. В число таких задач входит распознавание рукописных и печатных символов при оптическом вводе в ЭВМ, распознавание клеток крови, распознавание речи и др.

2. *Кластеризация данных.* Задача состоит в группировке входных данных по присущей им «близости». Количество кластеров заранее не известно.

3. *Аппроксимация функций.* Имеется набор экспериментальных данных $\{(x_1, y_1), \dots, (x_n, y_n)\}$, представляющий значения y_i неизвестной функции от аргумента x_i , $i = 1, \dots, n$. Требуется найти функцию, аппроксимирующую неизвестную функцию, и удовлетворяющую некоторым критериям. Эта задача актуальна при моделировании сложных систем управления и при создании систем управления сложными динамическими объектами.

4. *Прогнозирование.* Имеется набор $\{y(t_1), \dots, y(t_n)\}$ значений y , представляющий поведение системы в моменты времени t_i , $i = 1, \dots, n$. Требуется по предыдущему поведению системы предсказать ее поведение $y(t_{n+1})$, в момент времени t_{n+1} . Эта задача актуальна для систем принятия решения.

5. *Оптимизация.* Цель этих задач – найти решения NP-полной задачи удовлетворяющее ряду ограничений и оптимизирующее значение целевой функции. К числу таких задач относится, например, задача коммивояжера.

6. *Контекстно-адресуемая (ассоциативная) память.* Эта память позволяет считывать содержимое по частичному или искаженному представлению входных данных. Основной областью применения являются мультимедийные базы данных.

Сегодня это та область знаний, о которой должны иметь определенное представление специалисты, работающие в области компьютерных информационных технологий.

7.1. Формальная модель нейрона

Нейронные сети строятся на основе искусственного (или формального) нейрона, который является простым процессором и выполняет взвешенное суммирование своих входов с последующим нелинейным преобразованием результата. На рис. 7.1 показана модель нейрона, лежащего в основе искусственных нейронных сетей. В этой модели можно выделить три основных элемента [17].

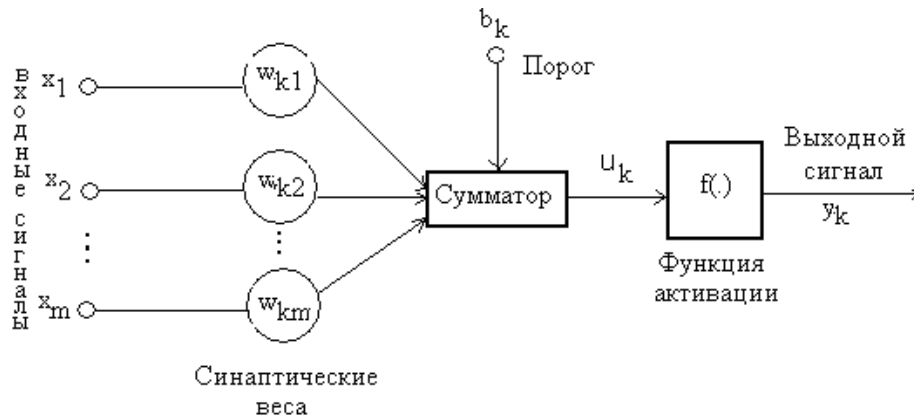


Рис. 7.1. Нелинейная модель нейрона

1. Набор *синапсов или связей*, каждый из которых характеризуется своим весом. Например, сигнал x_j на входе синапса j , связанного с нейроном k , умножается на вес w_{kj} . Синаптический вес искусственного нейрона может иметь как положительные, так и отрицательные значения.

2. *Сумматор* складывает входные сигналы, взвешенные относительно соответствующих синапсов нейрона. Эту операцию можно описать как линейную комбинацию. Отрицательное значение веса соответствует подавлению активности соответствующего элемента, положительное значение — усилению его активности.

3. *Функция активации* ограничивает амплитуду выходного сигнала нейрона. Эта функция также называется функцией сжатия. Обычно нормализованный диапазон амплитуд выхода нейрона лежит в интервале $[0, 1]$ или $[-1, 1]$.

В модель нейрона, показанную на рис.7.1, включен пороговый элемент, который обозначен символом b_k . Эта величина отражает увеличение или уменьшение входного сигнала, подаваемого на функцию активации.

В математическом представлении функционирование нейрона k можно описать следующей парой уравнений:

$$u_k = \sum_{j=1}^m w_{kj} x_j, \quad (7.1)$$

$$y_k = f(u_k + b_k), \quad (7.2)$$

где x_1, x_2, \dots, x_m — входные сигналы; $w_{k1}, w_{k2}, \dots, w_{km}$ — синаптические веса нейрона k ; u_k — линейная комбинация входных воздействий; b_k — порог; $f(.)$ — функция активации; y_k — выходной сигнал нейрона.

Структура связей обычно представляется в виде весовой матрицы W , в которой каждый элемент w_{kj} представляет величину весового коэффициента для связи, идущей от элемента k к элементу j . Матрица весов является памятью сети, хранящей информацию о том, как должна выполняться задача.

Использование порога обеспечивает эффект аффинного преобразования выхода линейного сумматора. В модели, показанной на рис. 7.1, выходной сигнал сумматора вычисляется следующим образом:

$$v_k = u_k + b_k. \quad (7.3)$$

В зависимости от того, какое значение принимает порог, положительное или отрицательное, *индуцированное локальное поле* или потенциал активации v_k нейрона k изменяется так, как показано на рис. 7.2.



Рис. 7.2. Преобразование, вызванное наличием порога

Здесь и далее для выхода линейного сумматора мы будем использовать термин «индуцированное локальное поле». Обратите внимание на результат преобразования. График v_k уже не проходит через начало координат, как график u_k .

Порог b_k является внешним параметром искусственного нейрона k . Принимая во внимание выражение (7.3), формулы (7.1) и (7.2) можно преобразовать к следующему виду

$$v_k = \sum_{j=0}^m w_{kj} x_j, \quad (7.4)$$

$$y_k = f(v_k) \quad (7.5)$$

В выражении (7.4) добавился новый синапс. Его входной сигнал $x_0 = +1$, а его вес: $w_{k0} = b_k$.

Это позволило трансформировать модель нейрона к виду, показанному на рис. 7.3. На этом рисунке видно, что в результате введения порога добавляется новый входной сигнал фиксированной величины $+1$, а также появляется новый

синаптический вес, равный пороговому значению. Модели, показанные на рис. 7.1 и 7.3, математически эквивалентны.

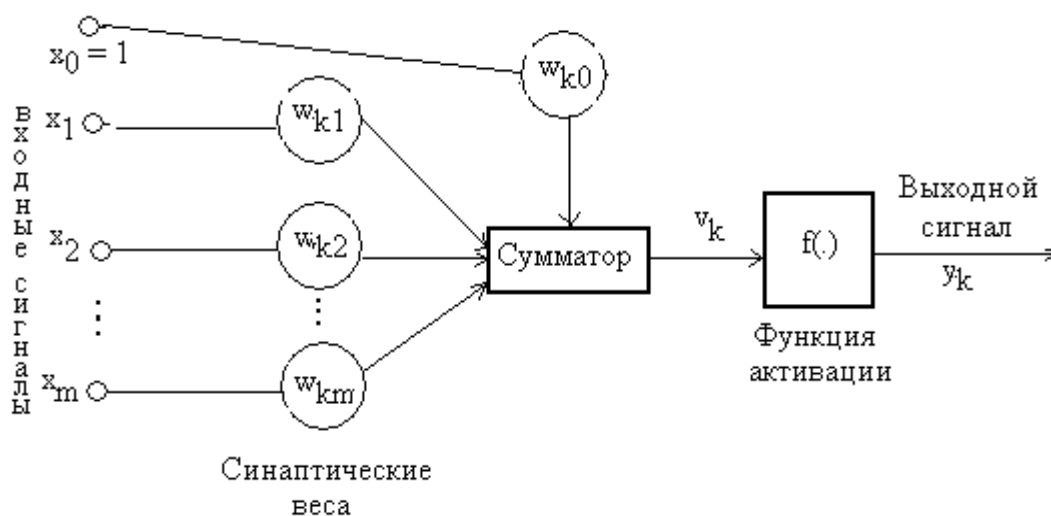


Рис. 7.3. Еще одна нелинейная модель нейрона

7.2. Типы функций активации

Функции активации, представленные в формулах как $f(v)$, определяют выходной сигнал нейрона в зависимости от значения v на выходе сумматора. Можно выделить три основных типа функций активации.

1. Функция *единичного скачка*, или *пороговая функция* (рис. 7.4). Этот тип функции показан на рис. 6 и описывается следующим образом:

$$f(v) = \begin{cases} 1, & \text{если } v_k \geq 0; \\ 0, & \text{если } v_k < 0. \end{cases}$$

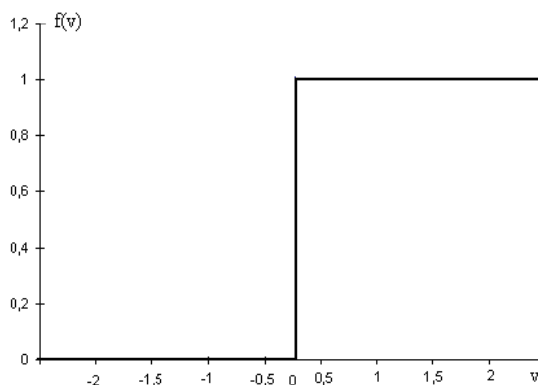


Рис. 7.4. Функция единичного скачка

Эта форма функции единичного скачка называется функцией *Хэвисайда*. Соответственно выходной сигнал такой функции можно представить как

$$y_k = \begin{cases} 1, & \text{если } v_k \geq 0; \\ 0, & \text{если } v_k < 0, \end{cases}$$

где $v_k = \sum_{j=1}^m w_{kj} x_j + b_k$.

Эту модель в литературе называют моделью *Мак-Каллока–Питца*. В этой модели выходной сигнал нейрона принимает значение 1, если индуцированное локальное поле этого нейрона не отрицательно, и 0 – в противном случае. Это выражение описывает свойство «все или ничего» модели Мак-Каллока–Питца.

2. *Кусочно-линейная функция*. Кусочно-линейная функция, показанная на рис. 7.5, описывается следующим выражением

$$f(v) = \begin{cases} 1, & v \geq +\frac{1}{2}, \\ |v|, & +\frac{1}{2} > v > -\frac{1}{2}, \\ 0, & v \leq -\frac{1}{2}, \end{cases}$$

где коэффициент усиления в линейной области оператора предполагается равным единице. Следующие два варианта можно считать особой формой кусочно-линейной функции.

- Если линейная область оператора не достигнет порога насыщения, он превращается в линейный сумматор.
- Если коэффициент усиления линейной области принять бесконечно большим, то кусочно-линейная функция вырождается в пороговую.

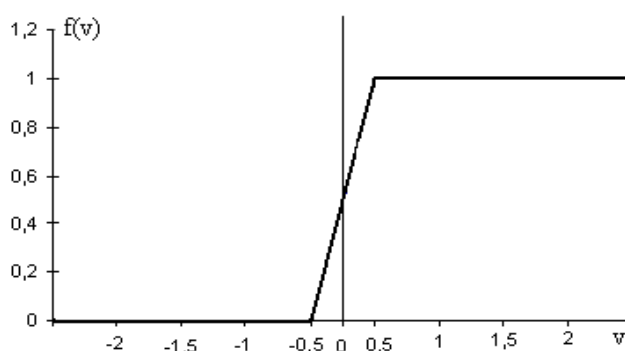


Рис. 7.5. Кусочно-линейная функция активации

3. *Сигмоидальная функция*. Сигмоидальная функция, график которой напоминает букву *S*, является одной из самых распространенных функций, используемых для создания искусственных нейронных сетей. Примером сигмоидальной функции является логистическая функция, задаваемая следующим выражением:

$$f(v) = \frac{1}{1 + \exp(-av)},$$

где *a* – параметр наклона сигмоидальной функции. Изменяя этот параметр, можно построить функции с разной крутизной (рис. 7.6). В пределе, когда параметр наклона достигает бесконечности, сигмоидальная функция вырождается в пороговую. Если пороговая функция может принимать только значения 0 и 1, то сигмоидальная функция принимает бесконечное множество значений в диапазоне от 0 до 1. При этом следует заметить, что сигмоидальная функция является дифференцируемой, в то время как пороговая – нет.

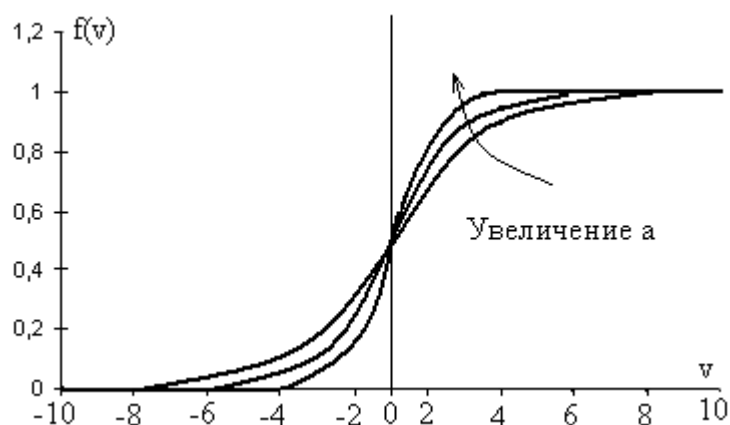


Рис. 7.6. Изменение сигмоидальной функции

7.3. Представление нейронных сетей с помощью графов

Блочные диаграммы, используемые нами для представления формального нейрона, обеспечивают функциональное описание различных элементов, из которых состоит модель нейрона. Внешний вид модели можно упростить, применив графы передачи сигналов.

Граф *передачи сигнала* – это сеть направленных связей, соединяющих отдельные узлы. С каждым узлом j связан сигнал x_j . Обычная направленная связь начинается в узле j и заканчивается в другом узле k . С ней связана *передаточная функция*, определяющая зависимость сигнала y_k узла k от сигнала x_j узла j . Прохождение сигнала по различным частям графа подчиняется трем основным правилам.

Правило 1. Направление прохождения сигнала вдоль каждой связи определяется направлением стрелки. При этом можно выделить два типа связей.

- *Синаптические связи.* Их поведение определяется линейным отношением вход-выход. Как показано на рис. 7.7, а, сигнал узла x_j умножается на синаптический вес w_{kj} , в результате чего получается сигнал узла y_k .

- *Активационные связи.* Их поведение определяется нелинейным отношением вход-выход. Этот вид связи показан на рис. 7.7, б, где $f(.)$ – нелинейная функция активации.

Правило 2. Сигнал узла равен алгебраической сумме сигналов, поступающих на его вход. На рис. 7.7, в это правило проиллюстрировано для случая *синаптической сходимости*.

Правило 3. Сигнал данного узла передается по каждой исходящей связи без учета передаточных функций исходящих связей. Это правило проиллюстрировано на рис. 7.7, г для *синаптической дивергенции* или расходимости.

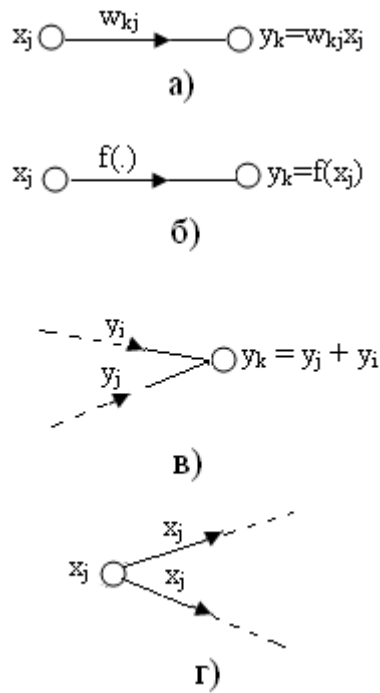


Рис. 7.7. Основные правила построения графов передачи сигналов

На рис. 7.8 показан пример графа передачи сигнала. Это модель нейрона, соответствующая блочной диаграмме, приведенной на рис. 7.3.

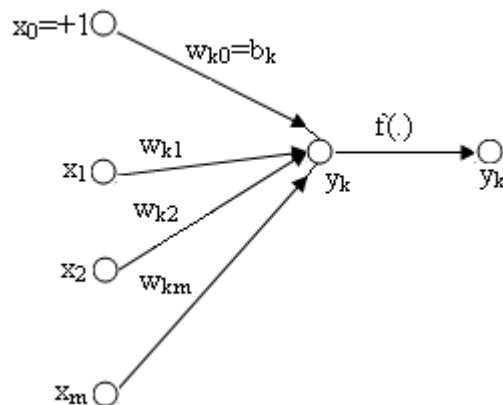


Рис. 7.8. Граф передачи сигнала одного нейрона

Принимая в качестве модели нейрона граф прохождения сигнала, показанный на рис. 7.8, можно сформулировать еще одно определение нейронной сети.

Определение

Нейронная сеть – это направленный граф, состоящий из узлов, соединенных синаптическими и активационными связями, который характеризуется следующими четырьмя свойствами.

1. Каждый нейрон представляется множеством линейных синаптических связей, внешним порогом и, возможно, нелинейной связью активации. Порог, представляемый входной синаптической связью, считается равным +1.

2. Синаптические связи нейрона используются для взвешивания соответствующих входных сигналов.

3. Взвешенная сумма входных сигналов определяет индуцированное локальное поле каждого конкретного нейрона.

4. Активационные связи модифицируют индуцированное локальное поле нейрона, создавая выходной сигнал.

Направленный граф, определенный указанным способом, является *полным*. Это значит, что он описывает не только прохождение сигнала между нейронами, но и передачу сигнала в самом нейроне. Если необходимо описать только прохождение сигнала между нейронами, то можно использовать сокращенную форму этого графа, опускающую детали передачи сигнала внутри нейрона. Такой направленный граф называется *частично полным*.

Определение

Частично полный граф характеризуется следующими свойствами.

1. Входные сигналы формируются узлами источника или входными элементами.

2. Каждый нейрон представляется одним узлом, который называется вычислительным.

3. Линии передачи сигнала, соединяющие узлы-источники и вычислительные узлы графа имеют веса. Они просто определяют направление прохождения сигнала на графе.

Частично полный направленный граф, определенный указанным выше способом, называется *архитектурным*. Он описывает структуру нейронной сети. Пример такого графа показан на рис. 7.9.

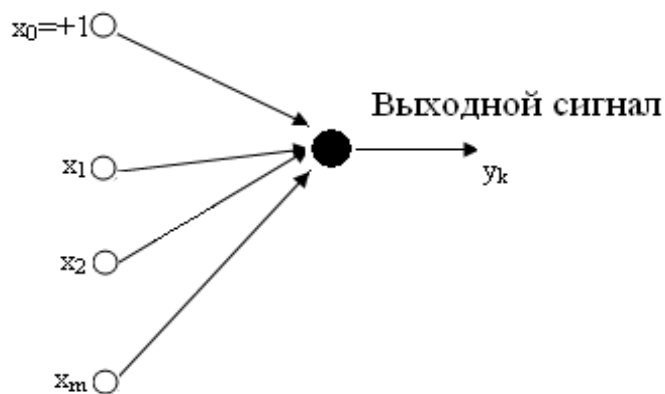


Рис. 7.9. Архитектурный граф нейрона

Обратите внимание, что на этом графе вычислительный узел обозначен заштрихованным кружком, а узлы источника – маленькими кружочками.

Подводя итог сказанному, можно выделить три графических представления нейронных сетей.

- Блочная диаграмма, описывающая функции нейронной сети.
- Граф прохождения сигнала, обеспечивающий полное описание передачи сигнала по нейронной сети.

- *Архитектурный* граф, описывающий структуру нейронной сети.

7.4. Архитектура нейронных сетей

Структура нейронных сетей тесно связана с используемыми алгоритмами обучения. Классификация алгоритмов обучения будет приведена дальше, здесь мы рассмотрим архитектуру (структуру) сетей.

По архитектуре связей нейронные сети можно сгруппировать в два класса: сети прямого распространения, в которых граф не имеет петель, и рекуррентные сети или сети с обратными связями (рис. 7.10).

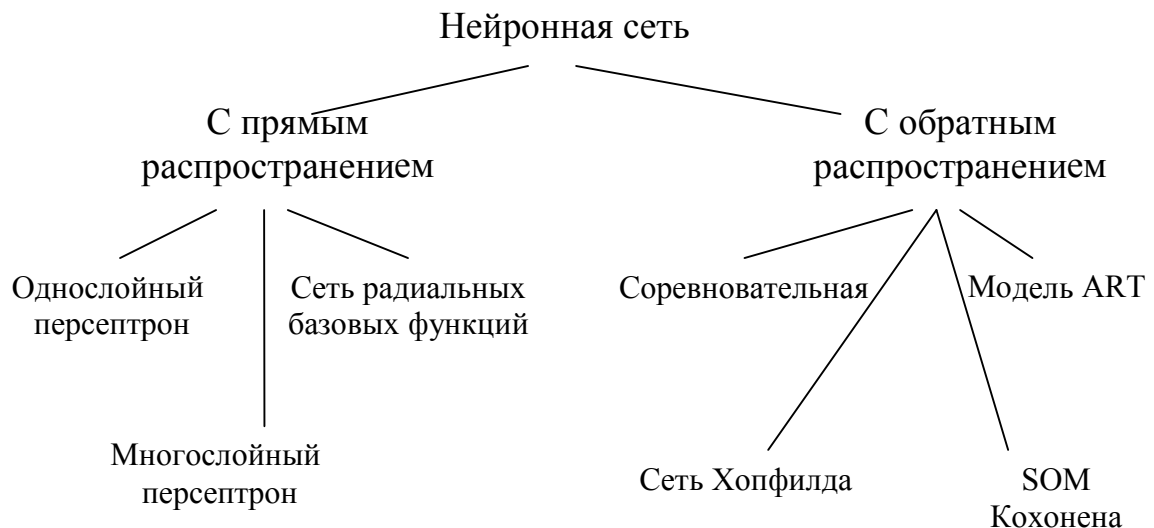


Рис. 7.10. Типы нейронных сетей

Среди сетей прямого распространения выделяют однослойные и многослойные сети.

В однослойных сетях *прямого распространения* имеется входной слой узлов источника, информация от которого передается на выходной слой нейронов, но не наоборот. Такая сеть называется сетью прямого распространения или ациклической сетью. На рис. 7.11 показана структура такой сети для случая трех узлов в каждом из слоев (входном и выходном).

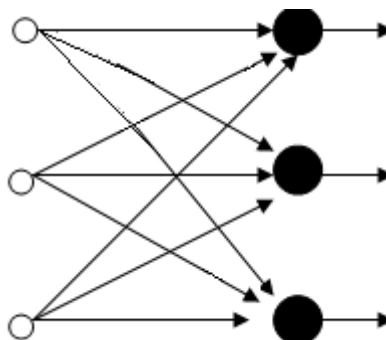


Рис. 7.11. Сеть прямого распространения с одним слоем нейронов

Такая нейронная сеть называется *однослойной*, при этом под единственным слоем подразумевается слой вычислительных элементов. При подсчете числа слоев мы не принимаем во внимание узлы источника, так как они не выполняют никаких вычислений.

Другой класс нейронных сетей прямого распространения характеризуется наличием одного или нескольких *скрытых слоев*, узлы которых называются скрытыми нейронами, или скрытыми элементами (рис. 7.12). Функция последних заключается в посредничестве между внешним входным сигналом и выходом нейронной сети. Добавляя один или несколько скрытых слоев, мы можем выделить статистики любого порядка.

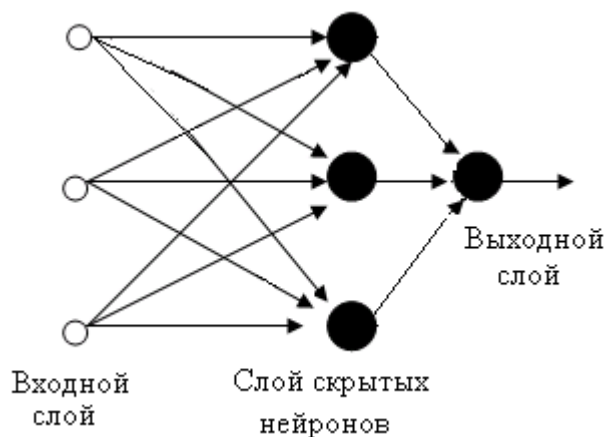


Рис. 7.12. Сеть прямого распространения с одним скрытым и одним входным слоем

Сеть, показанная на рис.7.12, является *полносвязной* в том смысле, что все узлы каждого конкретного слоя соединены со всеми узлами смежных слоев. Если некоторые синаптические связи отсутствуют, такая сеть называется *неполносвязной*.

Рекуррентная нейронная сеть отличается от сети прямого распространения наличием, по крайней мере, одной обратной связи. Например, рекуррентная сеть может состоять из единственного слоя нейронов, каждый из которых направляет свой входной сигнал на входы всех остальных нейронов слоя. Архитектура такой сети показана на рис. 7.13. Наличие обратной связи оказывает непосредственное влияние на способность сетей к обучению.

Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния. Рекуррентные сети являются динамическими, так как в силу обратных связей в них модифицируются входы нейронов, что приводит к изменению состояния сети.

Архитектура нейронных сетей задается матрицей весовых коэффициентов w_{ij} , характеризующих силу связей между элементами сети. Некоторые коэффициенты связей могут оставаться свободными и определяются при обучении сети.

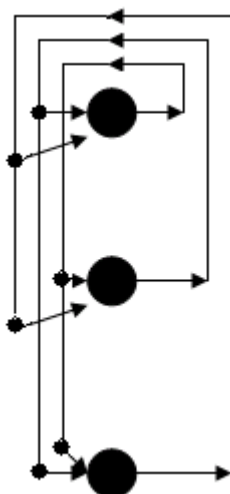


Рис. 7.13. Рекуррентная сеть без скрытых нейронов

7.5. Обучение нейронной сети

Одно из главных преимуществ нейронных сетей заключается в том, что они предполагают наличие правил, с помощью которых сеть может обучаться.

Процесс обучения рассматривается как настройка архитектуры сети и весов связей для эффективного выполнения задачи. Это свойство сетей делает их более привлекательными по сравнению с системами, которые следуют определенной системе правил, полученных от эксперта.

Определение

Обучение – это процесс, в котором свободные параметры нейронной сети настраиваются посредством моделирования среды, в которую сеть встроена. Тип обучения определяется способом подстройки этих параметров.

Для проведения процесса обучения необходимо иметь модель внешней среды, в которой функционирует сеть. Эта модель определяет *парадигму* обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети, т.е. какие *правила обучения* управляют процессом настройки.

Теория обучения рассматривает три фундаментальных свойства, связанных с обучением с помощью примеров: емкость, сложность образцов и вычислительная сложность. Под *емкостью* понимается, сколько образцов может запомнить сеть, и какие функции и границы принятия решений могут быть в ней сформированы. *Сложность* образцов определяет число обучающих примеров, необходимых для достижения способности сети к обобщению. *Вычислительная сложность* определяет сложность алгоритма, реализующего правило обучения сети.

Рассмотрим парадигмы обучения.

7.5.1. Парадигма обучения с учителем

Обучение с учителем можно рассматривать как наличие знаний об окружающей среде, представленных в виде пар *вход-выход*. При этом сама среда неизвестна обучаемой нейронной сети [13]. Теперь предположим, что учителю и

обучаемой среде подается обучающий вектор из окружающей среды. На основе встроенных знаний учитель может сформировать и передать обучаемой нейронной сети желаемый отклик, соответствующий данному входному вектору. Этот желаемый результат представляет собой оптимальные действия, которые должна выполнить нейронная сеть. Параметры сети корректируются с учетом обучающего вектора и сигнала ошибки. *Сигнал ошибки* – разность между желаемым сигналом и текущим откликом нейронной сети. Корректировка параметров выполняется пошагово с целью имитации нейронной сетью поведения учителя (рис. 7.14).

Описанная форма обучения с учителем является обучением на основе коррекции ошибок.



Рис. 7.14. Блочная диаграмма обучения с учителем

7.5.2. Парадигмы обучения без учителя

В Альтернативной парадигме обучения без учителя не существует маркированных примеров, по которым проводится обучение сети (рис. 7.15).



Рис.7.15. Блочная диаграмма обучения без учителя

Существует лишь независимая от задачи мера качества представления, которому должна научиться нейронная сеть, и свободные параметры сети оптимизируются по отношению к этой мере. После обучения сети на статистические закономерности входного сигнала она способна формировать внутреннее представление кодируемых признаков входных данных, и таким образом, автоматически создавать новые классы.

Для обучения без учителя можно использовать правило конкурентного обучения.

7.5.3. Алгоритмы обучения

В предыдущем разделе мы дали определения процессу обучения и рассмотрели парадигмы обучения нейронной сети. Напомним, что обучение – это процесс, в котором свободные параметры нейронной сети настраиваются посредством моделирования среды, в которую сеть встроена. Тип обучения определяется способом настройки этих параметров.

Это определение процесса обучения предполагает следующую последовательность событий.

1. В нейронную сеть поступают стимулы из внешней среды.
2. В результате этого изменяются внутренние свободные параметры нейронной сети.
3. После изменения внутренней структуры нейронная сеть отвечает на возбуждения уже иным образом.

Это список из трех правил решения проблемы обучения называется *алгоритмом обучения*. Не существует универсального алгоритма обучения, подходящего для любой архитектуры нейронных сетей. Существует лишь набор средств, представленный множеством алгоритмов обучения, каждый из которых имеет свои достоинства. Алгоритмы обучения отличаются друг от друга способом настройки синаптических весов нейронов. Еще одной отличительной характеристикой является способ связи обучаемой нейросети с внешним миром. Это определяет парадигму обучения, которая связана с моделью окружающей среды, в которой функционирует сеть.

Мы рассмотрим только идеи, заложенные в алгоритмах обучения. Все подробности оставляем на самостоятельное изучение.

Хотелось бы подчеркнуть, что алгоритм обучения на основе коррекции ошибки относится к алгоритмам обучения с учителем. Целью обучения является построение желаемого отображения входного сигнала в выходной, которое сеть должна аппроксимировать. Остальные рассматриваемые алгоритмы являются алгоритмами самоорганизующегося обучения или обучения без учителя. Целью алгоритмов самоорганизующегося обучения является выявление во множестве входных данных существенных образов или признаков.

Обучение, основанное на коррекции ошибок

Пусть имеем сеть прямого распространения с единственным нейроном в выходном слое k . Нейрон работает под управлением вектора входного сигнала $x(n)$, производимого одним или несколькими слоями скрытых нейронов. Эти слои получают сигнал от входного вектора возбуждения, передаваемого входному слою нейронной сети (рис. 7.16).

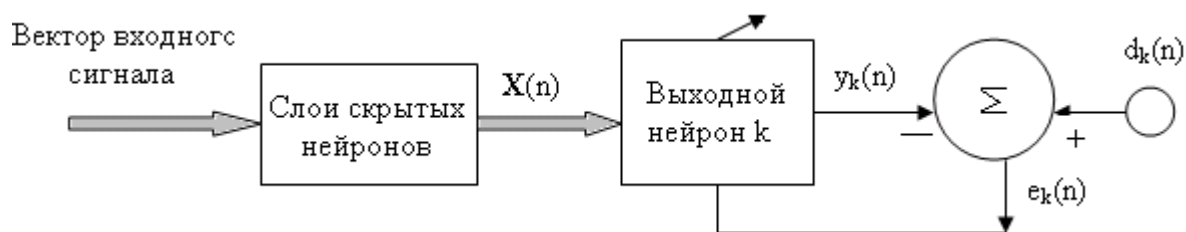


Рис. 7.16 Обучение, основанное на коррекции

Нейрон k работает под управлением вектора сигнала $x(n)$, который поступает из слоя скрытых нейронов. Эти слои получают информацию из входного вектора, передаваемого начальным узлом (входным слоем) нейронной сети. n – это дискретное время, или номер шага итеративного процесса настройки синаптических весов нейрона k . Выходной сигнал обозначен $y_k(n)$. Этот сигнал является единственным выходом нейронной сети. Он сравнивается с желаемым выходом, обозначенным $d_k(n)$. В результате получим сигнал ошибки $e_k(n)$. По определению

$$e_k(n) = d_k(n) - y_k(n).$$

Так как значение выходного сигнала нейрона сравнивается с желаемым, то используется парадигма обучения с учителем.

Сигнал ошибки поступает на механизм управления, который корректирует синаптические веса нейрона k так, чтобы приблизить выходной сигнал нейрона к входному. Эта цель достигается за счет минимизации *функции стоимости* или индекса производительности $E(n)$, которая определяется так:

$$E(n) = \frac{1}{2} e_k^2(n).$$

Пошаговая корректировка синаптических весов нейрона k продолжается до тех пор, пока система не достигнет устойчивого состояния (т.е. такого, при котором синаптические веса практически стабилизируются). В этой точке процесс обучения останавливается.

Описанный процесс называется *обучением, основанным на коррекции ошибок*. Минимизация функции стоимости $E(n)$ выполняется по дельта-правилу, или правилу Видроу-Хоффа. Введем обозначения: $w_{kj}(n)$ – текущее значение синаптического веса w_{kj} нейрона k , $x_j(n)$ – значение входного вектора $x(n)$ на шаге n . В соответствии с дельта-правилом изменение $\Delta w_{kj}(n)$, применяемое к синаптическому весу $w_{kj}(n)$ на шаге n , задается выражением:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n),$$

где η – положительная константа, определяющая скорость обучения. Константа используется при переходе от одного шага процесса обучения к другому. Она называется *параметром скорости обучения*. Таким образом, дельта-правило обучения можно сформулировать так:

Определение

Корректировка, применяемая к синаптическому весу нейрона, пропорциональна произведению сигнала ошибки на входной сигнал, его вызвавший.

Дельта-правило применимо к каждому нейрону и его применение не зависит от состояний других нейронов. Новое значение синаптического веса определяется по формуле:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n).$$

На рис. 7.16 видно, что обучение на основе коррекции ошибок – это пример замкнутой системы с обратной связью. Для сходимости итерационного процесса требуется подбор параметра скорости обучения η . Выбор этого параметра влияет также на точность и другие характеристики процесса обучения.

Обучение Хебба

Правило обучения использует постулат Хебба. Правило является самым старым и самым известным среди всех правил обучения. Из постулата следуют два правила:

1. Если два нейрона по обе стороны синапса (соединения) активизируются одновременно, то прочность связи возрастает.
2. Если два нейрона по обе стороны связи активизируются асинхронно, то прочность связи между ними ослабляется или вообще отмирает.

Функционирующий таким образом синапс называется синапсом Хебба. Для модификации такого синапса необходимо знать время возникновения предсинаптического (входного) и постсинаптического (выходного) сигналов.

Рассмотрим синаптический вес $w_{kj}(n)$ нейрона k с предсинаптическим и постсинаптическим сигналами x_j и y_k . Изменение синаптического веса $w_{kj}(n)$ в момент времени n можно выразить следующим соотношением:

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n)),$$

где $F(.)$ – некоторая функция, зависящая от предсинаптического и постсинаптического сигналов.

Простейшая форма обучения Хебба имеет следующий вид:

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n),$$

где η – положительная константа, определяющая скорость обучения. Графически эта формула проиллюстрирована на рис. 7.17.

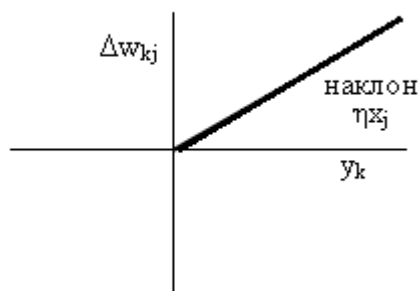


Рис.7.17. Иллюстрация гипотезы Хебба

На рис. 7.17 видно, что регулярное приложение входного сигнала $x_j(n)$ ведет к увеличению $y_k(n)$ и экспоненциальному росту активности, приводящему к насыщению синаптической связи.

Существует строгое физиологическое доказательство реализации постулата Хебба в области мозга, называемой гипоталамусом. Эта область играет важную роль в некоторых аспектах обучения и запоминания. Эти результаты подтверждают правильность постулата обучения Хебба.

Конкурентное обучение

Как следует из самого названия, в конкурентном обучении выходные нейроны нейронной сети конкурируют между собой за право быть активизированными. Если в нейронной сети, основанной на обучении Хебба, одновременно в возбужденном состоянии может находиться несколько нейронов, то в конкурентной сети в каждый момент времени может быть активным только один нейрон. Благодаря этому свойству конкурентное обучение очень удобно использовать для изучения статистических свойств, используемых в задачах классификации входных образов.

Правило конкурентного обучения основано на использовании трех основных элементов.

- Множество одинаковых нейронов со случайно распределенными синаптическими весами, приводящими к *различной* реакции нейронов на один и тот же входной сигнал.
- *Предельное значение* «силы» каждого нейрона.
- Механизм, позволяющий нейронам конкурировать за право отклика на данное подмножество входных сигналов и определяющий единственный активный выходной нейрон. Нейрон, победивший в этом соревновании, называют *нейроном-победителем*, а принцип конкурентного обучения формулируют в виде лозунга «победитель забирает все».

Простейшая нейронная сеть с конкурентным обучением содержит единственный слой входных нейронов, каждый из которых соединен с входными узлами. В такой сети могут существовать обратные связи между нейронами (рис. 7.18). В такой сети обратные связи нейрона стремятся «затормозить» связанные с ним другие нейроны.

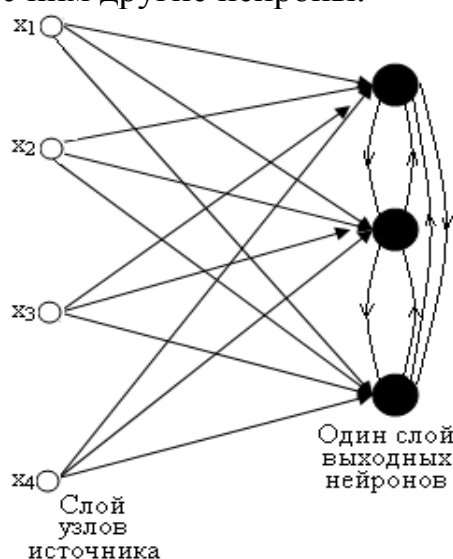


Рис. 7.18. Архитектурный граф простой сети конкурентного обучения

Для того, чтобы нейрон k победил в конкурентной борьбе, его индуцированное локальное поле v_k для заданного входного вектора x должно быть максимальным среди всех нейронов сети. Тогда выходной сигнал y_k нейрона-победителя k принимается равным единице. Выходные сигналы остальных нейронов при этом устанавливаются в значение нуль. Таким образом, можно записать:

$$y_k = \begin{cases} 1, & \text{если } v_k > v_j \text{ для всех } j, j \neq k, \\ 0 & \text{в остальных случаях,} \end{cases}$$

где индуцированное локальное поле v_k представляет собой возбуждение нейрона k , зависящее от всех входных сигналов и сигналов обратной связи.

Пусть w_{kj} – синаптический вес связи входного узла j с нейроном k . Предположим, что синаптические веса всех нейронов фиксированы (т.е. положительны), при этом $\sum_j w_{kj} = 1$, для всех k .

Тогда обучение этого нейрона состоит в смещении синаптических весов от неактивных к активным входным узлам. Если нейрон не формирует отклика на конкретный входной образ, то он и не обучается. Если некоторый нейрон выигрывает в конкурентной борьбе, то веса связей этого нейрона равномерно распределяются между его активными входными узлами, а связи с неактивными входными узлами ослабляются. Согласно *правилу конкурентного обучения* изменение Δw_{kj} синаптического веса w_{kj} определяется следующим выражением:

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}), & \text{если нейрон } k \text{ побеждает в соревновании,} \\ 0, & \text{если нейрон } k \text{ не побеждает в соревновании,} \end{cases}$$

где η – параметр скорости обучения. Это правило отражает смещение вектора синаптического веса w_k победившего нейрона k в сторону входного образа x .

Обучение Больцмана

Правило обучения Больцмана представляет собой стохастический алгоритм обучения, основанный на идеях, заложенных в термодинамике. Нейронная сеть, созданная на основе обучения Больцмана, получила название *машины Больцмана*.

Машина Больцмана использует скрытые и видимые нейроны, представляющие собой стохастические двоичные элементы. Нейроны находятся в одном из двух возможных вероятностных состояний. Эти двум состоянием формально можно присвоить значения $+1$ (соответствующее включенному состоянию) и -1 (соответствующее выключенному состоянию). Аналогично, можно принять значениями этих состояний $+1$ и 0 соответственно. Примем первое допущение. Еще одним отличительным свойством машины Больцмана является использование *симметричных синаптических связей* между нейронами. Использование этой формы синаптической связи обусловлено соглашениями статистической физики.

Стохастические нейроны машины Больцмана разбиваются на две функциональные группы: *видимые* и *скрытые* нейроны (рис. 7.19). Видимые нейроны предоставляют интерфейс между сетью и средой, в которой она работает. Во время обучения сети, видимые нейроны фиксируются в своих специфичных состояниях, определяемых средой. С другой стороны, скрытые нейроны всегда работают свободно – они используются для выражения ограничений, содержащихся во входных векторах.

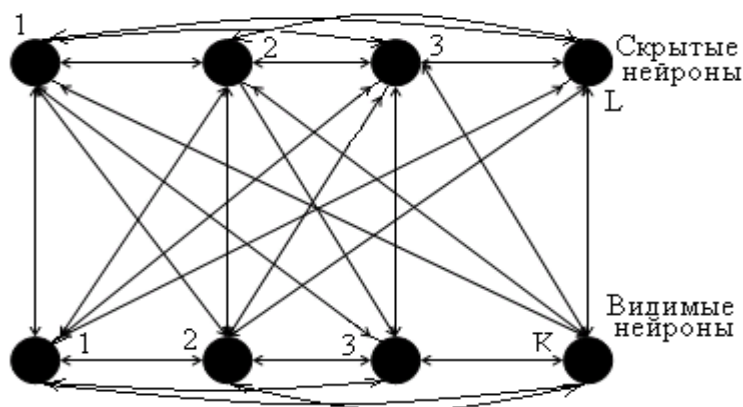


Рис. 7.19. Архитектурный граф машины Больцмана

Обозначим символом \mathbf{x} , состоящим из компонентов x_i , соответствующих состояниям нейронов i , вектор состояний машины Больцмана. Это состояние \mathbf{x} представляет собой реализацию случайного вектора \mathbf{X} . Синаптические связи между нейронами i и j обозначим символами w_{ji} . При этом

$$w_{ji} = w_{ij} \text{ для всех пар } (i, j) \quad 7.6$$

$$\text{и } w_{ii} = 0 \text{ для всех } i. \quad 7.7$$

Равенство (7.6) описывает свойство симметрии, а равенство (7.7) – отсутствие собственных обратных связей.

По аналогии с термодинамикой энергия машины Больцмана определяется следующим образом:

$$E(\mathbf{x}) = -\frac{1}{2} \sum_i \sum_{j, i \neq j} w_{ji} x_i x_j.$$

Работа такой машины заключается в случайном выборе некоторого нейрона (например, k -го) на определенном шаге процесса обучения и переводе этого нейрона из состояния x_k в состояние $-x_k$ при некоторой температуре T , с вероятностью

$$P(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(\Delta E_k / T)},$$

где ΔE_k – изменение энергии машины, вызванное в результате такой смены состояний. T – аналог температуры, используемой для управления уровнем шума, и, таким образом, степенью неопределенности переключения. При многократном применении этого правила машина достигает *термального равновесия*.

Однако, время, затраченное на достижение термального равновесия, может быть слишком большим. Для того, чтобы избежать этой проблемы, используют моделирование отжига для конечной последовательности температур $T_0, T_1, \dots, T_{\text{конечная}}$. Таким образом, начальная температура устанавливается в T_0 , обеспечивая быстрое достижение термального равновесия. После этого температура T медленно снижается до своего окончательного значения $T_{\text{конечная}}$ и в этой точке состояния нейронов достигнут (будем надеяться) своих желаемых граничных распределений.

Таким образом, посредством обучения распределение вероятности, представляемое нейронами, становится сходным с распределением вероятности среды.

7.6. Выводы

1. Нейронная сеть представляет собой совокупность простых обрабатывающих элементов, посылающих сигналы один другому по взвешенным сетям.

2. Типы связей, допустимых между элементами сети, зависят от конкретной модели сети.

3. Для каждого элемента сети имеется правило суммирования поступающих сигналов и правило вычисления выходного сигнала, посылаемого другим элементам сети. Правило вычисления выходного сигнала называется функцией активности.

4. Архитектуру нейронной сети можно представить в виде графа. По архитектуре связей существует три фундаментальных класса нейронных сетей: однослойные и многослойные сети прямого распространения и рекуррентные сети.

5. Для обучения нейронной сети необходимо иметь модель внешней среды, в которой функционирует сеть. Эта модель определяет парадигму обучения.

6. Существует две парадигмы обучения: с учителем и без учителя.

7. При обучении с учителем сеть настраивается так, чтобы минимизировать ошибку отклонения фактического отклика сети от желаемого.

8. Обучение без учителя с подкреплением целью обучения является минимизировать функции стоимости, определяемой как математическое ожидание кумулятивной стоимости действий, предпринятых в течение нескольких шагов обучения.

9. Обучение на основе самоорганизации позволяет использовать сеть для решения задач автоматической классификации.

10. Нейронные сети для решения конкретных примеров обучаются с использованием различных алгоритмов обучения. Не существует универсального алгоритма обучения, подходящего для любой архитектуры нейронных сетей. Выбор конкретного алгоритма обучения зависит от задач, решению которых следует обучить нейронную сеть.

11. Существует четыре класса алгоритмов обучения нейронной сети: обучение на основе коррекции ошибок, обучение с применением постулата Хебба, конкурентное обучение и стохастическое обучение с использованием машины Больцмана. Это общие названия целого семейства алгоритмов.

12. Алгоритмы на основе коррекции ошибки применяют для корректировки весовых коэффициентов дельта-правило Видроу-Хоффа.

13. Алгоритмы обучения Хебба, навеяны идеями нейробиологии. Они используют постулаты Хебба, согласно которым, если два нейрона по обе стороны синапса (соединения) активизируются одновременно, то прочность связи возрастает и если они активизируются асинхронно, то прочность связи между ними ослабляется.

14. В ходе конкурентного обучения нейроны конкурируют за право быть активизированными. Элемент-победитель, который является элементом, находящимся к входному сигналу ближе всех других, корректирует значения своих весовых коэффициентов так, чтобы стать к данному входному образцу еще ближе (т.е. стать еще более похожим на него).

15. Правило обучения Больцмана представляет собой стохастический алгоритм, основанный на идеях, заложенных в термодинамике. Нейронная сеть, созданная на основе обучения Больцмана, получила название *машины Больцмана*. Нейроны машины Больцмана находятся в одном из двух возможных вероятностных состояний. В результате обучения распределение вероятности, представляемое нейронами, становится сходным с распределением вероятности среды.

7.7. Вопросы для самоконтроля

1. Укажите классы задач, для решения которых применяются нейронные сети. Опишите их особенности.

2. Назовите основные компоненты нейронной сети.

3. Что определяет структура связей в нейронной сети?

4. Поясните правило комбинирования входных сигналов.

5. Что такое порог? Как он используется в искусственном нейроне?

6. Какие функции активации вы знаете. Укажите их отличительные характеристики.

7. Почему нужна нелинейная функция активации нейрона?

8. Назовите три способа изображения нейронных сетей. Поясните, когда они применяются.

9. Укажите отличия сетей прямого распространения от сетей с обратными связями.

10. Укажите особенности парадигмы обучения с учителем.

11. Охарактеризуйте парадигмы обучения без учителя.

14. Что понимается под емкостью и сложностью образцов в процессе обучения нейросети?

15. Как определяется вычислительная сложность алгоритма обучения?

16. Какие параметры нейросети модифицируются с помощью алгоритмов обучения?
17. Какова цель обучения алгоритма на основе коррекции ошибки?
18. Укажите цель обучения самоорганизующихся алгоритмов.
21. Поясните принципы, заложенные в обучение нейронной сети с помощью алгоритма коррекции по ошибке.
22. Поясните алгоритм обучения Хебба.
23. Какие принципы заложены в алгоритм конкурентного обучения?
24. Как обучается машина Больцмана?

ГЛОССАРИЙ

*Адаптация (лат.
приспособлять)*

В кибернетике процесс накопления и использования информации в системе, направленный на достижение определенного состояния или поведения системы при начальной неопределенности и изменяющихся внешних условиях.

Адаптивность

Способность интеллектуальной системы настраиваться в соответствии с данными. Синонимом этого понятия является способность к обучению.

*Аналитические
методы*

Методы, которые используют математические формулы для непосредственного получения оптимального решения или предсказания верного результата, в основном при решении структурированных задач.

*Антецедент (от лат.
предшествующий)*

Условное утверждение в левой части правила; в этой части определяются некоторые условия, которые должны выполняться в БД для того, чтобы были выполнены соответствующие действия.

*Архитектура (от
лат. зодчество,
искусство
проектировать)*

1) Организационная структура, в рамках которой происходит применение знаний и решение проблем; 2) принципы инженерии знаний, направляющие выбор подходящих структур для конкретных интеллектуальных систем.

*Атрибут (от лат.
наделенное,
присовокупленное)*

Существенное, необходимое свойство, признак объекта, предмета или явления.

*База данных (БД) (от
греч. основа чего
либо)*

1) Совокупность связанных файлов, таблиц, отношений и т.д., объединенных в систему хранения данных, обеспечивающую оперативный доступ к информации; 2) набор фактов, утверждений и заключений, используемых при сопоставлении с правилами в интеллектуальной системе.

<i>База знаний (БЗ)</i>	Основной компонент интеллектуальной системы, содержащий экспертные знания об определенной предметной области. Эти знания представляют собой собрание фактов, правил, эвристик и процедур, организованных различными схемами и моделями представления.
<i>Байесовское рассуждение</i>	Статистический подход к управлению неопределенностью в интеллектуальных и экспертных системах, основанных на байесовском правиле свидетельствования.
<i>Байесовское правило</i>	Статистический метод для модернизации вероятностей, связанных с определенными фактами в свете новых свидетельств.
<i>Вывод</i>	Процесс получения заключений из данных свидетельств. Достижение решения путем рассуждений.
<i>Вывод на предикатах</i>	В логической системе процесс получения знаний, выводимых из формул на основе аксиом с помощью правил вывода.
<i>Данные</i>	Элементарные описания предметов, событий, действий и транзакций, которые записаны, классифицированы и сохранены, но не организованы для передачи какого-либо смысла.
<i>Дедукция (от лат. выведение)</i>	Форма мышления, когда новая мысль выводится чисто логическим путем из предшествующих мыслей и аксиом.
<i>Декларативное представление знаний (от лат. заявление)</i>	Представление знаний, в котором описания выполняемых процедур не содержатся в явном виде. Декларативное представление это, как правило, множество фактов и утверждений, не зависящих от того, где они используются. Предметная область в такой форме представляется в виде синтаксического описания ее состояния.
<i>Демон (от гр. дух)</i>	Процедура, которая присоединена к слоту и активизируется при каждой попытке добавления или удаления данных из слота (по умолчанию). Демоны запускаются при обращении к соответствующему слоту.

	<p>Эта процедура выполняется каждый раз, когда атрибут изменяет свое значение. Демон обычно имеет структуру Если-Тогда. Существуют следующие типы процедур-демонов: ЕСЛИ-НЕОБХОДИМО ЕСЛИ-ИЗМЕНЕНО, ЕСЛИ-ДОБАВЛЕНО, ЕСЛИ-УДАЛЕНО.</p>
<i>Дерево вывода</i>	<p>Схематический просмотр процесса вывода, который показывает порядок, в котором были проверены правила.</p>
<i>Дерево поиска</i>	<p>Графическое представление, которое показывает задачу, ее альтернативные решения и процесс поиска наилучшего или приемлемого решения.</p>
<i>Дерево решений</i>	<p>Графическое представление последовательности взаимосвязанных решений, которое описывает данные древовидными структурами. Обычно используются в решении классификационных задач.</p>
<i>Динамическая система</i>	<p>Система, состояние которой изменяется с течением времени, т.е. изменение состояния является функцией времени или параметров системы.</p>
<i>Доска объявлений</i>	<p>1) Способ представления и управления знанием, основанный на использовании независимых групп правил, называемых источником знаний, которые общаются через общую область памяти, называемую доской объявлений; 2) глобально доступная база данных, используемая в некоторых интеллектуальных системах для записи промежуточных результатов решения задачи. В нее помещаются активные источники знаний, текущий план решения, промежуточные результаты и текущие данные для решения задачи.</p>
<i>Естественно-языковой интерфейс</i>	<p>Интерфейс пользователя, который использует естественный язык для взаимодействия.</p>
<i>Знания</i>	<p>Закономерности предметной области (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области.</p>
<i>Извлечение знаний</i>	<p>Выделение, формулирование, накопление, передача и</p>

	<p>преобразование знаний, полученных из различных источников. Извлечение знаний, т.е. процесс приобретения, изучения и организации знаний может использоваться в интеллектуальной системе.</p>
<i>Индукция (от лат. наведение)</i>	<p>В логике, форма мышления, посредством которой мысль наводится на какое-либо общее правило, общее положений, присущее всем единичным предметам какого-либо класса.</p>
<i>Инженер знаний</i>	<p>Участник создания экспертной системы, специалист, владеющий способами и приемами работы с экспертами методами выявления и представления знаний, знающий технологию создания экспертных систем.</p>
<i>Инженерия знаний</i>	<p>1) Теория, методология и технология интеллектуальных и экспертных систем, которые охватывают методы добычи, анализа и выражения в правилах знаний экспертов; 2) процесс построения интеллектуальной системы. Основные этапы этого процесса: оценка проблемы, приобретений данных и знаний, развитие прототипа системы, развитие завершенной системы, оценка и исправление системы, интеграция и сопровождение системы.</p>
<i>Инструментальные средства инженерии знаний</i>	<p>Системы программирования, которые упрощают работу по созданию и проектированию интеллектуальных систем, в особенности пакеты для часто встречающихся задач, и языки высокого уровня для эвристического программирования.</p>
<i>Интеллект (от лат. ум, рассудок)</i>	<p>Мыслительные способности человека к обучению, пониманию, рассуждению, умозаключениям, решению задач и принятию адекватных решений. Человек пытается повысить «интеллектуальные» возможности компьютера, передавая ему все более сложные функции по поиску и обработке информации. Машина мыслит интеллектуально, если она может достичь человеческого уровня выполнения некоторых когнитивных заданий.</p>
<i>Интеллектуальная информационная система</i>	<p>Компьютерная программа, состоящая из 5 основных компонентов: языковой подсистемы, информационной подсистемы, подсистемы знаний, подсистемы моделей и</p>

<i>Интенциональная семантическая сеть</i>	<p>подсистемы обработки и решения задач.</p> <p>Сеть, описывающая предметную область на обобщенном концептуальном уровне.</p>
<i>Информация (от лат. разъяснение, изложение, осведомленность)</i>	<p>Одно из наиболее общих понятий науки, обозначающее некоторые сведения, совокупность каких-либо данных, которые организованы в смысловые формы так, что они имеют значение и ценность для получателя.</p>
<i>Искусственная нейронная сеть (ИНС)</i>	<p>Математические модели, а также их программные и аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей. Она состоит из большого количества простых взаимодействующих процессоров, называемых нейронами, которые являются аналогами биологических нейронов мозга. Нейроны соединены связями с определенными весами, которые передают сигналы от одного нейрона к другому. ИНС обучают посредством повторяющихся регулировок весов. Эти веса хранят знания, необходимые для решения конкретной задачи.</p>
<i>Искусственный интеллект (ИИ)</i>	<p>Научное направление, в рамках которого ставятся и решаются задачи аппаратного или программного моделирования тех видов человеческой деятельности, которые традиционно считаются интеллектуальными (представление знаний, обучение, общение и т.п.).</p>
<i>Источники знаний</i>	<p>Совокупность знаний из предметной области, имеющая отношение к некоторой конкретной задаче. Понятие, находящее применение в интеллектуальной и экспертной системе. Источники знаний включают экспертов, учебники, справочники, мультимедийные документы, базы данных, специальные исследовательские отчеты и информацию, доступную через Интернет.</p>
<i>Консеквент (от лат. следствие)</i>	<p>Заключение или действие в правой части правила. Эта часть правила содержит описание действий, которые должны быть совершены над БД в случае выполнения соответствующих условий. В простейших продукционных системах они только определяют, какие элементы следует добавить (а иногда и удалить) в БД.</p>
<i>Конфликт (от лат. столкновение)</i>	<p>Состояние, при котором два или более продукционных правила подходят к данным в БД, но только одно</p>

	<p>правило в действительности может быть активизировано в данном цикле.</p>
<i>Коэффициент уверенности (от лат. содействующий)</i>	<p>Оценка в процентах, представляемая экспертной системой которая указывает вероятность того, что заключение, полученное системой, является правильным. Также, это степень уверенности эксперта в том, что заключение будет истинным если известная посылка истинна.</p>
<i>Лингвистическая переменная (от лат. язык)</i>	<p>Переменная, которая может иметь значения в виде элементов языка, таких как слова и фразы. В нечеткой логике термины лингвистическая переменная и нечеткая переменная являются синонимами.</p>
<i>Логика высказываний</i>	<p>Формальная логическая система рассуждений, в которой заключения выводятся из ряда утверждений в соответствии со строгим множеством правил.</p>
<i>Логика предикатов</i>	<p>Логическая система рассуждений, используемая в программах ИИ для обозначения отношений между группами данных. Логика предикатов является расширением логики высказываний, так как основным объектом здесь является высказывание (предикат), истинность или ложность которого зависит от значений его переменных.</p>
<i>Машинное обучение</i>	<p>Адаптивный механизм, который дает возможность компьютерам обучаться из опыта решения задач на примерах и по аналогиям. Способности к обучению постоянно улучшают характеристики интеллектуальной системы. Машинное обучение является основой адаптивных систем. Наиболее популярными подходами к машинному обучению являются искусственные нейронные сети и генетические алгоритмы.</p>
<i>Метазнание</i>	<p>1) Знание о знании; 2) знание об использовании и управлении предметными знаниями в экспертной системе.</p>
<i>Метаправило</i>	<p>Правило, которое представляет метазнание. Метаправило определяет стратегию использования предметно-ориентированных правил в интеллектуальной системе.</p>

Метод резолюций

Метод рассуждения и вывода в логических системах, используемых в системах искусственного интеллекта. Метод предложен Робинсоном.

Механизм вывода

Часть интеллектуальной системы, которая фактически выполняет функции рассуждения. Используя необходимые знания из БЗ и соответствующие данные, интеллектуальная система с использованием этого механизма может делать логический вывод, получая нужное знание, зачастую не хранящееся в явном виде в БЗ. В экспертных системах, основанных на правилах, его также называют управляющим механизмом или интерпретатором правил. Механизм вывода реализует общую встраиваемую схему поиска решения.

Многослойный персептрон

Наиболее общая топология ИНС, в которой персептроны связаны вместе для формирования слоев. Многослойный персептрон имеет входной слой, по крайней мере, один скрытый слой и один выходной слой. Наиболее популярным методом обучения является алгоритм обратного распространения ошибки.

«Мягкая» информация

Нечеткая, субъективная, неясная, подразумеваемая и неопределенная информация.

«Мягкие» вычисления

Когнитивная методология, объединяющая такие современные направления искусственного интеллекта, как искусственные нейронные сети и нейрокомпьютинг, нечеткую логику, индуктивную логику, эволюционные вычисления, генетическое программирование, интеллектуальный анализ данных.

Нейрон (от греч. нерв)

1) Клетка, которая способна обрабатывать информацию. Типичный нейрон имеет много входов и один выход. Человеческий мозг содержит приблизительно 10¹⁰–10¹² нейронов. 2) основной элемент искусственных нейронных сетей, который вычисляет взвешенную сумму входных сигналов и передает результат через передаточную функцию для генерации выхода.

Нейронная модель Маккалока и Питтса

Модель нейрона на основе бинарного порогового элемента. Этот математический нейрон вычисляет взвешенную сумму входных сигналов и формирует на выходе сигнал величины (+1), если эта сумма

	<p>превышает определенный порог, и (-1) – в противном случае.</p>
<i>Нейронная сеть</i>	<p>Система обрабатывающих элементов, называемых нейронами, связанных между собой для формирования сети. Искусственные нейронные сети обладают этой способностью обучаться на примерах посредством повторяющихся регулировок их весов.</p>
<i>Нейронная сеть прямого распространения</i>	<p>Топология ИНС, в которой нейроны одного слоя связаны с нейронами следующего слоя. Входные сигналы распространяются в прямом направлении, переходя от слоя к слою. Примером ИНС прямого распространения является многослойный персептрон.</p>
<i>Нейронная сеть с обратной связью (рекуррентная)</i>	<p>Топология ИНС, в которой нейроны имеют обратные петли от выходов к своим же входам. Примером сети с обратной связью является сеть Хопфилда.</p>
<i>Неопределенность</i>	<p>В контексте интеллектуальных информационных систем неопределенность относится к значению, которое не может быть определено в процессе консультаций. В основном определяют четыре источника неопределенных знаний: неизвестные данные, неточный язык, неявное смысловое содержание и трудности, связанные с сочетанием взглядов различных экспертов.</p>
<i>Нечеткий вывод</i>	<p>Процесс рассуждений, основанный на нечеткой логике.</p>
<i>Нечеткая логика</i>	<p>Способы рассуждения, которые могут справляться с неопределенной и неполной информацией. Эта логическая система разработана для представления условий, которые не могут быть просто описаны при помощи бинарных терминов «истина» и «ложь». Была предложена Л. Заде в 1965 г. В отличие от булевой логики, нечеткая логика является многозначной и использует понятие частичной истинности (значение истинности между «полностью истинно» и «полностью ложно»).</p>
<i>Нечеткая переменная</i>	<p>Величина, которая может принимать лингвистические значения.</p>

Нечеткая система

Множество нечетких правил, преобразующих входные данные в выходные.

*Нечеткое
множество*

Множество, элементы которого принадлежат ему в той или иной степени. Это множество имеет нечеткие границы. Для представления нечеткого множества на компьютере его выражают как функцию, а затем располагают элементы множества в соответствии с их функцией принадлежности.

Нечеткое правило

Условное высказывание вида «Если X есть A, Тогда Y есть B», где A и B – нечеткие множества. Правило образует связь между нечеткими множествами. Каждое правило определяет нечеткую область (декартово произведение A и B) в пространстве состояний системы. Чем обширнее множества A и B, тем обширнее и более неопределенная нечеткая область. Нечеткие правила нечетких систем являются блоками для построения базы знаний.

Обработка знаний

Обработка содержимого знаний правилами преобразования тех форм, которые описывают знания в компьютере. При обработке знаний наиболее фундаментальной и важной проблемой является, прежде всего, описание смыслового содержания задач широкого диапазона, а также наличие такой формы описания знаний, которая гарантирует, что их обработка формальными правилами преобразования будет осуществляться правильно. Эта проблема называется проблемой представления знаний. Обработка знаний является одной из областей обработки информации.

*Обратная цепочка
рассуждений*

Метод вывода, в котором система начинает с того, что хочет доказать некоторое утверждение и пытается установить факты, необходимые для доказательства этого утверждения; либо пытается установить условия, при которых возможна некоторая рассматриваемая ситуация.

Обучение

1) В общем смысле – это изменение правил или поведения с определенной целью, т.е. процесс улучшения характеристик работы путем изменения знаний или структуры управления; 2) процесс предъявления нейросетевой вычислительной системе

	<p>множества примеров-стимулов с целью достижения конкретной цели, определяемой пользователем. В простейшей форме обучение означает самонастройку ИНС на уровне процессорных элементов. Взвешенные связи между процессорными элементами или веса подстраиваются таким образом, чтобы получить конкретные результаты.</p>
<i>Обучающий алгоритм</i>	<p>Обучающая процедура, используемая в искусственных нейронных сетях.</p>
<i>Персептрон (от лат. восприятие, представление)</i>	<p>Модель процесса восприятия, осуществляемого при помощи нейронной сети. Это простейшая форма нейронной сети, предложенная Ф. Розенблаттом, операция персептрона основана на модели Маккаллока и Питтса. Он состоит из простого нейрона с регулируемыми синаптическими весами и пороговым элементом.</p>
<i>Планировщик</i>	<p>Компонент механизма вывода в продукционных интеллектуальных системах, который поддерживает управление выводом. Он оценивает результаты используемых правил вывода в свете их приоритетов или других критериев.</p>
<i>Поиск</i>	<p>Организованный процесс проверки пространства возможных решений, гарантирующий эффективное нахождение приемлемого решения.</p>
<i>Поиск решения методами перебора на дереве решений</i>	<p>К методам перебора при поиске решений относятся: поиск в глубину, поиск в ширину, поиск на основе стоимости дуг, поиск с возвратом (бэктрекинг).</p>
<i>Поиск решений методом редукций (от лат. возвращение)</i>	<p>При этом методе поиск необходимой совокупности данных для решения задачи сводится к решению более простых составляющих подзадач. Задачи могут описываться различными способами: списки, деревья, массивы.</p>
<i>Поиск решения эвристическими методами</i>	<p>Методы поиска, использующие эмпирические правила, которые позволяют сокращать объем просматриваемых вариантов решения.</p>
<i>Правило</i>	<p>Пара, состоящая из антецедентного условия и консеквентного предложения, которая может быть</p>

	использована в дедуктивном процессе типа прямой или обратной цепочки рассуждений и имеет форматы: предпосылка-заключение, условие-действие.
<i>Предикат (от лат. сказанное)</i>	Логическая функция; в логике предикатов функция любого числа аргументов, принимающая значение истинности 1 или 0.
<i>Представление знаний</i>	Структурирование предметных знаний в интеллектуальной системе с целью облегчения поиска решения задачи.
<i>Преобразующая (передаточная) функция</i>	Отношение между внутренним уровнем активации и выходом нейрона; вычисляет внутреннее возбуждение или активацию уровня ИНС. Основанный на этом уровне нейрон может (или не может) производить выход. Отношение между уровнем внутренней активации и выходом может быть линейным или нелинейным. Такое отношение выражается преобразующей (передаточной) функцией.
<i>Приближенное рассуждение</i>	Используется, когда интеллектуальная система должна принимать решения, основанные на частичной или неполной информации.
<i>Принятие решений</i>	Действия по выбору среди альтернатив.
<i>Приобретение знаний</i>	Извлечение и формулирование знаний, полученных из внешних источников, в особенности от экспертов.
<i>Продукционная модель</i>	Набор правил вида «условие-действие», где условиями являются утверждения о содержимом некой базы данных, а действия представляют собой процедуры, которые могут изменять содержимое БД.
<i>Продукция</i>	Термин, используемый для описания правила.
<i>Пространство поиска</i>	Множество всех возможных решений задачи.
<i>Процедурное представление знаний</i>	Представление знаний, в котором семантика непосредственно заложена в описание элементов базы знаний. Процедурные знания содержат в явном виде описание некоторых процедур, которые обрабатывают определенный участок базы знаний, за счет чего

	повышается эффективность поиска решений. При этом текущее состояние представляется в виде набора специализированных процедур.
<i>Процедуры (от лат. продвигаться)</i>	Набор инструкций о том, как комбинировать ранее описанные компоненты для того, чтобы обрабатывать информацию и генерировать требуемые выходы; 2) Отдельная произвольная часть компьютерного кода.
<i>Прямая цепочка рассуждений</i>	Метод вывода в продукционных системах, который начинается с известных данных и работает в прямом направлении, связывая факты из базы данных в продукционных правилах из базы правил, устанавливая новые факты до тех пор, пока не останется правил, которые могут быть активизированы. Это поиск, управляемый данными.
<i>Разрешение конфликта</i>	Метод выбора правила или процедуры из конфликтующего множества применимых конкурирующих правил или процедур для активизации, когда более чем одно правило подходит для активизации в данном цикле.
<i>Репозиторий</i>	База данных, где хранятся метаданные.
<i>Самоорганизующаяся карта Кохонена</i>	Специальный класс ИНС с соревновательным методом обучения, предложенная Т. Кохоненом в начале 1980-х гг. Карта Кохонена состоит из простого слоя вычислительных нейронов с двумя типами связей: прямые связи (от нейронов во входном слое к нейронам в выходном слое), и горизонтальные связи между нейронами в выходном слое. Горизонтальные связи используются для создания соревнования между нейронами.
<i>Семантическая сеть</i>	Метод представления знаний посредством сети узлов, соответствующих понятиям или объектам, связанных дугами, описывающими отношения между дугами.
<i>Сеть Хопфилда</i>	Нейронная сеть с обратной связью с простым слоем. В сети Хопфилда выход каждого нейрона связан с входами всех других нейронов (но не со своим входом).
<i>Слепой поиск</i>	Подход к поиску решений, при котором не

<i>(методы полного перебора)</i>	используются знаний или эвристики для ускорения процесса поиска решения. Реализация этого метода произвольного поиска требует значительных временных затрат в попытках исчерпать все возможности.
<i>Слой</i>	Основная архитектурная компонента ИНС, состоящая из множества процессорных элементов с одинаковыми функциональными возможностями и занимающая в сети положение, соответствующее определению стадии обработки.
<i>Слот (от англ. отверстие, щель)</i>	Признак или описание компоненты некоторого объекта во фрейме. Другими словами – слот это атрибут, связанный с узлом в системе, основанной на фреймах. Заполнение слота приводит к тому, что данный фрейм ставится в соответствие некоторой ситуации, явлению или объекту.
<i>Степень принадлежности</i>	Числовое значение от 0 до 1, которое представляет степень, с которой элемент принадлежит к какому-либо множеству.
<i>Субъективная вероятность</i>	Вероятность, которая оценивается лицом, принимающим решение без использования формальной модели.
<i>Сценарий</i>	Состояние предположений и возможных конфигураций относительно окружающей среды, в которой действует конкретная система, на отдельном определенном промежутке времени.
<i>Теория уверенности</i>	Теория для управления неопределенностью в интеллектуальных системах, основанная на неточном рассуждении. Она использует коэффициенты уверенности для представления уровня доверия к гипотезе, что данное событие наблюдалось. Основные принципы этой теории были предложены Бухананом и Шортлиффом в 1975 г.
<i>Теория нечетких множеств</i>	Теория, разработанная Л. Заде в 1965 г. Аппарат теории нечетких множеств предназначен для решения широкого круга проблем, в которых важное место занимают субъективные, трудно формализуемые знания человека. Применяется при разработке человеко-ориентированных социальных и управленческих

	систем, в частности, экспертных систем.
<i>Уверенность</i>	Условие, при котором предполагается, что будущие значения известны с определенной уверенностью и только один результат ассоциируется с данным действием.
<i>Узел</i>	Точка решения в дереве решений.
<i>Фреймы (от англ. рама, остов)</i>	Метод представления знаний, основанный на минимальных структурах информации, необходимых для представления класса объектов, явлений или процессов. Фреймы связывают свойства с узлами, представляющими понятия и объекты. Свойства описываются атрибутами, которые называются слотами и их значениями. Фреймы используются для представления знаний в интеллектуальных системах.
<i>Функция принадлежности</i>	Математическая функция, которая определяет нечеткое множество на пространстве рассуждений. Типичные формы функций принадлежности, используемых в нечетких интеллектуальных системах, это треугольники и трапеции.
<i>Эвристика</i>	Эмпирическое правило или стратегия, которая может использоваться при решении сложных задач, упрощая или ограничивая поиск решения в предметной области, которая является сложной или недоступной нашему пониманию.
<i>Эвристический поиск</i>	Метод поиска, который использует эвристики для направления процесса рассуждения, уменьшая пространство поиска.
<i>Эксперт (от лат. опытный)</i>	Человек, который имеет глубокие знания в конкретной предметной области и за годы обучения и практики научился эффективно решать задачи, относящиеся к этой предметной области.
<i>Экспертная система</i>	Система, которая использует человеческие знания и логику эксперта, встраиваемые в компьютер, для обеспечения высокоэффективного решения задач в некоторой узкой предметной области. Такие системы, как правило, представляют знания символически,

*Экстенциональная
семантическая сеть*

исследуют и объясняют сами процессы рассуждения и предназначены для предметных областей, в которых людям для достижения мастерства необходимы годы специального обучения и практики.

*Экстенциональные
знания*

Сеть, в которой производится конкретизация и наполнения ее фактическими знаниями.

Представляют собой данные, характеризующие конкретные объекты, их состояния, значения параметров в определенные моменты времени.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

- 1 Абдикеев, Н.М. Проектирование интеллектуальных информационных систем / Н.М. Абдикеев. – М.: Издательство «Экзамен», 2004. – 258 с.
2. Боженюк, А.В. Интеллектуальные интернет-технологии: учебник / А.В. Боженюк, Э.М. Котов, А.А. Целых. – Ростов н/Д: Феникс, 2009. – 384 с.
3. Гаврилова Т.А. Интеллектуальные технологии в менеджменте: инструменты и системы: учебн. пособие. 2-е изд. / Т.А. Гаврилова, Д.И. Муровцев. – СПб.: Изд-во «Высшая школа менеджмента»; изд. дом С.-Петерб. гос. ун-та, 2008. – 488 с.
4. Гаврилова, Т.А. Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2000. – 384 с.
5. Глухих, И.Н. Интеллектуальные информационные системы: учебн. пособие для высш. проф. образования / И.Н.Глухих. – М.: Изд. центр «Академия», 2010. – 112 с.
6. Джарратано, Дж., Райли Г. Экспертные системы: принципы разработки и программирование, 4-е изд. / Дж.Джарратано, Г.Райли Г. – М.: изд. дом «Вильямс», 2007; 1152 с.
- 7 Джексон, П. Введение в экспертные системы: пер. с англ. / П. Джексон. – М.: изд. дом «Вильямс», 2001. – 624 с.
8. Каллан, Р. Основные концепции нейронных сетей: пер. с англ. / Р. Калан. – М.: изд. дом «Вильямс», 2003. – 288с.
9. Кричевский, М.Л. Интеллектуальные методы в менеджменте / М.Л. Кричевский, СПб.: Питер, 2005. – 304 с.
10. Люггер, Д.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем: пер. с англ. / Д.Ф. Люггер. – М.: изд. дом «Вильямс», 2003; 863 с.

11. Попов, Э.В. Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ / Э.В. Попов. – М.: Наука, гл. ред. физ.-мат. лит., 1987. – 288 с.
12. Поспелов, Д.А. Ситуационное управление. Теория и практика / Д.А. Поспелов. – М.: Наука, 1987.
13. Рассел, С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг, 2-е изд., Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1408 с.
14. Романов, В.П. Интеллектуальные информационные системы в экономике: учебн. пособие / В.П. Романов. – М.: Издательство «Экзамен», 2003. – 496.
15. Тельнов, Ю.Ф. Интеллектуальные информационные системы в экономике: учебн. пособие / Ю.Ф. Тельнов. – М.: СИНТЕГ, 1998, 216 с.
16. Уткин В.Б. Информационные системы в экономике: Учебник для студ. высш. учебн. заведений / В.Б. Уткин, К.В. Балдин – М.: изд. центр «Академия», 2004. – 288 с.
17. Хайкин, С. Нейронные сети: полный курс, 2-е изд.: пер. с англ. / С. Хайкин. – М.: изд. дом «Вильямс», 2006. – 1104 с.

Электронное издание

Галина Андреевна Поллак

ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ
СИСТЕМЫ

Учебное пособие

Издательский центр Южно-Уральского государственного
университета

Подписано в печать 27.12.2011. Формат 60×84 1/16.
Усл. печ. л. 8,37. Заказ 492.
