7. БАЗЫ ЗНАНИЙ

7.1. Знания

Знания – это выявленные закономерности предметной области (принципы, связи, законы), позволяющие решать задачи в этой области.

Знание, которое существует в виде заранее известных фактов, называют экстенсивным знанием *(экстенсионалом)*, а базу данных – экстенсиональной базой.

Знание, выводимое из экстенсивного знания при помощи правил, называют интенсивным знанием *(интенсионалом)*, а базу данных – интенсиональной.

Интенсивная форма позволяет выразить данные в компактной форме, избежать избыточности данных.

Пример. Статическая база данных состоит из утверждений предиката **parts**, состоящего из объектов **subpart** (поддеталь) и **part** (деталь).

```
domains
        subpart, part = integer
predicates
        nondeterm parts (subpart, part)
        nondeterm componentOf (subpart, part)
clauses
        parts (200, 315).
        parts (250, 300).
        parts (250, 315).
        parts (300, 324).
        parts (315, 350).
        componentOf (X, Y) :-
                parts (X, Y).
goal    componentOf (250, X).
```

На поставленную цель будет получено два решения **(X=300** и **X=315)**, хотя, очевидно, что поддеталь 250 является частью деталей 324 и 350, то есть 250 – это поддеталь второго уровня.

Можно представить ещё один предикат **partsTwo** для хранения информации о таких деталях:

```
predicates
        partsTwo (subpart, part)
clauses
        partsTwo (200, 350).
        partsTwo (250, 324).
        partsTwo (250, 350).
```

Однако, доступен второй, более эффективный путь – использование интенсивной базы данных. При этом предикат **partsTwo** и связанные с ним факты не потребуются. Достаточно добавить правило вида:

```
componentOf (X, Z) :-
        parts (X, Y),
        parts (Y, Z).
```

7.2. База знаний

База знаний (knowledge base) – это особого рода база данных, разработанная для управления знаниями (их сбором, хранением, поиском и выдачей).

Система базы знаний – это компьютерная система, составляющими которой являются:

1. База данных, содержащая основные факты.
2. База данных, содержащая правила, которые позволяют делать выводы из базы данных фактов.
3. Система управления – программное обеспечение, которое поддерживает основные функции СУБД, а также управление процессом вывода в базе данных правил, оперирующих с базой данных фактов.

Можно утверждать, что представлению информации в базе данных присущ пассивный аспект: таблица, заполненная данными память. В базе знаний подчёркивается **активный** аспект представления. Операция *знать* становится активной операцией, позволяющей не только запоминать, но и *посредством логического вывода* получать новые знания. Эта возможность увеличивает приносимую базой данных пользу при определении, контроле и интерпретации поддерживаемых ею данных.

Для хранения данных используются базы данных, для хранения знаний – базы знаний. Для баз данных характерны большой объём и относительно небольшая удельная стоимость информации. Для баз знаний – небольшой объём, но исключительно дорогие информационные массивы.

Наиболее важный параметр баз знаний – качество содержащихся знаний. Лучшие базы знаний включают самую актуальную и достоверную информацию, имеют совершенные системы поиска, а также тщательно продуманную структуру и формат знаний.

В языке *Пролог* базы знаний описываются в форме конкретных фактов, а также правил и процедур логического вывода. Достоверность обобщённых сведений зависит от наличия необходимых фактов и достоверности данных в базах знаний.

Базы знаний могут использоваться для создания экспертных систем. Главная цель этих систем – помочь менее опытным людям найти существующее описание способа решения какой-либо проблемы из предметной области.

7.3. Модели представления знаний

Существует множество всевозможных моделей представления знаний для разных предметных областей. Большинство из них может быть сведено к следующим классам:

1. Формальные логические модели.
2. Продукционные модели.
3. Семантические сети.
4. Фреймы.

7.3.1. Формальные логические модели

Эта модель предполагает унифицированное описание объектов и действий в виде предикатов первого порядка. Отношения для объектов задаются явно в виде фактов, а действия описываются как правила, определяющие логическую формулу *вывода фактов из других фактов*.

Механизм вывода осуществляет дедуктивный перебор фактов, относящихся к правилу по принципу *сверху – вниз слева – направо* или обратный вывод *методом поиска в глубину*.

Для логической модели характерна строгость формального аппарата получения решения. Однако полный последовательный перебор всех возможных решений может приводить к комбинаторным взрывам, в результате чего поставленные задачи могут решаться недопустимо большое время.

Модель применима в небольших исследовательских системах, так как предъявляет высокие требования и ограничения к предметной области.

7.3.2. Продукционные модели

В модели основной единицей знаний служит *правило (продукция)* вида:

ЕСЛИ (условие), ТО (действие)

Под условием *(антецедентом)* понимается некоторое предложение, по которому осуществляется поиск в базе знаний, а под действием *(консеквентом)* – действие, выполняемое при успешном исходе поиска.

Действие может быть промежуточным, выступающим далее как условие и целевым (завершающим работу программы).

При помощи правил могут быть выражены пространственно-временные, причинно-следственные, функционально-поведенческие *отношения объектов*. Правилами могут быть описаны и сами объекты: *свойство – объект* или *объект – набор свойств*. Однако чаще описания объектов фигурируют только в качестве переменных внутри правил. В основном продукционная модель предназначена для описания последовательности разных действий и в меньшей степени для структурированного описания объектов.

Достоинствами продукционной модели являются:

- наглядность,
- высокая модульность,
- лёгкость модификации,
- простой механизм логического вывода.

Отличительная особенность продукционной модели – способность осуществлять выбор правил из конфликтного множества возможных на данный момент времени в зависимости от определённых критериев и характеристик предметной области. Подобная стратегия поиска решений называется *поиском в ширину*. Для её реализации в описание продукций вводятся предусловия и постусловия в виде:

$$\mathbf{A, B, C \rightarrow D, E}$$

, где $\mathbf{C \rightarrow D}$ (импликация) представляет собственно правило,
A – предусловие выбора класса правил,
B – предусловие выбора правила в классе,
E – постусловие правила, определяющее переход на следующее правило.

В предусловиях и постусловиях могут быть заданы дополнительные процедуры (например, по вводу и контролю данных или по их обработке). Введение предусловий и постусловий позволяет выбирать наиболее рациональную стратегию работы механизма вывода, существенно сокращая перебор относящихся к решению правил.

Продукционная модель предполагает более гибкую организацию работы механизма вывода по сравнению с формальной логической моделью. Так, в зависимости от направления вывода возможна как прямая аргументация, управляемая данными *(от данных к цели)*, так и обратная, управляемая целями *(от целей к данным)*.

7.3.2.1. Прямая цепочка рассуждений

При использовании прямой цепочки рассуждений решается задача: *по известным условиям найти последствия*.

Прямой вывод используется в продукционных моделях при решении, например, задач интерпретации, когда по исходным данным нужно определить сущность некоторой ситуации или в задачах прогнозирования, когда из описания некоторой ситуации требуется вывести все следствия.

Механизм, основанный на прямой цепочке, функционирует так:

1. Вводится условие.
2. Система ищет в базе знаний правило, в условной части которого содержится соответствующее условие, которое может быть доказано на основе имеющихся фактов.

3. В соответствии с консеквентом (частью **ТО**) найденное правило может генерировать новый факт, который добавляется к уже существующим фактам.
4. Система обрабатывает следующее правило, годное для выполнения, повторяя шаги 2 и 3. Рассуждения заканчиваются, когда больше нет необработанных правил.

Пример 7-1. Прямой вывод.

Для простоты предполагается, что правила не содержат переменных.

```
facts       вода (symbol)
            сухо (symbol)
            закрыто (symbol)
predicates  nondeterm утечка (symbol)
            nondeterm неисправность (symbol)
            nondeterm неПоступает (symbol, symbol)
clauses     % известные на данный момент факты
            вода (гостиная).
            сухо (ванная).
            закрыто (окно).
            % правила
            утечка (ванная) :-                    % 1
                  вода (гостиная),
                  сухо (кухня).
            утечка (кухня) :-                     % 2
                  неисправность (кухня),
                  неПоступает (вода, снаружи).
            неисправность (кухня) :-              % 3
                  вода (гостиная),
                  сухо (ванная).
            неПоступает (вода, снаружи) :-        % 4
                  закрыто(окно).
goal        утечка (кухня).
```

Поиск решения при прямой цепочке рассуждений:

1. На основе фактов, известных на момент решения задачи, можно решить только правило % 3. В результате состав фактов станет таким:

   вода (гостиная).
   сухо (ванная).
   закрыто (окно).
   неисправность (кухня).

2. Делается попытка решить очередное правило. Таким правилом станет правило %4. В результате к фактам добавиться ещё один:

   вода (гостиная).
   сухо (ванная).
   закрыто (окно).

неисправность (кухня).
неПоступает (вода, снаружи).

3. Становится возможным решение правила %2, и получается ответ **yes**.

7.3.2.2. Обратная цепочка рассуждений

Обратная цепочка рассуждений применяется для того, чтобы *по известному результату найти причины, которые его вызвали.*

При обратной цепочке рассуждений механизм вывода пытается выяснить, обусловлен ли результат другими известными фактами и правилами, то есть он пытается определить достоверность всех посылок в тех правилах, которые могут применяться для установления истинности заключения.

Обратный вывод применяется в задачах диагностики, когда нужно проверить определенную гипотезу или небольшое множество гипотез на соответствие фактам.

Механизм, основанный на обратной цепочке рассуждений, функционирует следующим образом:

1. Выдвигается гипотеза (цель).
2. Система ищет в базе знаний правило, в заголовке которого содержится цель.
3. В соответствии с телом найденного правила проверяется истинность конъюнкции всех входящих в него подцелей.
4. Каждая подцель обрабатывается как новое целевое утверждение, повторяя шаги 2 и 3. Рассуждения заканчиваются, когда больше нет необработанных правил в гипотезе.

Механизм вывода в *Прологе* основан на обратной цепочке рассуждений. Процесс выполнения программы сводится к установлению истинности определённого предложения и продолжается до тех пор, пока не будут найдены некоторые известные базовые истинные факты.

Пример 7-2. Обратный вывод.

Текст программы идентичен тексту из *Примера 7-1*.

Поиск решения при обратной цепочке рассуждений:

1. Согласно цели **утечка (кухня)** делается попытка доказать правило %2.
2. Доказывается новая цель **неисправность (кухня)**. Эта цель доказывается по правилу %3.
3. Доказывается следующая цель правила %2 **неПоступает (вода, снаружи)**. Эта цель подтверждается по правилу %4.
4. Все цели правила %2 согласованы, получается ответ **yes**.

7.3.4.3. Сравнение прямого и обратного логического вывода

Правила вывода образуют цепочки, по которым переход осуществляется слева направо. Элементы в левой части этих цепочек представляют собой входную информацию, а в правой – производную информацию.

Эти два вида информации могут иметь разные названия в зависимости от контекста, в котором они используются. Входная информация может называться *данными*, или *свидетельствами*, или *проявлениями*. Производная информация может называться *целями,* или *доказываемыми гипотезами*, или *причинами проявлений*, или *объяснениями*, которые позволяют трактовать имеющиеся факты. Поэтому цепочки этапов вывода соединяют информацию различных типов следующим образом:

данные	→	**...**	→	**цели**
свидетельства	→	**...**	→	**гипотезы**
факты, наблюдения	→	**...**	→	**объяснения, диагнозы**
проявления	→	**...**	→	**диагнозы, причины**

И прямой, и обратный логический вывод требуют поиска, но они отличаются друг от друга по направлению поиска. При обратном выводе поиск происходит от целей к данным, а при прямом – от данных к цели. Поскольку *обратный логический вывод* начинается с целей, принято считать, что он *управляется целями*. Соответственно *прямой логический вывод* начинается с данных и *управляется данными*.

Выбор зависит от конкретной решаемой задачи. Если необходимо проверить, является ли истиной некоторая определённая гипотеза, то следует выбрать логический вывод в обратном направлении. Если же имеется много конкурирующих гипотез, и нет причин начинать с одной, а не с другой, то лучше формировать логический вывод в прямом направлении. Например, прямой вывод рекомендуется в задачах текущего контроля, где непрерывно поступают новые данные и система должна определить, не возникла ли аномальная ситуация. Конкретному выбору направлению поиска может помочь *анализ правил*:

- Если узлов данных мало, а целевых узлов много, то предпочтительнее прямой логический вывод.
- Если количество целевых узлов гораздо меньше по сравнению с узлами данных, то – обратный логический вывод.

7.3.4.4. Механизм вывода

Механизмом вывода (inference engine) – это механизм, который управляет перебором правил и позволяет формировать логические выводы.

Механизм вывода выполняет две функции: вывод и управление.

Функция вывода обеспечивает просмотр правил и/или существующих фактов из рабочей памяти и сопоставление этих фактов с базой правил или добавление фактов по мере необходимости. Для пользователя сохраняется информация о полученных заключениях и/или запрашивается информация, когда для срабатывания очередного правила оказывается недостаточно данных.

Функция управления обеспечивает определение порядка просмотра и применения правил и выполняет следующие функции:

1. *Сопоставление*. Образец правила сопоставляется с имеющимися фактами.
2. *Выбор*. Если в конкретной ситуации может быть применено сразу несколько правил, то из них выбирается одно, наиболее подходящее по заданному критерию.
3. *Срабатывание*. Если образец правила при сопоставлении совпал с какими-либо фактами, то правило срабатывает.
4. *Действие*. Рабочая память подвергается изменению путём добавления в неё заключения сработавшего правила. Если в правой части правила содержится указание на какое-либо действие, то оно выполняется.

На рис. 7 показан цикл работы механизма вывода.

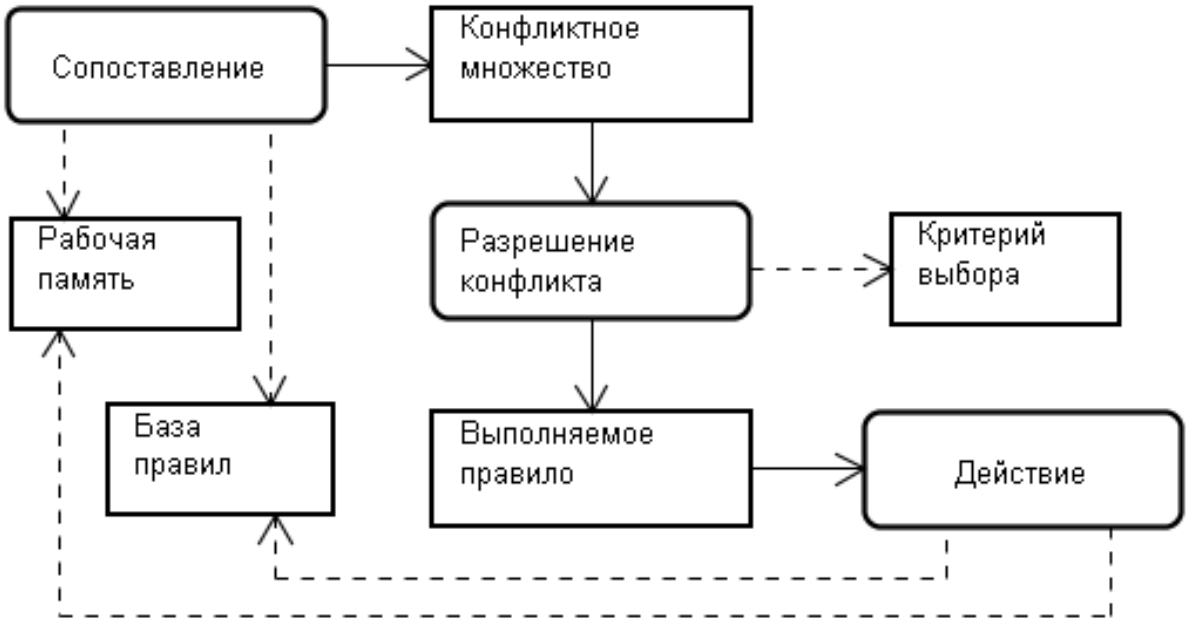

Рис. 7. Цикл работы механизма вывода

На практике обычно учитывается история работы, то есть поведение механизма вывода фиксируется в протоколе, чтобы объяснить пользователю полученное заключение.

Действие механизма вывода основано на применении *правила заключения*, называемого **modus ponens**:

Если известно, что истинно утверждение **A** и существует правило вида **ЕСЛИ A, ТО B**, тогда утверждение **B** тоже истинно:

$$\frac{\mathbf{A},\ \mathbf{A} \rightarrow \mathbf{B}}{\mathbf{B}}\ \textbf{modus ponens}$$

7.3.3. Семантические сети

Семантическая сеть (semantic network) – это модель, в которой знания о предметной области формализуются в виде ориентированного графа.

Вершинами графа являются понятия различных категорий (объекты, события, свойства, операции). Понятия могут быть конкретными или абстрактными.

Дуги представляют отношения между понятиями.

Возможные в семантических сетях отношения представлены в табл. 20.

Таблица 20

Отношения в семантических сетях

Отношения	Разновидности отношений	Примеры
логические		дизъюнкция конъюнкция отрицание импликация
теоретико-множественные		часть – целое множество – подмножество класс – элемент класса
функциональные	количественные	больше… меньше…
	временнЫе	раньше… позже… в течение…
	пространственные	далеко от… за… под…
	атрибутивные	объект – свойство свойство – значение
квантификационные	логические кванторы	всеобщность существование
	нелогические кванторы	много… несколько…

Характерной особенностью семантических сетей является обязательное наличие трех типов отношений:

1. класс – элемент класса;
2. свойство – значение;
3. пример элемента класса.

В качестве примера на рис. 8 показана очень простая семантическая сеть для представления объекта *Электровоз*.

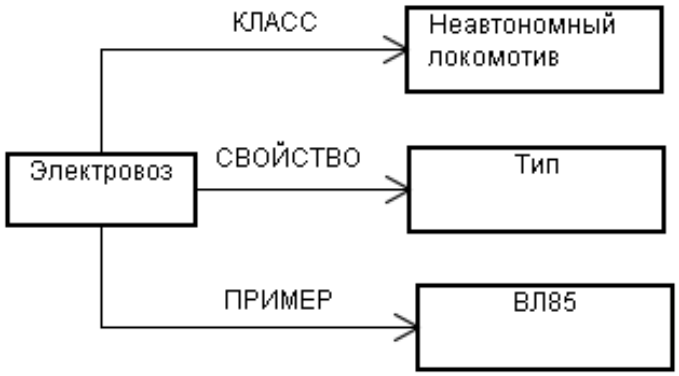

Рис. 8. Семантическая сеть представления объекта *Электровоз*

Семантические сети классифицируются по количеству типов отношений и по типу отношений.

По количеству типов отношений сети бывают:

1. *однородные* (с единственным типом отношений) и *неоднородные* (с различными типами отношений);
2. *бинарные* (отношения связывают два объекта) и **n**-парные (в них есть специальные отношения, связывающие более двух понятий).

В зависимости от типа отношений, используемых в однородной сети, различают квалифицирующие сети, функциональные сети и сценарии.

Квалифицирующие сети использую отношения структуризации, к которым, в частности, относятся:

– Отношение *is-a, A-Kind-Of (является)*. **A** *является* **B** для двух типов объектов **A** и **B** тогда и только тогда, когда экстенсиональное представление типа **A** есть часть экстенсионального представления **B** в любой допустимой интерпретации.
– *Отношение has, part of (имеет часть)*. Позволяет разбивать информацию по уровням детализации.

Отношения *является* и *имеет часть* определяют иерархическую структуру, где свойства понятий верхнего уровня иерархии автоматически переносятся на понятия нижнего уровня. Это позволяет избежать дублирования информации в сети.

Функциональные сети – это вычислительные модели.

Сценарии используют отношения типа *средство – результат*, *орудие – действие*.

Проблема поиска решения в базе знаний с использованием модели семантической сети сводится к задаче поиска фрагмента сети (подсети), которая соответствует поставленному вопросу. Подобного рода задачи решаются с помощью аппарата теории графов. Следует особо отметить роль фундаментальных признаков связей (рефлексивность, симметричность и транзитивность) в процессе вывода.

На рис. 9 приведён пример простой семантической сети, которая является описанием объекта *автомобиль* и ряда связанных с ним понятий.

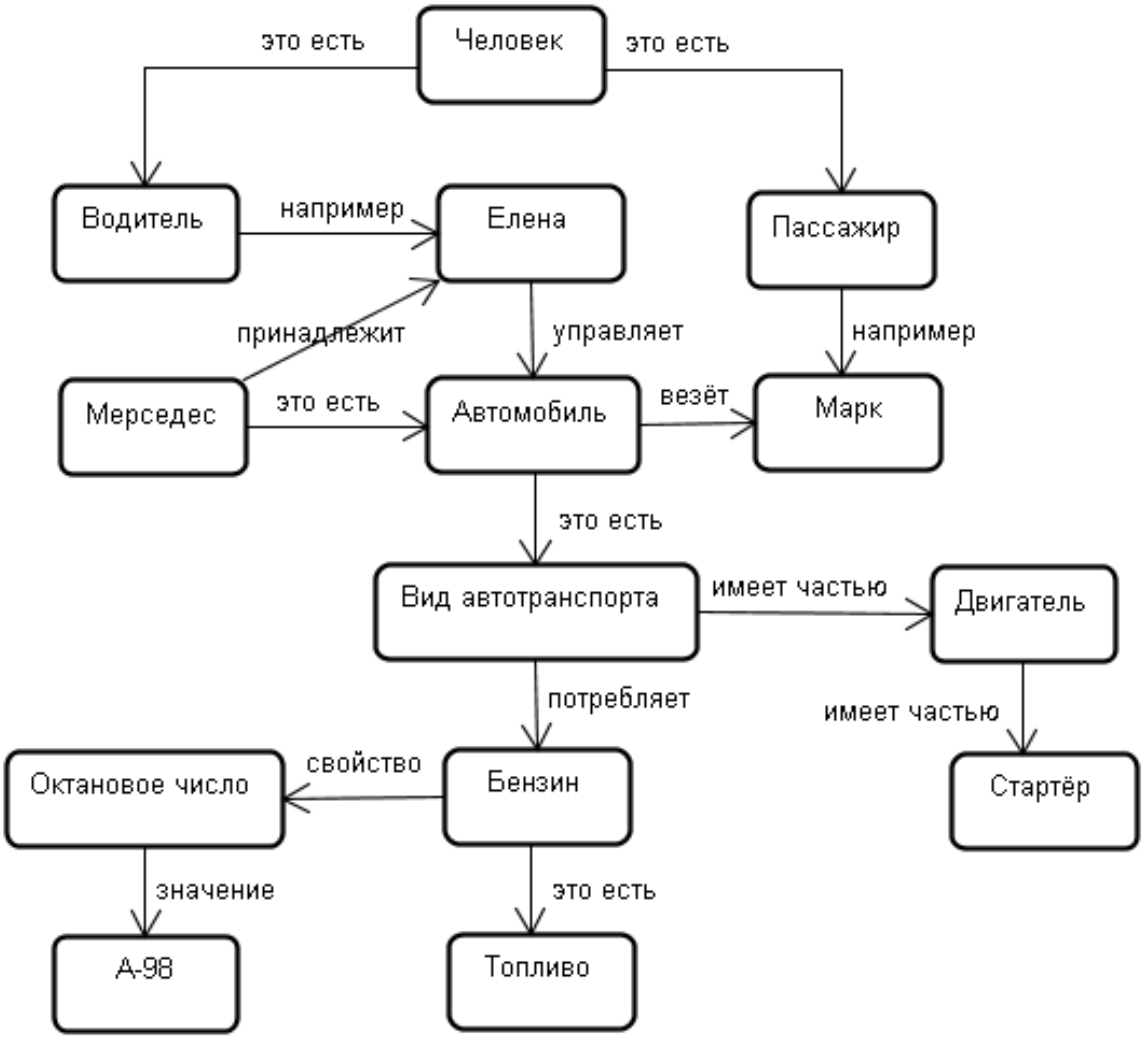

Рис. 9. Пример простейшей семантической сети

Здесь присутствует следующая цепочка понятий: *"Вид автотранспорта имеет частью двигатель"*, *"Двигатель имеет частью стартёр"*. В силу транзитивности отношения *имеет частью* можно вывести следующее утверждение *"Вид автотранспорта имеет частью стартёр"*. Аналогично

можно сделать вполне очевидные выводы: *"Мерседес является видом автотранспорта, который потребляет топливо и имеет частью двигатель"* или *"Водитель управляет автомобилем и везёт пассажира"*.

7.3.4. Фреймы

Фрейм – структура представления знаний, которая представляет собой абстрактное описание некоторого образа или ситуации.

Из описания ничего нельзя убрать, иначе оно престанет определять ту единицу знаний, для которой предназначено. Например, слово *комната* вызывает образ: жилое помещение с четырьмя стенами, полом, потолком, окнами, дверью и площадью. Если убрать окна, получится чулан, а не комната.

Различают *фреймы-образцы*, или прототипы, хранящиеся в базе знаний, и *фреймы-экземпляры*, которые создаются для отображения реальных ситуаций на основе поступающих данных.

Фрейм имеет определённую структуру (табл. 21), состоящую слотов.

Таблица 21

Структура фрейма

Имя фрейма			
имя слота	тип слота	значение слота	присоединённая процедура

Слот – составляющая фрейма, которая характеризует некоторое свойство или связь, описываемого фреймом понятия или объекта.

В качестве значения слота может выступать имя другого фрейма. Так образуют сети фреймов (рис. 10).

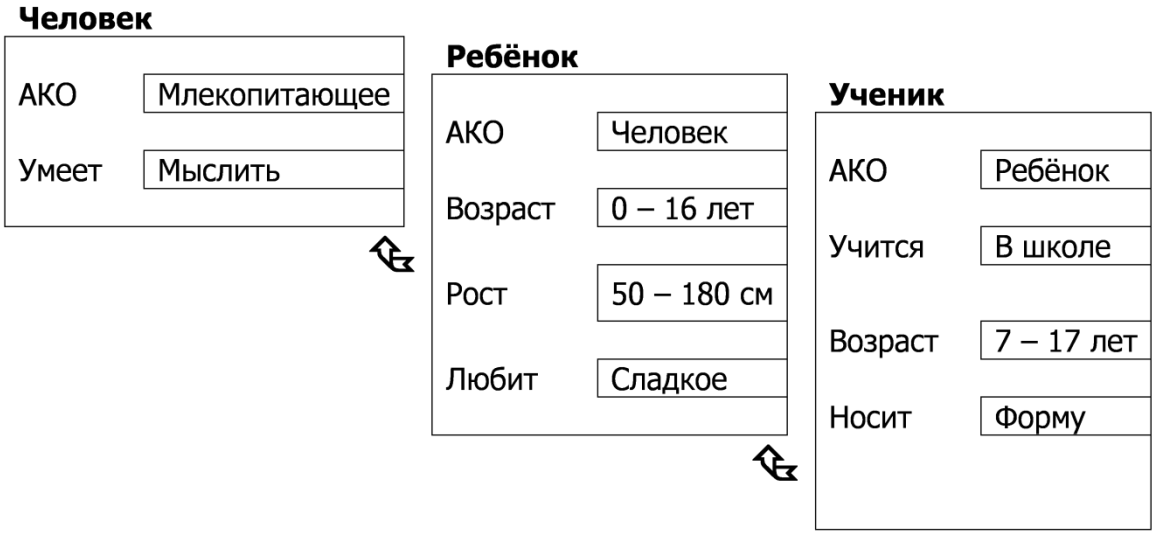

Рис. 10. Сеть фреймов

Способы получения значения слота:

- По умолчанию от фрейма-образца.
- Через наследование свойств, указанное в слоте A-Kind-Of (AKO).
- По указанной в слоте формуле.
- Через присоединённую процедуру.
- Явно из диалога с пользователем.
- Из базы знаний.

Важнейшим свойством теории фреймов является наследование свойств. Во фреймах (как и в семантических сетях) наследование происходит по связям АКО (ЭТО). Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются значения аналогичных слотов.

Фреймовая модель является достаточно универсальной, поскольку позволяет отобразить всё многообразие знаний о мире через:

- фреймы-структуры для обозначений объектов и понятий (залог, заём, акция);
- фреймы-роли (менеджер, кассир, клиент);
- фреймы-сценарии (банкротство, собрание акционеров);
- фреймы-ситуации (тревога, авария, рабочий режим устройства).