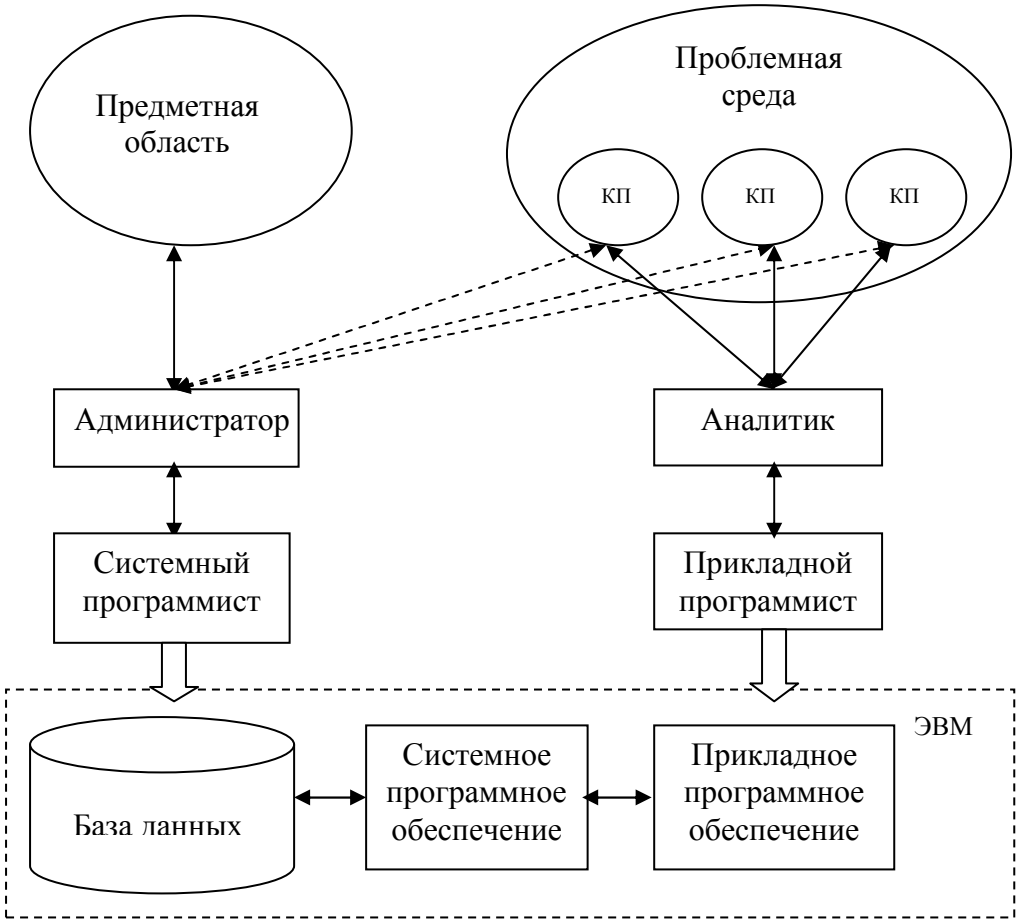
Искусственный интеллект (ИИ) - это одно из направлений информатики, целью которого является разработка аппаратно-программных средств, позволяющих пользователю-непрограммисту ставить и решать свои, традиционно считающиеся интеллектуальными задачи, общаясь с ЭВМ на ограниченном подмножестве естественного языка.

Автоматизированные банки знаний

Архитектура автоматизированного банка данных может быть представлена следующей схемой.

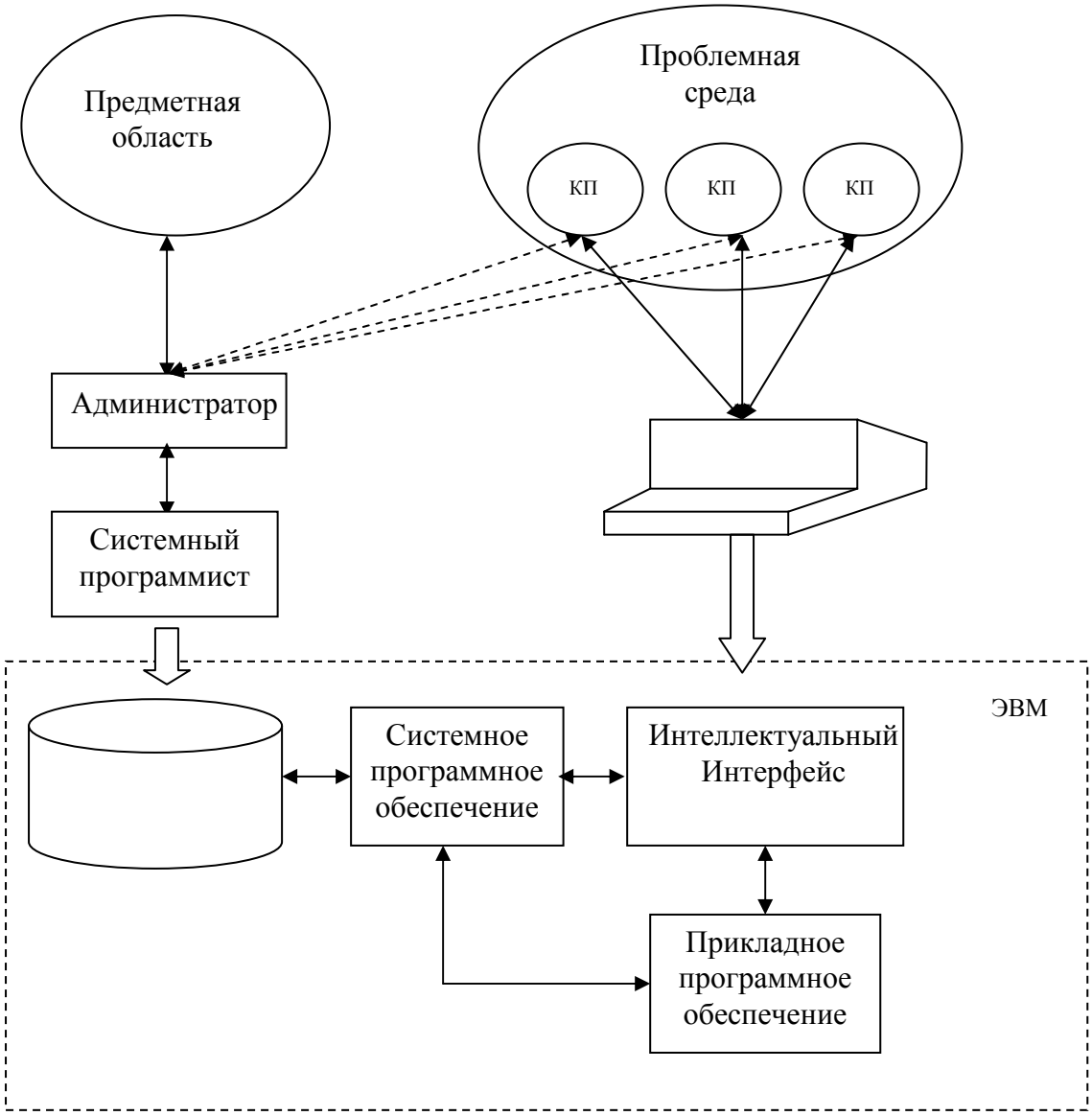


Технология решения задач в рамках такой информационной системы предполагает использование **посредников** (аналитиков и прикладных программистов) между конечными пользователями и ЭВМ, что приводит к повышению субъективного фактора при создании приложений. Действительно, качество работы программы полностью зависит от того, как аналитик понял задачу, сформулированную конечным пользователем, от адекватности и эффективности выбранных им моделей решения, от квалификации прикладного программиста в написании программного кода (реализации моделей).

Существенное снижение роли субъективной составляющей при решении задач в рамках интеллектуальных систем достигается путем исключения промежуточных звеньев между конечным пользователем и информационной системой. Ниже представлена общая архитектура автоматизированного банка знаний.

Конечный пользователь общается с системой через терминал. Между пользователем и банком информации присутствует новый посредник - Интеллектуальный Интерфейс (ИИ).

ИИ представляет собой программно-алгоритмический комплекс, автоматизирующий основные функции аналитика и прикладного программиста.



К функциям аналитика здесь отнесем:

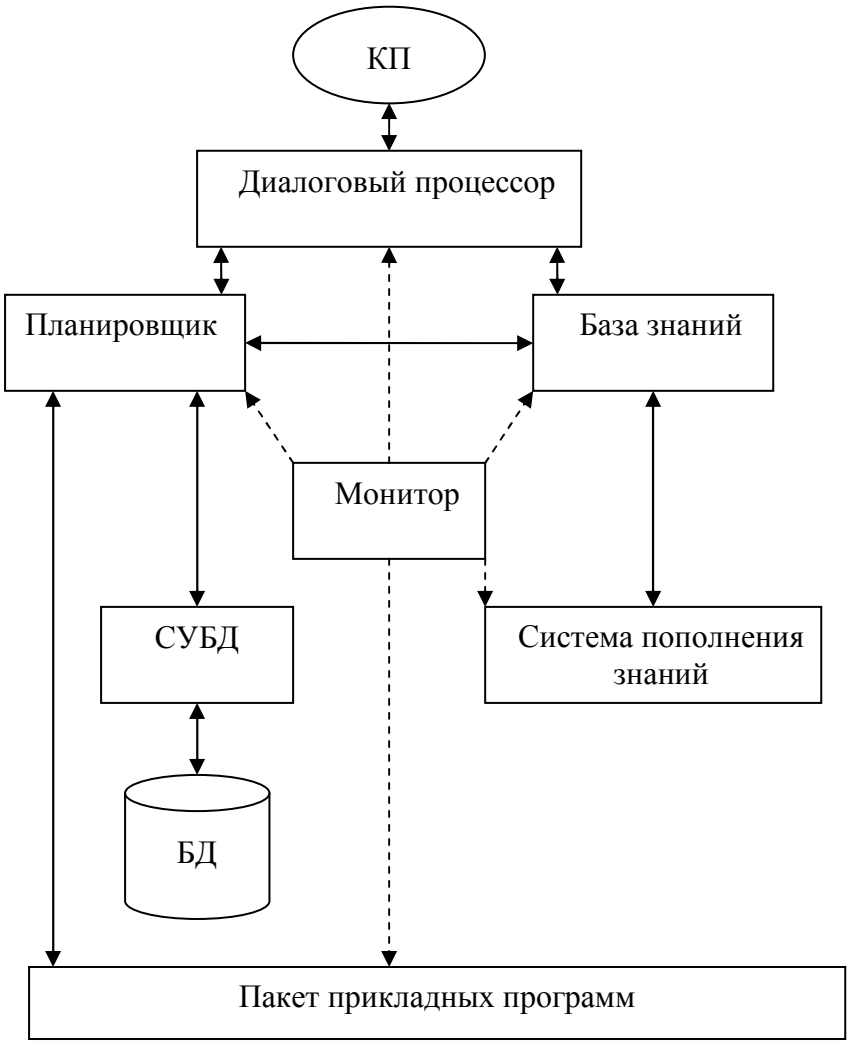
- общение с конечным пользователем при анализе постановки задачи;
- общение с конечным пользователем при интерпретации результатов полученного решения;
- перевод описания задачи на формализованный язык алгоритмов (выбор модели решения).

Функции прикладного программиста ИИ выполняет на основе библиотеки программных модулей, из которых синтезируются прикладные программы, обеспечивающие решение поставленной информационной задачи конечным пользователем.

Чтобы ИИ мог выполнить возлагаемые на него функции по общению с пользователем, «пониманию» постановки задачи, синтезу программ, обеспечивающих решение задачи, представление полученных результатов в виде, удобном для восприятия, необходимо предоставить ИИ следующую информацию:

- знания о закономерностях, существующих в предметной области (ПО), и позволяющие выводить новые факты (имеющие место в данном состоянии ПО, но незафиксированные в явном виде в базе данных (БД);
- знания о проблемной среде, в которой работает конечный пользователь;
- знания о структуре и содержании БД;
- лингвистические знания, обеспечивающие «понимание» входного языка.

АБИ, имеющий в своем составе ИИ, будем называть **банком знаний**.
Ниже приведена наиболее общая схема банка знаний.



Диалоговый (лексический) процессор - переводит сообщения конечного пользователя с естественного языка во внутреннее представление. Диалоговый процессор использует знания о входном языке, хранящиеся в базе знаний.
Планировщик (планирующая система) - преобразует поступающее на его вход формализованное описание исходной задачи в рабочую программу. Представляет собой специальный комплекс программ, во время работы постоянно контактирующих с базой знаний.
База знаний - хранит знания о предметной области и проблемной среде. Из базы знаний планировщик получает информацию:
о проблемной среде и способах решений задач;
о способе формирования рабочей программы;
о возможностях автоматического синтеза программ из набора базовых программных модулей, имеющихся в базе знаний.
Из **базы данных** через СУБД извлекается вся необходимая информация для выполнения синтезированной прикладной программы. Данные, получаемые из БД, выступают в этом случае как входные аргументы прикладной программы.
При отсутствии необходимой информации в базе данных и базе знаний знания качественного характера могут быть получены с помощью подсистемы пополнения знаний, которая реализует логические алгоритмы синтеза новых знаний. Кроме того, данная подсистема обеспечивает автоматическое пополнение и модификацию знаний, исходя из результатов

взаимодействия с конечными пользователями, то есть в соответствии с изменениями проблемной среды.

ДАННЫЕ И ЗНАНИЯ

Данные - это отдельные факты, характеризующие объекты, явления, процессы ПО, а также их свойства.

Знания - это закономерности ПО (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.
   При переходе к знаниям в память ЭВМ вводится информация о некоторой **протоструктуре** информационной единицы (например, специальное машинное слово, в котором указано, в каких разрядах хранятся сведения о фамилиях, годах рождения и т.д.). Все современные СУБД обеспечивают реализацию внутренней интерпретируемости всех информационных единиц, представляющих содержимое БД, поддерживая ведение Словарей-справочников.
2. **Структурированность**. Информационные единицы должны обладать **гибкой структурой**. Для них должен выполняться «принцип матрешки», то есть, рекурсивная вложенность одних информационных единиц в другие. Другими словами, должна существовать возможность произвольного установления между отдельными информационными единицами иерархических отношений типа **«часть-целое», «род-вид», «элемент-класс».**
3. **Связность**. В информационной базе между информационными единицами должна быть предусмотрена возможность установления связей различного типа. Прежде всего, эти связи могут **характеризовать отношения** между информационными единицами. Семантика отношений может носить **декларативный** (**описательный**) или **процедурный** характер (см. далее классификацию знаний).

Примеры. 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**» - функциональная связь (процедурное знание).

Очевидно, что между информационными единицами могут устанавливаться и иные связи, например, определяющие порядок выбора информационных единиц из памяти или указывающие на их несовместимость в одном описании.

Особенности 1÷3 позволяют ввести общую модель представления знаний - **семантическую сеть**.

4. **Семантическая метрика.** На множестве информационных единиц в некоторых случаях полезно задавать отношение, характеризующее ситуационную близость информационных единиц, то есть, **силу ассоциативной связи** (**отношение релевантности**) между информационными единицами. Отношение релевантности позволяет находить знания, близкие к уже найденным. Такое отношение дает возможность выделять в информационной базе некоторые типовые ситуации.
5. **Активность**. С момента появления ЭВМ и разделения используемых в ней информационных единиц на данные и команды создалась ситуация, при которой **данные пассивны**, а **команды активны**. Все процессы, протекающие в ЭВМ, инициируются командами, а данные используются этими командами в случае

необходимости. Для ИС эта ситуация неприемлема. Как и у человека, в ИС актуализация тех или иных действий способствуют знания, имеющиеся в системе. Таким образом, выполнение программ в ИС должно инициироваться текущим состоянием информационной базы. Появление в базе новых фактов или описаний событий, установление связей могут стать источником активизации системы.

Перечисление особенностей 1÷5 информационных единиц определяет ту грань, за которой данные превращаются в знания, а БД перерастает в БЗ.

Совокупность средств, обеспечивающих работу со знаниями, образуют СУБЗ.

Трансформация данных и знаний

При обработке на ЭВМ данные трансформируются, условно проходя следующие этапы:

1. Данные как результат измерений и наблюдений.
2. Данные на материальных носителях информации (таблицы, справочники).
3. Модели данных в виде диаграмм (ER-модель).
4. Данные в компьютере на языке описания данных.
5. Базы данных во внешней памяти ЭВМ.

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

При обработке на ЭВМ знания трансформируются аналогично данным:

1. Знания в памяти человека как результат мышления.
2. Материальные носители знаний (учебники, монографии).
3. **Поля знаний** - условное описание основных объектов ПО, их атрибутов и закономерностей, их связывающих.
4. Знания, описанные на языках представления знаний.
5. База знаний во внешней памяти ЭВМ.

КЛАССИФИКАЦИЯ ЗНАНИЙ

1. Поверхностные и глубинные знания

Как правило, в предметной области можно выделить иерархии объектов трех видов: **структурная** (строение системы), **каузальная** (**причинно-следственая**, отражающая поведение системы) и **функциональная (**связь системы с внешним миром - метасистемой). Знания, связанные с этими уровнями, могут быть определены как **поверхностные** и **глубинные** согласно следующей схеме (Рис 1.).

Пример. Предметная область - компьютер.

1. **Структурная иерархия**. Рассмотрение компьютера как совокупности иерархически организованных физических компонент (до плат и далее микросхем). Очевиден иерархический путь поиска неисправности в компьютере. При этом метод локализации неисправности (поиск дефектной микросхемы) не требует наличия у проверяющего никаких знаний об устройстве компьютера или сведений из электроники. Не являются необходимыми и данные о функционировании блоков, требуются только **поверхностные знания** о структуре системы.
2. **Причинно-следственная иерархия, или модель поведения**. Функционирование компьютера описывается в терминах «причина-следствие». Например, на самом верхнем уровне иерархии нажатие клавиши на клавиатуре (или кнопки меню) должно привести к выводу на печать. При отсутствии печати проверяется следующий уровень иерархии - работа принтера. Его действия тоже можно описать в терминах «причина-следствие». Ясно, что здесь требуются более глубокие знания, чем на структурном уровне.

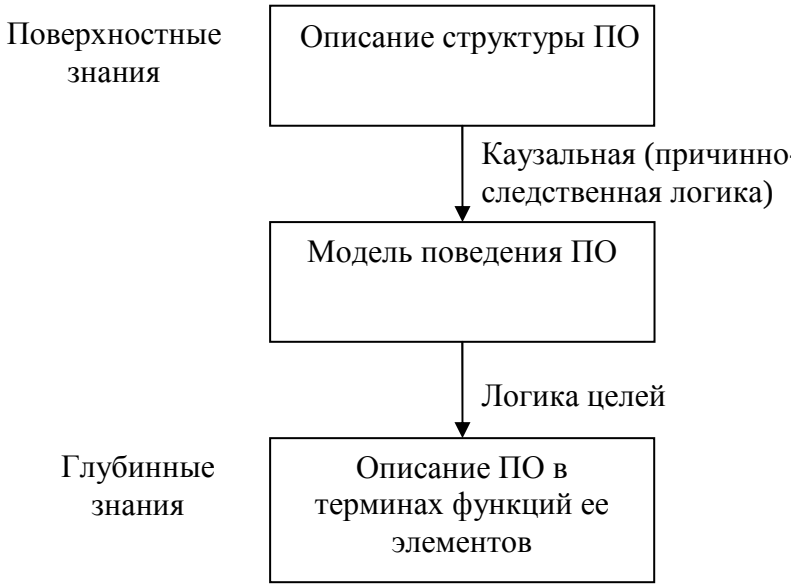


Рис 1.

3. **Функциональная иерархия**. На самом высоком уровне абстракции компьютер рассматривается как иерархия модулей или подсистем, выполняющих определенные функции. Если не выполняется основная функция компьютера - преобразование поступивших входных данных в требуемые выходные в соответствии с заданным алгоритмом, на низлежащем уровне имеется целый спектр функций (ввод данных, обработка, вывод результатов), требующих анализа. Очевидно, что проведение этого анализа требует наличия **глубинных знаний**.

Таким образом, поверхностные знания включают представление о ПО только как об иерархии составляющих ее физических компонент. Эти знания представляют собой эвристики и некоторые закономерности, устанавливаемые опытным путем и используемые при отсутствии общих теорий и методов.

Глубинные знания отражают совокупность основных закономерностей, аксиом и фактов о конкретной ПО. Глубинные знания представляют наиболее общие принципы, в соответствии с которыми развиваются все процессы в ПО, и свойства этих процессов.

Замечания.

Каждый вид иерархии соответствует определенной точке зрения на ПО.

Можно рассматривать два типа иерархий: объективно существующие в ПО и отраженные в некотором ее представлении.

Не во всех ПО может быть установлена иерархия какого бы то ни было вида.

К любой ПО применимо деление знаний на **поверхностные** и **глубинные**. Это деление носит общий характер.

2. Знания как элементы семиотической системы

В общем виде знания представляются некоторой знаковой (семиотической) системой. В любой семиотической системе выделяют три аспекта:

- **синтаксический** - описывающий внутреннее устройство знаковой системы, то есть правила построения и преобразования знаковых выражений;
- **семантический** - определяющий отношения (связи) между знаками и свойства знаков, то есть задает смысл или значения конкретных знаков;
- **прагматический** - определяющий знак с точки зрения конкретной сферы его применения либо с точки зрения субъекта, использующего данную знаковую систему.

Соответственно, выделяют 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.

Информация, с которой имеет дело ЭВМ, разделяется на **процедурную**, которая представлена программами, выполняемыми в процессе решения задачи, и **декларативную**, описывающую данных, обрабатываемые программами.

Информационную базу в процессе выполнения программ образует содержимое оперативной памяти. Машинное слово является основной характеристикой информационной базы, так как каждое машинное слово хранится в одной стандартной ячейке памяти, снабженной индивидуальным именем - адресом (свойство интепретируемости знаний). Параллельно с развитием структуры ЭВМ происходило развитие информационных структур для представления данных. Появились способы описания данных в виде векторов и матриц, возникли списочные структуры. В настоящее время в языках программирования высокого уровня используются **абстрактные** типы данных, структура которых задается самим пользователем. Следующим шагом в организации работы с декларативной информацией стало появление Баз Данных.

Концепция знаний объединила в себе многие черты процедурной и декларативной информации.

Два подхода к организации вычислительного процесса

С двумя формами представления знаний в ЭВМ - процедурной и декларативной, их относительным весом в отображении проблемной среды, взаимосвязями и ролью в процессе обработки информации тесно связана постоянная эволюция программного обеспечения.

По отношению к роли процедур и данных при обработке информации на ЭВМ можно говорить о двух подходах:

- традиционном, ориентированном на процесс выполнения программы при решении задачи; то есть цель - выполнение программы;
- ориентированном на данные; то есть единственной целью процесса обработки является получение требуемых данных.

Традиционный подход. Процесс - это выполнение программы. В каждый момент времени процесс находится в определенном **состоянии**. Состояние процесса включает в себя всю информацию, необходимую для прерывания выполнения процесса с последующим его возобновлением. Состояние процесса содержит по меньшей мере следующую информацию:

- программу;
- индикацию (адрес) команды, которая должна выполняться следующей за текущей;
- значения всех программных переменных и данных;
- состояние всех используемых УВВ.

По мере протекания процесса его состояние меняется. В целом изменяющиеся параметры характеризуются **вектором состояния** процесса, по которому осуществляется контроль процесса обработки информации и управление им.

Технология ИИ. Контроль и управление процессом обработки информации в вычислительной системе осуществляется по состоянию всей совокупности данных, то есть наличию или отсутствию в этой совокупности тех или иных данных.

Таким образом, можно сформулировать требования к общей организации процесса обработки информации в системах ИИ:

Предоставление знаний. Решение любой задачи рассматривается как предоставление потребителю информации по его запросу нужного знания, удовлетворяющего некоторой спецификации на знания, содержащиеся в запросе.

Замечание. Под термином **потребитель знания** здесь понимается как конечный пользователь, выдавший запрос с терминала, так и любой процесс, обрабатывающий знания.

Информационная потребность. Необходимость для потребителя в том или ином знании. Определяется как отсутствие у потребителя информации, необходимой для решения подзадачи в составе общей задачи.

Примечание. В целом человеко-машинная система трактуется как структура, состоящая из компонентов, каждый из которых выполняет две функции:

- решение некоторой задачи по пришедшему извне запросу;
- формирование запроса к другим компонентам на недостающее для решения этой задачи знание.

Таким образом, задача представляется как множество подзадач, выполняемых различными компонентами системы, а весь процесс решения задачи распадается на множество процессов удовлетворений запросов различных компонентов. При этом конечный пользователь трактуется как некоторый конечный потребитель знаний, информационные потребности которого инициируют работу всех процессов.

Активность декларативных знаний. Ход решения задачи в любой момент времени оценивается по состоянию системы знаний системы, в которой процедурная часть остается постоянной, а декларативная постоянно меняется.

МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Модели представления знаний образуют две группы - **эвристические** и **логические** модели.

Эвристические подходы к представлению знаний носят в большей степени характер искусства, в их использовании превалирует интуиция, опыт и мастерство разработчика.

К числу эвристических моделей относятся **фреймовые модели** представления знаний и **семантические сети**.

В основе логических моделей лежит строгое понятие **формальной системы**, то есть изобразительные средства, лежащие в основе этих моделей, имеют теоретическое обоснование. Состав логических моделей представления знаний образуют **логические** и **продукционные** модели.

Фреймовые модели представления знаний

Теория фреймов излагается ее автором Минским (1974 г) в следующем виде:

«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».

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

С каждым фреймом ассоциировано несколько видов информации; это, например, информация о том:

- как пользоваться данным фреймом,
- чего ожидать в следующий момент,
- что делать, если ожидания не подтвердятся,
- и т.д.

В целом фреймовая модель состоит из двух частей:

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

Структура фрейма

В состав фрейма входят характеристики описываемой стереотипной ситуации (**слоты**) и их значения (**заполнители слотов**). Общая организация фрейма может быть представлена в следующем виде:

имя_фрейма: имя_слота$_1$, значение_слота$_1$

имя_слота$_2$, значение_слота$_2$

…….

имя_слота$_к$, значение_слота$_к$

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

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). Говорят, что протофрейм представляет **интенсиональное** описание.

Заполненный фрейм (экземпляр, пример прототипа) называется **экзофреймом**. Говорят, что экзофрейм соответствует **экстенсиональному** представлению протофрейма.

Более подробно структуру фрейма можно представить следующей схемой:

имя_фрейма

Имя_слота	Указатель наследования	Указатель атрибутов данны	Значения слота	Присоедненная процедура
Слот 1				
Слот 2				
…….				
Слот К				

Комментарии:

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.

Важным свойством фреймовых систем является то, что они 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 должен 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).

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.

Таким образом, значением слота может быть объект произвольного типа и структуры (точнее, указатель на этот объект).

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.

Пример использования системы фреймов. Пусть имеется иерархическая система протофреймов, описывающая понятие об отчете по выполнению работ по научной теме.
Со слотами связаны следующие процедуры-демоны:

1. Слот **Автор**. Процедура «ЕСЛИ_ДОБАВЛЕНО»:

“Уведомить лицо из слота **Автор** , что ОТЧЕТ ПО ТЕМЕ из слота **Тема** ОБЪЕМОМ число страниц из слота **Объем** должен быть ПРЕДСТАВЛЕН к дате из слота **Дата представления”**.

2. Слот **Тема**. Процедура «ЕСЛИ ДОБАВЛЕНО»:

“Поместить в слот **Автор** ИМЯ РУКОВОДИТЕЛЯ ПРОЕКТА для ПРОЕКТ из слота **Тема”**

3. Слот **Дата представления**. Процедура «ЕСЛИ НУЖНО»:

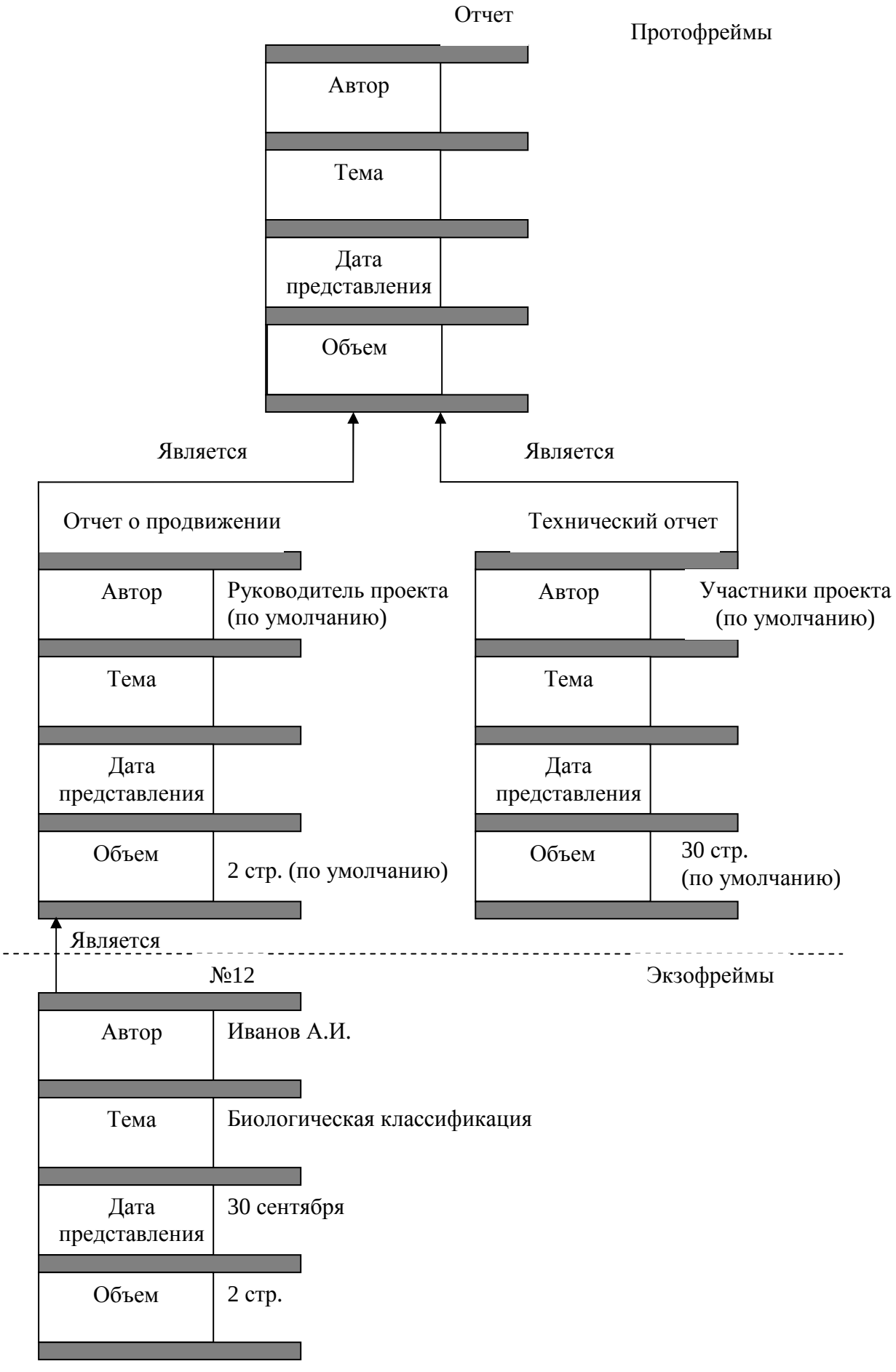
“Поместить в слот **Дата представления** либо 31 марта, либо 30 июня, либо 30 сентября, либо 31 декабря (дата ближайшего конца квартала).”

Использование организованными таким образом знания может производиться по следующему сценарию.

1. Через терминал в систему поступает запрос:

“Необходим отчет о продвижении проекта по биологической классификации”.

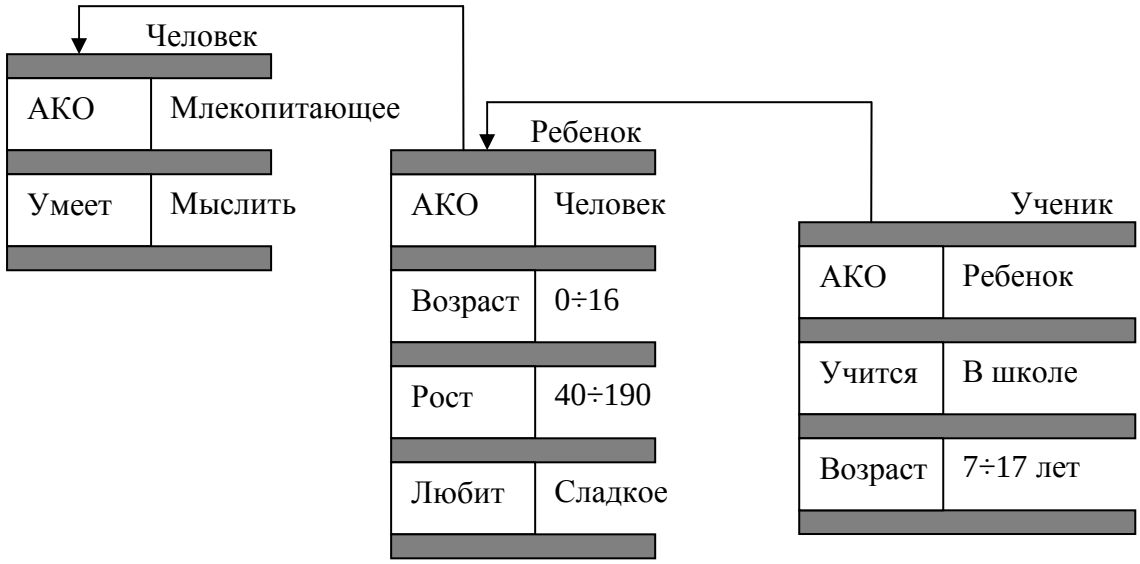
2. В списке зкзотермов отчетов о продвижении система находит пустой узел с номером 12.
3. Интерфейсная программа анализирует этот запрос и вносит строку “Биологическая классификация в слот **Тема** узла 12.
4. Запускается процедура «ЕСЛИ ДОБАВЛЕ/НО» слота **Тема**. Эта процедура осуществляет поиск в базе данных системы ФИО руководителя проекта по биологической классификации; пусть это будет Иванов А.И. Процедура вписывает эти данные в слот **Автор** узла 12.
5. Запускается процедура «ЕСЛИ ДОБАВЛЕНО» слота **Автор**. Эта процедура начинает составлять сообщение для отправки его Иванову А.И.:
   “Иванову А.И. Пожалуйста, окончите отчет о продвижении работ по проекту Биологическая классификация к ……. Предполагаемый объем отчета равен ….. страницам.”
   В процессе формирования сообщения процедура обнаруживает пустым слот **Дата представления**.
   Замечание. Формируемое сообщение представляет собой шаблон, в который вставляются значения слотов соответствующего экзотерма. Места вставок отмеченыстроками …..


Отчет
Протофреймы
Автор
Тема
Дата представления
Объем
Является
Является
Отчет о продвижении
Технический отчет
Автор
Руководитель проекта (по умолчанию)
Тема
Дата представления
Объем
2 стр. (по умолчанию)
Автор
Участники проекта (по умолчанию)
Тема
Дата представления
Объем
30 стр. (по умолчанию)
Является
№12
Экзофреймы
Автор
Иванов А.И.
Тема
Биологическая классификация
Дата представления
30 сентября
Объем
2 стр.

6. Активизируется процедура «ЕСЛИ НУЖНО» слота **Дата представления** (поскольку к слоту было обращение). Эта процедура по текущей дате и содержимому календаря в базе данных определяет, что подходит дата 30 сентября, которая и заносится в качестве значения в слот **Дата представления**.
7. Процедура «ЕСЛИ ДОБАВЛЕНО» слота **Автор** встраивает дату представления в сообщения:
   "Иванову А.И. Пожалуйста, окончите отчет о продвижении работ по проекту Биологическая классификация к 30 сентября. Предполагаемый объем отчета равен ….. страницам."
   Процедура обнаруживает пустым слот **Объем**.
8. Для слота Объем не предполагается запуск демонов. Поэтому процедура формирования сообщения обращается к родительскому фрейму для фрейма узла 12. Таким фреймом является протофрейм **Отчет о продвижении**. В этого фрейма указано значение по умолчанию 2 стр. Это значение согласно принципу наследования и заносится в слот **Объем** узла 12.
9. Процедура формирования заканчивает построение сообщения, заполнив поле объема значением слота **Объем**:
   "Иванову А.И. Пожалуйста, окончите отчет о продвижении работ по проекту Биологическая классификация к 30 сентября. Предполагаемый объем отчета равен двум страницам."

Организация логического вывода на фреймах

Заметим, что наличие иерархических связей позволяет в рамках фреймовой системы организовать, используя свойство транзитивности, логический вывод.
Пример. Пусть имеется система фреймов.



Слот **АКО (**A Kind Of**)** - это системный слот, значением которого является ссылка на родительский фрейм, из которого неявно наследуются некоторые свойства фрейма-потомка.
Вопрос: "Любит ли ученик сладкое?"
Ответ: "Да". Вывод сделан на основании наследования слота **Любит** фрейма **Ребенок**.
Вопрос: "Может ли ученик мыслить?"
Ответ: "Да". Вывод сделан на основании наследования слота **Умеет** фрейма **Человек**.

Достоинства и недостатки фреймовых моделей

Достоинства:

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:

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 фреймовской модели не являются языками, родственными естественным, а ближе к изобразительным средствам.
4. Отсутствует специальный механизм управления выводом, поэтому он реализуется с помощью присоединенных процедур.

СЕТЕВЫЕ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

В самом общем виде семантическая сеть есть 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.

Формально сетевую модель S (семантическую сеть) можно задать в виде

$S = \langle I, G_1, G_2, \ldots, G_N, R \rangle$, где

I - множество информационных единиц, представленных вершинами сети;

$G_1, G_2, \ldots, G_N$ - заданный набор типов отношений между информационными единицами;

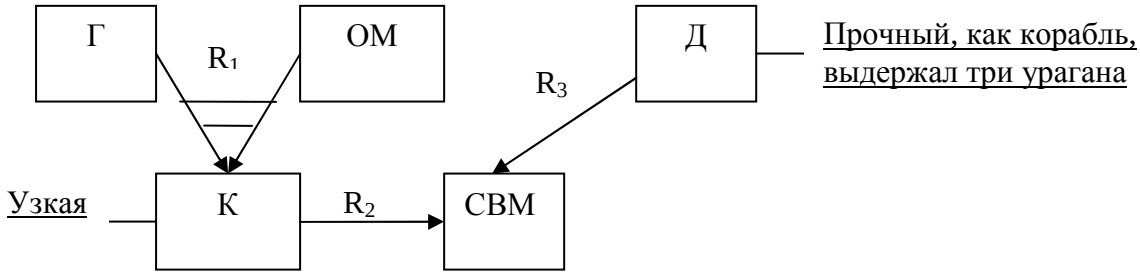
R - отображение, задающее между информационными единицами, входящими в I, связи из заданного набора типов связей.

Пример1. С помощью семантической сети моделируется следующий отрывок литературного произведения:

“Дом был построен на самом высоком месте узкой косы между гаванью и открытым морем. Построен он был прочно, как корабль, и выдержал три урагана. Его защищали от солнца высокие кокосовые пальмы, пригнутые пассатами, а с океанской стороны крутой спуск вел прямо от двери к белому песчаному пляжу, который омывался Гольфстримом.”

Введем систему понятий (информационных единиц):

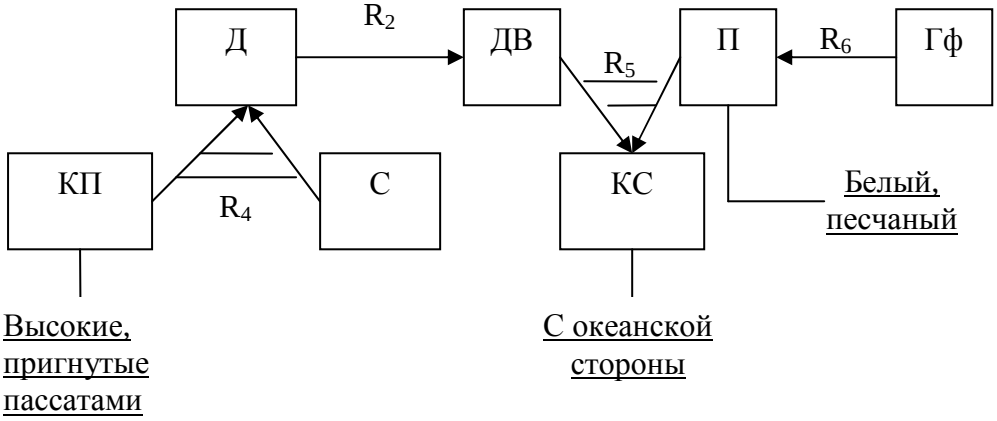
Д - дом; **СВМ** - самое высокое место; **К**- коса; **Г** - гавань; **ОМ** - открытое море; **КП** - кокосовые пальмы; **С** - солнце; **КС** - крутой спуск; **ДВ** - дверь; **П** - пляж; **Гф** - Гольфстрим.
Построим фрагмент (А) семантической сети, соответствующий первым двум фразам текста.



Где:
- отношение R_1 - “быть между”;
- отношение R_2 - “принадлежать”;
- отношение R_3 - “находиться на”;
- подчеркнутый текст соответствует описателю свойств вершины.

(А)

Остальной части текста соответствует фрагмент (В).



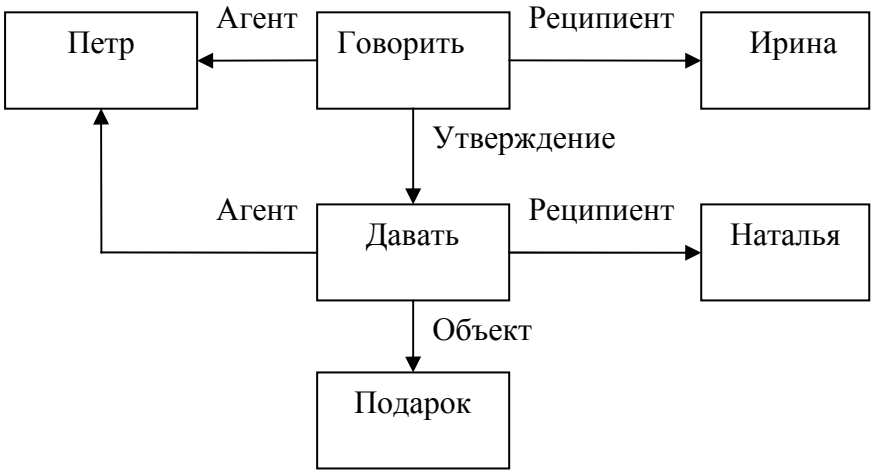
Где:
- отношение R_4 - “защищать от”;
- отношение R_5 - “соединять”;
- отношение R_6 - “омывать”.

(В)

Полное описание текста в виде семантической сети получится, если объединить фрагменты (А) и (В), соединив вершины, соответствующие понятию “Дом”.
Очевидно, что можно элементы сети, в частности, описатели вершин, представить и по-другому; например, для понятий “ураган” и “пассат” ввести отдельные вершины и включить их в общую структуру сети.
Пример 2. Семантическая сеть и грамматический разбор фраз естественного языка.
Семантические сети успешно используются в научных работах по естественному языку для представления сложных грамматических предложений:
“Петр сказал Ирине, что он отдал Наталье подарок.”
Этот пример иллюстрирует тот факт, что узлы сети могут соответствовать не только понятиям-сущностям, но и понятиям-процессам (“Говорить”, “Давать”).

Типы объектов в семантических сетях. Любая более или менее сложная модель любой предметной области отображает какие-либо **обобщенные**, **конкретные** или **агрегатные** объекты.

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.
Очевидно, что определение того или иного типа объекта условно и зависит от взгляда на ПО, другими словами, от проблемной среды.

Классификация отношений в семантических сетях. Классификацию отношений можно проводить по разным основаниям. Примеры таких классификаций приведены ниже.
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).

Наиболее часто в семантических сетях встречаются следующие **виды** отношений:

- иерархические («РОД-ВИД», «ЭЛЕМЕНТ-МНОЖЕСТВО», «ЧАСТЬ-ЦЕЛОЕ» и т.п.);
- функциональные («АРГУМЕНТ-ФУНКЦИЯ», а также связи, определяемые глаголами «влияет», «производит» и другими);
- 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»);
- каузальные (причинно-следственные;
- логические («И», «ИЛИ», «НЕ»);
- лингвистические;
- и др.

Проблема поиска решения в базе знаний типа семантической сети сводится к задаче поиска фрагмента сети, соответствующего некоторой подсети, отражающей поступивший в систему запрос.
В зависимости от смысловой нагрузки отношений семантические сети можно классифицировать по разным типам:

Классифицирующие сети. Используют отношения структуризации. Такие сети позволяют вводить в Базу Знаний иерархические отношения между информационными единицами.
Функциональные сети. Характеризуются наличием функциональных отношений (вычислительные модели).
Сценарии. Используют каузальные связи.

Иерархические сети. Являются важнейшим видом семантических сетей. Отметим два важных вида иерархических связей между информационными единицами.

1. Между двумя обобщенными объектами может существовать **родовая связь**. Наличие ее между обобщенными объектами А и В означает, что любой объект, отображаемый понятием В, отображается понятием А и существует такой объект, который отображается А, но не отображается В (А более общее, чем В). Так, например, понятие «млекопитающее» является родовым для понятия «человек».

Видовая связь является обратной к родовой. Если объект А является **родом** объекта В, то объект В является **видом** объекта А.

Ясно, что родовое понятие не охватывает всех свойств видового; видовое понятие в общем случае богаче содержанием. С другой стороны, все свойства родового понятия присущи и видовому (наследование свойств).

2. Между обобщенным и конкретным объектами может существовать связь «является представителем» (или, что аналогично, «быть элементом класса»).

Свойства, присущие обобщенному объекту, характеризуют и любой конкретный объект-представитель. Таким образом, множество свойств конкретного объекта содержит в себе подмножество свойств, которыми он наделяется как представитель тех или иных обобщенных объектов.

Замечание. Один и тот же конкретный объект может рассматриваться как представитель нескольких обобщенных объектов в одной и той же предметной среде. Так, например, конкретный объект «Третьяковская галерея» является представителем двух обобщенных объектов - «картинная галерея» и «здание в Москве». Причем объект «картинная галерея» находится в родовидовой связи с обобщенным объектом «музей», а объект «здание в Москве» нет, так как музей может размещаться в нескольких зданиях и просто на открытом воздухе, и не всякое здание в Москве является музеем.

Этот пример показывает, что к выводам, основанным на комплексном использовании родовидовых связей и связей подчиненности, следует подходить осторожно.

Организация поиска в иерархической сети.

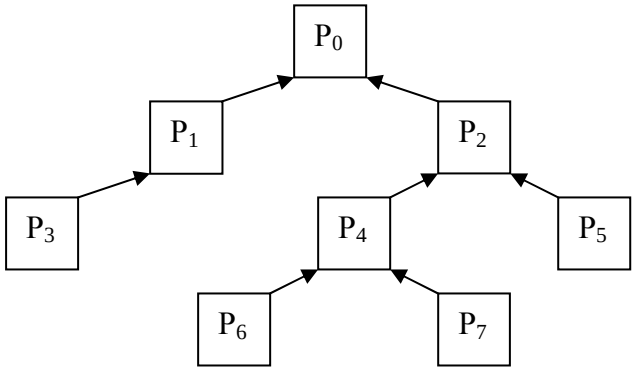
Назовем **простой** семантическую сеть, у которой для вершин не определена внутренняя структура.

Одним из основных отличий иерархических сетей от простых семантических состоит в том, что иерархические сети можно разделить на подсети (пространства) и установить отношения не только между вершинами, но и между пространствами. Очевидно, что все вершины и дуги являются элементами, по крайней мере, одного пространства.

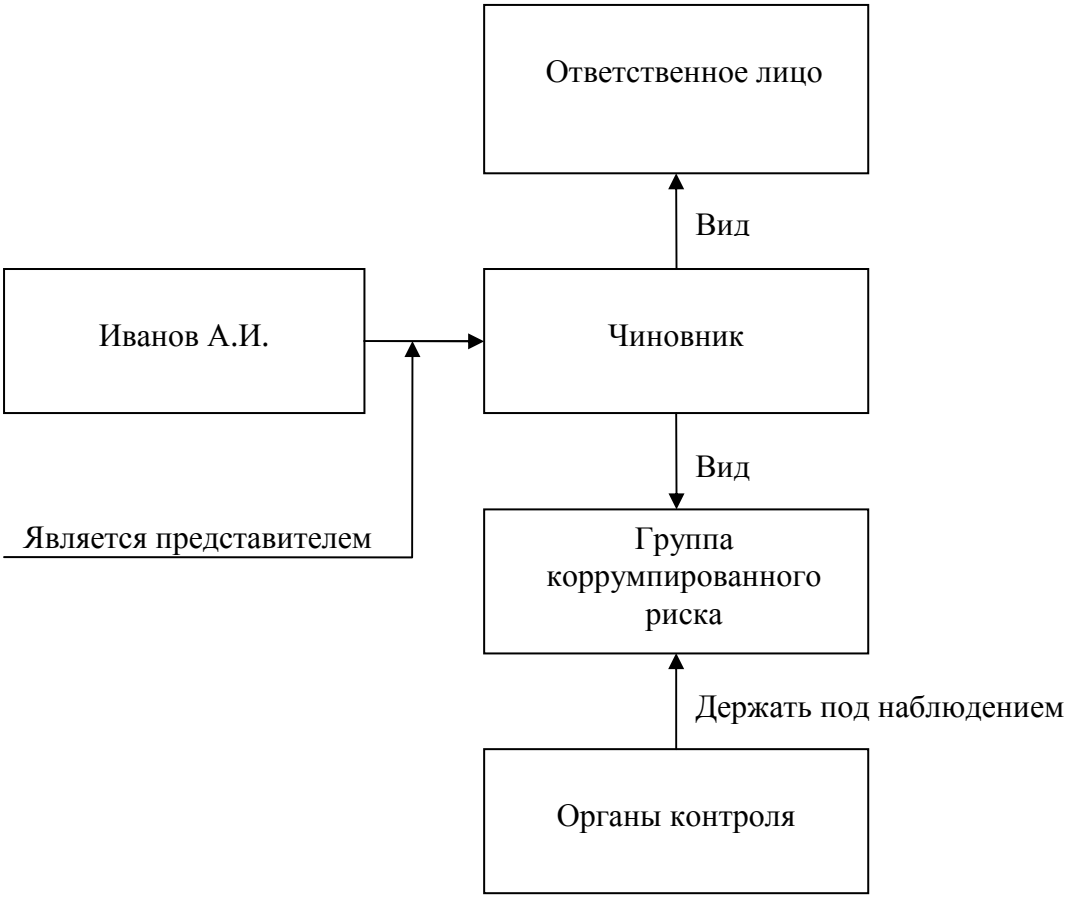
Замечание. Понятие пространства аналогично понятию скобок в математической нотации. Различные пространства, существующие в сети, могут быть упорядочены в виде дерева пространств, вершинам которого соответствуют пространства, а дугам - отношения «видимости».

Из пространства P_6 (пространство-потомок) видимы все вершины и дуги, лежащие в пространствах-предках P_4, P_2, и P_0, а остальные пространства «невидимы».

Отношение «видимости» позволяет сгруппировать пространства в упорядоченные множества - **перспективы**. Перспектива обычно используется для ограничения сетевых сущностей, «видимых» некоторой процедурой, работающей с сетью. Обычно в перспективу включают только иерархически сгруппированные пространства. Очевидно, что свойство «видимости» позволяет повысить эффективность операции поиска в сети.



Организация вывода на сети. Рассмотрим следующий пример семантической сети.



Вопрос “Находится ли под наблюдением органов контроля Иванов А.И.?”
Очевидно, что ответ зависит от семантики связи «Держать пол наблюдением». Если это связь иерархическая, то ответ положителен однозначно на основании свойства транзитивности иерархических связей. В противном случае все зависит от действия процедуры, реализующей эту связь.

Семантические сети и фреймы. Заметим, что в чисто сетевых моделях понятия **атрибута** объекта (вершины) как такового не существует, определение атрибутов объекта осуществляется через соответствующие отношения. Это:

1. усложняет структуру сети;
2. существенно затрудняет реализацию аппарата наследования.

Поэтому на практике вершины семантической сети представляются фреймами, а дуги - значениями соответствующих слотов.

Процедурные семантические сети. В целях введения единой семантики в семантические сети могут быть использованы **процедурные** семантические сети. В этом случае сеть строится на основе **класса** (понятия), а вершины, дуги (отношения) и процедуры представляются как объекты. Процедурами определяются все основные действия над вершинами и дугами (связями). В таких семантических сетях кроме декларативных возможно представление и процедурных знаний.

Достоинства и недостатки сетевой модели представления знаний

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).

ПРОДУКЦИОННЫЕ МОДЕЛИ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Продукционные модели наряду с фреймами являются наиболее популярными средствами представления знаний в ИС. Продукции, с одной стороны, близки к логическим моделям, что позволяет организовать на них эффективные процедуры вывода, а с другой стороны, более наглядно отображают знания, классические логические исчисления, что дает возможность изменять интерпретацию элементов продукции.

Формальное описание

В общем виде под **продукцией** понимается выражение вида: **(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

Ядра продукций можно классифицировать по различным основаниям. Прежде всего все ядра делятся на два больших типа: **детерминированные** и **недетерминированные**. Секвенция $\Rightarrow$ в детерминированных ядрах реализуется с **необходимостью**, а в недетерминированных - **с возможностью**.

Детерминированные ядра могут быть **однозначными** и **альтернативными.** Во втором случае 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 B_1,А С ВЕРОЯТНОСТЬЮ 0.4 B_2);

лингвистические (ЕСЛИ А, ТО ЧАЩЕ НАДО ДЕЛАТЬ B_1, РЕЖЕ B_2);

и т.п.

Возможность реализации в **недетерминированных** ядрах определяется **оценками реализации** ядра: **ЕСЛИ А, ТО ВОЗМОЖНО В**. Оценки могут быть:

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

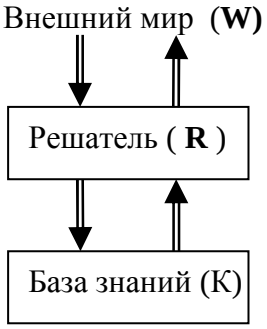
лингвистические (ЕСЛИ А, ТО С БОЛЬШОЙ ДОЛЕЙ УВЕРЕННОСТИ В);

и т.п.

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

ЕСЛИ А, ТО С ВЕРОЯТНОСТЬЮ p МОЖНО ОЖИДАТЬ В.

Классификацию ядер продукций можно провести, опираясь на типовую схему ИС.



В этой схеме используются продукции следующих типов:

1. **Aw $\Rightarrow$ B_R.** В левой части продукции указана информация, поступившая из внешнего мира, а в правой - сведения о вытекающих из этой информации изменениях в рассуждающей системе. Очевидно, что эти изменения сказываются на ходе рассуждений.

   Пример. Вечером договариваемся о воскресном выходе на природу. Утром узнаем из сводки погоды, что ожидается дождь (**Aw).** В ответ меняется весь ход рассуждений о предпочтительном варианте отдыха (**B_R).** Сама продукция могла бы выглядеть следующим образом:

   "ЕСЛИ на улице идет дождь или гроза или они ожидаются в течение дня ТО вместо прогулки лучше пойти в кино или музей"

В качестве **Aw** могут выступать: факты, имеющие место в **W**, сообщения о **W,** прямое воздействие из внешнего мира на рассуждающую систему. Вне зависимости от вида **Aw**, в правой части продукции стоят некоторые операторы, меняющие ход рассуждений.

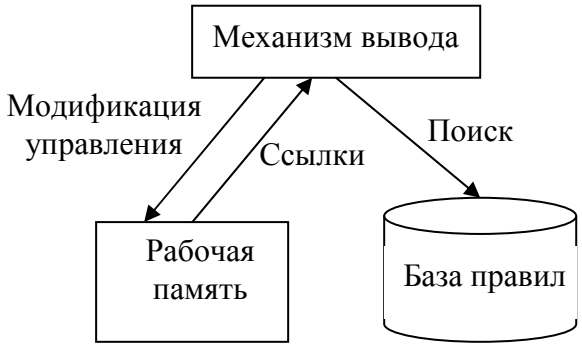
2. **Aw ⇒ Bk.** Такие продукции отражают ситуацию передачи некоторого сообщения из внешнего мира для запоминания в базе знаний.
   Пример. Приказание командира разведки: ”Все, что увидишь интересного в окрестности переправы, запомни, а потом передай через связного”
   При работе с продукциями такого типа решатель выступает в роли отделения связи, передающего сообщение от одного абонента другому.
   Замечание. Решатель может «вскрыть» корреспонденцию, воспользовавшись информацией для своих целей.
3. **Ak ⇒ Bw.** Решатель выступает в качестве отделения связи при выдаче сообщения из базы знаний во внешний мир.
   Пример. Обнаружение в базе знаний противоречивой информации. Здесь **Ak** - факт наличия противоречия, **Bw** - действия, принимаемые решателем в связи с обнаружением данного факта.
4. **A_R ⇒ Bk.** Некоторый факт, полученный решателем, передается на хранение в базу знаний.
5. **Ak ⇒ B_R**. 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. **A_R⇒ Bw.** Описание воздействий во внешний мир, которые порождает результат работы решателя.
   Пример. “Подумай, прежде чем сделать” - совет воспользоваться продукцией данного типа, а не продукцией непосредственного отклика.
8. **A_R⇒ B_R**. 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.

Очевидно, что возможны и другие интерпретации. Например, продукции типа **Aw ⇒ Bk** можно трактовать как способ описания шагов общения между пользователем и системой в диалоговом режиме. Тогда **Aw** будет трактоваться как вопрос пользователя, а **Bk** - ответ системы. При смене инициаторов диалога рассматриваются продукции типа **Ak ⇒ Bw.**

Структура продукционной системы и стратегии вывода

Структурно продукционная система состоит из трех основных компонентов.



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.
Существует две общие стратегии логического вывода: **прямой** и **обратный**.
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.**

Пример. Пусть данные, записанные в РП, представляют собой **образцы** в виде наборов символов, например, следующие текстовые данные (факты):

- “намерение - отдых”
- “место отдыха - горы”

Правила, накапливаемые в базе знаний, отражают содержимое РП. В их условной части находятся либо одиночные образцы, либо образцы, соединенные предлогами **И** и **ИЛИ,** а в правой части - образцы, которые должны быть зарегистрированы в РП (то есть, должны быть занесены в РП, если их там нет после выполнения соответствующего правила).
Пусть в базе знаний хранятся следующие правила:
Правило 1. **ЕСЛИ** (“намерение - отдых” **И** “дорога ухабистая”) **ТО** “использовать джип”
Правило 2. **ЕСЛИ** “место отдыха - горы” **ТО** “дорога ухабистая”
Общая схема работы механизма прямого вывода.

1. Механизм вывода сопоставляет образцы из условных частей правил, хранящихся в БЗ. С образцами, занесенными в РП.
   Если для какого-либо правила в РП имеются все образцы, необходимые для того, чтобы условная часть этого правила принимала значение **истина**, правило выполняется и его правая часть (факт) заносится в РП, если его еще там не было.
   В нашем примере выполниться может только Правило 2. После его выполнения содержимое РП принимает вид:
   - “намерение - отдых”
   - “место отдыха - горы”
   - “дорога ухабистая
2. Сработавшее на очередном цикле работы механизма вывода правило в следующих циклах блокируется и поиск подходящего для выполнения правила ведется среди оставшихся правил. В нашем примере для анализа на применимость остается одно Правило 1. Для его реализации в РП есть все данные, поэтому оно срабатывает и содержимое РП принимает следующий вид:
   - “намерение - отдых”

- “место отдыха - горы”
- “дорога ухабистая
- “использовать джип”

3. Очевидно, что в общем случае механизм заканчивает свою работу в случае, когда в БЗ нет ни одного применимого для текущего состояния РП правила, либо когда все правила БЗ выполнились, что имеет место в нашем примере.

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.

Пример.

1. Пусть исходная цель - “использовать джип”.
2. В условиях предыдущего примера для достижения этой цели достаточно на основании Правила 1 подтвердить факт “дорога ухабистая”.
3. Этот факт в РП отсутствует, поэтому становится новой целью (подцелью).
4. Исследуется возможность применения Правила 2. Правило применимо, подцель “дорога ухабистая” достигнута, этот факт заносится в РП.
5. В РП достаточно фактов для достижения первоначальной цели.

В случае обратного вывода условия остановки системы очевидны: либо достигается первоначальная цель, либо исчерпываются правила, применимые для достижения цели в ходе вывода.

Обратный вывод часто называют **выводом, управляемым целями.** Такая стратегия эффективна, когда цели заранее известны и их сравнительно немного.

Прямой вывод чаще применяют в системах **диагностики**, а обратный - в системах **прогнозирования**.

На сочетании ограниченного прямого и обратного выводов основывается часто применяемый на практике метод **циклического вывода**.

ЭКСПЕРТНЫЕ СИСТЕМЫ

Назначение и особенности.

Классифицируем знания специалистов как: **формализованные** (точные) и **неформализованные** (неточные).

Неточные знания характеризуются **неконкретностью**, **субъективностью**, **приблизительностью**. Знания этого рода являются результатом обобщения многолетнего опыта работы и интуиции специалистов и обычно представляют собой многообразие эмпирических (эвристических) приемов и правил.

Назовем **формализованными** задачи, основанные на **точных** знаниях, и **неформализованными** - на **неточных**.

К **неформализованным** можно отнести задачи, обладающие одной или несколькими следующими **характеристиками**:

- задачи не могут быть заданы в числовой форме;
- цели не могут быть выражены в терминах точно определенной целевой функции;
- не существует алгоритмического решения задачи;
- алгоритмическое решение существует, но его нельзя использовать из-за ограничения ресурсов вычислительной системы.

Неформализованные задачи обычно обладают следующими **особенностями**:

- ошибочность, неоднозначность, неполнота и противоречивость данных;
- ошибочность, неоднозначность, неполнота и противоречивость знаний о ПО;
- динамически изменяемые данные и знания;

- большой размер пространства состояний, не допускающий простой перебор при поиске решения.

Особенности экспертных систем. Определение экспертной системы

А) Отличие систем искусственного интеллекта (ИС) и, в частности, экспертных систем (ЭС) от систем обработки данных (СОД) состоит в том, что в этих системах используются:

- символьный, а не числовой способ представления данных;
- символьный вывод (поиск решения);
- эвристический поиск решения, а не готовое решение (алгоритм).

В) Специфика ЭС по сравнению с другими типами ИС состоит в следующем:

1. ЭС применяется для решения только практически трудных задач.
2. по качеству и эффективности решение ЭС не уступают решениям человека-эксперта.
3. решения ЭС обладают “прозрачностью”, то есть могут быть объяснены пользователю на качественном уровне (в отличие от решений, полученных числовыми, в особенности, статистическими методами). Это качество ЭС обеспечивается их способностью рассуждать о своих знаниях и умозаключениях.
4. Знания, позволяющие ЭС получать качественные и эффективные решения задач, являются в основном эвристическими, экспериментальными, неопределенными, правдоподобными. Причина этого - решаемые задачи являются не- или слабо формализованными. ЭС способны пополнять свои знания в ходе диалога с экспертом.
5. Обеспечение “дружественного”, как правило, ЕЯ-интерфейса с пользователем.
6. Мощность ЭС обусловлена в первую очередь мощностью базы знаний, и только во вторую - используемыми методами (процедурами) вывода. То есть важнее иметь разнообразные специальные знания, а не общие процедуры вывода.

Таким образом, ЭС называют машинную систему, достигающую высокого уровня работы в таких областях, в которых человеку необходимо потратить несколько лет на образование и приобретение опыта.

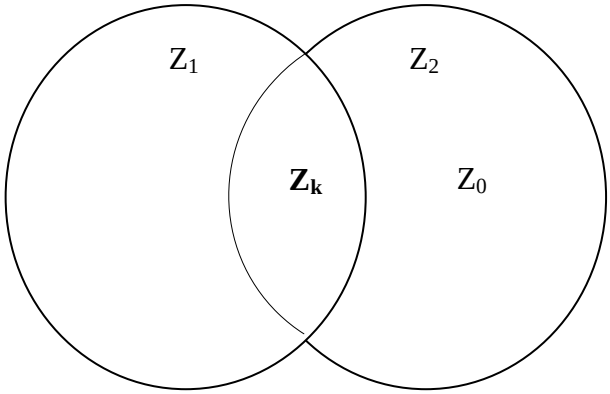
ЭС - это специализированная компьютерная система, способная к накоплению и обобщению эмпирического опыта высококвалифицированных экспертов в заданной ПО с последующей консультацией рядовых специалистов в их повседневной деятельности, включая объяснение логики получения решения.

ЭС в процессе работы способна оперативно находить ответы на вопросы о том, **что**, **где**, **когда** и **почему** случилось, как поступить в данном)текущем) состоянии, насколько эффективны будут предпринятые действия, каков будет результат текущего действия, как поступить, если ожидания не оправдаются и т.п.

В связи со сказанным ключевая роль в накоплении знаний в базе знаний системы принадлежит **инженеру знаний.** В его задачу входит выявление знаний экспертов-специалистов, их четкая формулировка (формализация), компьютеризация и введение в ЭВМ.

Состав знаний экспертной системы

В общем случае знания Z некоторой группы экспертов, заносимые в БЗ, можно представить в виде двух множеств Z_1 и Z_2 ($Z= Z_{1\cup} Z_2$).

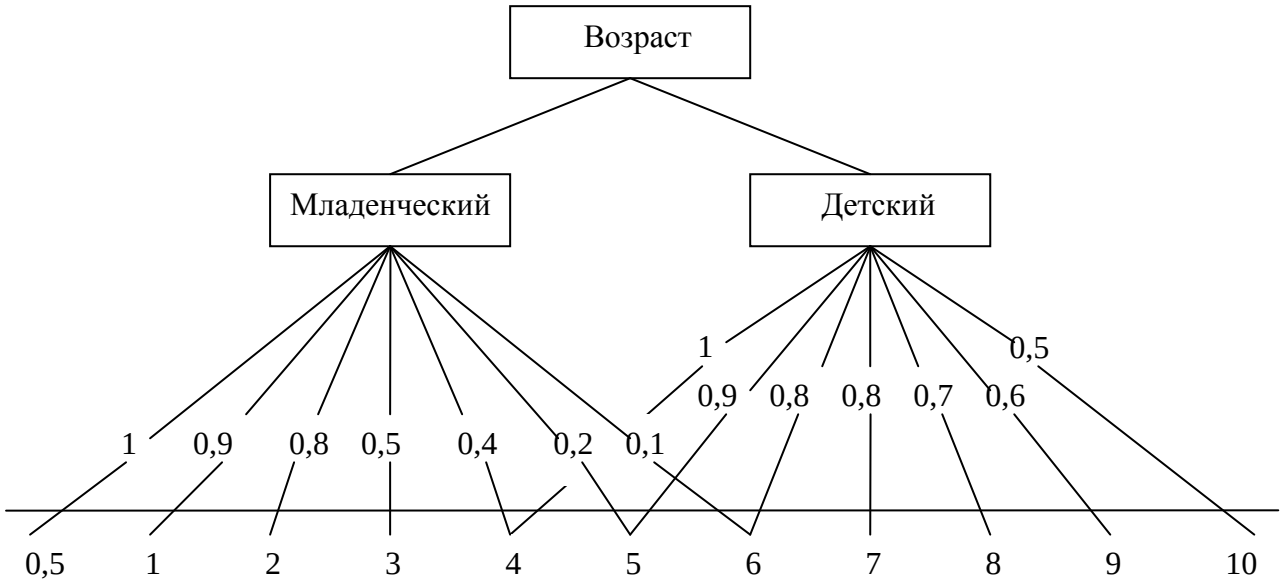


где:
Z_1 - множество знаний, общепринятых в данной ПО и содержащихся в учебниках, справочниках, монографиях по данному вопросу (то есть, «общие» знания);
Z_2 - множество знаний, приобретенных специалистами (экспертами) в данной ПО в процессе их профессиональной деятельности («личные» знания).
Множество знаний Z_2 включает в себя подмножества личных знаний отдельных экспертов.
Пересечение Z_k множеств Z_1 и Z_2 ($Z_k = Z_1 \cap Z_2$) представляет собой канонизированную часть личных знаний - то, что усвоено экспертами из специальной литературы и в чем нет расхождений между различными экспертами.
Множество Z_0, которое не имеет пересечения с Z_1, представляет собой ту часть личных знаний, которая обусловлена профессиональным опытом и интуицией экспертов.
В ЭС традиционно особую ценность имеют слабо формализованные знания типа Z_2.
В свою очередь Z_1 - обычно хорошо структурировано и сравнительно легко формализовано.
Замечание 1. В общем случае для представления Z_1 и Z_2 в базе знаний могут применяться разные модели (способы).
Замечание 2. Любая теория - это в некотором смысле идеализация ПО, а, следовательно, и ее упрощение. Эмпирические же знания конкретны, отсюда они сложнее и многообразнее, шире и глубже описывают ПО.
Ясно, что деление знаний на множества Z_1 и Z_2 не абсолютно. Во-первых, это обусловлено наличием множества Z_k, а, во-вторых, со временем наиболее плодотворные и подтвержденные практикой гипотезы переходят из Z_0 **в** Z_k, а следовательно, в Z_1.

Нечеткие знания

При попытке формализовать человеческие знания исследователи вскоре столкнулись с проблемой, затруднявшей использование традиционного математического аппарата для их представления. Существует целый класс описаний, оперирующих качественными характеристиками объектов (**много**, **мало**, **сильный**, **очень сильный** и т.п.). Эти характеристики обычно **размыты** и не могут быть однозначно интерпретированы, однако содержат важную информацию (**высокая** температура - один из возможных признаков гриппа). Кроме того, в задачах, решаемых интеллектуальными системами, часто приходится пользоваться неточными знаниями, которые не могут быть интерпретированы как **полностью истинные** или **ложные**, то есть, существуют знания, достоверность которых выражается некоторой промежуточной цифрой, например, 0,7. Возникает очевидная проблема - как, не разрушая свойства размытости и неточности, представлять подобные знания формально?
В начале 1970-х годов Лотфти Заде (США) предложил формальный аппарат **нечеткой алгебры** и **нечеткой логики**. Позднее это направление положило начало одной из ветвей искусственного интеллекта - **мягким вычислениям**.

Согласно Заде, назовем **лингвистической** переменную (ЛП), значения которой определяются набором вербальных (словесных) характеристик некоторого свойства.
Например, ЛП «рост» определяется через набор значений:
{**карликовый**, **низкий**, **средний**, **высокий**, **очень высокий**}
В свою очередь, сами значения ЛП определяются через, так называемые, **нечеткие множества** (НМ), которые, в свою очередь, определяются на некоторой **базовой числовой шкале**, имеющей размерность соответствующей ЛП.
Пусть U - базовая шкала (*универсальное множество*, *универсум*), определенная для некоторого НМ X (точнее, для ЛП, значением которой является X).
Говорят, что X **задано**, если задана **функция принадлежности $\mu_B(u)$**, $\forall u \in U$, принимающая значения на [0,1].
То есть, НМ X - это совокупность пар вида **$(\mu_x(u),u)$**, где $u \in U$.
Таким образом, для задания нечеткого множества необходимо задать **базовую шкалу** и **функцию принадлежности**.
Функция принадлежности определяет субъективную **степень уверенности** эксперта в том, что данное конкретное значение базовой шкалы соответствует определяемому НМ.
Замечание. Функцию принадлежности не следует путать с вероятностью, носящей **объективный** характер.
Пример. Определим «возраст» как ЛП со множеством значений
{**младенческий**, **детский**, **юный**, **молодой**, **зрелый**, **преклонный**, **старческий**}
Пусть для ЛП «возраст» базовая шкала представлена числовой шкалой [0,120].
Определим функцию принадлежности как степень уверенности в том, что данное количество лет можно отнести к конкретной возрастной категории (значению ЛП). Мнение эксперта по определению НМ, соответствующих значениям **младенческий** и **детский**,
может выглядеть следующим образом.



НМ «младенческий» возраст можно определить в следующих форматах:
Младенческий = {(1,0.5), (0.9,1), (0.8,2), (0.5,3), (0.4,4), (0.2,5), (0.1,6)}
Младенческий = {1/0.5, 0.9/1, 0.8/2, 0.5/3, 0.4/4, 0.2/5, 0.1/6},
Младенческий = 1/0.5+0.9/1+0.8/2+0.5/3+0.4/4+0.2/5+0.1/6.