Методы представления знаний

Кудрявцев К.Я.
к.т.н., доцент

Данные и знания

Данные - **это отдельные факты, характеризующие объекты, явления, процессы предметной области, а также их свойства**.

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

Знания - **это хорошо структурированные данные, или данные о данных, или метаданные**.

Особенности знаний

1. Внутренняя интерпретируемость.

Каждая информационная единица должна иметь уникальное имя, по которому система находит ее, а также отвечает на запросы, в которых это имя упомянуто.

2. Структурированность.

Информационные единицы должны обладать гибкой структурой. Для них должен выполняться «ПРИНЦИП МАТРЕШКИ», то есть, рекурсивная вложенность одних информационных единиц в другие.

3. Связность.

В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа.

Связи могут характеризовать ОТНОШЕНИЯ между информационными единицами.

Семантика отношений может носить ДЕКЛАРАТИВНЫЙ (описательный) или ПРОЦЕДУРНЫЙ характер.

Две и более информационных единицы могут быть связаны ОТНОШЕНИЕМ:

- **«ОДНОВРЕМЕННО» - временная связь (декларативное знание);**
- **«ПРИЧИНА-СЛЕДСТВИЕ» - причинно-следственная (каузальная) связь (декларативное знание);**
- **«БЫТЬ РЯДОМ» - пространственная связь (декларативное знание);**
- **«АРГУМЕНТ-ФУНКЦИЯ»**

Особенности знаний

4. Семантическая метрика

Характеризует ситуационную близость информационных единиц, то есть, силу ассоциативной связи (отношение релевантности) между информационными единицами. Отношение релевантности позволяет находить знания, близкие к уже найденным.

5. Активность

Выполнение программ в Интеллектуальных системах должно инициироваться ТЕКУЩИМ СОСТОЯНИЕМ информационной базы. Появление в базе новых фактов или описаний событий, установление связей могут стать источником активизации системы.

Особенности 1÷5 информационных единиц определяет ту грань,

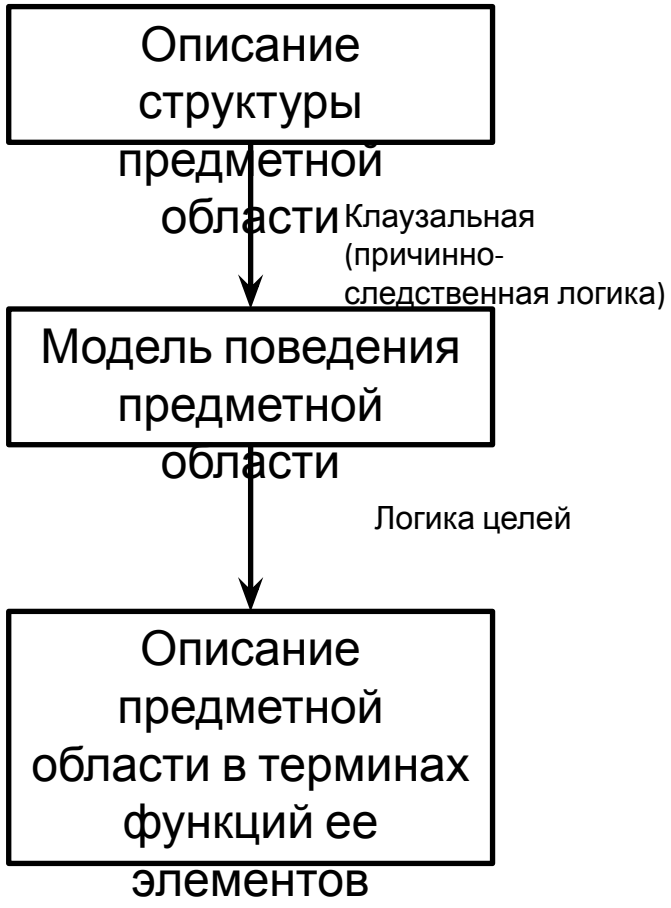
за которой данные превращаются в знания,

а БАЗА ДАННЫХ перерастает в БАЗУ ЗНАНИЙ.

Классификация знаний

1. Поверхностные и глубинные знания

Поверхностные знания



Глубинные знания

Классификация знаний

2. Знания как элементы семиотической системы

Знания представляются некоторой знаковой (семиотической) системой.

В любой семиотической системе выделяют три аспекта:

Три типа знания:

- **СИНТАКСИЧЕСКИЕ - характеризуют синтаксическую структуру описываемого объекта или явления, не зависящую от смысла и содержания используемых при этом понятий;**
- **СЕМАНТИЧЕСКИЕ - содержат информацию, непосредственно связанную со значениями и смыслом описываемых явлений и объектов;**
- **ПРАГМАТИЧЕСКИЕ - описывают объекты и явления с точки зрения решаемой задачи.**

3. Процедурные и декларативные знания

МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

1. Эвристические модели (интуиция, опыт и мастерство разработчика)

- фреймовые модели представления знаний
- семантические сети

2. Логические модели (исчисление предикатов)

- Логические (Prolog)
- Продукционные (CLIPS)

Классы задач, решаемых ЭС

- диагностика;
- прогнозирование;
- идентификация;
- управление;
- проектирование;
- мониторинг

Области деятельности ЭС

- медицина;
- вычислительная техника;
- военное дело;
- радиоэлектроника;
- юриспруденция;
- экономика;
- экология;
- геология;
- математика.

ПРИМЕРЫ ЭКСПЕРТНЫХ СИСТЕМ

1. MYCIN – ЭС диагностики кишечных заболеваний;
2. PUFF – ЭС диагностики легочных заболеваний;
3. МОДИС – ЭС диагностики различных форм гипертонии;
4. DENDRAL – ЭС для распознавания структуры сложных органических молекул по результатам их спектрального анализа;
5. MOLGEN – ЭС для выработке гипотез о структуре ДНК на основе экспериментов с ферментами;
6. PROSPECTOR – ЭС для консультаций при поиске залежей полезных ископаемых;
7. AIRPLANE – экспертная система для помощи летчику при посадке на авианосец;
8. МИДАС – ЭС для идентификации и устранения аварийных ситуаций в энергосистемах;

ЭС в робототехнике 1

Действия робота в непредсказуемых условиях:

- оценка обстановки (самостоятельная обработка информации о внешней среде, без участия человека),
- принятие (на основании этой оценки) решений,
- управление исполнительным механизмом.

Связь с внешней средой является необходимым условием эффективной работы манипуляционного робота. В тех случаях, когда в контуре управления присутствует человек, именно он осуществляет эту связь. (Однако в ряде случаев присутствие человека нежелательно, а часто
невозможно).

Действия в экстремальных условиях, когда оператор не в состоянии осуществлять непосредственное управление роботом - в силу сложности манипуляционной задачи или дефицита времени.

ЭС в робототехнике 2

Разработка проблемно ориентированных экспертных систем (ЭС) для интеллектуализации отдельных уровней управления роботом (приводного, тактического и стратегического), а также подсистемы его очувствления.

Инициация тех программно-реализованных алгоритмов управления или распознавания, выбор которых в той или иной ситуации представляется наиболее оправданным.

- Определение объема выполняемых функций,
- синтез архитектуры построения,
- формирование базы алгоритмов и соответствующей базы знаний

ЭС в робототехнике 3

Практическое воплощение проекта по созданию экспертного регулятора для системы управления электрическими приводами роботов требует, в частности, решения следующих задач:

- разработки алгоритмического и программного обеспечения;
- создание базы знаний системы управления электрическими приводами робота;
- Разработки алгоритмического и программного обеспечения для моделирования процессов в системе управления электрическими приводами роботов.

Фреймовые модели представления знаний

Теория фреймов (автор Минский):

«Когда человек сталкивается с новой ситуацией (или существенно меняет точку зрения на прежнюю задачу), он извлекает из памяти определенную структуру, называемую фреймом. Эту хранящуюся в памяти систему следует при необходимости привести в соответствие с реальностью путем изменения ее деталей».

Фрейм - это структура данных, предназначенная для представления знаний о стереотипной ситуации, причем детали фрейма с изменением текущей ситуации могут меняться.

Структура фрейма

Фреймовая модель состоит из двух частей:

- **набора фреймов, составляющих библиотеку внутри представляемых знаний,**
- **механизмов преобразования фреймов, их связывания и т.д.**

Общая организация:

имя_фрейма: имя_слота1, значение_слота1
имя_слота2, значение_слота2
…….
имя_слотак, значение_слотак

слоты - это некоторые незаполненные подструктуры фрейма, заполнение которых приводит к тому, что данный фрейм ставится в соответствие некоторой ситуации, явлению или объекту.

Структура фрейма

Незаполненный фрейм (оболочка, образец, прототип фрейма), в котором отсутствуют заполнители слотов, называется протофреймом.

Наличие такой оболочки позволяет осуществить процедуру внутренней интерпретации, благодаря которой данные в памяти системы не безлики, а имеют вполне определенный, известный системе смысл (обладают семантикой).

Протофрейм представляет интенсиональное описание.

Заполненный фрейм (экземпляр, пример прототипа) называется экзофреймом. экзофрейм соответствует экстенсиональному

Структура фрейма

Имя фрейма

Имя_слота	Указатель наследования	Указатель атрибутов данных	Значения слота	Присоединенная процедура
Слот 1				
Слот 2				
…				
Слот к				

Свойства фрейма

1. Каждый фрейм должен иметь уникальное имя в данной фреймовой системе.

2. Фрейм состоит из произвольного количества слотов.

Некоторые из них обычно определяются самой системой для выполнения специфических функций, а остальные - самим пользователем.

Фреймы могут представлять иерархические структуры, то есть реализовывать принцип наследования. Реализация механизма наследования основана на использовании системных слотов. Так, в число системных слотов входит слот, указывающий на фрейм-родитель и слот-указатель на дочерние фреймы.

3. Указатель наследования.

Эти указатели касаются только фреймовых систем иерархического типа. Они показывают, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты-потомки.

Типичными указателями наследования являются:

- **U (Unique) - уникальный. Фреймы-потомки должны иметь различные уникальные значения этого слота.**
- **S (Same) - такой же. Значение слота у всех потомков должно быть равным значению соответствующего слота фрейма-прародителя.**
- **R (Range) - интервал. Значение слота лежит в некоторых границах.**
- **O (Override) - игнорировать. Одновременное выполнение функций указателей U и S. При отсутствии значения слота у фрейма-потомка этим значением становится значение слота фрейма верхнего уровня (S), но допустимо и указание нового значения слота у фрейма- потомка (U).**

Свойства фрейма

4. Указатель атрибутов (типов) данных.

К возможным типам данных относятся:

a) Литеральные константы - INTEGER, REAL, BOOL, CHAR, …

b) TEXT, LIST (список), TABLE, EXPRESSION, …

c) LISP (присоединенная процедура),

d) FRAME (фрейм)

и др.

5. Значение слота.

a) Тип значения должен совпадать с типом указателя атрибута данного.

b) В качестве значений могут выступать выражения, содержащие обращения к функциям, имена таблиц, списков, других фреймов.

6. Присоединенная процедура.

Выделяют два типа присоединенных процедур - процедуры-слуги и процедуры-демоны. Процедуры-слуги активизируются только при выполнении условий, определенных при создании фрейма.

Процедуры-демоны активизируются при каждой попытке обращения к слоту. Среди разновидностей демонов можно отметить следующие:

- **«ЕСЛИ-НУЖНО» - активизируется, если в момент обращения к слоту его значение не было задано.**
- **«ЕСЛИ-ДОБАВЛЕНО» - запускается при занесении в слот значения.**
- **«ЕСЛИ-УДАЛЕНО» - запускается при стирании значения слота.**

Достоинства фреймовых моделей

1. Представление знаний, основанное на фреймах, дает возможность хранить родовую иерархию понятий в Базе знаний в явной форме.
2. Принцип наследования позволяет экономно расходовать память, проводить анализ ситуации при отсутствии ряда деталей.
3. Фреймовая модель является достаточно универсальной, поскольку позволяет отобразить все многообразие знаний о реальном мире через:

- фреймы-структуры, использующиеся для обозначения объектов и понятий (залог, вексель);
- фреймы-роли (клиент, менеджер);
- фреймы-сценарии (банкротство, собрание);
- фреймы-ситуации (авария, рабочий режим устройства);

и др.

4. С помощью присоединенных процедур фреймовая система позволяет реализовать любой механизм управления выводом.

Недостатки фреймовых моделей

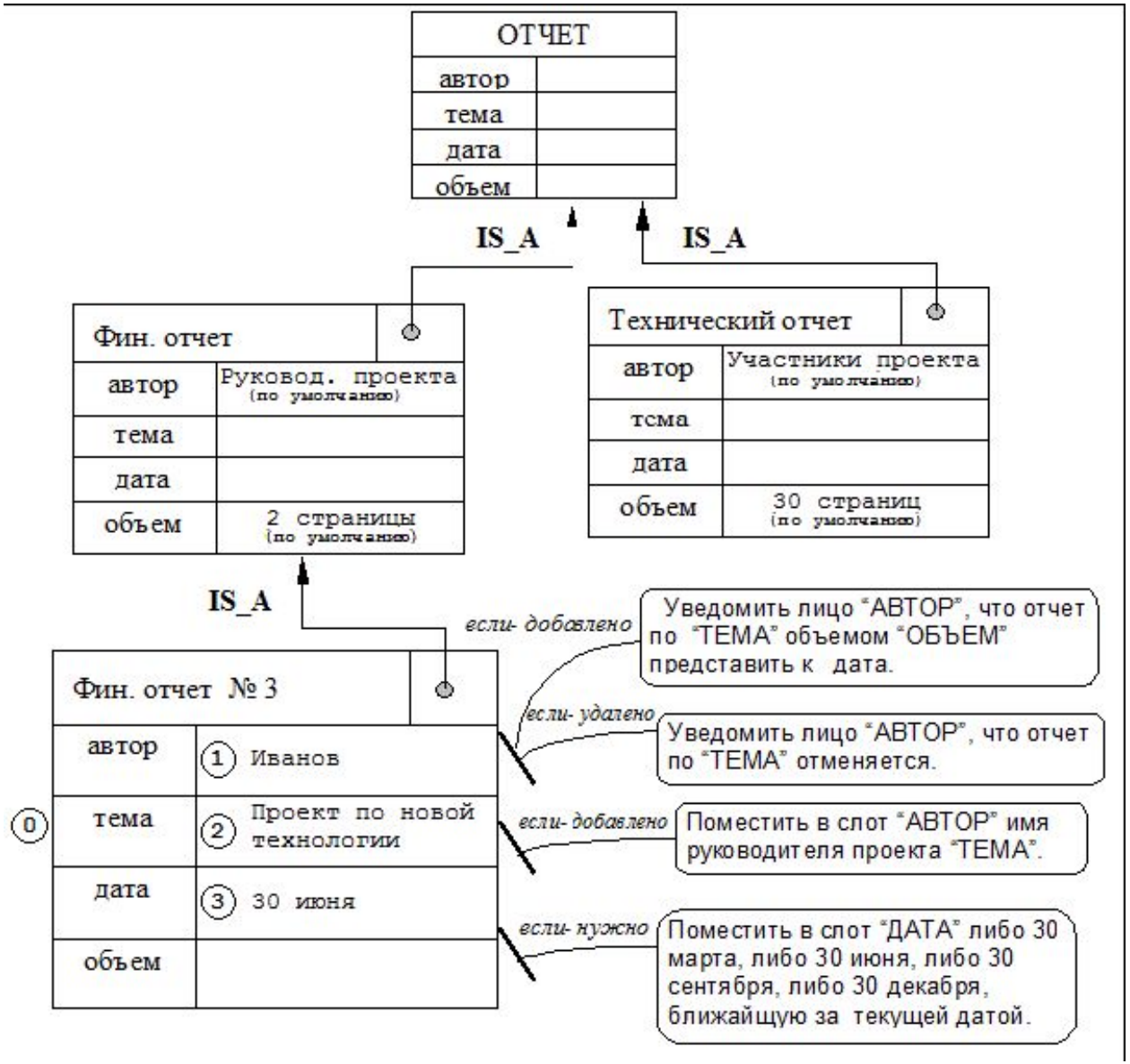
1. Относительно высокая сложность фреймовых систем, что проявляется в снижении скорости работы механизма вывода и в увеличении трудоемкости внесения изменений в родовую иерархию.

2. Во фреймовых системах затруднена обработка исключений. Наиболее ярко достоинства фреймовых систем представления знаний проявляется в том случае, если родовидовые связи изменяются нечасто и предметная область насчитывает немного исключений.

3. Разрозненные части информации, объединенные во фрейм, не могут быть выстроены в последовательность высказываний, иначе говоря, языки описания знаний во фреймовой модели не являются языками, родственными естественным, а ближе к изобразительным средствам.

4. Отсутствует специальный механизм управления выводом, поэтому он реализуется с помощью присоединенных процедур.

Пример реализации фреймовой модели



Пример реализации фреймовой модели

Запрос: «Нужен финансовый отчет о выполнении **проекта по новой технологии**».

Анализируя структуру модели и на основе фрейма–образца «Фин. отчет» добавим в свою структуру новый пустой фрейм-экземпляр «Фин. отчет №_» (узел №3), а после его создания в слот «ТЕМА» этого фрейм-экземпляра, на основании исходного запроса, внесем текст «**Проект по новой технологии**».

Далее срабатывают **присоединенные процедуры**:

1. **Процедура «ЕСЛИ–ДОБАВЛЕНО»**, связанная со слотом «**ТЕМА**», осуществляет поиск по своей базе данных руководителя этого проекта (например это Иванов). Процедура вписывает его фамилию в слот «АВТОР» финансового отчета №3. Если руководитель этой темы не будет найден, в слот «АВТОР» будет наследовано значение класса, а именно текст «РУКОВОДИТЕЛЬ ПРОЕКТА».

2. **Процедура «ЕСЛИ–ДОБАВЛЕНО»,** связанная со слотом «**АВТОР**», начинает выполняться, т.к. в слот только что было вписано новое значение. Эта процедура начинает составлять сообщение, чтобы отправить его Иванову, но тут же обнаруживает, что нет нужной даты исполнения.

3. **Процедура «ЕСЛИ–ДОБАВЛЕНО»**, просматривая слот «**ДАТА**» и найдя его пустым, активирует процедуру «если–нужно», связанную с этим слотом. Эта процедура, анализируя текущую дату, например 09.02.17, решит, что «30 июня» ближайшее к ней окончание финансового квартала и впишет эту дату в слот «ДАТА».

4. **Процедура «ЕСЛИ–ДОБАВЛЕНО»,** связанная со слотом «**АВТОР**», найдет, что еще одно значение, которое нужно включить в сообщение, т.е. объем отчета, отсутствует. Слот «ОБЪЕМ» не связан с процедурами и ничем помочь не может. Однако выше узла № 3 существует узел общей концепции финансового отчета, содержащий значение объема. Процедура, используя концепцию наследования свойств класса, использует значение объема и составляет следующее сообщение: «Господин Иванов, подготовьте финансовый отчет по проекту новой технологии к 30 июня объемом 2 страницы».

Пример реализации фреймовой модели

FRL (Frame Representation Language)

```
(frame СТОЛ
        (purpose (value(размещение предметов для деятельности рук)))
        (type (value(письменный)))
        (colour (value(коричневый)))
)
```

KRL (Knowledge Representation Language),
Фреймовая оболочка Kappa,

PILOT/2

```
[ Person is_a prototype;
        Name string, if_changed ask_why();
        Age int, restr_by >=0;
        Sex string, restr_by (==”male” || ==”female”), by_default “male”;
        Children {frame} ];
[ John is_a Person; if_deleted bury();
        Name = ”Johnson”;
        Age = 32;
        Children = {Ann, Tom} ];
[ Mary is_a Person;
        without Age;
        Name = ”Smirnova”;
        Sex = ”female”;
        Children = empty ];
```

Семантическая сеть

СЕМАНТИЧЕСКАЯ СЕТЬ - множество вершин, каждая из которых соответствует определенному **ПОНЯТИЮ, ФАКТУ, ЯВЛЕНИЮ ИЛИ ПРОЦЕССУ**; а между вершинами заданы различные отношения, представляемые дугами.

Вершины помечены именами и описателями, содержащими нужную для понимания семантики вершины информацию.

Дуги также снабжены именами и описателями, задающими семантику отношений.

S = <I, G1, G2,…, GN, R>

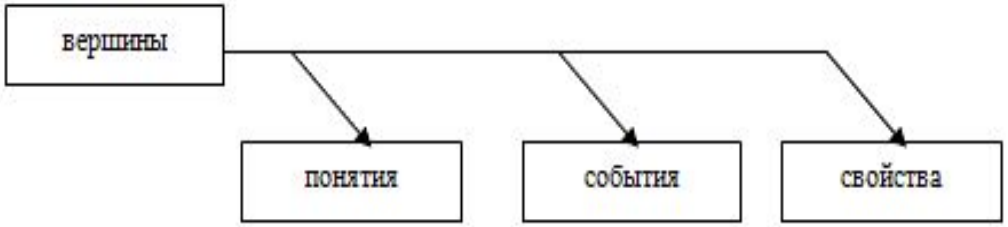
I - множество информационных единиц, представленных вершинами сети;

G1, G2,…,GN - заданный набор типов отношений между информационными единицами;

R - отображение, задающее между информационными единицами, входящими в **I**, связи из заданного набора типов связей.

Элементы семантической сети

Вершины семантической сети



Понятия — представляют собой сведения об абстрактных или физических объектах предметной области или реального мира.

События — представляют собой действия происходящие в реальном мире и определяются:

- указанием типа действия;
- указанием ролей, которые играют объекты в этом действии.

Свойства — используются для уточнения понятий и событий. Применительно к понятиям они описывают их особенности и характеристики (цвет, размер, качество), а применительно к событиям — продолжительность, время, место.

Типы объектов

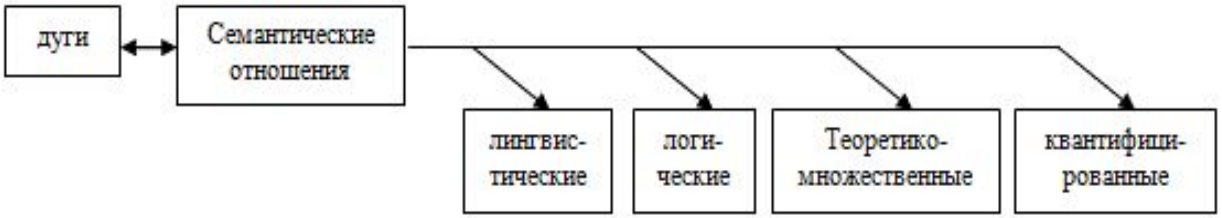
Обобщенный объект - некоторое известное и широко используемое в ПО понятие, представляющее определенным образом некоторый **тип объектов.**

Конкретный (индивидуальный) объект – выделенная одиночная сущность (экземпляр объектов некоторого типа).

Агрегатный (составной)объект - составлен тем или иным способом из других объектов, являющихся его частями.

Элементы семантической сети

Дуги графа семантической сети — отображают многообразие семантических отношений



Лингвистические отношения — отображают смысловую взаимосвязь между событиями, между событиями и понятиями или свойствами. Лингвистические отношения бывают:

- глагольные (время, вид, род, залог, наклонение);
- атрибутивные (цвет, размер, форма);
- падежными (см. ниже).

Логические отношения — это операции, используемые в исчислении высказываний (алгебре логики): дизъюнкция, конъюнкция, инверсия, импликация.

Теоретико-множественные — это отношение подмножеств, отношение части и целого, отношение множества и элемента. Примерами таких отношений являются "IS-A" и "PART-OF".

Квантифицированные отношения — это отношения, которые используют логические кванторы общности и существования.

Классификация отношений

По количеству типов отношений выделяют:

- **однородные сети** (с единственным типом отношений),
- **неоднородные сети** (с различными типами отношений).

Выделяют следующие **типы отношений:**

- **бинарные** (отношения связывают два объекта);
- **N-арные** (отношение связывает более двух обектов).

Виды отношений:

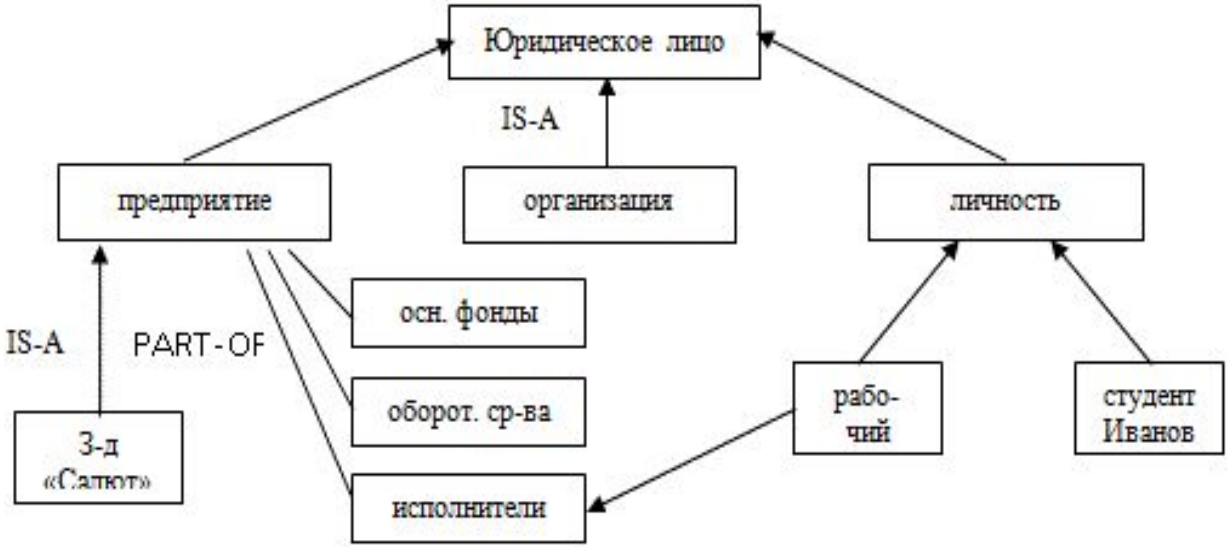
- **иерархические** («РОД-ВИД», «ЭЛЕМЕНТ-МНОЖЕСТВО», «ЧАСТЬ-ЦЕЛОЕ» и т.п.);
- **функциональные** («АРГУМЕНТ-ФУНКЦИЯ», а также связи, определяемые глаголами «влияет», «производит» и др.);
- **количественные** («больше», «меньше», и т.д.);
- **пространственные** («далеко от», «близко от», «над», «под» и т.д.);
- **временные** («раньше», «позже», «в течение»);
- **атрибутивные** («иметь свойство», «иметь значение»);
- **клаузальные** (причинно-следственные);
- **логические** («И», «ИЛИ», «НЕ»);
- **лингвистические**.

Представление структуры понятий семантической сетью

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

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

Связи понятий образуют структуру, в общем случае сетевую, в которой используется как минимум два типа связей: **"IS - A"** и **"PART – OF"**.



Представление событий семантической сетью

При представлении **событий** предварительно выделяются **простые отношения**, которые характеризуют основные компоненты события. В первую очередь из события выделяется **действие**, которые обычно описываются **глаголом**.

Далее определяются:

- **объекты, которые действуют**;
- **объекты, над которыми эти действия выполняются**.

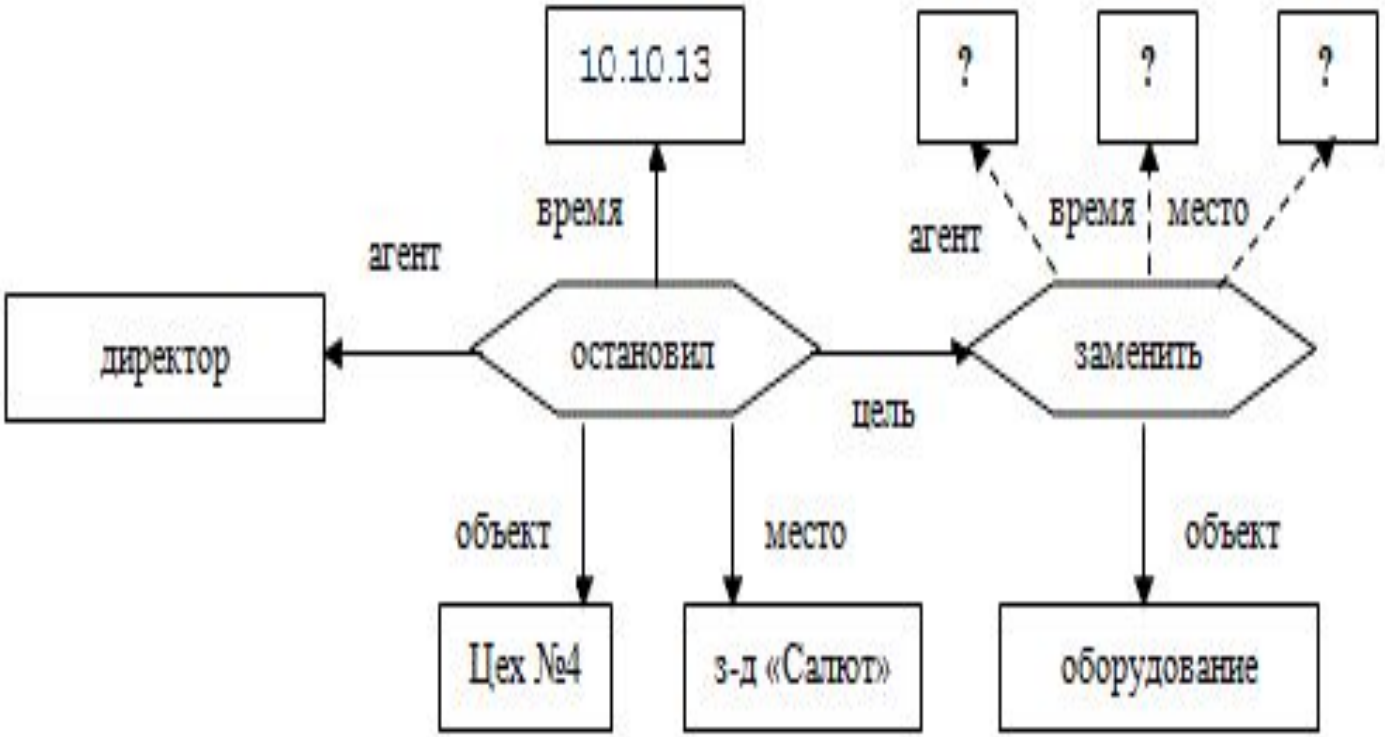
Все связи понятий, событий и свойств с действием (глаголом) называют **падежами** или **падежными отношениями**, которые относятся к классу лингвистических отношений.

Основные лингвистические отношения (падежи).

Падежс:	**Лингвистическое (падежное) отношение**, определяющее **связь действия**
Агент	- предметом, являющимся **инициатором** действия
Объект	- предметом, подвергающимся действию
Источник	- размещение предмета **перед** действием
Приемник	- размещение предмета **после** действия
Время	- **моментом выполнения** действия
Место	- **местом проведения** действия
Цель	- **действием другого события**

Семантическая структура знания о событии

Директор завода «Салют» остановил 10.10.13 цех №4, чтобы заменить оборудование



Представление знаний семантической сетью

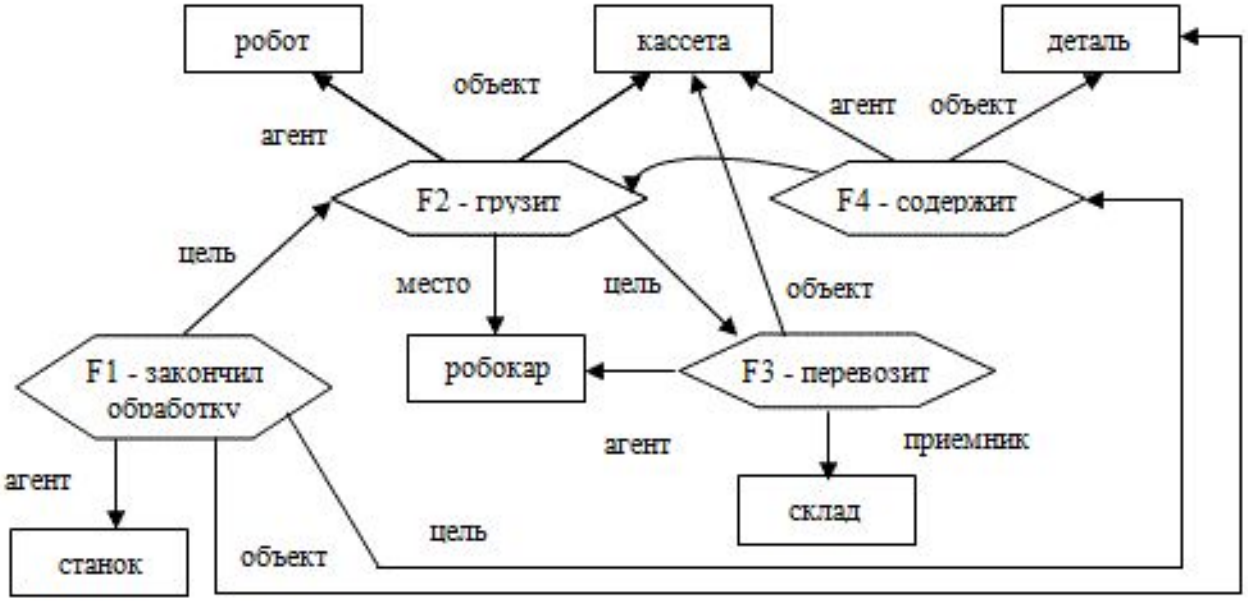
Если станок закончил обработку, робот грузит кассету с деталями на робокар, который перевозит их на склад.

Понятия: "Станок", "Деталь", "Кассета", "Робот", "Робокар" и "Склад".

События: "Закончил", "Грузит", "Перевозит".

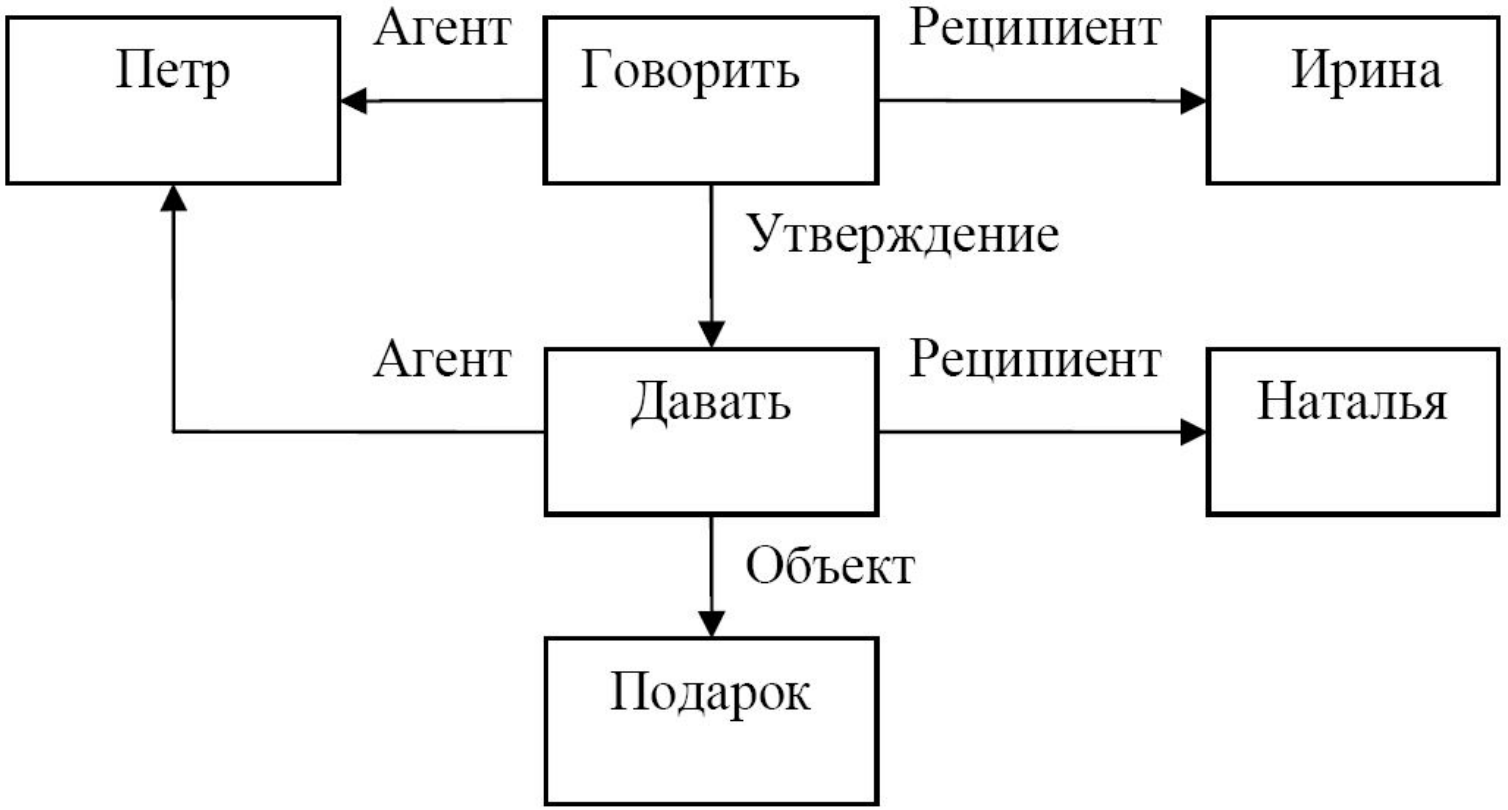
Описание элементарных действий:

- **F1** - станок закончил обработку (детали)
- **F2** - робот грузит (кассету на робокар)
- **F3** - робокар перевозит (кассету на склад)
- **F4** - кассета содержит (детали)



Грамматический разбор фраз естественного языка

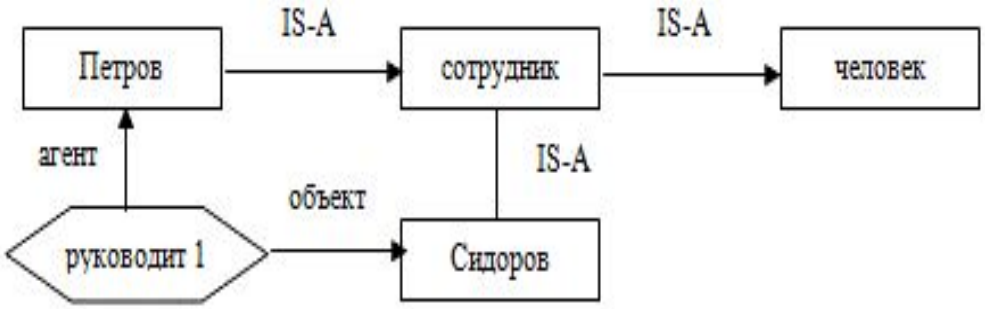
Петр сказал Ирине, что он отдал Наталье подарок



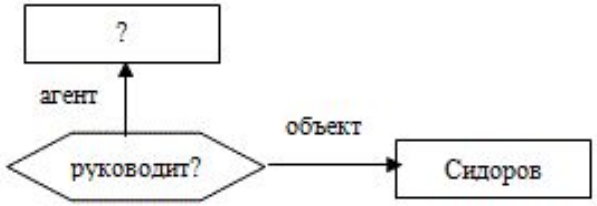
Получение вывода с помощью семантической сети

При формировании запроса к базе знаний

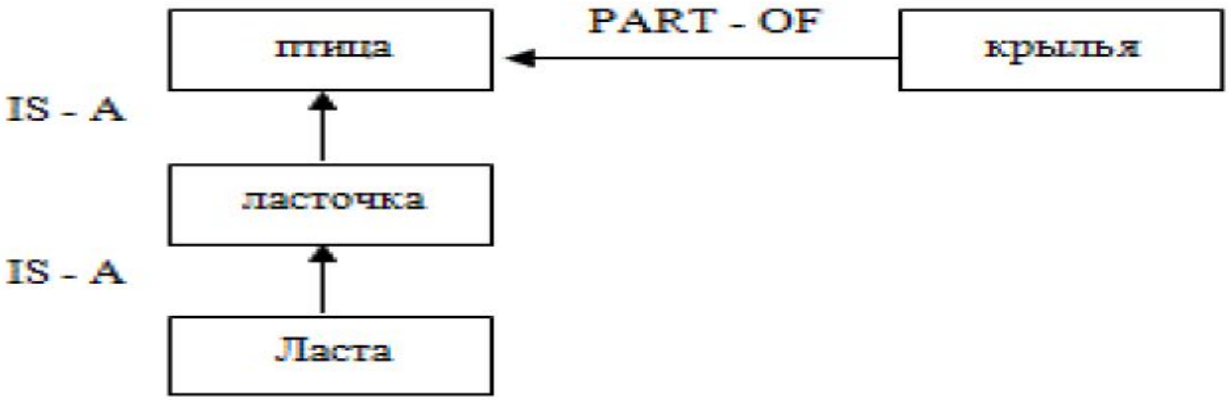
- **строится семантическая сеть, отражающая структуру запроса;**
- **вывод обеспечивается за счет сопоставления общей сети БЗ и сети для запроса.**



Запрос: «Кто руководит Сидоровым?»



Семантическая сеть как Prolog программа




```
Files   Edit   Run   Compile   Options   Setup
Editor
 Line 11   Col 43   C:\LASTA.PRO   Indent   Insert
predicates
  is_a(string,string)
  part_of(string,string)
  to_be(string,string)
clauses
  is_a("ласточка","птица").
  is_a("Ласта","ласточка").
  to_be(X,Y) :- is_a(X,Y).
  to_be(X,Y) :- is_a(Z,Y),to_be(X,Z).
  part_of("крылья","птица").
  part_of(X,Y) :- to_be(Y,Z),part_of(X,Z).
```

```
Dialog
Goal: to_be("Ласта",Q).
Q=ласточка
Q=птица
2 Solutions
Goal: part_of(Q,"птица")
Q=крылья
1 Solution

Goal: part_of(Q,"Ласта")
Q=крылья
Q=крылья
2 Solutions
```

Достоинства и недостатки семантических сетей

Достоинства:

1. Большие выразительные возможности, естественность и наглядность системы знаний, представленной графически.

2. Близость структуры сети, представляющей систему знаний, семантической структуре фраз естественного языка.

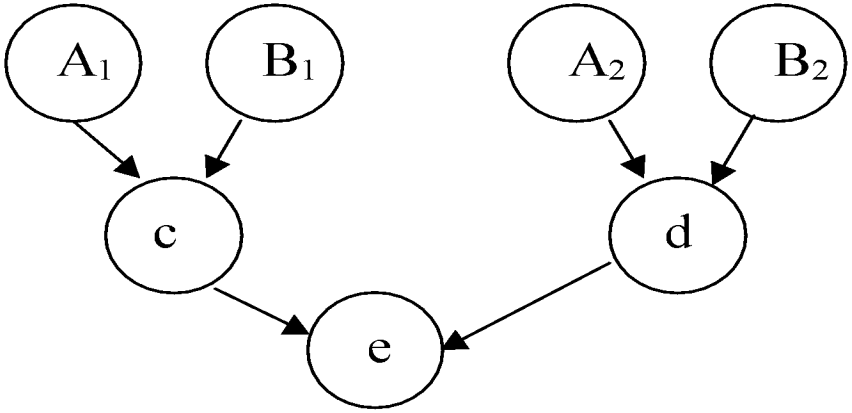
3. Данная модель более других соответствует современным представлениям об
организации долговременной памяти человека.

Недостатки:

1. Громоздкость и неэффективность представления знаний только аппаратом
семантической сети.

2. Сложность организации процедуры поиска нужного знания (как фрагмента сети).

Байесовские сети доверия



$$p(c_i) = \sum_m \sum_n p(c_i | A_{1m}, B_{1n}) \times p(A_{1m}) \times p(B_{1n})$$

$$p(d_j) = \sum_m \sum_n p(d_j | A_{2m}, B_{2n}) \times p(A_{2m}) \times p(B_{2n})$$

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 (ТУВ) или функцией условных вероятностей (ФУВ);
- для вершин без “родителей” вероятности её состояний являются безусловными (маргинальными).

В байесовских сетях доверия вершины представляют собой случайные переменные, а дуги – вероятностные зависимости, которые определяются через таблицы условных вероятностей. Таблица условных вероятностей каждой вершины содержит вероятности состояний этой вершины при условии состояний её “родителей”

Логические модели

Логическая (формальная) модель представления знаний - **совокупность фактов и правил (утверждений)**.

Факты (правила) представляются как **формулы в некоторой логической системе.**

База Знаний - совокупность формул.

Формулы неделимы и при модификации БЗ могут лишь добавляться и удаляться.

Для представления знаний в математической логике пользуются **исчислением предикатов, которое имеет** ясную формальную семантику и для него разработаны механизмы вывода (**метод резолюций**).

Исчисление предикатов

n-местным предикатом $P(x_1, x_2, ..., x_n)$ называется функция P: $\boldsymbol{M}^n \to \boldsymbol{B}$, где $\boldsymbol{M}$ — произвольное множество, а $\boldsymbol{B} = \{1, 0\}$.

$\boldsymbol{M}$ называется предметной областью, x_i — предметными переменными.

Кванторы

Пусть $P(x_1, x_2, ..., x_n)$ — предикат, определенный на $\boldsymbol{M}^n$. Предметные переменные $x_1, x_2, ..., x_n$ называют *свободными*.

Выражение

$$\forall x_i P(x_1, x_2, ..., x_n)$$

означает: "Для всех $x_i \in \boldsymbol{M}$ предикат $P(x_1, x_2, ..., x_n)$ принимает значение истина".

Выражение

$$\exists x_i P(x_1, x_2, ..., x_n)$$

означает: "Существует $x_i \in \boldsymbol{M}$ такое, что $P(x_1, x_2, ..., x_n)$ принимает значение истина".

Теоремы исчисления предикатов

Формальная теория в которой всякая доказуемая формула тождественно истинна, а любая общезначимая формула доказуема – называется полной

Теорема (Геделя) о полноте исчисления предикатов

Исчисление предикатов первого порядка — полная теория.

Формальная теория называется *разрешимой*, если существует алгоритм, который для любой формулы определяет, является ли она теоремой (т. е. может ли быть построен вывод этой формулы в данной теории) или нет.

Теорема (Черча)

Исчисление предикатов неразрешимо.

Несмотря на это, есть случаи, когда разрешающий алгоритм построить удается. Это имеет место для исчисления, содержащего только одноместные предикаты, и для формул, которые не содержат кванторов.

Исчисление предикатов. Примеры

Основное тригонометрическое тождество: $\sin^2(x) + \cos^2(x) = 1$.

Эта формула является теоремой тригонометрии (основное тригонометрическое тождество).

$\forall x Равно(сумма(квадрат(sin(x)), квадрат(cos(x))), 1)$

Здесь x — переменная для обозначения некоторого произвольного числа из множества вещественных чисел, "1" — предметная константа.

Равно — имя предиката, *сумма*, *квадрат*, *sin*, *cos* — имена функций.

Большая (или великая) теорема Ферма утверждает, что для любого целого $n > 2$ не существует натуральных чисел x, y, z, удовлетворяющих равенству $x^n + y^n = z^n$. Если этому равенству поставить в соответствие предикат $P_F(x, y, z, n)$, истинный тогда и только тогда, когда оно выполняется, а через $N(x)$ обозначить предикат "x — натуральное число", то теорема Ферма формулируется так:

$$\forall x \forall y \forall z \forall n (N(x) \,\&\, N(y) \,\&\, N(z) \,\&\, N(n) \,\&\, (n > 2) \rightarrow \neg P_F(x, y, z, n))$$

Логический вывод в исчислении предикатов

Пусть есть множество формул исчисления предикатов $S = \{F_1, ..., F_n\}$. Если каждая из формул этого множества истинна при данной интерпретации, то говорят, что данная интерпретация удовлетворяет этому множеству или является *моделью* этого множества формул.

Так же как и в исчислении высказываний, формула F логически следует из некоторого множества формул $S = \{F_1, ..., F_n\}$, если при каждой интерпретации, когда все F_i истинны (т. е., когда истинна их конъюнкция), истинной является и F, что будем записывать или $\{F_1, ..., F_n\} \Rightarrow F$, или $S \Rightarrow F$. По-другому: каждая интерпретация, удовлетворяющая S, удовлетворяет также и F.

Если $S \Rightarrow F$, то любая интерпретация, удовлетворяющая S, удовлетворяет F или, по-другому, никакая интерпретация, удовлетворяющая S, не удовлетворяет $\neg F$ и, следовательно, множество формул $S \cup \{\neg F\}$ является *неудовлетворимым* или *противоречивым*.

То же самое может быть выражено следующим утверждением: формула $F_1 \& ... \& F_n \& \neg F \rightarrow$ `false` будет истинной в любой интерпретации или $F_1 \& ... \& F_n \& \neg F \Rightarrow$ `false`.

Преобразование формул. Предваренная нормальная форма

Предваренной формой называется формула вида $Q_1x_1Q_2x_2 \ldots Q_nx_nM$, где Q_i — один из двух возможных кванторов, x_i — переменные, входящие в формулу, M — формула, не содержащая кванторов. $Q_1x_1Q_2x_2 \ldots Q_nx_n$ — называют *префиксом*, M — *матрицей* формулы.

Для любой формулы исчисления предикатов существует логически эквивалентная ей предваренная форма.

1. Исключить связки эквивалентности и импликации:

   $F \equiv G \Leftrightarrow (F \to G) \,\&\, (G \to F)$

   $F \to G \Leftrightarrow \neg F \vee G$

2. Уменьшить область действия отрицания так, чтобы знак отрицания был применен не более чем к одной предикатной букве, для этого надо использовать следующие эквивалентные преобразования:

   - $\neg \forall x F \Leftrightarrow \exists x \neg F$
   - $\neg \exists x F \Leftrightarrow \forall x \neg F$
   - $\neg (F \,\&\, G) \Leftrightarrow \neg F \vee \neg G$
   - $\neg (F \vee G) \Leftrightarrow \neg F \,\&\, \neg G$
   - $\neg \neg F \Leftrightarrow F$

Предваренная нормальная форма

3. Переименовать, если это необходимо, связанные переменные так, чтобы в области действия каждого квантора каждая связанная им переменная имела бы индивидуальное имя, и имена свободных и связанных переменных не совпадали. Это можно сделать, т. к. имя связанной квантором переменной не влияет на значение формулы:

   $\forall x F(x, y) \,\&\, \exists x G(x, y) \vee P(x) \Leftrightarrow \forall x P(x, y) \,\&\, \exists z G(z, v) \vee P(w)$

4. Удалить кванторы, если область их действия не содержит переменной, связываемой квантором (квантифицированной переменной):

   $\forall x \forall z P(x, y, a) \Leftrightarrow \forall x P(x, y, a)$

5. Переместить кванторы в начало формулы:

   $\exists x F(x) \vee \exists x G(x) \Leftrightarrow \exists x (F(x) \vee G(x))$

   $\forall x F(x) \,\&\, \forall x G(x) \Leftrightarrow \forall x (F(x) \,\&\, G(x))$

6. Если R — формула, не содержащая свободных вхождений x, то будут иметь место следующие эквивалентности:

   $R \,\&\, \forall x F(x) \Leftrightarrow \forall x (R \,\&\, F(x))$

   $R \,\&\, \exists x F(x) \Leftrightarrow \exists x (R \,\&\, F(x))$

Предваренная нормальная форма. Примеры

Пример 1

$\forall x(P(x) \;\&\; \forall y \exists x(\neg Q(x, y) \to \forall z R(a, x, y)))$

1. $\forall x(P(x) \;\&\; \forall y \exists x(Q(x, y) \vee \forall z R(a, x, y)))$
2. $\forall x(P(x) \;\&\; \forall y \exists u(Q(u, y) \vee R(a, u, y)))$
3. $\forall x \forall y(P(x) \;\&\; \exists u(Q(u, y) \vee R(a, u, y)))$
4. $\forall x \forall y \exists u(P(x) \;\&\; (Q(u, y) \vee R(a, u, y)))$

Пример 2

$\forall x(P(x) \to (\forall y(P(y) \to P(f(x, y))) \;\&\; \neg \forall y(Q(x, y) \to P(y))))$

1. $\forall x(\neg P(x) \vee (\forall y(\neg P(y) \vee P(f(x, y))) \;\&\; \neg \forall y(\neg Q(x, y) \vee P(y))))$
2. $\forall x(\neg P(x) \vee (\forall y(\neg P(y) \vee P(f(x, y))) \;\&\; \exists y \neg (\neg Q(x, y) \vee P(y))))$
3. $\forall x(\neg P(x) \vee (\forall y(\neg P(y) \vee P(f(x, y))) \;\&\; \exists y(Q(x, y) \;\&\; \neg P(y))))$
4. $\forall x(\neg P(x) \vee (\forall y(\neg P(y) \vee P(f(x, y))) \;\&\; \exists z(Q(x, z) \;\&\; \neg P(z))))$
5. $\forall x \forall y \exists z(\neg P(x) \vee ((\neg P(y) \vee P(f(x, y))) \;\&\; (Q(x, z) \;\&\; \neg P(z))))$

Преобразование формул. Скулемовская форма

Рассмотрим следующую замкнутую предваренную форму:

$$F = \exists x \forall y \exists z \forall u \exists v \forall w F(x, y, z, u, v, w)$$

1. Обозначим значение x, которое существует в соответствии с первым квантором, предметной константой, например буквой a, отбросив при этом квантор существования:

$$F = \forall y \exists z \forall u \forall v \exists w F(a, y, z, u, v, w)$$

2. Тот факт, что по всякому y может быть найдено значение для z, может быть выражен некоторой функцией $z = f(y)$, называемой *скулемовской*.

$$F = \forall y \forall u \forall v \exists w F(a, y, f(y), u, v, w)$$

3. Тот факт, что для любых y, u и v найдется w, можно выразить скулемовской функцией $w = g(y, u, v)$, которая подставляется вместо w:

$$F = \forall y \forall u \forall v F(a, y, f(y), u, v, g(y, u, v))$$

Так как все переменные должны быть связанными, кванторы существования исключены, а порядок кванторов всеобщности не влияет на значение формулы, то кванторы всеобщности отбрасывают, предполагая, что все переменные ими связаны.

$$S_F = F(a, y, f(y), u, v, g(y, u, v))$$

Для первого примера:

$$F = \forall x \forall y \exists u (P(x) \ \& \ (Q(u, y) \vee R(a, u, y)))$$
$$S_F = P(x) \ \& \ (Q(f(x, y), y) \vee R(a, f(x, y), y))$$

$u = f(x, y)$ — скулемовская функция.

Преобразование формул. Клоузальная форма

Клаузальной формой называется скулемовская форма, матрица которой является конъюнктивной нормальной формой, т. е. конъюнкцией дизъюнкций.

Для первого примера:

$$S = \{P(x);\ Q(f(x, y), y) \vee R(a, f(x, y), y)\}$$

Для второго примера: приведение к конъюнктивной нормальной форме аналогично приведению в исчислении высказываний, используется формула:

$$A \vee (B\ \&\ C) \Leftrightarrow (A \vee B)\ \&\ (A \vee C)$$

1. $(\neg P(x) \vee \neg P(y) \vee P(f(x, y)))\ \&\ (\neg P(x) \vee (Q(x, g(x))\ \&\ \neg P(g(x))) \Leftrightarrow$
2. $(\neg P(x) \vee \neg P(y) \vee P(f(x, y)))\ \&\ (\neg P(x) \vee Q(x, g(x)))\ \&\ (\neg P(x) \vee \neg P(g(x)))$

Получена клаузальная форма, которую можно записать в виде множества *дизъюнктов* (*предложений* или *клаузов* — от англ. *clause*):

$$S = \{\neg P(x) \vee \neg P(y) \vee P(f(x, y));\ \neg P(x) \vee Q(x, g(x));\ \neg P(x) \vee \neg P(g(x))\}$$

Метод резолюций. Вывод

- $\{a \vee F, \bar{a} \vee G\} \Rightarrow F \vee G$
- $(a \vee F)\&(\bar{a} \vee G) \longrightarrow F \vee G$
- $\overline{(a \vee F)\&(\bar{a} \vee G)} \vee (F \vee G)$
- $\overline{(a \vee F)} \vee \overline{(\bar{a} \vee G)} \vee F \vee G$
- $\bar{a} \,\&\, \overline{F} \vee a \,\&\, \overline{G} \vee F \vee G$
- $(\bar{a} \,\&\, \overline{F}) \vee F \vee (a \,\&\, \overline{G}) \vee G$
- $(\bar{a} \vee F)\&(\overline{F} \vee F) \vee (a \vee G)\&(\overline{G} \vee G)$
- $\bar{a} \vee F \vee a \vee G = \boldsymbol{TRUE}$

Метод резолюций. Пример

$$\{(U \vee V), (W \vee \neg V)\} \Rightarrow (U \vee W) \text{ или } \{V, (\neg V \vee W)\} \Rightarrow W$$

Пример 1

Всякий, кто находится в здравом уме, может понимать математику. Ни один из сыновей Гегеля не может понимать математику. Сумасшедшие не допускаются к голосованию. Следовательно, никто из сыновей Гегеля не допускается к голосованию.

Введем предикаты:

- $Z(x)$ — x находится в здравом уме;
- $M(x)$ — x может понимать математику;
- $G(x)$ — x является сыном Гегеля;
- $N(x)$ — x не допускается к голосованию.
- первое предложение можно записать так: $\forall x(Z(x) \rightarrow M(x))$;
- второе: $\neg \exists x(G(x) \ \& \ M(x))$;
- третье: $\forall x(\neg Z(x) \rightarrow N(x))$;
- четвертое предложение, являющееся заключением из первых трех: $\neg \exists x(G(x) \ \& \ \neg N(x))$.

Метод резолюций. Пример

Опустим знак квантора, стоящий перед всей формулой, предполагая, что доказательство будет проведено для некоторого фиксированного x, но т. к. x — любое значение, то оно будет справедливо для всех случаев. Заменим импликацию на дизъюнкцию, "внесем" знак отрицания, стоящий перед квантором существования, заменив его на квантор всеобщности, и получим следующие предложения:

1. $\neg Z(x) \vee M(x)$
2. $\neg G(x) \vee \neg M(x)$
3. $Z(x) \vee N(x)$
4. $\neg G(x) \vee N(x)$

Докажем:

$$\{(\neg Z(x) \vee M(x)), (\neg G(x) \vee \neg M(x)), (Z(x) \vee N(x))\} \Rightarrow \neg G(x) \vee N(x)$$

Для доказательства методом резолюций возьмем отрицание заключения $\neg(\neg G(x) \vee N(x)) \Leftrightarrow G(x)\ \&\ \neg N(x)$, добавим его к множеству посылок и докажем противоречивость полученного множества формул:

$$\{(\neg Z(x) \vee M(x)), (\neg G(x) \vee \neg M(x)), (Z(x) \vee N(x)), G(x), \neg N(x)\} \Rightarrow \texttt{false}$$

Метод резолюций. Пример

Вывод:

1. $\{(\neg G(x) \vee \neg M(x)),\ G(x)\} \Rightarrow \neg M(x)$
2. $\{\neg M(x),\ (\neg Z(x) \vee M(x))\} \Rightarrow \neg Z(x)$
3. $\{\neg Z(x),\ (Z(x) \vee N(x))\} \Rightarrow N(x)\}$
4. $\{N(x),\ \neg N(x)\} \Rightarrow$ `false`

Другой вариант вывода:

1. $\{(\neg Z(x) \vee M(x)),\ (Z(x) \vee N(x))\} \Rightarrow M(x) \vee N(x)$
2. $\{(M(x) \vee N(x)),\ \neg N(x)\} \Rightarrow M(x)$
3. $\{M(x),\ (\neg G(x) \vee \neg M(x))\} \Rightarrow \neg G(x)$
4. $\{\neg G(x),\ G(x)\} \Rightarrow$ `false`

Таким образом, логическое рассуждение является верным.

Метод резолюций. Теория

Для того чтобы доказать, что формула F логически следует из некоторого множества формул $S_1 = \{F_1, ..., F_n\}$, можно доказать, что множество формул $S_2 = S_1 \cup \{\neg F\}$, является противоречивым (неудовлетворимым), т. е. не существует интерпретации, в которой бы оно удовлетворялось.

$$S_2 = S_1 \cup \{\neg F\} = \{F_1, ..., F_n, \neg F\}$$

Тогда множество S противоречиво тогда и только тогда, когда:

$$\neg S_2 \Leftrightarrow \neg (F_1 \,\&\, ... \,\&\, F_n \,\&\, \neg F) \Leftrightarrow \neg (F_1 \,\&\, ... \,\&\, F_n) \vee F \Leftrightarrow (F_1 \,\&\, ... \,\&\, F_n) \rightarrow F \Leftrightarrow S_1 \rightarrow F$$

является общезначимой (истинной в любой интерпретации) формулой или, иначе, что $S_1 \Rightarrow F$.

Таким образом, необходимо найти способ доказательства противоречивости множества предложений S_2.

Для того чтобы определить интерпретацию множества формул, необходимо задать область интерпретации. Проблема заключается в том, что таких областей может быть бесконечно много.

Универсум Эрбрана

В качестве такой области берут множество H_S, называемое *универсумом Эрбрана*, обладающее следующим важным свойством:

если на H_S множество S неудовлетворимо, то оно неудовлетворимо на любой другой области интерпретации.

Универсум Эрбрана строится (рекурсивно) следующим образом:

1. Множество всех предметных констант (константных букв), входящих в S, принадлежит H_S. Если ни одной предметной константы не содержится в S, то H_S содержит произвольную предметную константу, называемую любым именем.
2. Если термы t_i ($i = 1, ..., n$) принадлежат H_S и f_j — какой-либо функциональный *n-арный* символ, встречающийся в S, то $f_j(t_1, ..., t_n)$ также принадлежит H_S.

Примеры:

Пусть $S = \{P(a) \vee Q(x, b);\ \neg P(x) \vee Q(b, y);\ \neg R(x, a, c)\}$,

тогда $H_S = \{a, b, c\}$.

Пусть $S = \{P(x) \vee Q(x, y);\ P(f(x));\ \neg P(f(x)) \vee \neg R(g(y))\}$,

тогда $H_S = \{a, f(a), g(a), f(f(a)), f(g(a)), g(f(a)), g(g(a)), ...\}$.

Эрбрановская база

Эрбрановской базой для S называется множество всех константных частных случаев для всех атомарных формул, входящих в дизъюнкты из S и получаемых при подстановке элементов из H_S.

Примеры

$S = \{P(x) \vee Q(a, y); \neg P(a); \neg Q(x, y)\}$,

то $H_S = \{a\}$ база Эрбрана $E_S = \{P(a), Q(a, a)\}$.

$S = \{P(x, a) \vee Q(f(x)); P(a, y)\}$,

$H_S = \{a, f(a), f(f(a)), f(f(f(a))), \ldots\}$,

$E_S = \{P(a, a), P(a, f(a)), P(f(a), a), P(a, f(f(a))), P(f(f(a)), a), P(f(f(a)), f(a)), \ldots\}$.

Метод резолюций основывается на теореме Эрбрана, доказанной в 1930 году.

Теорема Эрбрана

Множество дизъюнктов S невыполнимо тогда и только тогда, когда имеется конечное невыполнимое множество S_c константных частных случаев этих дизъюнктов на H_S.

Другими словами, множество S невыполнимо, если можно найти такие подстановки констант вместо предметных переменных, при которых полученное множество дизъюнктов будет противоречивым.

Унификация

Процесс подстановок с целью получения подходящих резолюций дизъюнктов называется *унификацией*.

Пусть имеется атомарная формула (литерал) $P(x, y, z)$. Термы литерала могут быть предметными переменными, предметными константами или выражениями, содержащими функциональные буквы. Результат подстановки вместо переменных различных термов называется *частным случаем*, *подстановочным частным случаем*, или *примером* данного литерала. Для $P(x, f(y), b)$ это могут быть:

- $P(z, f(v), b)$;
- $P(x, f(a), b)$;
- $P(g(z), f(a), b)$;
- $P(c, f(a), b)$.

Первый вариант называется *алфавитным вариантом исходного литерала*, т. к. переменные в нем заменены на другие переменные. Последний (из четырех) вариант называется *константным частным случаем* или *атомом*, т. к. ни один из термов не содержит переменных.

Подстановку можно представить в виде множества упорядоченных пар:

$\theta = \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\}$

или:

$\theta = \{t_1 \mid v_1, t_2 \mid v_2, \ldots, t_n \mid v_n\}$

где (t_i, v_i) или $t_i \mid v_i$ означает, что повсюду переменная v_i заменяется на терм t_i.

Алгоритм унификации

Множество литералов $\{L_i\}$ $i = 1, ..., n$ называется *унифицируемым*, если существует подстановка θ такая, что $L_1\theta = L_2\theta = ... = L_n\theta$. Подстановка θ в этом случае называется *унификатором* для $\{L_i\}$.

L_0 — исходное множество литералов, $\theta_0 = \varepsilon$ —"единичная" (ничего не меняющая) подстановка.

На k-м шаге:

- если $D_k = \varnothing$ (т. е. нет различающихся литер), то конец алгоритма, θ_k, результирующая подстановка, — наиболее общий унификатор;
- если D_k не содержит переменных, то конец алгоритма, множество литералов не унифицируемо;
- если множество рассогласования содержит переменную v_k и терм t_k, и если v_k входит в t_k, то конец алгоритма, множество литералов не унифицируемо.

В противном случае переходим к следующему $(k + 1)$-му шагу.

На $(k + 1)$-м шаге алгоритма: $\theta = \{t_k \mid v_k\}$, $\theta_{k+1} = \theta\theta_k$ выполняем подстановку для каждого из литералов, входящих в L_k:

$$L_{k+1} = L_k\theta \text{ (или } L_{k+1} = L_0\,\theta_{k+1})$$

Пример унификации

Пример 1

$L_0 = \{P(x, f(y, g(a)), h(x)), P(x, f(y, x)), y\}$, $D_0 = \{g(a), x\}$.

1. $D_0 \neq \varnothing$, $x \in D_0$, $g(a) \in D_0$, $x \notin g(a)$, т. е. не входит в состав букв $g(a)$.
2. $\theta = \{g(a)|x\}$, $\theta_1 = \theta\theta_0 = \theta$;

   $L_1 = L_0\theta_1 = \{P(x, f(y, g(a)), h(x))\theta_1, P(x, f(y, x), y)\theta_1\} =$

   $= \{P(g(a), f(y, g(a)), h(g(a))), P(g(a), f(y, g(a)), y)\}$.
3. $L_1 = \{P(g(a), f(y, g(a)), h(g(a))), P(g(a), f(y, g(a)), y)\}$; $D_1 = \{h(g(a)), y\}$.
4. $D_1 \neq \varnothing$, $y \in D_1$, $y \notin h(g(a))$.
5. $\theta = \{h(g(a))|y\}$, $\theta_2 = \theta\theta_1 = \{g(a)|x, h(g(a))|y\}$;

   $L_2 = L_1\theta_2 = \{P(g(a), f(h(g(a)), g(a)), h(g(a))), P(g(a), f(h(g(a)), g(a)), h(g(a)))\}$;
   $D_2 = \varnothing$.
6. $D_2 = \varnothing$, $L = \{P(g(a), f(h(g(a)), g(a)), h(g(a)))\}$, $\theta_p = \{g(a)|x, h(g(a))|y\}$ — наиболее общий унификатор, конец алгоритма.

Исчисление метода резолюций

Пусть имеется такое множество предложений S и дизъюнкт F, для которого требуется доказать, что $S \Rightarrow F$. Возьмем предложения из S в качестве аксиом исчисления T_S.

Единственное правило вывода (правило *резолюции*) следующее. Пусть имеются предложения D_1 и D_2 и унифицируемые литералы L_1 и L_2 такие, что $L_1\theta = L_2\theta$, тогда:

$$\frac{D_1 \vee L_1,\ D_2 \vee \neg L_2}{D_1\theta \vee D_2\theta}$$

В основе этого правила лежат два правила вывода исчисления предикатов: правило подстановки и правило заключения *(modus ponens)*:

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

По правилу подстановки $L_1(x) \vdash L_1(t)$ и $L_2(x) \vdash L_2(t)$, причем унифицируемость L_1 и L_2 означает, что имеется подстановка θ, при которой $L_1\theta = L_2\theta = L$, тогда:

$(D_1 \vee L_1)\theta$, $(D_2 \vee \neg L_2)\theta$ то же самое, что $D_1\theta \vee L\theta$, $L\theta \to D_2\theta$, откуда по обобщению правила *modus ponens*:

$$\frac{F \vee E,\ F \to G}{G \vee E}$$

получим $D_1\theta \vee L\theta,\ L\theta \to D_2\theta \vdash D_1\theta \vee D_2\theta$.

Prolog –Programming Language

В Прологе используют метод резолюций для хорновских предложений (дизъюнктов), содержащих не более одного положительного (без отрицания) литерала:

$$F \vee \neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_n$$

что эквивалентно

$$F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n \rightarrow F$$

и записывается на Прологе в виде:

$$F :\!- F_1, F_2, \ldots, F_n$$

Предложения такого вида называют в Прологе *правилами*. Если литералов с отрицанием (отрицательных литералов) нет, то такие предложения называют *фактами*. Множество *правил* и *фактов* (конъюнкция соответствующих предложений) составляет программу, описание данной предметной области в виде *множества аксиом* прикладного исчисления предикатов.

Предложение, не содержащее положительных литералов,

$$\neg Q_1 \vee \neg Q_2 \vee \ldots \vee \neg Q_n$$

что эквивалентно

$$\neg (Q_1 \,\&\, Q_2 \,\&\, \ldots \,\&\, Q_n)$$

называется *целью* (*goal*) или запросом к программе и представляет собой отрицание *теоремы*, присоединяемой к множеству предложений (правил и фактов).

Линейная резолюция

Выполнение программы по методу *линейной резолюции* осуществляется следующим образом. Берется Q_1 и первый в программе дизъюнкт $F \vee \neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_m$ такой, что F и Q_1 унифицируемы: существует НОУ (Наиболее Общий Унификатор) θ такой, что $F\theta = Q_1\theta$, тогда по резолютивному правилу вывода

$$\frac{F \vee \neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_m,\ \neg Q_1}{(\neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_m)\theta}$$

получится новый запрос:

$$(\neg F_1 \vee \neg F_2 \vee \ldots \vee \neg F_m \vee \neg Q_2 \vee \ldots \vee \neg Q_n)\theta$$

В этом новом запросе правило линейной резолюции будет применяться к $(\neg F_1)\theta$ и т. д. до тех пор, пока не будет получен пустой запрос (пустой дизъюнкт).

Prolog –Programming Language

Терм (синтаксический объект):

- **Константа** ,
- **Переменная** предназначена для установления соответствия между **термами предикатов, действующих в пределах одной фразы** (предложения), **а не местом памяти** для хранения данных,
- **Функция** (составные термы или структуры), состоящие из имени функции и списка аргументов-термов, имена функций начинаются со строчной буквы.

Пролог и логика предикатов

В Прологе используется специальная **клаузальная** форма записи – запись в виде набора клауз Хорна.

Правило P:-Q,R,T.

-формула логики предикатов

Q&R&T→P,

равносильная формуле

¬ Q ∨ ¬ R ∨ ¬ T ∨ P

– дизъюнкту, в котором все слагаемые кроме одного отрицательны.

Подобные формулы называются **хорновскими**.

Любое пролог-предложение является хорновским диъюнктом,

Программирование на Прологе – программирование в хорновских дизъюнктах.

Пример программы на Прологе

Предложение - фраза Хорна (Хорновский дизъюнкт), в которой посылки связаны между собой логическим «И», а дизъюнкты- логическим «ИЛИ».

Простейшим типом предложения является факт.

Факт является простым предикатом, который записывается в виде функционального терма

Цель – это средство формулировки задачи, которую должна решать программа

мать(мария, анна).
мать(мария, юлия).
мать(анна, петр).
отец(иван, анна).
отец(иван, юлия).

Вопрос*: Является ли* ***иван дедом петра****, можно задать в виде следующей цели:*

отец(иван, X), мать(X, петр).

Решение: X связывается с константой ***анна***

Факты и Правила в Прологе

Правило:

H: - P_1, P_2, …, P_n.

Символ «**:-**» читается как «**если**»

H - заголовок правила (следует из тела правила), заголовок истинен, если истинны все посылки в теле *правила*

P_1, P_2, …, P_n - последовательность предикатов (посылки) - **тело правила**.

Приведенное правило является аналогом **хорновского дизъюнкта**

¬P_1 v ¬ P_2, …, v ¬ P_n v H.

мать(мария, анна).
мать(мария, юлия).
мать(анна, петр).
отец(иван, анна).
отец(иван, юлия).
дед (X, Y): - отец(X, Z), мать(Z, Y).
дед (X, Y): - отец(X, Z), отец(Z, Y).

***Вопрос**: **является ли иван дедом петра**, можно задать в виде следующей цели:*
дед(иван, петр).

Принципы доказательства целей

Цели, образующие составную цель **P1, P2, … Pn** доказываются последовательно, слева направо.

Для доказательства **простой цели P** выполняется просмотр предложений пролог-программы, **начиная с первого и далее**, с целью поиска применимого предложения.

Применимое предложение – это

- **предложение-факт**, сопоставимый (согласующийся) с целью P;
- **предложение-правило**, заголовок которого сопоставим (согласуем) с P.

Цель сопоставима с фактом или заголовком правила, если у входящих в них предикатов **совпадают имена и арность** (количество аргументов), а также **попарно сопоставимы аргументы этих предикатов.**

Сопоставление аргументов предикатов

Возможны следующие варианты пар: **атом-атом, атом-переменная, переменная-переменная**:

- атомы сопоставимы, если они тождественно равны;
- переменная сопоставима с любым атомом – в этом случае *переменная конкретизируется значением сопоставленного атома; далее этим значением конкретизируется все вхождения этой переменной в доказываемые цели (т.е. в ходе доказательства везде вместо этой переменной подставляется её значение);*
- переменная сопоставима с любой переменной – в этом случае переменные становятся *сцепленными; и если в ходе дальнейшего доказательства одна из связанных переменных конкретизируется некоторым значением, то такое же значение получает и другая переменная.*

Простая цель *P достижима (доказуема), если найденное применимое предложение:*

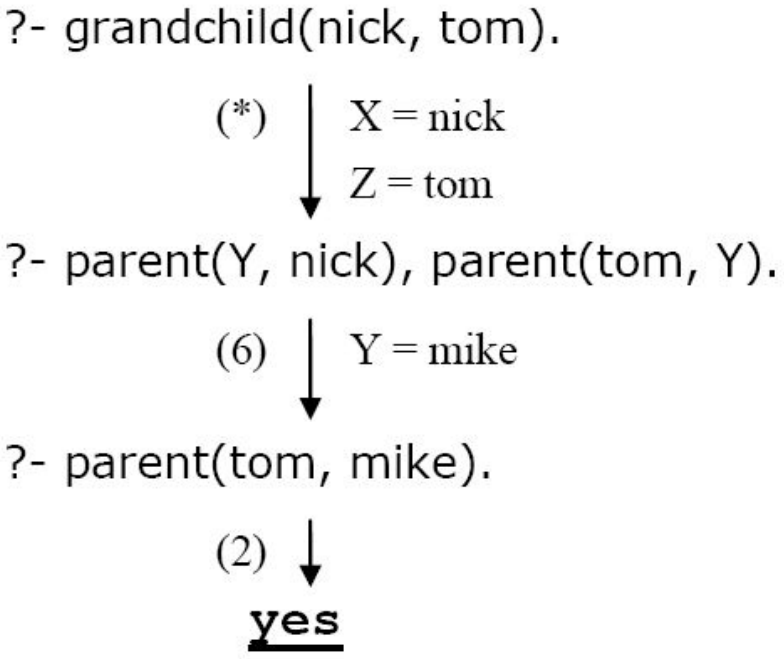
- либо является фактом;
- либо является правилом, и при этом достижимы все цели тела этого правила при соответствующих (найденных при сопоставлении *P и заголовка правила) конкретизациях входящих в них переменных.*

Если при доказательстве очередной простой цели *Pi в составе сложной цели P1, P2, …Pn оказывается, что Pi не достижима, то пролог-система инициирует* ***бектрекинг и осуществляет возврат к строго предыдущей цели Pi-1 и пытается её доказать вновь, найдя другой вариант решения.***

При бектрекинге теряют свои значения переменные, конкретизированные при доказательстве цели *Pi-1 и в ходе неуспешного доказательства цели Pi; в случае успеха повторного доказательства Pi-1 в общем случае будут найдены другие конкретизирующие значения этих переменных.*

Доказательство цели-вопроса с предикатом grandchild(nick, tom))

grandchild(X,Z) :- parent(Y,X), parent(Z,Y).




```
(1) parent(tom, ann).
(2) parent(tom, mike).
(3) parent(mary, mike).
(4) parent(mike, sue).
(5) parent(mike, john).
(6) parent(mike, nick).
(7) parent(john, jane).
(8) parent(john, jim).
```

Рекурсивные правила в Прологе

Предикат «предок» определяется **рекурсивно**:

предок(x, y): - мать(x, y).
предок(x, y): - отец(x, y).
предок(x, y): - мать(x, z), предок(z, y).
предок(x, y): - отец (x, z), предок(z, y).

Рекурсивное определение предиката обязательно **должно содержать нерекурсивную часть**, иначе оно будет логически некорректным и программа зациклится.

Чтобы избежать зацикливания, следует также позаботиться о **порядке выполнения предложений**, поэтому практически полезно, а порой и необходимо придерживаться принципа: ***«сначала нерекурсивные выражения».***

SLD - Резолютивный вывод

Selected Linear Defined Resolution

Linear resolution with **S**election function for **D**efinition clauses

Отборная Линейная Определенная резолюция

1) Par(cAdam, cCain).
2) Par(cEve, cCain).
3) Par(cAdam, cAbel).
4) Par(cEve, cAbel).
5) Par(cCain, cEnoch).

6) Pred(x, z) :- Par(x, z),
7) Pred(x, z) :- Par(x, y), Pred(y, z).

Запрос: Pred(x, cEnoch), Par(x, cAbel)

8) ¬Pred(x, cEnoch) v ¬Par(x, cAbel).

SLD – резолютивный вывод

1. Последовательно просматриваем правила начиная с первого. Найдем правило 6. Унифицирующая подстановка: {x/x1, cEnoch/z1}

Резолюция:

{Pred(x,cEnoch) ∨ ¬Par(x,cEnoch), ¬Pred(x, cEnoch) ∨ ¬Par(x, cAbel)} ⇒

Резольвента (новый запрос): ¬Par(x,cEnoch) ∨ ¬Par(x, cAbel)

2. Снова последовательно просматриваем правила начиная с первого. Найдем правило 5. Унифицирующая подстановка: {x/cCain}

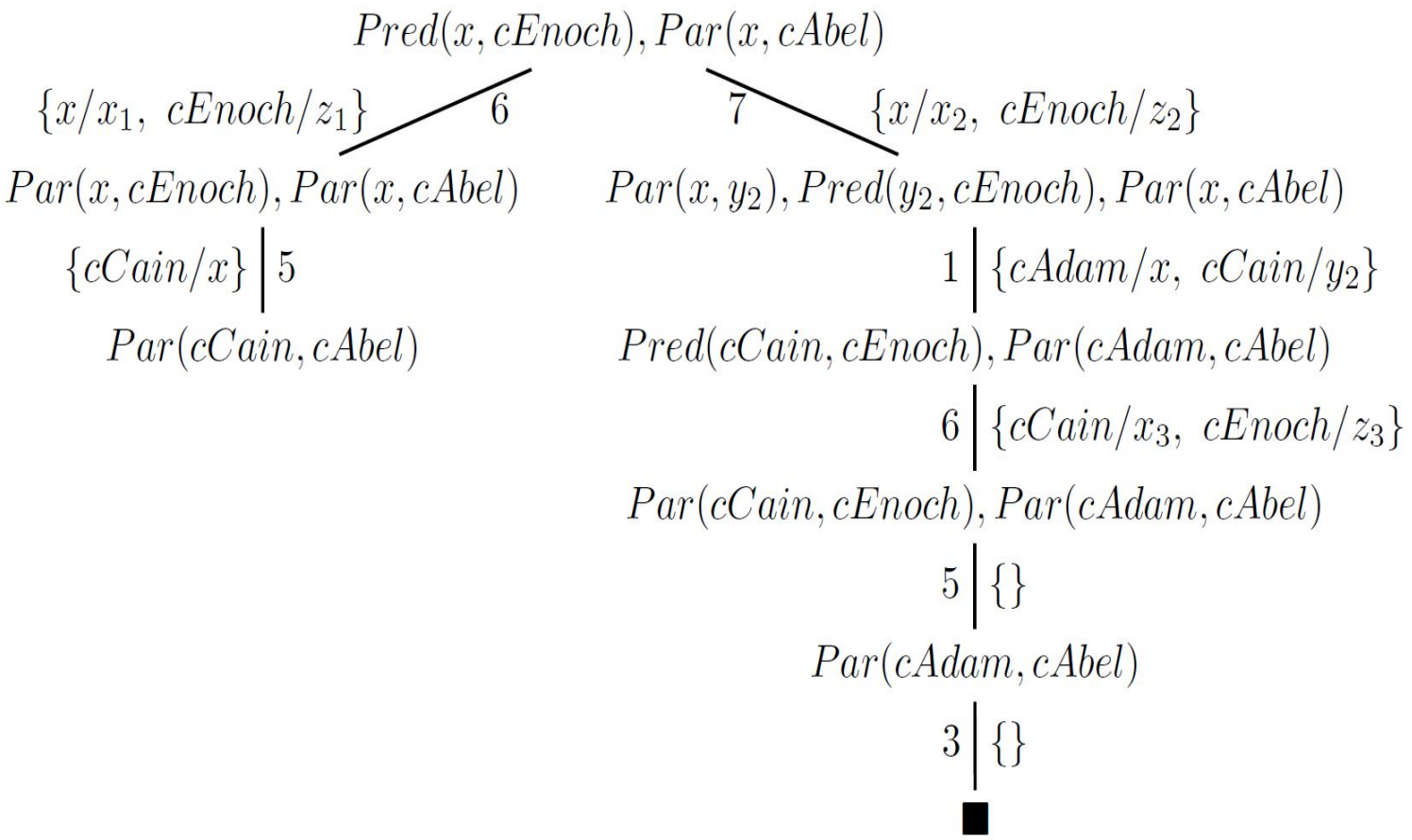
Резолюция:

{¬Par(cCain,cEnoch) ∨ ¬Par(cCain, cAbel), Par(cCain, cEnoch)} ⇒

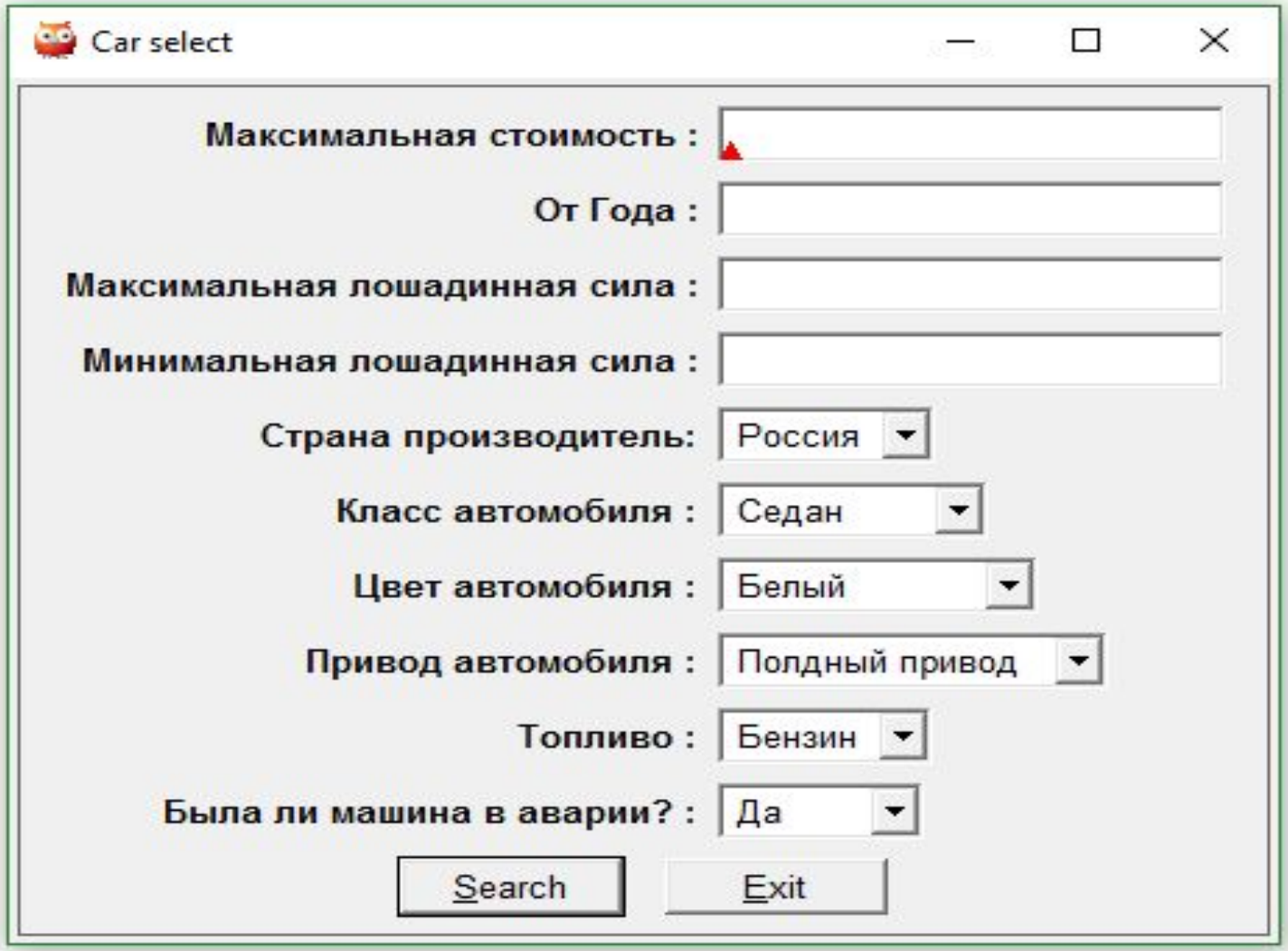
Резольвента (новый запрос): ¬Par(cCain, cAbel)

3. Нет правил для унификации нового запроса. Пустой дизъюнкт не получить.

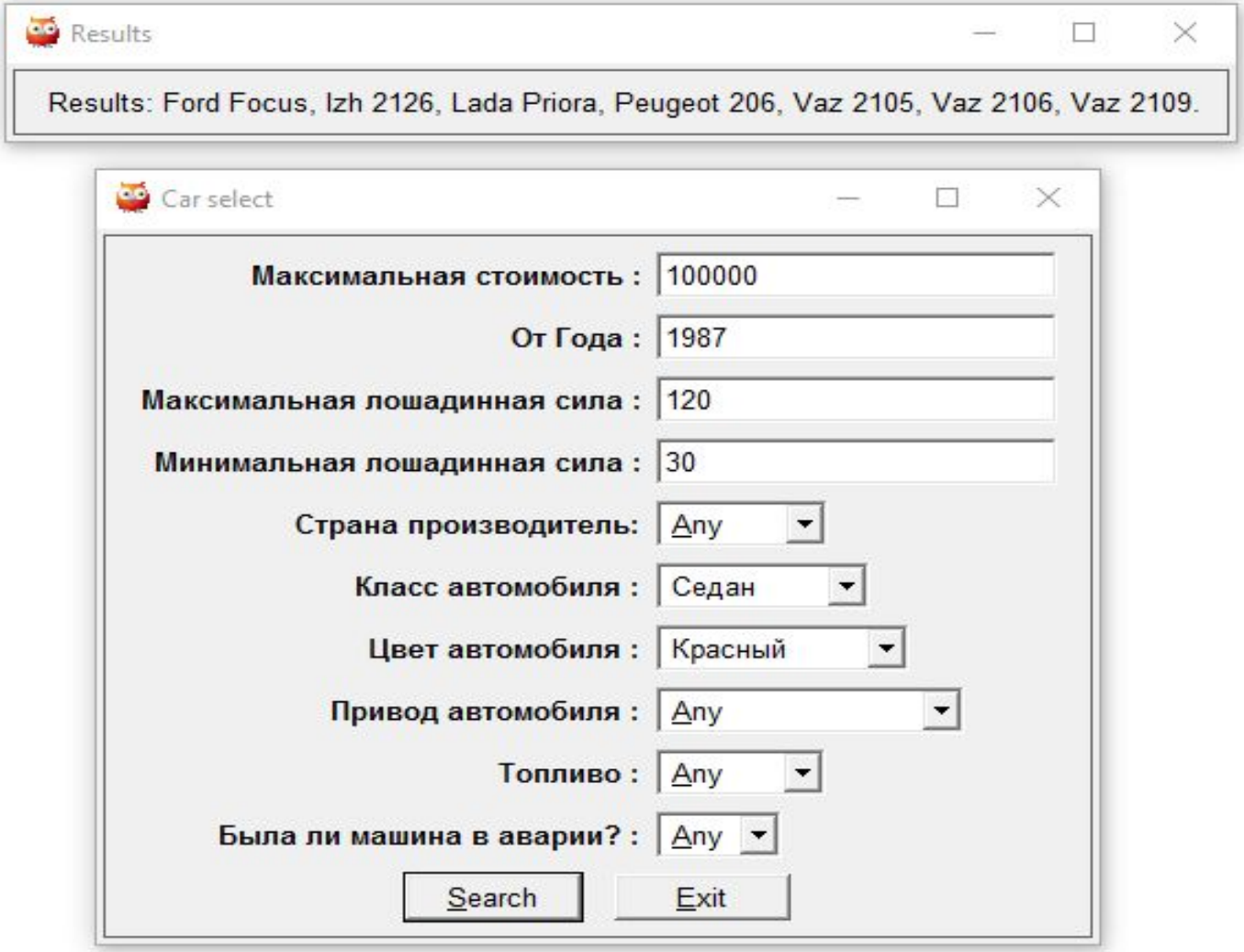
SLD дерево


$Pred(x, cEnoch), Par(x, cAbel)$
$\{x/x_1,\ cEnoch/z_1\}$
6
7
$\{x/x_2,\ cEnoch/z_2\}$
$Par(x, cEnoch), Par(x, cAbel)$
$Par(x, y_2), Pred(y_2, cEnoch), Par(x, cAbel)$
$\{cCain/x\}$
5
1
$\{cAdam/x,\ cCain/y_2\}$
$Par(cCain, cAbel)$
$Pred(cCain, cEnoch), Par(cAdam, cAbel)$
6
$\{cCain/x_3,\ cEnoch/z_3\}$
$Par(cCain, cEnoch), Par(cAdam, cAbel)$
5
$\{\}$
$Par(cAdam, cAbel)$
3
$\{\}$

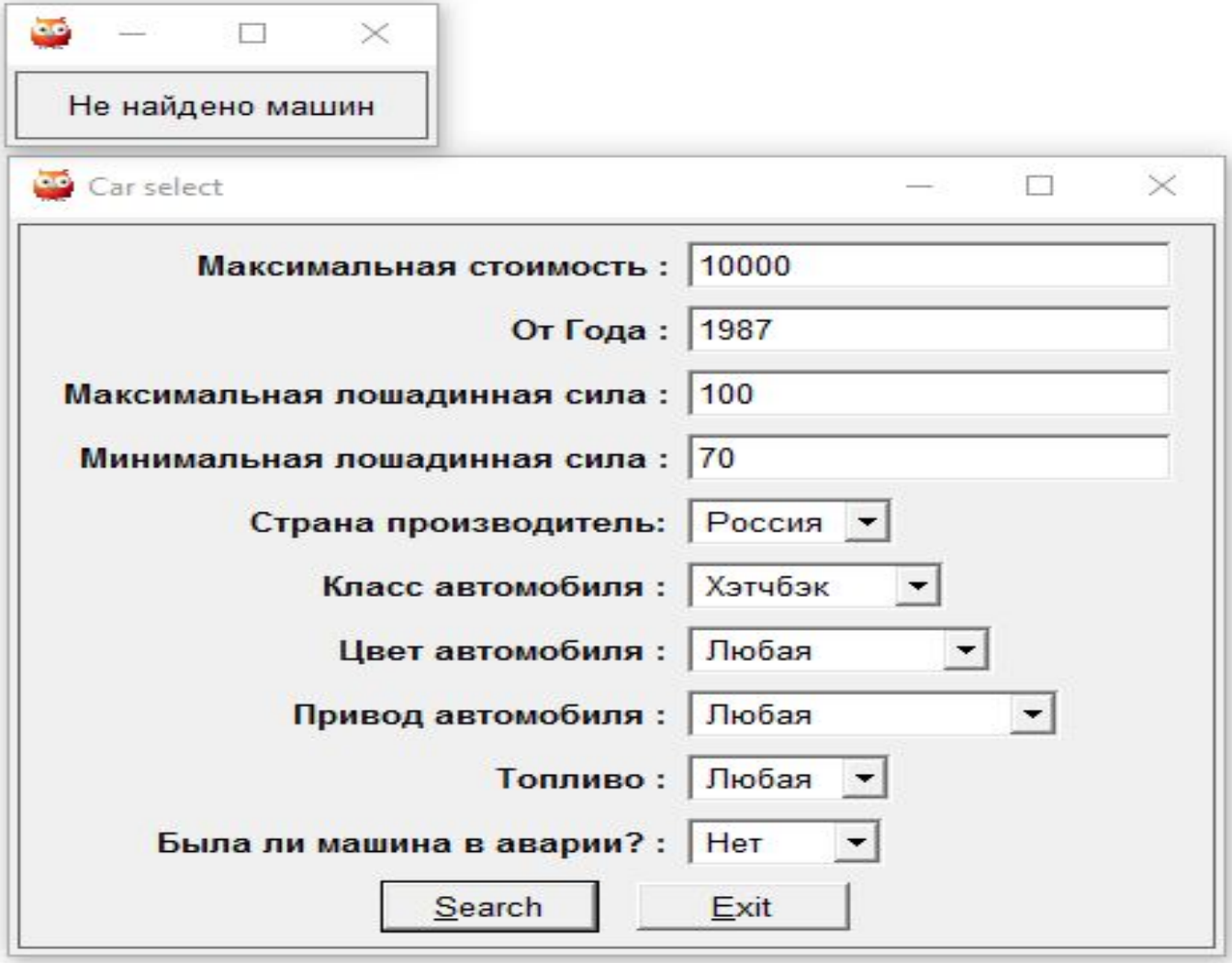
Экспертная система «Выбор авто»



Экспертная система «Выбор авто»



Экспертная система «Выбор авто»



Реализация ЭС «Основное меню выбора автомобиля»

```
selectCarDialog:-
   new(D, dialog('Car select')),
   send(D, append, new(Max_cost, text_item('Максимальная стоимость '))),
   send(Max_cost, type, int),
   send(D, append, new(Year, text_item('От Года '))),
   send(Year, type, int),
   send(D, append, new(Max_HP, text_item('Максимальная лошадинная сила '))),
   send(Max_HP, type, int),
   send(D, append, new(Min_HP, text_item('Минимальная лошадинная сила '))),
   send(Min_HP, type, int),
   send(D, append, new(Country, menu('Страна производитель', cycle))),
   send(D, append, new(Class, menu('Класс автомобиля ', cycle))),
   send(D, append, new(Color, menu('Цвет автомобиля ', cycle))),
   send(D, append, new(Drive, menu('Привод автомобиля ', cycle))),
   send(D, append, new(Fuel, menu('Топливо ', cycle))),
   send(D, append, new(Crash, menu('Была ли машина в аварии? ', cycle))),
   send_list(Country, append, ['Россия', 'Европа', 'Сша', 'Япония', 'Любая']),
   send_list(Class, append, ['Седан', 'Хэтчбэк', 'Универсал', 'ЛифтБэк', 'Любая']),
   send_list(Color, append, ['Белый', 'Серебрянный', 'Черный', 'Зеленый', 'Синий', 'Красный', 'Любая']),
   send_list(Drive, append, ['Полдный привод', 'Задний привод', 'Передний привод', 'Любая']),
   send_list(Fuel, append, ['Бензин', 'Дизель', 'Любая']),
   send_list(Crash, append, ['Да','Нет', 'Любая']),
   send(D, append,
      button(search, message(@prolog, writeResults,
                             Max_cost?selection,
                             Year?selection,
                             Min_HP?selection,
                             Max_HP?selection,
                             Country?selection,
                             Class?selection,
                             Color?selection,
                             Drive?selection,
                             Fuel?selection,
                            Crash?selection))),
   send(D, default_button, search),
   send(D, append, button(exit, message(D,destroy))),
   send(D, layout_dialog),
   send(D, open).

:-selectCarDialog.
```

Реализация ЭС «Факты и правила»

```
%car(Название,Стоимость,Год,Лошадинная сила,Страна,Класс автомобиля,Цвет,Привод,
Топливо,Авария)
car('Moskvitch 412',50000,1989,89,'Россия','Седан','Зеленый','Задний привод','Бензин','Да').
car('Izh 2125',55000,1987,89,'Россия','Хэтчбэк','Синий','Задний привод','Бензин',no).
car('Izh 2126',60000,1995,95,'Россия','Хэтчбэк','Белый','Задний привод','Бензин','Да').
car('Vaz 2121',75000,1990,90,'Россия','ЛифтБэк','Красный','Передний привод','Бензин','Нет').
car('Vaz 2101',50000,1979,80,'Россия','Седан','Красный','Задний привод','Бензин','Нет').

% Стоимость
costsLessThan(Name,MaxCost) :-
        (   MaxCost > 0 ->
        car(Name,Cost,_,_,_,_,_,_,_,_),
        not(Cost > MaxCost);
        true).

% Выбор по Стоимости
selectByCost(MaxCost,R) :-
        findall(Name,
              (car(Name,_,_,_,_,_,_,_,_,_),costsLessThan(Name,MaxCost)),
              R).
```

Реализация ЭС «Выбор по параметрам»

```
% Предикат Выбора по параметрам
selectByParameters(MaxCost,Year,MinHP,MaxHP,Country,Class,Color,Drive,Fuel,Crash,R) :-
        selectByCost(MaxCost,ByCost),
        selectByYear(Year,ByYear),
        selectByMinHorsepower(MinHP,ByMinHP),
        selectByMaxHorsepower(MaxHP,ByMaxHP),
        selectByCountry(Country,ByCountry),
        selectByClass(Class,ByClass),
        selectByColor(Color,ByColor),
        selectByDrive(Drive,ByDrive),
        selectByFuel(Fuel,ByFuel),
        selectByCrash(Crash,ByCrash),

        intersection(ByCost,ByYear,R1),
        intersection(ByMinHP,R1,R2),
        intersection(ByMaxHP,R2,R3),
        intersection(ByCountry,R3,R4),
        intersection(ByClass,R4,R5),
        intersection(ByColor,R5,R6),
        intersection(ByDrive,R6,R7),
        intersection(ByFuel,R7,R8),
        intersection(ByCrash,R8,R).
```

Реализация ЭС «Вывод результатов»

```
%Предикат вывода результатов
writeResults(MaxCost,Year,MinHP,MaxHP,Country,Class,Color,Drive,Fuel,Crash) :-

    selectByParameters(MaxCost,Year,MinHP,MaxHP,Country,Class,Color,Drive,Fuel,Crash,Buf),
    sort(Buf,R),
    new(D, dialog('Results')),
    length(R, Length),
    send(D, append, new(Text, text(''))),
   (   Length > 0 ->
          send(Text, append, 'Results: '),
          foreach(  member(X,R), (
                              send(Text, append, X),
                              (not(last(R,X)) ->
                                    send(Text, append, ', ');
                                    send(Text, append, ' . ')
                              )
                                    )
                    );
        send(Text, append, ' No cars found ')),
        send(D, open).
```

Достоинства и недостатки логических моделей

Достоинства:

- Наличие единообразной формальной процедуры доказательства теорем (организация вывода) и, как следствие,
- Возможность непосредственно запрограммировать механизм вывода синтаксически правильных высказываний.

Недостатки:

- Сложность использования при построении доказательств эвристик, отражающих специфику конкретной предметной области.
- Только с помощью правил, задающих синтаксис языка, нельзя установить истинность или ложность высказываний: синтаксически правильные высказывания могут быть семантически абсолютно бессмысленными.
- Отсутствие средств структурирования используемых элементов, без таких средств большая модель превращается в плохо обозримый конгломерат независимых фактов, трудно поддающихся анализу и обработке.
- Недопустимость противоречий.

Продукционные модели

Продукция:

(i); Q ; P; A⇒B ; N,

где

i - идентификатор продукции, с помощью которого осуществляется поиск продукции в базе знаний; в качестве имени может выступать некоторая лексема, отражающая суть продукции, например «покупка книги», или просто порядковый номер;

Q - элемент характеризует сферу применения продукции. Разделение знаний на отдельные сферы позволяет экономить время на поиск нужных знаний.

A⇒B - основной элемент продукции, ее **ядро**, часто называемое **правилом**. Интерпретация ядра продукции может быть различной и зависит от того, что стоит слева и справа от знака секвенции ⇒.

Обычное прочтение ядра продукции выглядит следующим образом:

ЕСЛИ A, ТО B;

более сложные конструкции ядра допускают в правой части альтернативный выбор, например,

ЕСЛИ A_1 ТО B_1, ИНАЧЕ B_2.

Секвенция может истолковываться в обычном логическом смысле как знак логического следования B из A (если A ложно, то о B ничего сказать нельзя). Однако возможны и другие интерпретации ядра продукций, например, A описывает некоторое условие, необходимое для совершение действия B.

P - условие применимости ядра продукции. Обычно P представляет собой **логическое выражение** (как правило, **предикат**). Когда P принимает значение истина, ядро продукции активизируется, в противном случае ядро продукции не может быть использовано. Например, если в продукции

“**НАЛИЧИЕ ДЕНЕГ**; ЕСЛИ ХОЧЕЩЬ КУПИТЬ ВЕЩЬ X, ТО ОПЛАТИ В КАССЕ ЕЕ

СТОИМОСТЬ И ОТДАЙ ЧЕК ПРОДАВЦУ” условие применимости ядра ложно (денег нет), применить ядро продукции невозможно.

N - элемент описывает постусловия продукции. Они актуализируются только в том случае, если ядро продукции реализовалось. Постусловия описывают действия и процедуры, которые необходимо выполнить после реализации B. Например, после покупки некоторой вещи в магазине необходимо уменьшить количество товара данного типа в описи имеющихся товаров.

Классификация ядер продукции

Детерминированные

Секвенция ⇒ в детерминированных ядрах реализуется с необходимостью,

Недетерминированные

Секвенция ⇒ в недетерминированных - с возможностью.

Детерминированные ядра

- **Однозначные**
- **Альтернативные**

В правой части ядра указываются альтернативные возможности выбора, которые оцениваются специальными **весами выбора.**

В качестве таких весов могут использоваться оценки разных типов:

- **Вероятностные**

(ЕСЛИ А, ТО С ВЕРОЯТНОСТЬЮ 0.6 НАДО ДЕЛАТЬ В1,А С ВЕРОЯТНОСТЬЮ 0.4 В2);

- **Лингвистические**

(ЕСЛИ А, ТО ЧАЩЕ НАДО ДЕЛАТЬ В1, РЕЖЕ В2);

Недетерминированные ядра

Возможность реализации определяется **оценками реализации** ядра:

ЕСЛИ А, ТО **ВОЗМОЖНО** В.

Оценки могут быть:

- **вероятностные**

(ЕСЛИ А, ТО С **ВЕРОЯТНОСТЬЮ 0.8** НАДО ДЕЛАТЬ В);

- **лингвистические**

(ЕСЛИ А, ТО **С БОЛЬШОЙ ДОЛЕЙ УВЕРЕННОСТИ** В);

и т.п.

Особым типом являются прогнозирующие продукции, в которых описываются последствия, ожидаемые при реализации **А:**

ЕСЛИ А, ТО **С ВЕРОЯТНОСТЬЮ p** МОЖНО ОЖИДАТЬ В.

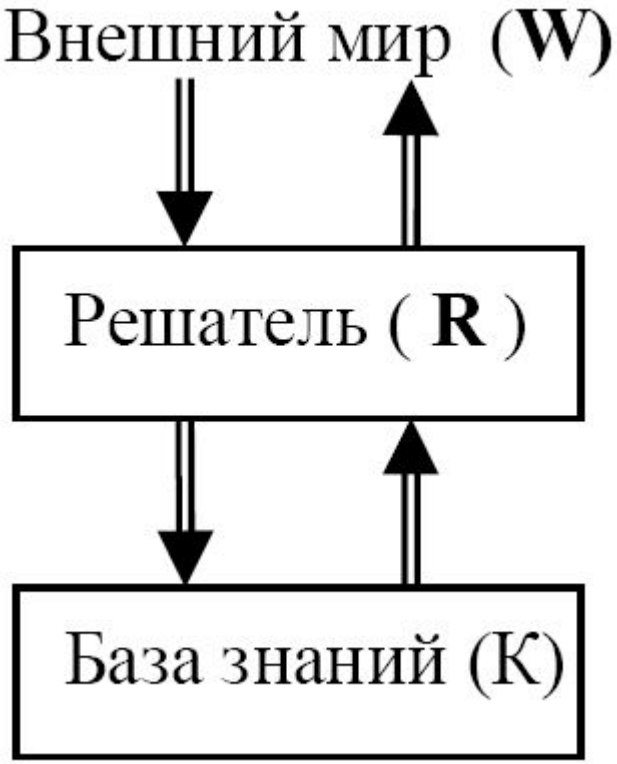
Классификация ядер продукций

1. **Aw ⇒ BR.** В левой части продукции указана информация, поступившая из внешнего мира, а в правой - сведения о вытекающих из этой информации изменениях в рассуждающей системе.

2. **Aw ⇒ Bk.** Такие продукции отражают ситуацию передачи некоторого сообщения из внешнего мира для запоминания в базе знаний.

3. **Ak ⇒ Bw.** Решатель выступает в качестве отделения связи при выдаче сообщения из
базы знаний во внешний мир.

AR ⇒ Bk. Некоторый факт, полученный решателем, передается на хранение в базу
знаний.



Классификация ядер продукций

5. **Ak ⇒ BR. Описание обмена информацией при работе решателя.** Некоторая информация выбирается из базы знаний и передается для обработки в решатель.

6. **Aw⇒ Bw. Продукции непосредственного отклика.**

Aw - описание некоторой наблюдаемой ситуации во внешнем мире или воздействие внешнего мира на решатель.

Bw - Описание действия, которое поступает от системы в окружающий мир. Решатель в этих случаях не успевает сработать, он лишь транслирует информацию от адресата **Aw к адресату Bw (эффект «отдергивания руки»).**

7. **AR⇒ Bw. Описание воздействий во внешний мир,** которые порождает результат работы решателя.

8. **AR⇒ BR. Внутренние продукции рассуждающей системы.** Описывают промежуточные шаги процесса вывода и не влияют непосредственно на содержимое Базы знаний и состояние внешнего мира. Эти продукции описывают единичные шаги многошаговых процессов рассуждений.

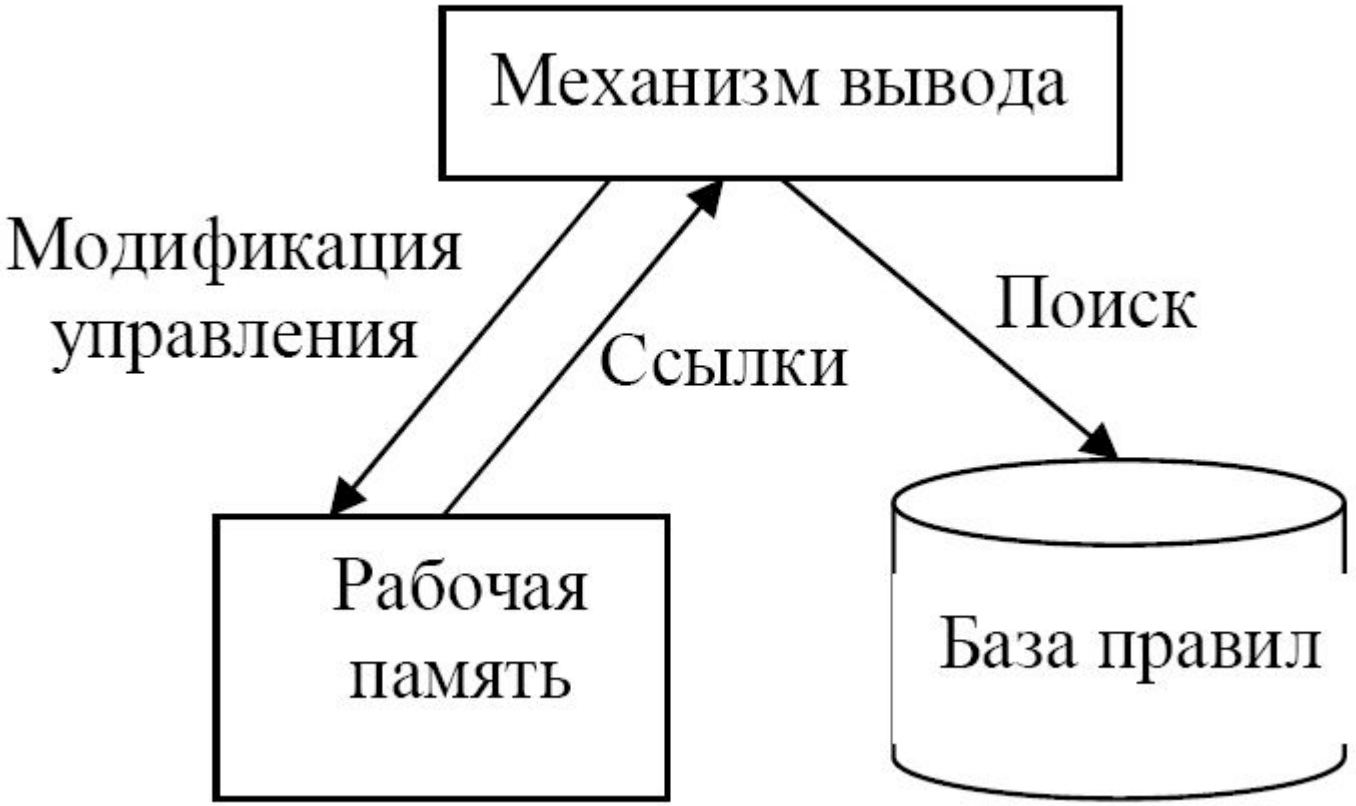
9. **Ak ⇒ Bk. Описание процедур преобразования знаний в базе знаний:**

- обобщение знаний;
- получение новых знаний из ранее известных с помощью логического вывода;
- установление закономерностей между знаниями на основании обработки сведений о единичных фактах; хранящихся в базе знаний;

и т.п.

Решатель в этом случае используется лишь в качестве инструмента, с помощью которого происходит изменение состояния базы знаний.

Структура продукционной системы



Структура продукционной системы

База правил - набор правил, используемый как база знаний.

Рабочая память (память для кратковременного хранения) - хранит предпосылки**,** касающиеся конкретных задач ПО, и результаты выводов, полученных на их основании.

Механизм логического вывода - использует правила в соответствии с содержимым рабочей памяти.

Стратегии логического вывода

1. Стратегия прямого вывода.

В системах с прямым выводом по известным фактам отыскивается заключение, которое из этих фактов следует.

Если такое заключение удается найти, оно заносится в рабочую память (РП).

Прямой вывод часто называют **выводом, управляемым данными.**

Стратегии логического вывода

2. Стратегия обратного вывода.

В системах с обратным выводом в начале выдвигается некоторая **гипотеза (цель).**

Если эта цель сразу достигнута быть не может, ставится **промежуточная** цель, **детализирующая первую** и являющаяся по отношению к ней **подцелью**, то есть механизм вывода в процессе работы как бы возвращается назад, переходя от цели к фактам и пытаясь найти среди них те, которые позволяют достичь цель.

Достигнутые подцели позволяют подтвердить гипотезы более высоких порядков вплоть до достижения первоначальной цели.

Обратный вывод часто называют **выводом, управляемым целями.**

Достоинства продукционных моделей

1. Подавляющая часть человеческих знаний может быть представлена в виде продукций.
2. Системы продукций являются модульными. За небольшим исключением удаление или добавление продукций в базу знаний не приводит к изменениям в остальных продукциях.
3. При необходимости системы продукций могут реализовывать любые алгоритмы и, следовательно, способны отражать любое процедурное знание, доступное ЭВМ.
4. Наличие в продукциях указателей на сферу применения продукции позволяет эффективно организовывать память, сократив время поиска в ней необходимой информации. Классификация сфер может быть многоуровневой, что еще более повышает эффективность поиска знаний, так как позволяет наследовать информацию в базе знаний.
5. При объединении систем продукций и сетевых представлений получаются средства, обладающие большой вычислительной мощностью. В таких моделях декларативные знания описываются в сетевой компоненте модели, а процедурные знания – в продукционном. В этом случае говорят о работе продукционной системы над семантической сетью.
6. Естественный параллелизм в системе продукций, асинхронность их реализации делают продукционные системы удобной моделью вычислений для ЭВМ параллельных архитектур.

Недостатки продукционных моделей

1. При большом количестве продукций становится **сложной проверка непротиворечивости** системы продукций (эту проверку приходится делать при каждом добавлении новой продукции).

2. Из-за присущей системе недетерминированности (неоднозначного выбора выполняемой продукции из фронта активных продукций) возникают принципиальные **трудности при проверке корректности работы системы**.

3. **Нехватка строгой теоретической основы**.