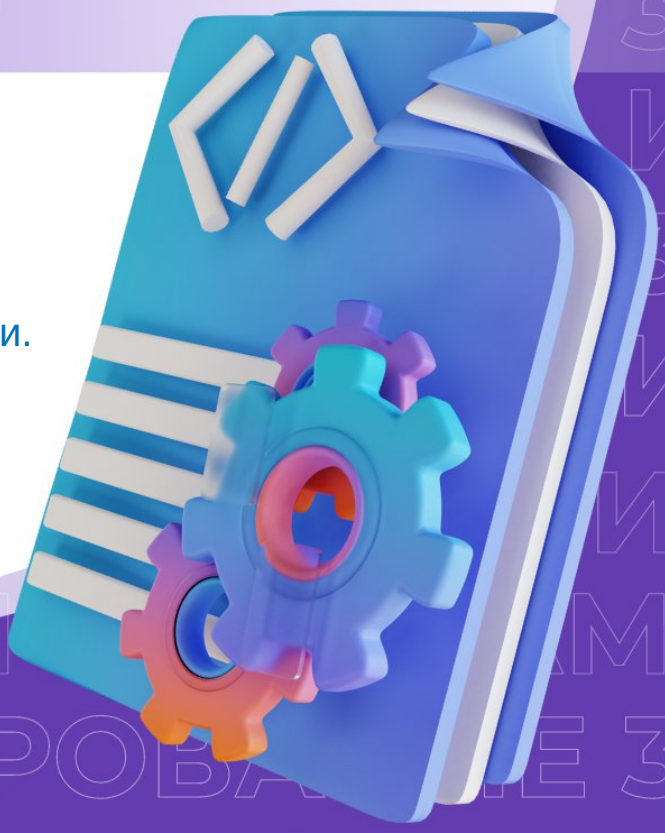
СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В АПК

приоритет2030
лидерами становятся

ТЕМА:
Представление знаний, рассуждений и задач в интеллектуальных информационных системах

КОНСПЕКТ ЛЕКЦИИ



Преподаватель:

Аникуев Сергей Викторович
к.т.н., доцент, доцент кафедры электротехники, автоматики и метрологии.

приоритет2030^
лидерами становятся

Тема 1 Представление знаний, рассуждений и задач в интеллектуальных информационных системах

Вопрос 1. Представление в базе знаний

Вопрос 2. Исчисление предикатов

Вопрос 1. Представление в базе знаний

Данные - это описание и значения свойств объектов, представленные в заданной форме. В частности, данные - это значения констант и переменные.

Знания - это представленные на естественном языке описания объектов, их характеристик в соответствии с фактами предметной области, представленные отношениями с помощью каузальной (причинно-следственной) зависимости между отношениями с целью констатации фактов или принятия решения.

Декларативные знания - это факты, правила, отношения.

Процедурные знания - это правила и процедуры.

Совокупность правил, соединенных союзами & (и), or (или), not (не), является в программе **предложениями**.

Совокупность предложений, не связанных союзами &, or, not, образует понятие **утверждение**.

Утверждение характеризуется *пространством утверждений*. Введем обозначения:

a, c - имена отношений;

D - действие;

- переменные, константы;

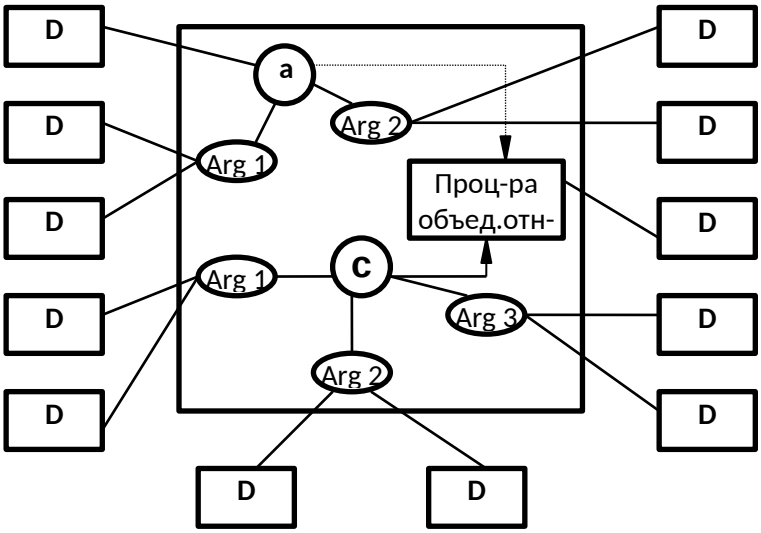


Рис. 3.1. Формирование утверждений

Представление знаний в рабочей памяти складывается из двух компонентов:

- текущих значений данных;
- агента (списка используемых в данный момент правил).

В соответствии с текущей ситуацией утверждение на основе значений данных (рис.3.1) настраивается на текущую ситуацию. В соответствии с этим приобретают смысл отношения пространства утверждения, которые объединены в агенте.

Пространство утверждения содержит совокупность релевантных (соответствующих) правил для выполнения заданной в утверждении процедуры, которая реализуется с помощью внешних ссылок пространства на значения данных.

Интеллектуальность экспертной системы оценивается на основе *релевантности*, здесь релевантность заключается в выборе необходимых правил для принятия решения.

Релевантность обеспечивается:

- ***используемой технологией*** (например, объектно-ориентированной);

вся совокупность объектов подразделяется на классы;

выбор знаний из базы знаний осуществляется на основе определения требуемых объектов. Выбор происходит в 2 этапа:

- на первом этапе осуществляется сопоставление искомого объекта с объектами, хранящимися в базе знаний (например, по имени утверждения; см. Рис.3.2);
- на втором этапе выбор объекта уточняется на основе значений данных, содержащихся в рабочей памяти и во внешних ссылках пространства утверждения, если значения данных совпадают, значит, используется данный объект для данного утверждения.
- ***механизмом доступа к базе знаний***;
- ***механизмом сопоставления с образцами, хранящимися в базе знаний*.**

Механизм сопоставления может быть: синтаксический, параметрический, функциональный и принудительный.

Синтаксический механизм сопоставления: выделенные значения данных (понятий) совпадают со значениями данных пространства утверждения по написанию или по значению.

Параметрический механизм сопоставления: выделенные значения данных совпадают со значениями пространства утверждения по заданным параметрам или понятием предметной области.

Функциональный механизм сопоставления: заданные значения данных совпадают по выполняемым функциям.

Принудительный механизм сопоставления: выбирается (назначается) разработчиком экспертной системой.

Формализация знаний о проблемной области

Описание проблемной информации начинается с представления всех объектов, классов, объединенных объектов и понятий, их определяющих до значений данных, которые могут иметь место.

Для представления всей этой информации используется ***таксономическая классификационная схема***. В корне таксономической схемы лежит основное понятие, описывающее проблемную область. Основное понятие таксономической схемы не должно иметь предшествующих понятий.

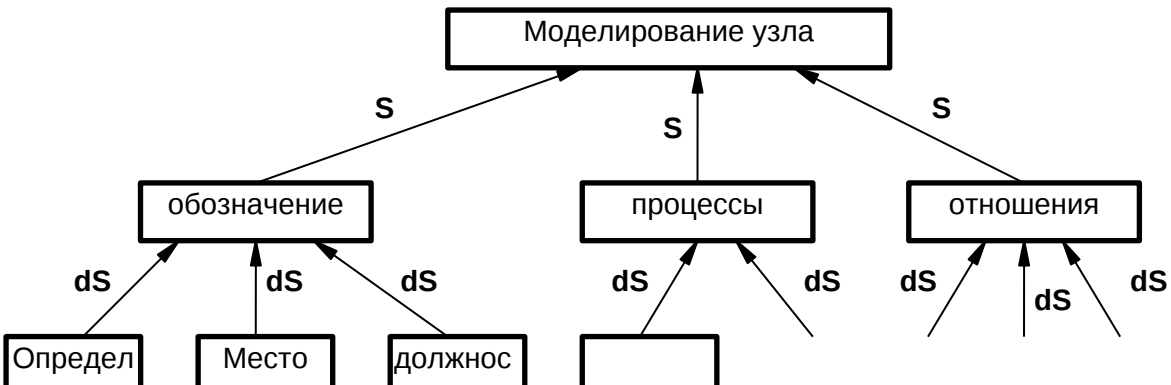


Рис. 3.2.

Корень таксономической схемы - основной узел. Совокупность узлов подсетей и элементов определяет таксономическая схема (таксономия). Дуги таксономической схемы имеют заданные значения S/E - определяют назначение текущего узла, как «*является множеством*»/«*состоит из элементов*». dS/dE - определяют назначение узла, как «является подмножеством»/ «является элементом», с направленными вверх стрелками дуг.

В начале выбор осуществляется на основе внутренней структуры объектов, а затем на основе внешних ссылок пространства.

Модели представления знаний

Основными классическими моделями на сегодняшний день являются фреймы, семантические сети, исчисления предикатов первого порядка, комбинация трех перечисленных моделей, продукционные системы.

Фреймы

Фрейм - структура данных для представления стереотипной ситуации.

Фрейм определяется имением и совокупностью слотов. Слоты могут иметь различные значения и оставаться незаполненными для конкретной ситуации. Слоты могут характеризовать объекты, классы и утверждения. Значением слота может быть имя другого фрейма.

Слоты фрейма, кроме декларативного и процедурного назначения, могут определяться принудительно следующим образом:

- слоты, которые приводят к выполнению действий в зависимости от полученных фреймом значений при сопоставлении, назовем их *слотами 1;*
- слоты, которые активизируются вне зависимости от значений других слотов, назовем их *слотами 2.*

Таким образом, фрейм может определять исходные состояния, например, состояние *A* и переход в новое состояние *B* на основе исходного описания с помощью фрейма *A'* в новое состояние, описываемое фреймом *B'*.

При использовании фреймового представления предполагается, что значения верхних слотов фрейма заданы, а значения нижних слотов фрейма заполняются в соответствии с ситуацией. Использование фреймов можно сравнить с использованием процедур в традиционных языках программирования. Описание процедуры в программе представляет собой *фрейм-прототип*. Действие, выполняемое процедурой, соответствует понятию *фрейм-реализации*.

Использование фреймов предполагает реализацию в информационной системе на основе объектно-ориентированного подхода, где классы объединяют в совокупности однотипных объектов. Объекты имеют свойства, характерные для данного класса. При построении иерархии классов устанавливается соответствие между классами и фреймами и характеристиками объектов со слотами. Родственные свойства объектов наследуются, то есть объект, находящийся на нижнем уровне классификации, имеет все свойства, присущие данному классу объектов.

Свойства объектно-ориентированной технологии:

1. **инкапсуляция -** скрытие информации, то есть доступ к определенным значениям слотов осуществляется при появлении или отсутствии значений данных в слотах 1 - 2;
2. **полиморфизм** - использование объектно-ориентированной технологии с определенно заданными методами, то есть методы доступа к информации определяются дополнительно;
3. **наследование.**

В отличие от описания процедур в традиционных языках программирования использование фреймов осуществляется не только по имени, но и по состоянию фрейма, описывающего ситуацию.

Таким образом можно получить модель состояния *B*’, имея только часть информации о состоянии *A*’.

Все модели представления данных предназначены для воплощения в правилах логического вывода. Этот основное назначение модели. Модели в правилах представляются утверждениями.

Фрейм — информационная структура, содержащая описание объекта в виде атрибутов и их значений. В отличие от моделей других типов во фреймовых моделях фиксируется жесткая структура информационных единиц, которая называется ***протофреймом***. В общем виде она выглядит следующим образом:

(Имя фрейма:
Имя слота 1(значение слота 1)
Имя слота 2(значение слота 2)
. . . . . .
Имя слота K (значение слота K)).

Значением *слота* может быть практически что угодно (числа или математические соотношения, тексты на естественном языке или программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов). В качестве значения слота может выступать набор слотов более низкого уровня, что позволяет во фреймовых представлениях реализовать "принцип матрешки".

При конкретизации фрейма ему и слотам присваиваются конкретные имена и происходит заполнение слотов. Таким образом, из протофреймов получаются *фреймы - экземпляры.* Переход от исходного протофрейма к фрейму - экземпляру может быть многошаговым за счет постепенного уточнения значений слотов.

Пример. Рассмотрим структуру фрейма табл. 4.1:

Таблица 4.1

Фамилия	Год рождения	Специальность	Стаж
Попов	1965	Слесарь	5
Сидоров	1946	Токарь	20
Иванов	1925	Токарь	30
Петров	1937	Сантехник	25

(Список работников:
Фамилия (значение слота 1);
Год рождения (значение слота 2);
Специальность (значение слота 3);
Стаж (значение слота 4)).

Если в качестве значений слотов использовать данные табл. 4.1, то получится фрейм – экземпляр.

(Список работников:
Фамилия (Попов - Сидоров - Иванов - Петров);
Год рождения (1965 - 1946 - 1925 - 1937);
Специальность (слесарь - токарь - токарь - сантехник);
Стаж (5 - 20 - 30 - 25)).

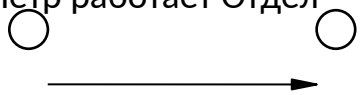
Связи между фреймами задаются значениями специального слота с именем "Связь".

Фрейм-реализация - это внешние ссылки пространства и настройка на конкретную ситуацию.

Семантические сети

В традиционном понимании **семантическая сеть** - это конструкция двух основных компонентов: узлов и дуг. Узлы моделируют понятия предметной области, дуги моделируют отношения между парой понятий.

Пример: Петр работает Отдел



С помощью семантической сети можно промоделировать достаточно сложные отношения между различного рода понятиями, описывающими достаточно много состояний проблемной области. Рассмотрим Рис. 4.1.

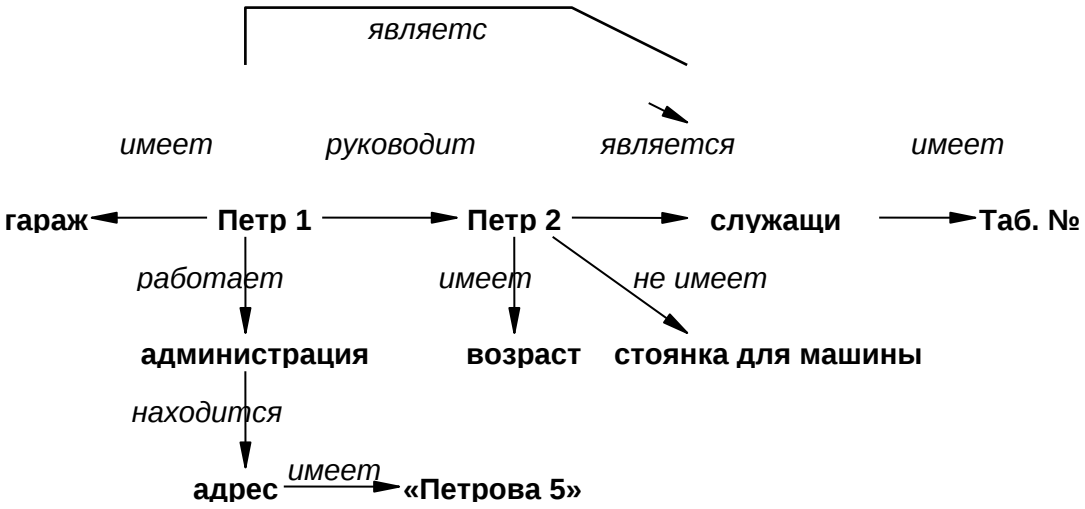


Рис.4.1. Пример семантической сети

Отношения между объектами в семантической сети можно установить как «все со всеми», то есть каждый узел семантической сети может иметь отношение с каждым ее узлом.

Семантическую сеть можно представить в виде бинарного предиката, или в виде отношения с двумя аргументами (работает (Петр, отдел), в общем виде – работает (X,T)). Представить эту информацию можно в списковой структуре данных.

Кроме того, описание семантической сети можно представить в виде *n* - аргументного отношения между понятиями при условии, что имя отношения будет иметь одно и то же значение.

При реализации семантической сети в Базе Данных (БД) информация описывается в виде троек: объект, атрибут, значение (то, что присуще БД).

Недостатки использования семантической сети:

- трудно реализуется свойство наследования между понятиями;

- сложная реализация временных отношений между понятиями.

Семантические сети используются в основном для представления декларативной информации.

Основные виды отношений, используемых в моделях представления знаний:

• *каузальные:*

а) причина и следствие (в прямой цепочке логических рассуждений);

б) следствие и причина (в обратной цепочке логическихрассуждений);

1. *логико-арифметические*:

а) арифметические;

б) логические;

1. *теоретико-множественные:*

а) отношение принадлежности - это часть целого или целое части;

- классификационные;
- *характеристические (присущие всем БД)*:

а) атрибутивные иметь свойство

быть свойством

б) идентифицирующие быть именем

иметь имя

1. *квантифицирующие*:

а) иметь количественное значение;

б) иметь лингвистическое значение;

• *динамические:*

а) изменить положение;

б) изменить ориентацию (на 90^0);

1. *временные*:

а) быть раньше;

б) быть позже;

в) быть одновременно;

- *пространственные*:

а) быть расположенным;

б) расстояние между;

1. *инструментальные* ~~вып~~олняться посредством

выполняться с помощью

Вопрос 2. Исчисление предикатов

Основные компоненты данной модели являются:

• *Обработка фактов.*

Факт является основной единицей модели представления знаний.

Факт - это отношение (предикат), которое представляет объект и совокупность свойств объекта или процедур, которые определяются аргументами.

Аргументы в предикатах задаются с помощью констант или с помощью переменных.

Пример: *является (Петр, Х)*

живет (У, Йошкар-Ола, Т)

где *Х, У, Т* - переменные, которые определяют интерпретацию факта в утверждении. Аналогом утверждения в исчислении предикатов является понятие «высказывание».

Высказывание - это элементарное утверждение для использования в сложном предложении.

Правила формирования предикатов:

- каждый предикат имеет имя;
- каждый предикат обязательно имеет аргументы;
- количество аргументов предиката не ограниченно;
- последовательность аргументов в задании предиката определяется последовательностью их интерпретаций в заданной проблемной области, и последовательность заранее известна программисту и не меняется на протяжении всей программы.

• *Правила.*

Простейшим видом утверждений в модели представления знаний являются правила.

В общем смысле, **правила** - это сложные умозаключения или высказывания, которые формируются из простых высказываний с помощью соответствующих связок &(and), U(or), ~(not), ® (импликация).

Связка импликация (®) предназначена для формирования каузального отношения.

Пример:

• работает (Петр, IBM) ® оператор (Петр);

• разработал (Петр, программа) ~ работает (программист) ® исправить (Петр, программа, вечером) U передать (программа, У).

• *Сложные правила.*

Используют для утверждения истинности на основе фактов, заданных в исходных данных (база данных) или на основе интерпретации значений переменных при доказательстве. Мерой истинности переменных, используемых при доказательстве правила или сложного высказывания являются **кванторы.**

" (X) - квантор общности; $(X) - квантор существования.

Квантор общности читается: «Для всех *X*» и определяет область «все».

Квантор существования читается, как «Существует значение переменных *X*» и определяется как «некоторые».

Замечание: между кванторами не допустимо использование связок. Связки используются только между высказываниями.

Пример: - " (X) (Y) руководит *(Y,X)* & отчитывается *(X,Y)*

-" (Y) (X) руководит *(Y,X)* & отчитывается *(X,Y)*.

4.*Утверждения*

На языке логики предикатов утверждения могут быть представлены:

1. правилами, используемыми для доказательства;
2. правилами вывода.

Правила вывода - это сложные предложения, которые определяют новые правила и факты, то есть правила вывода предназначены для задания новых стратегий с использованием новых фактов.

Рассмотрим выводы:

-" *(X)* руководит *(X,Y)* ® отчитывается *(Y,X)* (правило)

-" *(X,Y,Z)* руководит *(X,Y)* & руководит *(X,Z)* ®

отчитывается *(Z,Y)* (правило вывода),

то есть отсюда выводится новое правило: руководит*(Y,Z)*

Логика предикатов, называемая также логикой первого порядка, допускает четыре типа выражений.

1. *Константы. Они служат именами индивидуумов*: объектов, людей или событий. Константы представляются символами, например, Жак_2 (добавление 2 к слову Жак указывает на вполне определенного человека среди людей с таким именем), Книга_22, Посылка_8.

2. *Переменные. Обозначают имена совокупностей*, таких как человек, книга, посылка, событие. Символ Книга_22 представляет вполне определенный экземпляр, а символ книга указывает либо множество "всех книг", либо "понятие книги". Символами ***x, y, z*** представлены имена совокупностей (определенных множеств или понятий).

3. *Предикатные имена. Они задают правила соединения констант и переменных*, например, правила грамматики, процедуры, математические операции. Для предикативных имен используются символы, например: Фраза, Посылать, Писать, Плюс, Разделить. Предикатное имя иначе называется предикатной константой.

4. Функциональные имена *они представляют такие же правила, как и предикаты*. Чтобы не спутать с предикатными именами, функциональные имена пишут одними строчными буквами: фраза, посылать, писать, плюс, разделить. Их называют так же функциональными константами.

Символы, которые применяются для представления констант, переменных, предикатов и функций, не являются "словами русского языка". Они суть символы некоторого представления - слова "объектного языка" (в нашем случае языка предикатов).

Представление должно исключать всякую двусмысленность языка. Поэтому имена индивидуумов содержат цифры, приписываемые к именам совокупностей. Жак_1 и Жак_2 представляют двух людей с одинаковыми именами. Эти представления суть конкретизации имени совокупности "Жак". **Предикат - это предикатное имя вместе с подходящим числом термов.**

Примеры применения логики для представления знаний

Проиллюстрируем синтаксис логики предикатов, сопоставляя нескольким русским фразам их перевод на язык логического формализма.

1. По русски: Жак посылает книгу Мари,

Логически: Посылка (Жак_2, Мари_4, Книга_22).

2. По русски: Каждый человек прогуливается,

Логически: " x (Человек(x) É Прогуливается(x)).

3. По русски: Некоторые люди прогуливаются,

Логически: $ x (Человек(x) Ù Прогуливается(x)).

Квантор — общее название для логических операций, ограничивающих область истинности какого-либо **предиката**.

Сравнивая два последних примера, видим, что замена прилагательного "каждый" на "некоторые" влечет при переводе не только замену квантора " на $, но и замену связки É на Ù. Это иллюстрирует тот факт, что перевод фразы естественного языка на логический, вообще говоря, не является трафаретной операцией.

По русски: Ни один человек не прогуливается,

Логически: Ø ($ x (Человек(x) Ù Прогуливается(x))).

В исчислении предикатов имеется множество правил вывода. В качестве примера приведем классическое правило отделения :

(A, A -> B) / B

которое читается так "если истинна формула A и истинно, что из A следует B, то истинна и формула B ".Формулы, находящиеся над чертой, называются посылками вывода, а под чертой - заключением. Это правило вывода формализует основной закон дедуктивных систем: **из истинных посылок всегда следуют истинные заключения**. Аксиомы и правила вывода исчисления предикатов первого порядка задают основу формальной дедуктивной системы, в которой происходит формализация схемы

рассуждений в логическом программировании. Можно упомянуть и другие правила вывода.

1. Если из A следует B и B ложно, то и A ложно.

2. Если A и B не могут одновременно быть истинными и A истинно, то B ложно.

3. Если либо A, либо B является истинным и A не истинно, то B истинно.

Решаемая задача представляется в виде утверждений (аксиом) f_1, f_2... f_n исчисления предикатов первого порядка. Цель задачи B также записывается в виде утверждения, справедливость которого следует установить или опровергнуть на основании аксиом и правил вывода формальной системы. Тогда решение задачи (достижение цели) сводится к выяснению логического следования (выводимости) целевой формулы B из заданного множества формул (аксиом) f_1, f_2... f_n. Такое выяснение равносильно доказательству значимости (тождественно- истинности) формулы

$f_1 \& f_2 \& \ldots \& f_n \rightarrow B$

или невыполнимости (тождественно- ложности) формулы

$f_1 \text{È} f_2 \text{È} \ldots f_n \text{È} \varnothing B$

Из практических соображений удобнее использовать доказательство от противного, то есть доказывать невыполнимость формулы. На доказательстве от противного основано и ведущее правило вывода, используемое в логическом программировании, - ***принцип резолюции***. Робинсон открыл более сильное правило вывода, чем modus ponens, которое он назвал *принципом резолюции* (или правилом резолюции). При использовании *принципа резолюции* формулы исчисления предикатов с помощью несложных преобразований приводятся к так называемой

дизъюнктивной форме, то есть представляются в виде набора дизъюнктов. При этом под дизъюнктом понимается дизъюнкция литералов, каждый из которых является либо предикатом, либо отрицанием предиката.

Приведем пример дизъюнкта:

$$\forall x(Q(x,c_2) \supset P(x,c_1)).$$

Пусть P- предикат уважать, c_1- Ключевский, Q- предикат знать, c_2- история. Теперь данный дизъюнкт отражает факт "каждый, кто знает историю, уважает Ключевского".

Приведем еще один пример дизъюнкта:

$$\forall x(P(x,c_1)P(x,c_2)).$$

Пусть P- предикат знать, c_1- физика, c_2- история. Данный дизъюнкт отражает запрос "кто знает физику и историю одновременно".

Таким образом, условия решаемых задач (факты) и целевые утверждения задач (запросы) можно выразить в дизъюнктивной форме логики предикатов первого порядка. В дизъюнктах кванторы всеобщности $\forall, \exists$, обычно опускаются, а связки $\supset, \neg, \wedge$ заменяются на $\rightarrow$ импликацию.

Вернемся к *принципу резолюции*. Главная идея этого правила вывода заключается в проверке того, содержит ли множество дизъюнктов R пустой (ложный) дизъюнкт. Обычно резолюция применяется с прямым или обратным методом рассуждения. Прямой метод из посылок A, A -> B выводит заключение B (правило modus ponens). Основной недостаток прямого метода состоит в его не направленности: повторное применение метода приводит к резкому росту промежуточных заключений, не связанных с целевым заключением. Обратный вывод является направленным: из желаемого заключения B и тех же посылок он выводит новое подцелевое заключение A. Каждый шаг вывода в этом случае связан всегда с первоначально поставленной целью. Существенный недостаток *метода*

резолюции заключается в формировании на каждом шаге вывода множества резольвент - новых дизъюнктов, большинство из которых оказывается лишними. В связи с этим разработаны различные модификации *принципа резолюции*, использующие более эффективные *стратегии поиска* и различного рода ограничения на вид исходных дизъюнктов. В этом смысле наиболее удачной и популярной является система ПРОЛОГ, которая использует специальные виды дизъюнктов, называемых дизъюнктами Хорна.

Процесс доказательства *методом резолюции* (от обратного) состоит из следующих этапов:

1. Предложения или аксиомы приводятся к дизъюнктивной нормальной форме.

2. К набору аксиом добавляется отрицание доказываемого утверждения в дизъюнктивной форме.

3. Выполняется совместное разрешение этих дизъюнктов, в результате чего получаются новые основанные на них дизъюнктивные выражения (резольвенты).

4. Генерируется пустое выражение, означающее противоречие.

5. Подстановки, использованные для получения пустого выражения, свидетельствуют о том, что отрицание отрицания истинно.

Рассмотрим примеры применения *методов поиска решений на основе исчисления предикатов*. Итак, заданы утверждения 1- 4 в левом столбце таблица 3.2. Требуется ответить на вопрос: "Существует ли человек, живущий интересной жизнью?" В виде предикатов эти утверждения записаны во втором столбце таблицы. Предполагается, что $\forall X(smart(X) = \neg stupid(X))$ и $\forall Y(wealthy(Y) = \neg poor(Y))$. В третьем столбце таблицы записаны дизъюнкты.

Таблица 3.2. Интересная жизнь		
Утверждения и заключение	Предикаты	Предложения(дизъюнкты)
1. Все небедные и умные люди счастливы	$\exists X(\neg poor(X) \wedge smart(X) \rightarrow happy(X))$	$poor(X) \wedge \neg smart(X)$ $happy(X)$
2. Человек, читающий книги, - неглуп	$\forall Y(read(Y) \rightarrow smart(Y))$	$\neg read(Y) smart(Y)$
3. Джон умеет читать и является состоятельным человеком	$read(John) \wedge \neg poor(John)$	$3a\ read(John)$ $3b\ \neg poor(John)$
4. Счастливые люди живут интересной жизнью	$\forall Z(happy(Z) \rightarrow exciting(Z))$	$\neg happy(Z) \vee exciting(Z)$
5. Заключение: Существует ли человек, живущий интересной жизнью?	$\exists W(exciting(W))$	exciting(W)
6. Отрицание заключения	$\neg\exists W(exciting(W))$	$\neg exciting(W)$

Отрицание заключения имеет вид (строка 6): $\neg\exists W(exciting(W))$

Символ `NIL` означает, что база данных выражений содержит противоречие и поэтому наше предположение, что не существует человек, живущий интересной жизнью, неверно.

В *методе резолюции* порядок комбинации дизъюнктивных выражений не устанавливался. Значит, для больших задач будет наблюдаться экспоненциальный рост числа возможных комбинаций. Поэтому в процедурах резолюции большое значение имеют также эвристики поиска и различные *стратегии*. Одна из самых простых и понятных *стратегий* - *стратегия* предпочтения единичного выражения, которая гарантирует, что

резольвента будет меньше, чем наибольшее родительское выражение. Ведь в итоге мы должны получить выражение, не содержащее литералов вообще.

Исследования, связанные с доказательством теорем и разработкой алгоритмов опровержения резолюции, привели к развитию языка логического программирования PROLOG (Programming in Logic). PROLOG основан на теории предикатов первого порядка. Логическая программа - это набор спецификаций в рамках формальной логики.

Основные недостатки модели исчисления предикатов

Доказательство на языке логики предикатов происходит монотонно и аддитивно, то есть рассуждения с помощью описания их предикатами не соответствует мыслительной деятельности эксперта.

В языке логики предикатов монотонность определяет последовательное применение всех правил в надежде, что какое-то из правил приведет к результату.

Однако, это может привести к комбинаторному взрыву, кроме того в применении логики отсутствует операционное варьирование правилами, то есть какое правило должно использоваться в данный момент не фиксируется.

Основные формализмы (инструментальные средства) представления знаний.

Язык обработки списков LISP

В языке LISP осуществляется обработка списка аргументами, которыми могут быть объекты, свойства объектов и функции. В свою очередь каждая функция может быть представлена списком, среди аргументов которого может быть другая функция.

Списками в языке LISP определяются все предикаты.

Пример: (руководит, X,Y(отчитывается,X,Y))

Работа формализма ориентирована на обработку функций. Управляющая структура языка LISP находит среди аргументов списка описание состояния объекта или ситуации и выполняет процедуру, включенную в этот список. Если процедура содержит список, то он сопоставляется с текущей ситуацией. *Управляющая структура языка LISP ориентирована на сопоставление с образцом, а модель на исчисление предикатов.*

Управляющая структура - это механизм применения правил в заданной модели с целью получения решений.

Язык логического программирования PROLOG

Prolog реализует управляющую структуру в виде обратной цепочки логического вывода, то есть используется доказательство от противного. Этим частично исключается монотонность.

Язык FORLOG.

Язык предназначен для обработки математических системных областей, которые можно описать с помощью математических формул.

- *Правила продукции*

Одна из областей используемых для данной модели является так называемая система сопоставления образцов. Данная модель используется при разработке системы искусственного интеллекта, по обработке изображений, лингвистической обработки текста, распознавания речи. Основным элементом модели является функция.

Функция - это элементарное предложение, простое для обработки и понимания его экспертом.

Функции связаны между собой в сеть. Значения данных объединены в структуре данных. Описание структуры определяют значения данных для формирования информационного ядра. Элементы информационного ядра являются аргументами функции. Та функция, которой соответствует информационное ядро при сопоставлении, загружается. Сеть функций дополняет доказательство для получения решений.

Стратегии поиска решений

Стратегия поиска решений задается цепочкой логического вывода и управляющей структурой. Таким образом, основными компонентами организации логического вывода в экспертных системах являются управляющая структура и стратегия поиска решений. Как правило, метод взаимодействия правил определяется управляющей структурой, а детализация принятия решения определяется стратегией поиска решения. Таким образом, стратегия поиска решений – это:

1. задание метоправил по поиску решений;
2. использование специфических эвристик;
3. механизм, который позволяет усовершенствовать метод поиска решений.

Организация логического вывода в экспертной системе, опираясь на управляющую структуру, использует соответствующий механизм вывода. *Механизм вывода в экспертной системе называют интерпретатором*. Работа интерпретатора осуществляется на основе данных, описывающих ситуацию, и знаний в виде правил, которые предполагают при объединении получение решения.

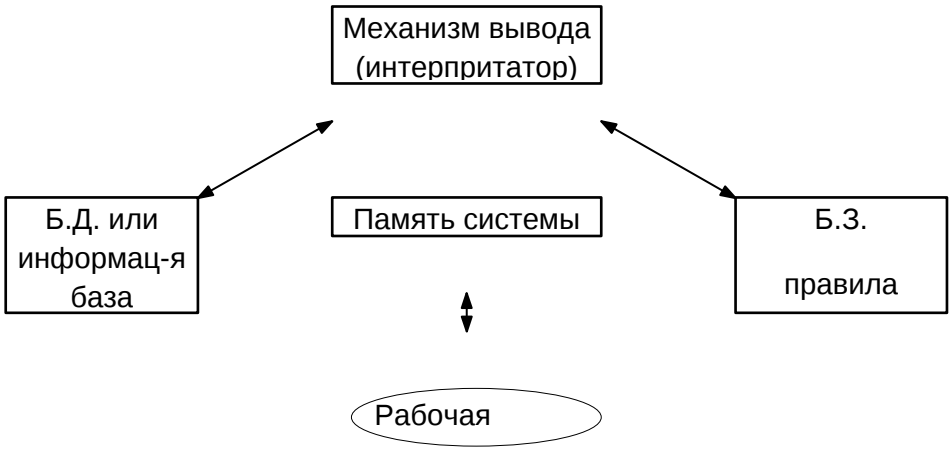


Рис.4.2

Этап выполнения интерпретации заключается в выполнении следующих действий: выборка, сопоставление, разрешение конфликтов, выполнение.

Выборка осуществляется двумя фазами:

1. синтаксическая фаза выборки;
2. семантическая фаза выборки.

Результат выборки - совокупность активных правил из базы знаний.

Синтаксическая выборка - это грубое определение правила по состоянию рабочей области памяти и переход к семантической выборке, то есть к конкретизации правил.

Семантическая выборка определяет соответствие выбранных правил, текущей цели или подцели для заданной предметной области. Отобранные правила переходят на стадию сопоставления, результатом которой является конфликтная совокупность правил. На этапе сопоставления выполняется заполнение выбранных правил текущими данными, при этом возможны конфликты.

Разрешение конфликтов - это этап интерпретации, который в зависимости от принятой стратегии используемого метода или цепочки логического вывода (прямой/обратной) формирует адженту (агенду) - список активных правил. После выбора агенды начинается ее выполнение. Выполнение предполагает реализацию правил в последовательности правил или модуля при сопоставлении с образцом. Результатом выполнения

является изменение состояния рабочей области, выполнения операций ввода/вывода, изменение памяти системы (рис.4.3).

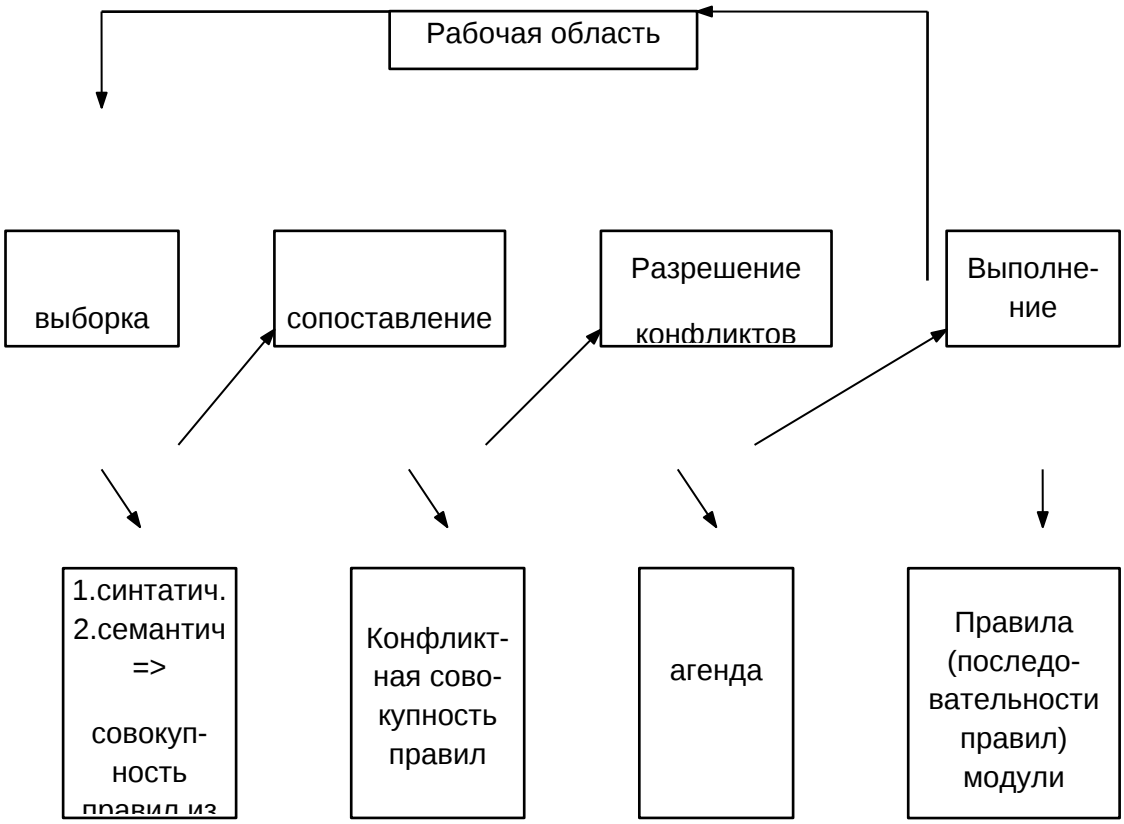


Рис.4.3.

Память системы (рис.4.2) содержит сведения о всех сеансах интерпретации при поиске решения.

Данная последовательность шагов (рис.4.3) может быть реализована в экспертных системах, представляемых двумя архитектурами:

- первая архитектура основана на управлении правилами; в ней рабочая область - это рабочая область(текущее состояние памяти), источник знаний - данные, агенда - совокупность конфликтующих в текущей ситуации правил, правила- все активные правила;
- вторая архитектура: правила- модули, источник знаний- классная доска, агенда- список конфликтующих правил, политические правила- правила для выполнения.

В системах с архитектурой второго состояния рабочей области памяти сравнивается:

1. с данными, представленными на классной доске(в базе данных или информационной базе);
2. с сопоставленными образцами, представленными также в базе данных.

В архитектуре второго состояния кроме правил, которые определяют принятие решений, используются общие правила (метоправила), которые позволяют увеличить эффективность функционирования системы в десятки раз. Если в первой архитектуре сформулированные правила могут быть прочитаны экспертом, даже не являющимся программистом, то во второй архитектуре метоправила могут быть прочитаны и использованы только программистом. **Метоправила** - это правила работы с правилами. Выбранный механизм обработки и структура метоправил являются наиболее общими и позволяют охватить при обработке больший объем данных, а последовательное применение частных правил в первой архитектуре может

привести к росту базы знаний и из-за путаницы в частных правилах увеличивается база знаний, что приведет к неадекватным решениям.

В традиционных языках программирования при вызове модуля или процедуры используется имя процедуры. При разработке экспертных систем выбор модуля (образца или правила) осуществляется на основе текущих правил.

Приведем классификацию стратегий принятия решений:

• *Локальная и глобальная классификации.*

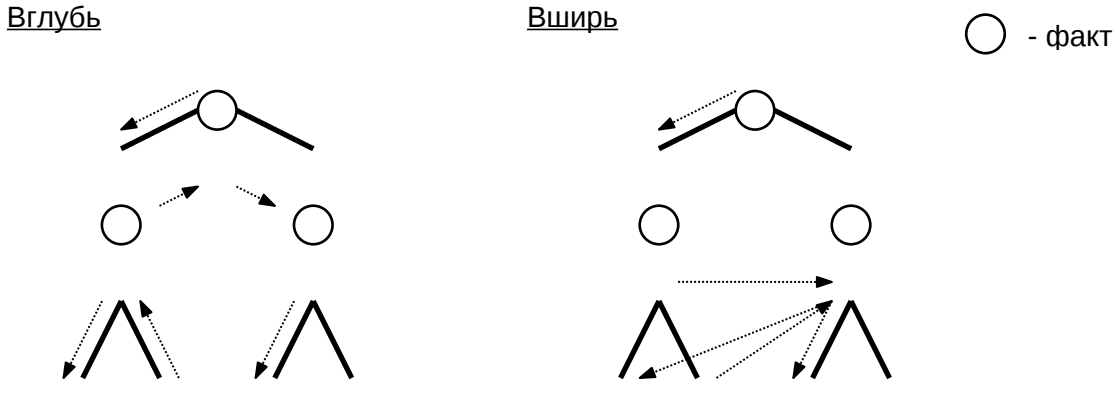
Локальная стратегия используется для частных правил, глобальная стратегия- для общих правил;

• *Скрытая и открыта.;*

Открытая стратегия позволяет вмешиваться в процесс поиска принятия решения, стратегия закрытая является жестко заданной для предметной области;

1. *По форме принятия решения:*

стратегия поиска принятия решения вглубь и поиска принятия решения вширь:



Методы поиска, реализованные в экспертных системах

Методы поиска различаются:

1. по определению предметной области;
2. по представлению результатов.

Методы по определению предметной области классифицируются следующим образом:

- по размерности пространства;
- по количеству пространств и определению места и времени;
- по модели, описывающих предметную область;
- по совокупности моделей;
- по определению неопределенности, то есть точности задания данных, размытости представления информации и так далее.

Методы по представлению результатов классифицируются следующим образом:

- количество представленных результатов(один, несколько, все);
- полнота представления информации и результат.

приоритет2030^
лидерами становятся