Экспертная система (*далее по тексту — ЭС*) — это информационная система, назначение которой частично или полностью заменить эксперта в той или иной предметной области. Подобные интеллектуальные системы эффективно применяются в таких областях, как логистика, управление воздушными полетами, управление театром военных действий. Основною направленной деятельностью предсказание, прогнозирование в рамках определенного аспекта в предметной области.

Экскурс в историю экспертных систем

История экспертных систем берет свое начало в 1965 году. Брюс Бучанан и Эдвард Фейгенбаум начали работу над созданием информационной системы для определения структуры химических соединений.

Результатом работы была система под названием Dendral. В основе системы формировалась последовательность правил подобных к «IF – THEN». Информационная система не перестала развиваться и получила множество наследников, таких как ONCOIN – информационная система для диагностики раковых заболеваний, MYCIN – информационная система для диагностики легочных инфекционных заболеваний.

Следующим этапом стали 70-е годы. Период не выделялся особыми разработками. Было создано множество разных прототипов системы Dendral. Примером служит система PROSPECTOR, областью деятельности которой являлась геологические ископаемые и их разведка.  
В 80-ых годах появляются профессия – инженер по знаниям. Экспертные системы набирают популярность и выходят на новый этап эволюции интеллектуальных систем. Появились новые медицинские системы INTERNIS, CASNE.

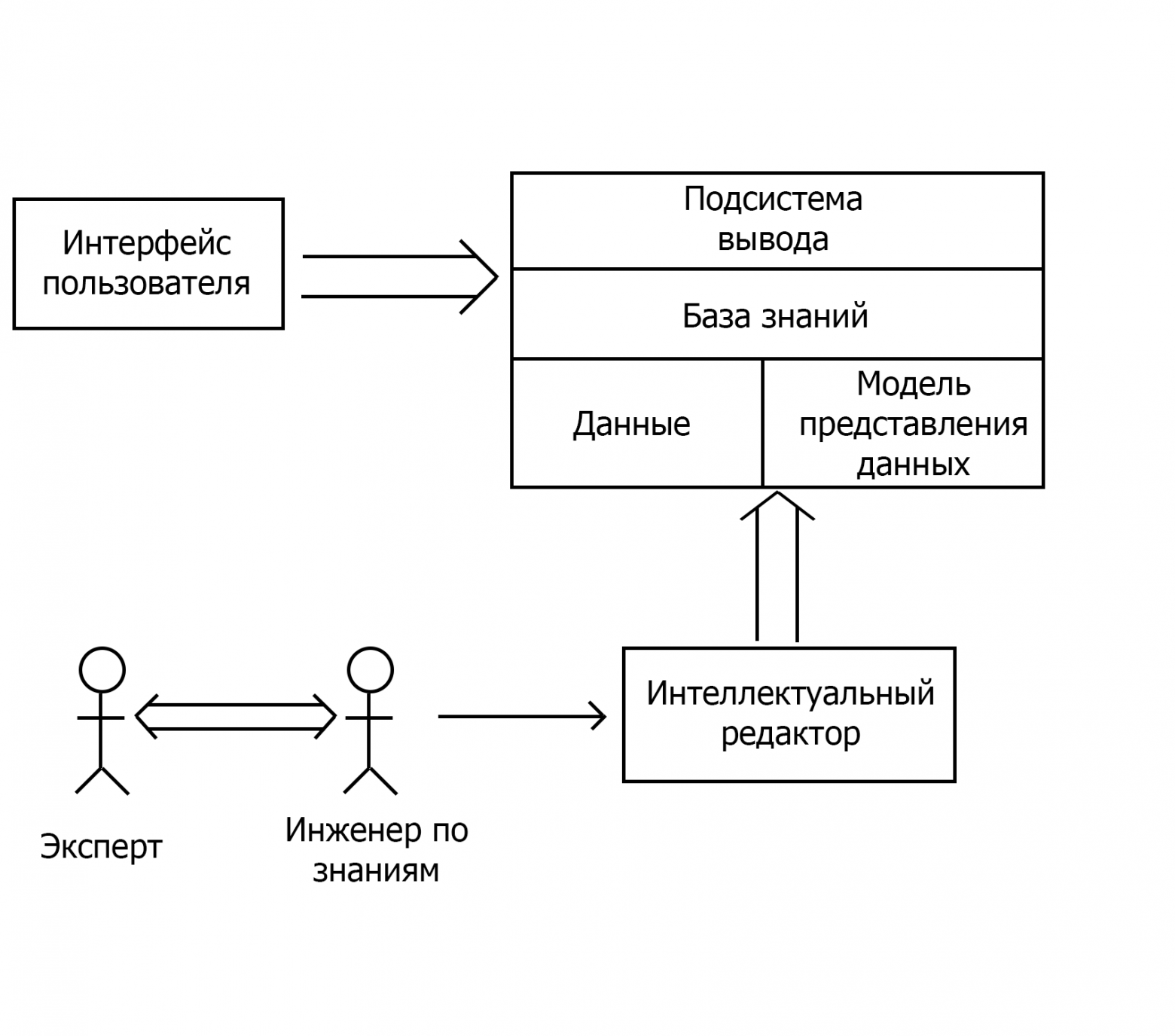
С 90-ых годов развитие интеллектуальных систем приобретает новые и новые методы и особенности. Нововведением становится парадигма проектирования эффективных и перспективных систем. Гибкость, четкость решения поставленных задач дало новое название – мультиагентных систем. Агент – фоновый процесс который действует в целях пользователя. Каждый агент имеет свою цель, «разум» и отвечает за свою область деятельности. Все агенты в совокупности образуют некий интеллект. Агенты вступают в конкуренцию, настраивают отношения, кооперируются, все как у людей.

В 21 век, интеллектуальной системой уже не удивишь никого. Множество фирм внедряет экспертные системы в области своей деятельности.

Быстродействующая система [OMEGAMON](https://www.ibm.com/us-en/marketplace/omegamon-for-cics-on-zos) разрабатывается c 2004 года с IBM, цель которой отслеживание состояния корпоративной информационной сети. Служит для моментального принятия решений в критических или неблагоприятных ситуациях.

[G2](http://dev.gensym.com/) – экспертная система от фирмы Gensym, направленная на работу с динамическими объектами. Особенность этой системы состоит в том, что в нее внедрили распараллеливание процессов мышления, что делает ее быстрее и эффективней.

Структура экспертной системы



**1. База знаний**  
Знания — это правила, законы, закономерности получены в результате профессиональной деятельности в пределах предметной области.  
База знаний — база данных содержащая правила вывода и информацию о человеческом опыте и знаниях в некоторой предметной области. Другими словами, это набор таких закономерностей, которые устанавливают связи между вводимой и выводимой информацией.

**2. Данные**  
Данные — это совокупность фактов и идей представленных в формализованном виде.  
Собственно на данных основываются закономерности для предсказания, прогнозирования. Продвинутые интеллектуальные системы способные учиться на основе этих данных, добавляя новые знания в базу знаний.

**3. Модель представления данных**  
Самая интересная часть экспертной системы.  
Модель представления знаний *(далее по тексту — МПЗ)* — это способ задания знаний для хранения, удобного доступа и взаимодействия с ними, который подходит под задачу интеллектуальной системы.

**4. Механизм логического вывода данных*(Подсистема вывода)***  
Механизм логического вывода*(далее по тексту — МЛВ)* данных выполняет анализ и проделывает работу по получению новых знаний исходя из сопоставления исходных данных из базы данных и правил из базы знаний. Механизм логического вывода в структуре интеллектуальной системы занимает наиболее важное место.  
Механизм логического вывода данных концептуально можно представить в виде <A,B,C,D>:  
А — функция выбора из базы знаний и из базы данных закономерностей и фактов соответственно  
B — функция проверки правил, результатом которой определяется множество фактов из базы данных к которым применимы правила  
С — функция, которая определяет порядок применения правил, если в результате правила указаны одинаковые факты  
D — функция, которая применяет действие.

Какие существуют модели представления знаний?

Распространены четыре основных МПЗ:

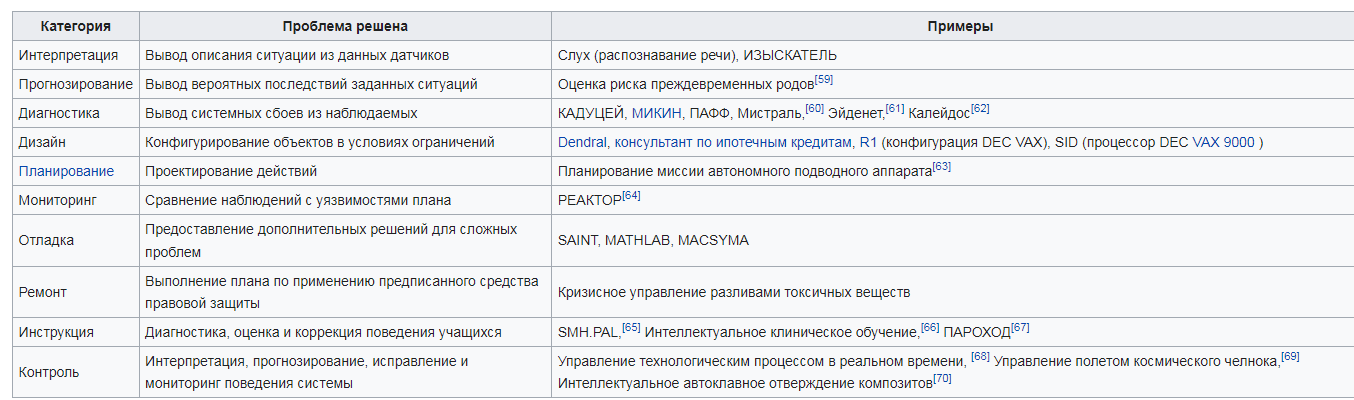
* Продукционная МПЗ
* Семантическая сеть МПЗ
* Фреймовая МПЗ
* Формально логическая МПЗ

Подводя итог преимуществам использования экспертных систем, можно выделить следующее: [[39]](https://translated.turbopages.org/proxy_u/en-ru.ru.1f153c6a-6301e92f-b63dd3d5-74722d776562/https/en.wikipedia.org/wiki/Expert_System#cite_note-:0-39)

1. Повышенная доступность и надежность: Экспертные знания могут быть доступны на любом компьютерном оборудовании, и система всегда завершает ответы вовремя.
2. Множественная экспертиза: несколько экспертных систем могут быть запущены одновременно для решения проблемы. и получить более высокий уровень знаний, чем у эксперта-человека.
3. Пояснение: Экспертные системы всегда описывают, как была решена проблема.
4. Быстрое реагирование: Экспертные системы быстры и способны решать проблему в режиме реального времени.
5. Снижение стоимости: стоимость экспертизы для каждого пользователя значительно снижается.

Наконец, можно суммировать следующие недостатки использования экспертных систем: [[39]](https://translated.turbopages.org/proxy_u/en-ru.ru.1f153c6a-6301e92f-b63dd3d5-74722d776562/https/en.wikipedia.org/wiki/Expert_System#cite_note-:0-39)

1. Экспертные системы обладают поверхностными знаниями, и простая задача потенциально может стать дорогостоящей в вычислительном отношении.
2. Экспертные системы требуют, чтобы инженеры по знаниям вводили данные, сбор данных очень сложен.
3. Экспертная система может выбрать наиболее неподходящий метод для решения конкретной проблемы.
4. Проблемы этики при использовании любой формы ИИ очень актуальны в настоящее время.
5. Это закрытый мир со специфическими знаниями, в котором нет глубокого понимания концепций и их взаимосвязей, пока эксперт не предоставит их.



1**. ПРОЛОГ – язык логического программирования**

1.1. Создание баз знаний в ПРОЛОГе

**Краткие теоретические сведения**

ПРОЛОГ (ПРОграммирование в ЛОГике) - язык логического

программирования, предназначен для решения задач из области искусственного

интеллекта. Он используется для обработки естественного языка и разработке

систем, основанных на знаниях. Методы поиска, используемые в нем,

принципиально отличаются от традиционных. Вместо детальных инструкций,

предписывающих как решать ту или иную задачу, программист на языке Prolog

уделяет основное внимание описанию задачи.

Основные конструкции ПРОЛОГа заимствованы из логики. ПРОЛОГ

относится не к процедурным, а к декларативным языкам программирования. Он

ориентирован не на разработку решений, а на систематизированное и

формализованное описание задачи с тем, чтобы решение следовало из

составленного описания.

Современной средой, в основе которой лежит язык Пролог, является

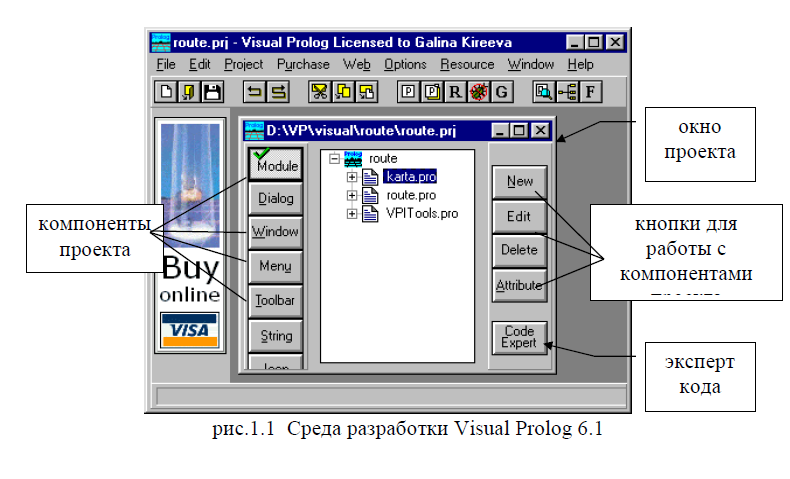
Visual Prolog. В среде Visual Prolog используется подход, получивший название

«визуальное программирование», при котором внешний вид и поведение

программ определяются с помощью специальных графических средств

проектирования без традиционного программирования на алгоритмическом

языке.



Интерфейс Visual Prolog включает: главное меню, панель инструментов,

окно проекта. Если во время последнего использования системы Visual Prolog

там был открытый проект, то система автоматически вновь откроет этот проект.

На рис.1.1 изображен внешний вид среды Visual Prolog после запуска. В

окне проекта отображаются модули открытого проекта route.prj: karta.pro,

route.pro, VPITools.pro.

Левая панель кнопок в окне проекта позволяет выбирать нужный

компонент проекта: модуль, окно, меню и т.д. С помощью кнопок правой

панели выбранный компонент можно редактировать(кнопка Edit),

удалять(кнопка Delete), а также добавлять новый(кнопка New).

Пункт меню File содержит команды для работы с файлами. Чтобы

создавать новое окно редактирования, можно использовать команду File | New.

Эта команда создаст новое окно редактора с заголовком "NONAME".

В меню Edit представлены команды, позволяющие редактировать текст

программы. Встроенный редактор системы по интерфейсу похож на обычный

текстовый редактор. Можно производить вырезку, копирование и вставку

текста, операции Отмена/Восстановление, которые можно активизировать из

меню Edit. Также меню Edit показывает "горячие клавиши", связанные для этих

действий.

Пункт меню **Project** содержит команды для работы с проектом: создать

новый, открыть, запустить и т.д. Запуск проекта на исполнение выполняется

нажатием кнопки <R> на панели инструментов (или F9, или с помощью команд

меню Project | Run).

Команды меню **Options** позволяют выполнять настройку проекта,

устанавливать необходимые параметры.

Среда Visual Prolog позволяет протестировать программу без создания

проекта. Для этого используется утилита Test Goal. Достаточно создать новый

файл, набрать текст программы и активизировать Test Goal нажатием кнопки

<G> на панели инструментов. Автономно исполняемый файл при этом не

создается. Утилита Test Goal компилирует только тот код, который определен

в активном окне редактора (код в других открытых окнах или модулях

проектов, если они есть, игнорируются). Test Goal находит *все* возможные

решения задачи и автоматически выводит значения *всех* переменных.

Программа на ПРОЛОГе состоит из предложений, которые могут быть

фактами, правилами или запросами. Факт – это утверждение о том, что со-

блюдается некоторое конкретное соотношение между объектами. Факт

используется для того, чтобы показать простую взаимосвязь между данными.

Структура факта:

<имя\_отношения>( t1,t2,...,tn) ), t1,t2,...,tn– объекты

Примеры фактов:

учится (ира, университет). % Ира учится в университете

родитель(иван, алексей). % Иван является родителем Алексея

язык\_программирования (пролог). % Пролог – это язык программиро-

вания

Набор фактов составляет базу данных. В виде факта в программе

записываются данные, которые принимаются за истину и не требуют

доказательства.

Правила используются для того, чтобы установить отношения между

объектами на основе имеющихся фактов.

Структура правила:

<имя\_правила> :- <тело правила> или

<имя\_правила > if <тело правила>

Левая часть правила вывода называется головой правила, а правая часть -

телом. Тело может состоять из нескольких условий, перечисленных через

запятую или точку с запятой. Запятая означает операцию «логическое И», точка

с запятой – операцию «логическое ИЛИ». В предложениях используются

переменные для обобщенной формулировки правил вывода. Переменные

действуют только в одном предложении. Имя в разных предложениях

указывает на разные объекты. Все предложения обязательно заканчиваются

точкой.

Примеры правил:

мать (Х, У) :- родитель (Х, У), женщина(Х).

студент (X) :- учится (X, институт); учится (X, университет).

Правило отличается от факта тем, что факт - всегда истина, а правило

является истинным, если выполняются все утверждения, составляющие тело

правила. Факты и правила образуют базу знаний.

Переменные служат для обозначения объектов, значения которых меня-

ются в ходе выполнения программы. Имена переменных начинаются с

заглавных букв или знака «\_» Область действия переменной – предложение.

Одноименные переменные в разных предложениях могут иметь разные зна-

чения.

Специальным знаком «\_» обозначается анонимная переменная, которая

используется тогда, когда конкретное значение переменной не существенно для

данного предложения. Значение анонимной переменной не выводится на

печать.

Если имеется база данных, то можно написать запрос (цель) к ней. За-

прос — это формулировка задачи, которую программа должна решить. Его

структура такая же, как у правила или факта. Существуют запросы с

константами и запросы с переменными.

Запросы с константами позволяют получить один из двух ответов: “да”

или “нет”. Если в запрос входит переменная, то интерпретатор пытается найти

такие ее значения, при которых запрос будут истинным. Запросы мо-гут быть

составными, т.е. состоять из нескольких простых запросов. Они могут

объединяться знаком ‘,’, который понимается как логическая связка И или

знаком ‘;’(логическое ИЛИ).

Простые запросы называются подцелью, составной запрос принимает

истинное значение тогда, когда истинна каждая подцель.

Как правило, программа состоит из четырех разделов.

7

DOMAINS – секция описания доменов(типов). Секция применяется, если

в программе используются нестандартные домены.

PREDICATES – секция описания предикатов. Секция применяется, если в

программе используются нестандартные предикаты.

CLAUSES – секция предложений. Именно в этой секции записываются

предложения: факты и правила вывода.

GOAL – секция цели. В этой секции записывается запрос.

**Пример 1.**

Имеется база данных, содержащая следующие факты:

родитель(илья, марина).

родитель(марина, ира).

родитель(елена, иван).

родитель(николай, ира).

родитель(ольга, алексей).

родитель(марина, саша).

родитель(сергей, иван).

Определить:

1) верно ли, что Марина является родителем Саши;

2) верно ли, что Алексей является родителем Ольги;

3) кто является ребенком Николая;

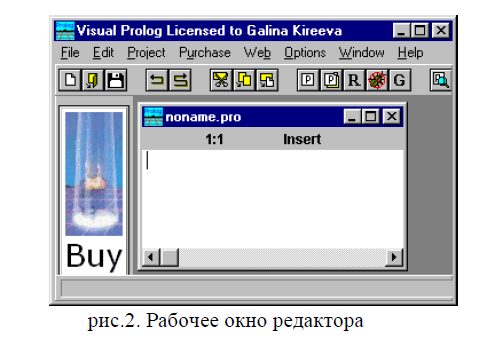
4) кто родители Ивана;

5) всех родителей и их детей.

**Решение.**

1. Запустите среду Visual Prolog. Закройте окно проекта (если оно

открыто) и откройте новый файл (**File|New**) (рис.2)



В появившемся окне наберите текст программы, содержащий разделы:

PREDICATES (описание предиката ***родитель***), CLAUSES (перечисляются

имеющиеся факты) и GOAL (запрос).

DOMAINS

имя=symbol

PREDICATES

nondeterm родитель(имя, имя)

CLAUSES

родитель(илья, марина).

родитель(марина, ира).

родитель(елена, иван).

родитель(николай, ира).

родитель(ольга, алексей).

родитель(марина, саша).

родитель(сергей, иван).

GOAL

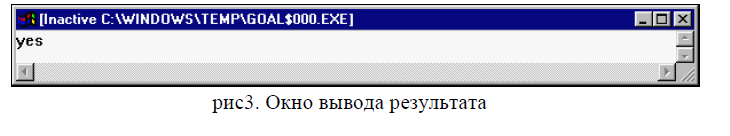
родитель(марина, саша) .

Запустите и протестируйте программу с помощью команды **Project | Test**

**Goal** (можно использовать кнопку на панели инструментов **<G>** или сочетание

клавиш **<Ctrl>+<G>**). Результат выполнения программы будет выведен в

отдельном окне

**Указание**: перед следующим запуском программы следует закрыть это

окно.

2. Для ответа на вопрос: верно ли, что Алексей является родителем

Ольги, измените запрос:

GOAL

родитель(алексей, ольга).

После запуска программы (Project | Test Goal) будет получен ответ:

no

3. Для ответа на вопрос: кто является ребенком Николая, запишите цель:

GOAL

родитель(николай, X) .

**Результат:**

X=ира

1 Solution

4. Для ответа на вопрос: кто родители Ивана, укажите запрос:

GOAL

9

родитель(X, иван), родитель(Y, иван), X<>Y.

**Результат:**

X=елена, Y=сергей

X=сергей, Y=елена

2 Solutions

5. Для определения всех родителей и их детей, запишите:

GOAL

родитель(X, Y).

**Результат:**

X=илья, Y=марина

X=марина, Y=ира

X=елена, Y=иван

X=николай, Y=ира

X=ольга, Y=алексей

X=марина, Y=саша

X=сергей, Y=иван

7 Solutions

**Пример 2**

Имеются факты вида: *родитель(имя, имя)* и *женщина(имя).*

а) составить правило ***мать*** и определить, кто мать Маши.

б) составить правило ***бабушка*** и определить, кто бабушка Ирины.

в) составить правило ***внучка*** и определить, сколько внучек у Ольги и

как их зовут

**Решение:**

DOMAINS

имя= symbol

PREDICATES

nondeterm родитель(имя, имя)

женщина(имя)

nondeterm мать(имя,имя)

nondeterm бабушка(имя,имя)

nondeterm внучка(имя,имя)

CLAUSES

родитель(марина,ирина).

родитель(елена, анна).

родитель(ольга,марина).

родитель(ольга,татьяна).

родитель(татьяна,катя).

родитель(анна, маша).

10

женщина(ольга).

женщина(маша).

женщина(ирина).

женщина(елена).

женщина(анна).

женщина(марина).

женщина(татьяна).

женщина(катя).

мать(X,Y):-родитель(X,Y),женщина(X).

бабушка(X,Z):-мать(X,Y),родитель(Y,Z).

внучка(X,Y):-бабушка(Y,X),женщина(X).

А)GOAL

мать(Кто,маша).\_\_

1.6 Создание экспертных систем средствами ПРОЛОГа

**Краткие теоретические сведения**

Экспертные системы (ЭС) - это сложные программные комплексы,

аккумулирующие знания специалистов в конкретных предметных областях и

тиражирующие этот эмпирический опыт для консультаций менее

квалифицированных пользователей.

41

При разработке экспертной системы на ПРОЛОГе база знаний

записывается в виде предикатов, которые описывают обобщённые и

конкретные сведения об объектах данной предметной области, выражают

правила определения понятий, а формулируют запросы.

В программе на ПРОЛОГе в разделе clauses перечисляются факты,

содержащие сведения, которые являются постоянными для данной предметной

области. Такая база данных является статичной, т.к. факты невозможно удалить

или добавить новые в ходе выполнения программы. Для работы с данными,

которые ЭС получает от пользователя в процессе работы и сохраняет их в

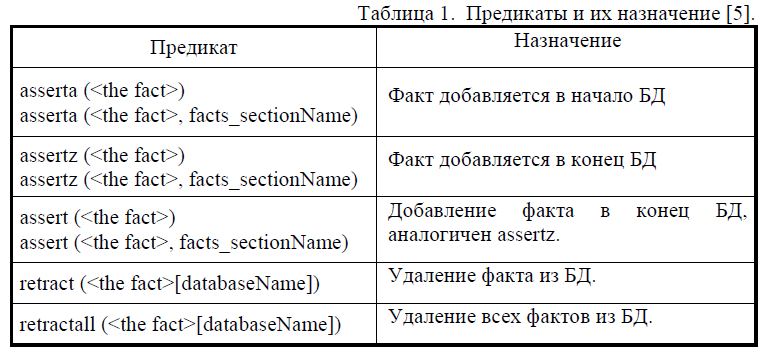
рабочей памяти, используется внутренняя (динамическая) база данных.

Динамическая база данных состоит из фактов, которые в ходе

выполнения программы можно добавлять и удалять из базы, кроме того в

ПРОЛОГе есть средства для сохранения их в файл и загрузки из файла в БД.

Во внутренней базе данных могут храниться только факты, а не правила.



Построим небольшую экспертную систему, которая будет определять

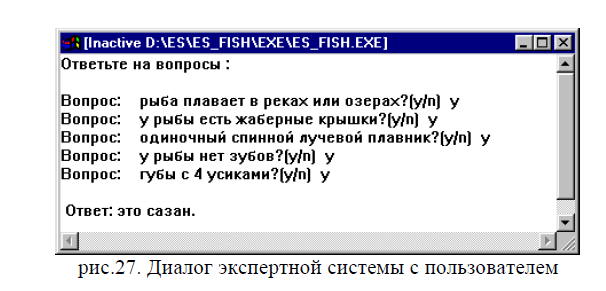
одну из нескольких рыб по признакам, указанным пользователем. Система

будет задавать вопросы и строить логические выводы на основе полученных

ответов.

Типичный диалог экспертной системы с пользователем может выглядеть

следующим образом (рис.27):



Первым шагом построения такой системы является обеспечение ее

знаниями, необходимыми для выполнения рассуждений. Программа должна во

время консультаций выводить заключения из информации, имеющейся в базе

знаний, а также использовать новую информацию, полученную от

пользователя. Поэтому минимальная ЭС должна включать:

базу знаний;

механизм вывода;

пользовательский интерфейс.

Разработку любой ЭС следует начать с исследования предметной

области. Пусть на основе бесед с экспертом были получены следующие

эмпирические правила:

1) ЕСЛИ

это отряд карпообразные и

И

у рыбы желто-золотистый окрас

И

губы с 4 усиками

ТО

это сазан

2) ЕСЛИ

это отряд карпообразные

И

у рыбы плавники с розовыми перьями

TО

это плотва

3) ЕСЛИ

спинной плавник узкий

И

у рыбы желто-золотистый окрас

И

это отряд карпообразные

TО

43

это лещ

4) ЕСЛИ

у рыбы нет зубов

И

одиночный спинной лучевой плавник

И

это костная рыба

И

это пресноводная рыба

TО

это отряд карпообразные

5) ЕСЛИ

у рыбы есть костный скелет

ИЛИ

у рыбы есть жаберные крышки

TО

это костная рыба

6) ЕСЛИ

рыба плавает в озерах

ИЛИ

рыба плавает в реках

ТО

это пресноводная рыба

Для создания базы знаний используем предикаты:

fish(symbol)

otrajd(symbol)

vid(symbol)

priznak(symbol)

Базу знаний будут составлять следующие правила:

fish("это сазан"):-

otrajd("отряд карпообразные"),

priznak("губы с 4 усиками").

fish("это плотва"):-

otrajd("отряд карпообразные"),

priznak("плавники с розовыми перьями").

fish("это лещ"):-

otrajd("отряд карпообразные"),

priznak("у рыбы желто-золотистый окрас"),

priznak("у рыбы спинной плавник узкий").

Необходимо предусмотреть, что искомой рыбы в базе знаний нет:

44

fish("Данной рыбы в базе знаний не обнаружено").

otrajd("отряд карпообразные"):-

vid("пресноводная рыба"),

vid("костная рыба"),

priznak("одиночный спинной лучевой плавник"),

priznak("у рыбы нет зубов").

vid("костная рыба"):-

priznak("у рыбы есть жаберные крышки");

priznak("у рыбы есть костный скелет").

vid("пресноводная рыба"):-

priznak("рыба плавает в реках или озерах").

Для хранения информации, полученной от пользователя, используются

предикаты **yes** и **no**, составляющие внутреннюю базу фактов. Предикат **yes**

служит для хранения фактов, соответствующих положительному ответу, а

предикат **no** – для хранения отрицательных ответов. Т.е. предикат **yes**

утверждает наличие какого-либо признака у рыбы, а **no** – отсутствие

указанного признака. Эти предикаты объявляются в разделе внутренней базы

фактов:

global facts

yes (symbol)

no (symbol)

Добавить новые факты во внутреннюю базу можно с помощью правила

add\_to\_database, состоящего из двух частей. Первая часть добавляет факты,

соответствующие положительному ответу (с клавиатуры вводится ‘y’). Вторая

часть правила добавляет факты, указывающие на отсутствие данного признака

у рыбы.

add\_to\_database (Y,'y') :- assertz (yes (Y)).

add\_to\_database (Y,'n') :- assertz (no (Y)),fail.

Необходимо предусмотреть очистку внутренней базы фактов. Для этого

создадим правило:

clear\_from\_database :- retract(yes(\_)),fail.

clear\_from\_database :- retract(no(\_)),fail.

Для проверки наличия у рыбы определенного признака создадим

правило priznak (Y):

priznak (Y) :- yes (Y),!.

priznak (Y) :- not(no (Y)),

question (Y).

45

Формулировка вопроса, ввод ответа и сохранение соответствующего

правила осуществляется с помощью правил:

answer :- fish(X),!,nl,

save("BF1.dbf"),

write (" Ответ: ",X,"."),nl.

question(Y) :-

write ("Вoпрос: ",Y,"?(y/n) "),

otvet(X),

write(X),nl,

add\_to\_database (Y,X).

otvet(C):-readchar(C).

И, наконец, правило begin, запускающее сеанс консультации:

begin :- write ("Ответьте на вопросы :"),nl,nl,

answer,

clear\_from\_database,

nl,nl,nl,nl,

exit.

Полный листинг программы выглядит следующим образом:

GLOBAL FACTS

yes (symbol)

no (symbol)

PREDICATES

fish(symbol)

otrajd(symbol)

vid(symbol)

begin

answer

question(symbol)

add\_to\_database(symbol,char)

otvet(char)

clear\_from\_database

priznak(symbol)

46

GOAL

begin.

CLAUSES

begin :-

write ("Ответьте на вопросы :"),nl,nl,

answer,

clear\_from\_database,

nl,nl,nl,nl,

exit.

answer :-

fish(X),!,nl,

save("BF1.dbf"),

write (" Ответ: ",X,"."),nl.

question(Y) :-

write ("Вoпрос: ",Y,"? "),

otvet(X),

write(X),nl,

add\_to\_database (Y,X).

otvet(C):-

readchar(C).

priznak (Y) :-

yes (Y),!.

priznak (Y) :-

not( no (Y)),

question (Y).

add\_to\_database (Y,'y') :-

assertz (yes (Y)).

add\_to\_database (Y,'n') :-

assertz (no (Y)),fail.

clear\_from\_database :- retract (yes(\_)),fail.

clear\_from\_database :- retract (no(\_)),fail.

fish("это сазан"):-

47

otrajd("отряд карпообразные"),

priznak("губы с 4 усиками").

fish("это плотва"):-

otrajd("отряд карпообразные"),

priznak("плавники с розовыми перьями").

fish("это лещ"):-

otrajd("отряд карпообразные"),

priznak("у рыбы желто-золотистый окрас"),

priznak("у рыбы спинной плавник узкий").

fish("Данной рыбы в базе знаний не обнаружено").

otrajd("отряд карпообразные"):-

vid("пресноводная рыба"),

vid("костная рыба"),

priznak("одиночный спинной лучевой плавник"),

priznak("у рыбы нет зубов").

vid("костная рыба"):-

priznak("у рыбы есть жаберные крышки");

priznak("у рыбы есть костный скелет").

vid("пресноводная рыба"):-

priznak(«рыба плавает в реках или озерах»).

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Реализуйте данную программу в среде Visual Prolog и протестируйте ее.

2. Расширьте базу знаний экспертной системы, добавив следующие правила:

1) ЕСЛИ

у рыбы есть электрические органы

И

это отряд скаты

TО

это электрический скат

2) ЕСЛИ

у рыбы на хвосте ядовитый шип

И

это отряд скаты

48

TО

это скат-хвостокол

3) ЕСЛИ

у рыбы серо-коричневый окрас

И

у рыбы коническая морда

И

это отряд акулы

TО

это гиганская акула

4) ЕСЛИ

это отряд акулы

И

рыба нападает на людей

И

у рыбы молотообразная морда

TО

это рыба молот

5) ЕСЛИ

у рыбы нет хвостового плавника

И

у рыбы тонкий длинный хвост

И

это хрящевая рыба

И

это морская рыба

TО

это отряд скаты

6) ЕСЛИ

это морская рыба

И

это хрящевая рыба

И

плавники не гибкие

И

хвост ассиметричный

TО

это отряд акулы

7) ЕСЛИ

у рыбы нет плавательного пузыря

ИЛИ

у рыбы есть хрящевый скелет

ТО

это хрящевая рыба

49

8) ЕСЛИ

рыба плавает в морях

ТО

это морская рыба

3. Протестируйте полученную экспертную систему.\_\_