2. МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ.

Системы ИИ обрабатывают знания. И что бы их можно было обрабатывать программно, необходимо знаниям предварительно придать подходящую для этих целей форму. Как раз такими формами и являются модели представления знаний.

Модель представления знаний - это формализм, предназначенный для отображения объектов и отношений проблемной области, иерархии понятий и изменений отношений между объектами, т.е. статики и динамики проблемной области.

Если в процессе создания и изучения моделей представления знаний поменять причину со следствием, то можно получить простой критерий, по которому легко различать данные и знания, а также ответить на вопрос о том, является ли система интеллектуальной. Если программная система обрабатывает информацию, используя модели представления знаний, то обрабатываемая информация является знаниями, а программная система - интеллектуальной.

Рассмотрим основные модели, используемые для представления знаний, в порядке их популярности.

2.1. Правила продукций

Правила продукций являются самой распространенной моделью в системах ИИ. Они обеспечивают формальный способ представления рекомендаций, указаний или стратегий; они часто подходят в тех случаях, когда предметные знания возникают из эмпирических ассоциаций, накопленных за годы по решению задач в данной области.

Термин "продукция" принадлежит американскому логику Э.Посту (Post E.,1943г.). Запись "ЕСЛИ A, ТО B" трактуется как оператор замены цепочки A цепочкой B в некотором входном слове.

Правила продукций имеют следующую форму:

***ЕСЛИ** условие (или предпосылка), **ТО** следствие (или заключение)*

и является импликацией (логическое "если...то"). **Импликацией** называется логический союз, объединяющий два исходных суждения таким образом, что истинность первого исключает ложность второго, в логике обозначается:

p → q (читается: Если p, то q). Первую часть импликации, соответствующую переменной p, называют антецедентом, вторую - консеквентом. Напомним, что **суждение** есть выраженная предложением или группой предложений единица мышления, содержащая некоторое утверждение или отрицание и обладающая вследствие этого определенным значением истинности. Приведем несколько примеров правил продукции**:**

Правило (26)
ЕСЛИ минимизируемая функция является нелинейной и
задача содержит систему ограничений,
ТО необходимо применять методы условной оптимизации.
Правило (72)
ЕСЛИ разрабатывается демонстрационный прототип и
есть подходящая "оболочка",
ТО использовать "оболочку" в качестве программного средства.

Правила устанавливают взаимосвязь между фактами. Под **фактом** понимаются предложение или данные, которые считаются достоверными, т.е. то, что известно о предметной области в данный момент. В выше приведенных примерах фактами являются утверждения**:**

"минимизируемая функция является нелинейной"
"задача содержит систему ограничений"
"есть подходящая "оболочка"" и т.п.

Системы ИИ, построенные на базе правил, называются продукционными. Продукционная система состоит из трех компонентов**:**

- базы знаний, содержащей правила продукций;
- рабочей области, которая отображает текущее состояние некоторой (решаемой) задачи и содержит факты;
- управляющей структуры (машина вывода; интерпретатор в терминологии программирования), решающей, какое из правил продукции надлежит применить следующим.

Когда часть правила ЕСЛИ удовлетворяет фактам рабочей области, то действие, указанное в части ТО, выполняется. Когда это происходит, то говорят, что правило выполнено или сработало. Интерпретатор правил сопоставляет части правил ЕСЛИ с фактами рабочей области и выполняет то правило, часть ЕСЛИ которого согласуется с фактами.

Действия правил могут состоять**:**

- в модификации набора фактов в рабочей области, например, в добавлении нового факта. Новые факты,

добавленные в рабочую область, сами могут быть использованы для сопоставления с частями правил ЕСЛИ;

- во взаимодействии с внешней средой (во влиянии на реальный мир): выдать рекомендацию, задать вопрос для ввода данных, выдать управляющее воздействие на оборудование и т.п.;
- вызвать на выполнение процедуру (программу).

Существует два способа (две основные стратегии) использования правил в продукционной системе (направления вывода)**:** прямая цепочка рассуждений и обратная цепочка рассуждений.

При использовании прямой цепочки вывод осуществляется от данных к цели, т.е. первоначально в рабочую область вводится один или несколько фактов, которые инициируют логический вывод. Затем определяется правило, которое содержит в части "ЕСЛИ" факты, находящиеся в рабочей области. Такое правило срабатывает, т.е. выводится заключение, в результате чего в рабочую область добавляются новые факты, а правило помечается как сработавшее. Далее определяется следующее подходящее правило и т.д. до тех пор, пока не будет исчерпан список всех подходящих правил. Все полученные факты содержатся в рабочей области и пользователь, делая необходимые запросы к рабочей области, получает ответы о наличии или отсутствии в ней соответствующих фактов. Например, пусть база знаний содержит следующие три правила:

П1.ЕСЛИ компьютер стоит дешево,
ТО скорее всего он имеет маломощный процессор и
жесткий диск малого объема и/или
низкое качество изготовления.
П2.ЕСЛИ компьютер имеет маломощный процессор,
ТО возможно решение, в основном, простых задач.
П3.ЕСЛИ компьютер имеет жесткий диск малого объема,
ТО возможна обработка малых объемов данных и
увеличение объемов данных приводит к усложнению процедуры решения.

Если ввести в рабочую область факт "компьютер стоит дешево", то сработает правило П1 и в рабочую область добавятся факты, составляющие заключение этого правила. После чего могут сработать правила П2 и П3. Таким образом, начальное условие: "компьютер стоит дешево" через правила П1, П2, П3 порождает ряд следствий, сформулированных в частях "ТО" правил.

Прямая цепочка рассуждений позволяет определить весь спектр последствий, вызываемых одним или несколькими фактами. Она используется, когда количество исходных фактов не велико и необходимо сделать прогноз последствий тех или иных действий. Например, получить ответ на вопрос: "Изменятся ли каким-либо образом характеристики блока А, если в схему внести такие-то изменения?" или "Каким образом изменятся функциональные характеристики устройства В, если оно будет эксплуатироваться в условиях повышенной влажности и при изготовлении материал С будет заменен на материал Д?".

Обратная цепочка рассуждений осуществляет вывод от цели к данным, т.е. когда необходимо определить причины имевшего место конкретного факта или установить возможность существования конкретного факта в конкретных условиях или когда необходимо получить определенный результат (например, составить конфигурацию вычислительной системы, т.е. конфигураций может быть очень много и они зависят от требований пользователя, а необходимо выбрать одну). Обратная цепочка рассуждений инициируется заданием цели, т.е. вопроса, на который система должна подыскать ответ. Например, для фрагмента базы знаний:

П1.ЕСЛИ процессор выбран и
жесткий диск выбран,
ТО конфигурация выбрана.
П2.ЕСЛИ компьютер будет использоваться для подготовки данных,
ТО объем данных небольшой и
программы обработки простые.
П3.ЕСЛИ объем данных небольшой,
ТО достаточно жесткого диска объемом 10 Мб.
П4.ЕСЛИ программы обработки простые,
ТО достаточно процессора типа XT.
П5.ЕСЛИ тип процессора определен,
ТО процессор выбран.
П6.ЕСЛИ объем жесткого диска известен,
ТО жесткий диск выбран.

В качестве цели может быть задан вопрос: "Какова конфигурация ЭВМ?". Затем в базе данных осуществляется поиск правила, которое в части "ТО" содержит ответ на этот вопрос и исследуется часть "ЕСЛИ" этого правила. Таким правилом является П1. Условия правила П1 также не определены, поэтому необходим поиск правил, устанавливающих эти условия. Такими правилами являются П3 и П4, которые, в свою очередь, зависят от правила П2. Правила, которое определяло бы условие правила П2, в базе знаний

нет, поэтому в таких случаях система делает запрос пользователю на ввод факта. Если, действительно, цель использования компьютера - это подготовка данных, то правило П2 срабатывает, затем срабатывают правила П3 и П4, далее П5 и П6 и, наконец, П1, которое завершает процесс логического вывода. Результатом будет ответ: "Конфигурация ЭВМ: процессор XT и жесткий диск, объем 10 Мб".

В ЭВМ правила, обычно, представляются списками фактов: один список для части "ЕСЛИ", второй - для части "ТО". Факт, часто, представляется тройкой: <объект, атрибут, значение>. Например, <Иванов, вес, 80_кг>, <Петров, стаж, 10_лет>.

В основе правил продукций лежит простой принцип: они определяют набор разрешенных преобразований, с помощью которых происходит продвижение от начального состояния до окончательного решения поставленной задачи. Текущее (промежуточное) состояние отражается с помощью множества фактов, фиксируемых в рабочей области (базе данных). В ходе решения задачи происходит сопоставление одной из частей правила с содержимым рабочей области. В реальной системе возможна ситуация, когда могут быть применены несколько правил. Поэтому возникает необходимость в управляющей структуре, которая должна решать, какие из подходящих правил применить.

Стратегии управления (выбором) правил.

1. *Последовательный перебор правил.*

2. *Принцип "стопки книг".* Основан на идее, что наиболее часто используемое правило является наиболее полезным. Подходящие для текущего состояния рабочей области правила как бы образуют "стопку", в которой порядок определяется накопленной частотой использования продукций в прошлом. На самом верху "стопки" находится правило, которое используется чаще всех. При актуализации некоторого фронта готовых правил для их применения выбирается то правило (или те правила при наличии параллельных технических устройств), у которых частота использования максимальна.

Подобный принцип управления особенно хорош, когда частота использования подсчитывается с учетом некоторой ситуации, в которой ранее исполнялось правило, и его применение было удачным. При такой обратной связи метод "стопки книг" может превратиться в обучающуюся процедуру, адаптирующуюся к тем задачам, которые возникают во внешней среде. Управление по принципу "стопки книг" целесообразно применять, если правила относительно независимы друг от друга, например, когда каждое из них есть имеет вид "ситуация (А) => действие (В)". Именно такой случай имеет место в планирующих системах для роботов.

3. *Принцип метапродукций*. Он основан на идее ввода в систему продукций специальных метапродукций (метаправил), задачей которых является организация управления в системе продукций.

Достоинства продукционных систем:

1) подавляющая часть человеческих знаний может быть записана в виде продукций;

2) системы продукций являются модульными. За небольшим исключением удаление или добавление продукций не приводят к изменениям в остальных продукциях. Правила являются, в основном, независимыми и выражают самостоятельные "куски" знаний, поэтому легко осуществляется модификация знаний;

3) при необходимости системы продукций могут реализовать любые алгоритмы и, следовательно, способны отражать любое процедурное знание, доступное ЭВМ;

4) естественный параллелизм в системе продукций, асинхронность их реализации делают продукционные системы удобной моделью вычислений для ЭВМ новой архитектуры, в которой идея параллельности и асинхронности является центральной.

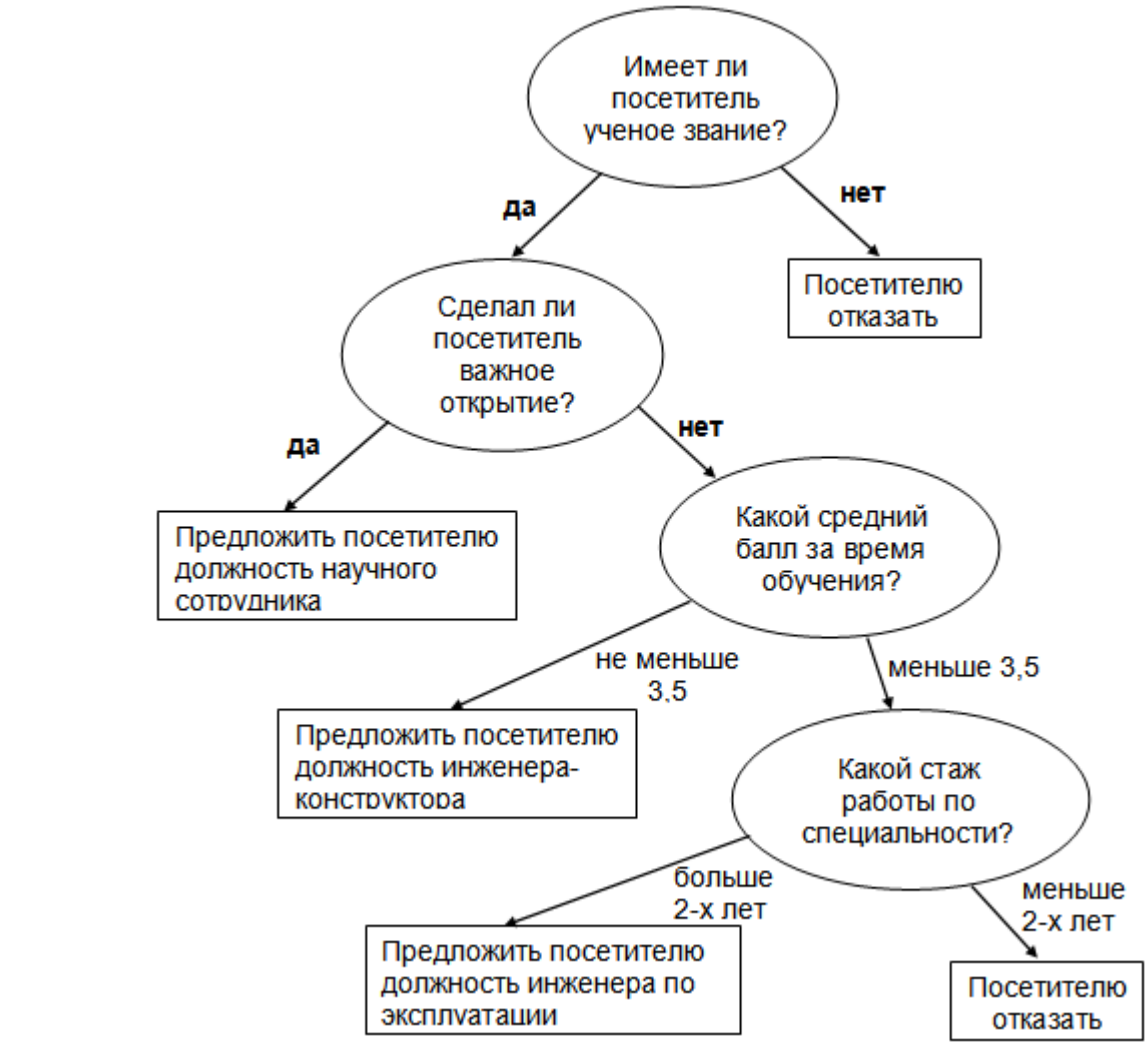
Основной недостаток продукционных систем состоит в том, что при большом количестве правил становится сложной проверка на непротиворечивость системы продукций. Это заставляет при добавлении новых правил тратить много времени на проверку непротиворечивости новой системы.

2.2. Разработка базы знаний: дерево решений.

Рассмотрим задачу, вытекающую из следующих ситуаций**:** к директору крупной технической фирмы пришел человек, желающий устроиться на работу. Директор располагает сведениями о его квалификации, о потребности фирмы в специалистах и общем положении дел в фирме. Ему нужно решить, какую должность в фирме может занять посетитель.

На первый взгляд задача не очень сложная, но на решение директора влияет много факторов. Допустим, претендент работает в данной области недавно, но уже сделал важное открытие, или он закончил учебное заведение с посредственными оценками, но несколько лет работал по специальности. Если директора этот человек устраивает, то для него нужно подобрать подходящую должность. Директору необходимо задать такие вопросы, ответы на которые дадут возможность сделать правильный выбор.

Итак, задача поставлена. Теперь нужно наглядно ее представить. Для описания подобных задач обычно используются диаграммы, которые называются деревьями решений. Деревья решений дают необходимую наглядность и позволяют проследить ход рассуждений. Ветви деревьев решений заканчиваются логическими выводами (результатами). Для рассматриваемого примера вывод заключается в том, что предложит ли директор должность поступающему на работу, и если да, то какую.



П(1) ЕСЛИ звание = нет,
ТО должность = нет.

П(2) ЕСЛИ звание = да и
открытие = да,
ТО должность = научн сотр

П(3) ЕСЛИ звание = да и
открытие = нет и
ср балл >= 3,5,
ТО должность = инж-констр

П(4) ЕСЛИ звание = да и
открытие = нет и
ср балл < 3,5 и
стаж >= 2_лет,
ТО должность = инж экспл

Кружки, содержащие вопросы, называются вершинами решений. Прямоугольники содержат цели диаграммы и означают логические выводы.

Процесс формирования правил состоит из следующих шагов:

Шаг 1. Выбрать из дерева решений вершину вывода (прямоугольник) и зафиксировать ее.

Шаг 2. Найти вершину, расположенную перед зафиксированной вершиной, т.е. связанную с ней ветвью, и зафиксировать ее (включить в путь и перейти к ней).

Шаг 3. Если текущая вершина не имеет предшествующих или текущая вершина является вершиной вывода, то перейти к шагу 4, иначе - к шагу 2.

Шаг 4. Каждая вершина, входящая в путь, - это одна из составляющих предпосылки правила. Эти составляющие объединяются логическим оператором "И".

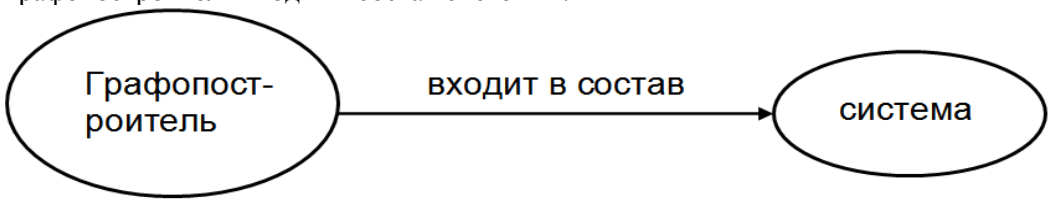
Шаг 5. Выбранный логический вывод является заключением правила.

2.3 Семантические сети

Понятие семантической сети основано на древней и очень простой идее о том, что "память" формируется через ассоциации между понятиями. Понятие "ассоциативная память" появилось еще во времена Аристотеля и вошло в информатику в связи с работами по использованию простых ассоциаций для представления значения слов в базе данных. С тех пор этот формализм был всесторонне развит для представления многих классов данных, используемых в различных предметных областях. К таким

областям относятся пространственные связи в простых физических системах, операции по управлению механизмами, причинные и функциональные связи в приборах и взаимосвязи между симптомами в медицине. Изначально семантическая сеть была задумана как модель представления структуры долговременной памяти в психологии, но в последствии стала одним из основных способов представления знаний в инженерии знаний.

Базовым функциональным элементом семантической сети служит структура из двух компонентов - "узлов" и связывающих их "дуг". Каждый узел представляет некоторое понятие, а дуга – отношение между парами понятий. Можно считать, что каждая из таких пар отношений представляет простой факт. Например: "Графопостроитель входит в состав системы".



Дуга имеет направленность, благодаря чему между понятиями в рамках определенного факта выражается отношение "субъект/объект". Любой из узлов может быть соединен с любым числом других узлов; в результате этого обеспечивается формирование сети фактов.

С позиций логики базовую структуру семантической сети можно рассматривать в качестве эквивалента предиката с двумя аргументами (бинарный предикат); эти два аргумента представляются двумя узлами, а собственно предикат - направленной дугой, связывающей эти узлы.

Особенность семантической сети (которая в то же время является ее недостатком) заключается в целостности системы, выполненной на ее основе, не позволяющей разделить базу знаний и механизм выводов. Обычно интерпретация семантической сети определяется с помощью использующих ее процедур. Эти процедуры основаны на нескольких методах, но наиболее типичный из них - это способ сопоставления частей сетевой структуры. Он основан на построении подсети, соответствующей вопросу, и сопоставлении ее с базовой сетью. При этом для исчерпывающего сопоставления с вершинами переменных подсети присваиваются гипотетические значения.

2.4. Фреймы

В области искусственного интеллекта термин "фреймы" относится к специальному методу представления общих концепций и ситуаций. Знания о конкретном объекте можно представить в виде набора признаков. А знания о сложной ситуации, кроме этого, должны еще содержать возможные действия, а также условия, определяющие, когда эти действия должны выполняться. Марвин Минский предложил гипотезу о том, что знания группируются в "сгустки" или модули, которые он назвал фреймами и описывает их следующим образом**:**

"Фрейм - это структура данных, представляющая стереотипную ситуацию, вроде нахождения внутри некоторого рода жилой комнаты, или сбора на вечеринку по поводу рождения ребенка. К каждому фрейму присоединяется несколько видов информации. Часть этой информации - о том, как использовать фрейм. Часть о том, чего можно ожидать далее. Часть о том, что следует делать, если эти ожидания не подтвердятся".

Фрейм по своей организации во многом похож на семантическую сеть (фактически и семантические сети, и фреймы можно рассматривать как сетевые модели). Фрейм является сетью узлов и отношений, организованных иерархически, где верхние узлы представляют общие понятия, а нижние узлы более частные случаи этих понятий.

В отличие от семантической сети в системе, основанной на фреймах, понятие в каждом узле определяется набором атрибутов (например, имя, должность, возраст) и значениями этих атрибутов (например, Сидоров, менеджер, 35_лет), а атрибуты называются слотами. Каждый слот может быть связан с процедурами (произвольными машинными программами), которые выполняются, когда информация в слотах (значения атрибутов) меняется. Пример такого узла показан на рис.2.1

С каждым слотом можно связать любое число процедур. Следующие три типа процедур чаще всего связываются со слотами:

1. Процедура "*если - добавлено*" Выполняется, когда новая информация помещается в слот.
2. Процедура "*если - удалено*" Выполняется, когда информация удаляется из слота.
3. Процедура "*если - нужно*" Выполняется, когда запрашивается информация из слота, а он пустой.

Эти процедуры могут следить за приписыванием информации к данному узлу и проверять, что при изменении значения производятся соответствующие действия.

Системы, основанные на фреймах, хороши в тех предметных областях, где важную роль играют форма и содержание данных, например, интерпретация визуальной информации или понимание речи.

Достоинство системы, использующей фреймы, заключается в том, что те элементы, которые традиционно используются в описании объекта или события, группируются и благодаря этому могут

извлекаться и обрабатываться как единое целое.

2.5. Логические модели

Логика имеет дело, главным образом, с выявлением обоснованности утверждений, т.е. с методами, позволяющими доказать, можно ли данное заключение обоснованно вывести исходя из известных фактов. Для представления математического знания в математической логике давно пользуются логическими формализмами - главным образом, исчислением предикатов, которое имеет ясную формальную семантику и операционную поддержку в том смысле, что для него разработаны механизмы вывода. Поэтому исчисление предикатов было первым логическим языком, который применили для формального описания предметных областей, связанных с решением прикладных задач. Описания предметных областей, выполненные на логических языках, называются ***логическими моделями.***

В исчислении предикатов именам отношений соответствует термин "предикаты", объектом - "аргументы". Все используемые в исчислении предикатов логические выражения называются высказываниями и в классическом исчислении предикатов должны иметь значение либо ИСТИНА, либо ЛОЖЬ:

отчитывается (Иванов, Петров) состоит_из (агрегат_B, узел_C)

руководит (Петров, Иванов) входит_в (деталь_D, узел_C).

Объект может быть представлен либо как "константа", т.е. как конкретный индивидуум или класс индивидуумов, либо как "переменная", в результате чего конкретный индивидуум или класс индивидуумов остаются незаданными. Например, одна из интерпретаций выражения

является (X, программист)

может быть такой: "имеется некоторый объект X, который является программистом".

Порядок аргументов должен всегда задаваться в соответствии с интерпретацией предиката, принятой в рамках определенной предикатной области. Это значит, что программист должен принять решение о фиксированном, приемлемом для интерпретации порядке записи аргументов и соблюдать его с начала и до конца. Предикат может иметь произвольное число аргументов.

Отдельные высказывания могут объединяться в сложные высказывания с помощью следующих "логических связок":

И (**&**), ИЛИ (**V**) , НЕ (~) и импликация (→, Если .... То...).

Например, высказывание "Если Семенов написал программу, и она не работает, то Семенову следует отладить программу на следующий день" может быть записано как

написал (Семенов, программа) **&** ~работает (программа) →

отладить (Семенов, программа, следующий_день).

Для того чтобы в исчислении предикатов можно было манипулировать переменными, потребовалось ввести дополнительную структуру - "квантор". Кванторы служат для указания меры, в какой экземпляры переменных должны быть истинны для того, чтобы в целом высказывание стало истинным. Различают: "квантор общности", обозначаемый символом и "квантор существования", обозначаемый символом . Для квантора общности **(X)** все значения переменной в скобках, относящиеся к некоторой предметной области должны быть "истинны". При применении квантора общности можно перед высказыванием мысленно добавлять слово "все". Для квантора существования **(X)** требуется, чтобы только некоторые из таких значений были "истинны". При применении квантора существования можно перед высказыванием мысленно добавлять слова "некоторые", "существует такое". Например: "Все специалисты по ЭВМ являются программистами"

(X) (специалист по ЭВМ (X) → программист (X)).

Порядок, в соответствии с которым вводятся квантифицируемые переменные, может влиять на смысл утверждения. Например,

(X) (Y) (служащий (X) →руководитель (Y,X))

может быть интерпретировано как: "У каждого служащего есть руководитель". Если же изменить порядок кванторов, например

(Y) (X) (служащий (X) → руководитель (Y,X)),

то изменится и утверждение: "Есть такое лицо, которое руководит всеми". Используя данный механизм представления, можно выразить в стандартном виде многие сложные предложения, свойственные разговорному языку. При этом можно избежать двусмысленности таких предложений. Средства исчисления предикатов позволяют, не изменяя смысла, преобразовывать их в форму, удобную для обработки с помощью ЭВМ. Такое преобразование становится возможным благодаря тому, что одни логические связки могут быть выражены через другие:

A & B равносильно ~(~A ~B)

A -> B равносильно ~A **V** B.

Рассмотрим на примере процедуру логического вывода. Пусть даны факты:

факт 1: имеет_в_составе (комплекс_A, агрегат_B)

факт 2: имеет_в_составе (агрегат_B, узел_C)

факт 3: имеет_в_составе (узел_C, деталь_D)

и два правила:

правило 1: **(X,Y)** (имеет_в_составе (X,Y) → входит_в (Y,X))

правило 2: **(X,Y,Z)** (имеет_в_составе (X,Y) **&** входит_в (Z,Y) → входит_в (Z,X)).

Эти правила можно интерпретировать так: "Для всех значений X и Y, если X имеет в составе Y, то Y входит в X", "Для всех значений X, Y, Z, если X имеет в составе Y и Z входит в Y, то Z входит и в X". Посмотрим, сможем ли мы с помощью приведенных выше правил сделать заключение о том, что "деталь_D входит в агрегат_B". Для решения этой задачи нужно, на основании правила 1 и факта 3 сделать промежуточный логический вывод**:** входит_в (деталь_D, узел_C), а затем, используя этот факт, правило 2 и факт 3, сделать заключение об интересующем факте: входит_в (деталь_D, агрегат_B).

При обработке знаний возникает проблема, связанная с различной природой процесса рассуждений в исчислении предикатов и ходом рассуждений, основанных на здравом смысле. Например, когда доказано, что A → C, то C остается истинным и в случае появления любого дополнительного факта B, т.е. A & B → C. Исчисление предикатов "монотонно", и отсюда следует, что любые умозаключения аддитивны и нет необходимости в их пересмотре. Совершенно ясно, что такая монотонность неприемлема в реальном мире. В реальной жизни мы часто вынуждены изменять умозаключение или отказываться от него при появлении новых фактов. О формальных системах, для которых следует это предусмотреть, говорят, что они "немонотонны". Из-за монотонности исчисления предикатов логические модели отошли на второй план, уступив место продукционным системам и системам, основанным на фреймах, при описании предметных областей из реального мира.

2.6. Модели нечетких знаний.

Знания не всегда могут быть описаны точно. Часто встречаются так называемые "нечеткие" знания. Люди повседневно решают проблемы и делают заключения в среде нечетких знаний, и для того, чтобы интеллектуальные системы вышли за рамки простых символьных выводов и приблизились к мышлению человека, обладая такими возможностями, как гибкость, широкий кругозор, адаптируемость, необходимы модели представления нечетких знаний и механизм выводов, работающий в их среде.

Человек обладает самыми различными формами знаний, и пока еще не совсем ясно, каким образом они структурированы в его мозгу. Однако для использования знаний в компьютерах они должны быть формализованы и описаны. В этом состоит проблема представления знаний. Пока знания не формализованы и не описаны, они не могут быть использованы в компьютерах, точно так же не могут быть использованы техническими средствами и нечеткости, предварительно не описанные в некоторой форме.

Все нечеткости, с которыми до сих пор приходилось иметь дело в инженерии знаний, можно классифицировать следующим образом: 1) недетерминированность выводов; 2) многозначность; 3) ненадежность; 4)неполнота; 5) нечеткость или неточность.

Недетерминированное управление выводом наиболее характерно для систем ИИ. Такое управление необходимо потому, что знания накапливаются фрагментарно, и нельзя априори определить цепочку логических выводов, в которых они используются. Другими словами, необходимо методом проб и ошибок выбрать некую цепочку выводов, и в случае неуспеха организовать перебор с возвратами для поиска другой цепочки и т.д. Такое управление является предпосылкой проявления гибкости и интеллектуальных способностей, позволяющих найти выход в самых различных ситуациях. Поскольку эффективность простого поиска низка, возникает необходимость определения пути, по которому следует начать поиск в первую очередь.

Многозначность интерпретации - обычное явление при понимании естественных языков и распознавании изображений и речи. При понимании естественных языков большими проблемами становятся многозначность смысла слов, многозначность их подчиненности, многозначность местоимений в контексте и т.п. Как правило, устранение многозначности обеспечивается за счет более широкого контекста и семантических ограничений. При обработке изображений часто многозначна интерпретация элементов изображения (контуры, области и т.п.). В общем случае устранить многозначность помогают более широкие пространственные отношения и другие способы.

2.6.1. Ненадежные знания и выводы.

В задачах, которые решают интеллектуальные системы, иногда приходится применять ненадежные знания и факты, представить которые двумя значениями - истина и ложь (1 или 0) - трудно. Существуют знания, достоверность которых, скажем 0.7. Такую ненадежность в современной физике и технике представляют вероятностью, подчиняющейся законам Бейеса, для удобства назовем ее байесовской вероятностью, но достоверность знания, т.е. степень незнания не носит вероятностный характер.

Дуда, Харт и Нильсон видоизменили формулы Байеса для выводов в инженерии знаний и предложили метод выводов, названный субъективным байесовским методом. Из формул Байеса следуют

соотношения:

P(A |X) – ?????????? ??????? ? ??? ???????, ??? ??????? ? ?????????.

??????? ?????????

?(?*?) = ?(?)*?(??) = ?(?)*?(?/?)

??????????? ???????????? ???? ??????? ????? ????????????? ??????????? ?????? ?? ???? ??????? ?? ???????? ??????????? ??????? ??? ?

$$P(A \mid X) = \frac{P(X \mid A) * P(A)}{P(X)}$$

$$P(\overline{A} \mid X) = \frac{P(X \mid A) * P(A)}{P(X)}$$

откуда

$$\frac{P(A \mid X)}{P(\overline{A} \mid X)} = \frac{P(X \mid A) * P(A)}{P(X \mid \overline{A}) * P(\overline{A})}$$

где $\overline{A}$ - дополнение множества **A**.

Понятие вероятности также используется для определения такого показателя, характеризующего неопределенность, как шансы.

$$O(A) = \frac{P(A)}{P(\overline{A})} = \frac{P(A)}{1 - P(A)}$$

шансы (odds)

и апостериорные шансы **A** при получении доказательства **X**

$$O(A \mid X) = \frac{P(A \mid X)}{P(\overline{A} \mid X)} = \frac{P(A \mid X)}{1 - P(A \mid X)}.$$

При этом вероятность **P** и шансы **O** связаны отношением:

$$P = O/(O+1),$$

если определены шансы, то можно получить вероятность.

Шортлифф (E.Shortliff) разработал схему, основанную на так называемых коэффициентах уверенности, которые он ввел для измерения степени доверия к любому данному заключению, являющемуся результатом полученных к этому моменту свидетельств. Коэффициент уверенности (КУ) - это разность между двумя мерами:

КУ[h/e] = MD[h/e] - MнD[h/e],

где

КУ[h/e] -уверенность в гипотезе **h** с учетом свидетельств **e**;

MD[h/e] -мера доверия **h** при свидетельстве **e**;

MнD[h/e] - мера недоверия **h** при свидетельствах **e**.

КУ может принимать любое значение на отрезке [-1,1]: -1 (абсолютная ложь), 0 (полное незнание), 1 (абсолютная истина). MD и MнD могут изменяться лишь от 0 до 1. Таким образом, КУ - это простой способ взвешивания свидетельств "за" и "против". Ни КУ, ни MD, ни MнD не являются вероятностными мерами. MD и MнD не являются выборками из какой-нибудь популяции и, следовательно, им нельзя дать статистическую интерпретацию.

Часто бывает необходимо отличать случай противоречивых свидетельств (MD и MнD обе велики) от случая недостаточной информации (MD и MнD обе малы), для этого следует использовать сумму MD и MнD, если **0.5*(MD + MнD) ≈1**, то случай противоречивых свидетельств, а при **0.5*(MD + MнD) ≈ 0** - случай недостаточной информации.

Для объединения свидетельств применяется формула, по которой новую информацию можно непосредственно сочетать со старыми результатами. Формула для MD выглядит следующим образом:

MD[h/e1,e2] = MD[h/e1] + MD[h/e2] * (1-MD[h/e1]),

где запятая между e1 и e2 означает, что e2 следует за e1. Аналогичным образом уточняются значения MнD. Смысл формулы состоит в том, что эффект второго свидетельства (e2) на гипотезу h при заданном свидетельстве e1 сказывается в смешении в сторону полной определенности на расстояние, зависящее от второго свидетельства.

Свойства формулы:

- Она симметрична в том смысле, что порядок e1 и e2 не существенен; по мере накопления подкрепляющих свидетельств MD (или MнD) движется к определенности.
- Объединенная мера доверия оказывается выше, чем при учете каждого свидетельства, взятого отдельно. Это согласуется с нашей интуицией, что несколько показывающих одно и то же направление свидетельств подкрепляют друг друга.

Схема Шортлиффа допускает также возможность того, что, как и данные, правила могут быть ненадежными. Каждое правило снабжено "коэффициентом ослабления", числом от 0 до 1, показывающим

надежность этого правила.

2.6.2. Вероятностная логика

Нильсон предложил идею расширения логики и ввел понятие вероятностной логики, в которой всем логическим формулам приписывается вероятность. Здесь вероятность вновь соответствует законам Байеса. Рассмотрим три логические формулы в логике высказываний A, A>B, B. Представим следующие вертикальные векторы:

1	2	3	4	
1	1	0	0	<———— истина и ложь A ,
1	0	1	1	<———— истина и ложь A ——>B,
1	0	1	0	<———— истина и ложь B,

где

1 - вариант истинности A, A ——>B, B;
2 - вариант истинности A и лжи A ——>B, B;
3 - вариант лжи A и истинности A ——>B, B;
4 - вариант лжи A, B и истинности A ——>B;

Эти три логические формулы подобраны так, что возможны только четыре указанных выше случая (когда нет противоречий). Это так называемые возможные варианты, например, A,A ——>B истина, B ложь - это варианты, содержащие противоречия.

Если выбрать один из возможных вариантов, то образуется традиционная двузначная логика. В вероятностной логике рассматриваются состояния, когда одновременно с некоторой вероятностью могут существовать несколько возможных вариантов. Например, пусть вероятность, с которой возможен вариант 1, равна 0.4, а вероятности вариантов 2, 3, 4 соответственно равны 0.3, 0.2, 0.1 (сумма вероятностей всех возможных вариантов равна 1), тогда представим следующим образом вектор вероятностей возможных вариантов:

$$P=\begin{bmatrix}0.4\\0.3\\0.2\\0.1\end{bmatrix}$$

Если построить матрицу M, элементами которой служат вертикальные векторы, представляющие возможные варианты, то с помощью матричной операции MP=V можно вычислить вероятности выбора каждой логической формулы. В данном примере

$$M\times P=\begin{bmatrix}1&1&0&0\\1&0&1&1\\1&0&1&0\end{bmatrix}\times\begin{bmatrix}0.4\\0.3\\0.2\\0.1\end{bmatrix}=\begin{bmatrix}0.7\\0.7\\0.6\end{bmatrix}=V,$$

т.е. эти вероятностные возможные варианты имеют состояние "истина" с вероятностью 0.7(A), 0.7(A →B), 0.6(B).

2.6.3. Неполные знания и немонотонная логика

В реальных ситуациях часто бывает чрезвычайно сложно описать полностью задачу. Например, знание "птицы летают" - верное, однако встречаются и нелетающие птицы, т.е. это неполное знание. Или же задача о "миссионерах и туземцах", т.е. задача о переправе в одной лодке через реку, она становится неразрешимой, если вдруг нет весел или на дне лодки дыра. Подобных причин может быть множество, а раз так, то полностью описать их невозможно. Исходя из здравого смысла, считают, что раз существует лодка, то ею можно пользоваться. Так же можно перечислить все предметы, которые находятся в комнате, но того, чего в ней нет, перечислить невозможно, поскольку это бесчисленное множество предметов. Точно так же можно перечислить верные знания (в некоторой предметной области), но перечислить неверные знания и разумно их определить невозможно.

Поэтому удобно в базе знаний определять исключительно правильные знания, а все, что не определено, считать заведомо неверным. Утверждения, которые не упомянуты ни как истинные, ни как ложные, принято относить к ложным. Это называют гипотезой закрытого мира.

Классическая логика исходит из предпосылки, что набор определенных в ней аксиом (знаний) полон, и правильный вывод не меняется, даже если впоследствии добавлена новая аксиома. Такое свойство называют монотонностью. Если допустить, что в базу знаний добавлено такое знание: "как правило, птицы летают (за некоторым исключением)", то обнаружится свойство немонотонных выводов. А именно, при добавлении новой аксиомы иногда возможно отрицание вывода, который считался верным в некоторой системе аксиом (базе знаний). Рассмотрим систему аксиом, состоящую из следующих знаний:

а) "как правило, птицы летают (за некоторым исключением)";
б) "пингвины не летают"
в) "Пикколо есть птица"

Из этой системы аксиом можно сделать вывод, "Пикколо летает". Однако впоследствии получена более подробная информация, выяснилось, что Пикколо это пингвин. И в систему аксиом внесено добавление "Пикколо есть пингвин". Теперь вывод, полученный ранее "Пикколо летает", отрицается и делается новый вывод, что "Пикколо не летает". Это пример немонотонности выводов.

К системам, связанным с неполными знаниями и управлением такими знаниями, относится система поддержания значений истинности. В базе знаний этой системы, неполной и содержащей противоречия, все знания делятся на достоверные и недостоверные, и предусмотрено упорядочение базы с целью устранения недостоверных знаний. В этой системе достоверно истинные знания относятся классу "IN", а знания, истинность которых недостоверна либо в истинность которых нет повода верить, - к классу "OUT". Если при добавлении новых значений возникает противоречие, то выполняется повторная проверка классов знаний.