Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)

Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Е.К. Пугачев

Методические указания по выполнению лабораторных работ по дисциплине «Проектирование интеллектуальных систем»

Часть 2

Исследование методов представления и обработки знаний

Москва 2018

Оглавление

Введение

Проектирование сложных систем с целью связано с задачами, отличительными признаками которых являются: принятие решений на основе неполной и «зашумленной» информации, необходимость учета опыта предыдущих решений, выработка стратегии для получения однозначно оптимального решения и др. Наиболее известными задачами являются задачи интерпретации, диагностики, контроля, прогнозирования, планирования, проектирования и т.д.

Системы с целью обладающие возможностями накапливать и получать новые знания, осуществлять рассуждения, характерные для человека, относят к классу интеллектуальных систем (ИС).

Можно выделить следующие основные черты интеллектуальных систем:

- наличие человека, который взаимодействует с ней;
- возможность накапливать и получать новые знания;
- способность осуществлять рассуждения, характерные для человека.

Основная проблема при создании интеллектуальных систем состоит в том, что во многих областях человеческой деятельности объекты, с которыми оперируют специалисты, не могут быть сведены к чисто синтаксическим объектам, характерным для математики.

Экспертные системы (ЭС) - это самостоятельное направление в области искусственного интеллекта. Мощность ЭС обусловлена в первую очередь мощностью базы знаний (БЗ) и возможностью ее пополнения, а во вторую очередь - используемыми алгоритмами достижения целей.

В данных методических указаниях рассмотрены подходы к проектированию диагностических интерактивных экспертных систем (ДИЭС), в составе которых должны быть интерфейсные, обрабатывающие компоненты, компоненты данных и знаний.

Опыт показал, что важнее иметь разнообразные специальные знания, а не общие процедуры вывода. Однако, представляемые знания экспертов в основном эвристические, эмпирические и неопределенные.

Одним из важных требований является возможность взаимодействия пользователя с ЭС. При этом предъявляются следующие требования: общение на языке близком к пользователю; линия рассуждения должна быть понятна пользователю; способность ЭС объяснять свои выводы.

При проектировании и разработке ЭС необходимо ответить на ряд следующих вопросов:

1. Какие способы представления знаний лучше всего использовать?

2. Какой механизм логического вывода лучше подходит для получения наиболее правильного решения?

3. Как пользователи различной квалификации должны взаимодействовать с ЭС?

4. Какие сервисные функции должны быть реализованы в ЭС?

5. Какие инструментальные средства лучше всего использовать при реализации ЭС и т.д.?

В системах, основанных на знаниях, правила (или эвристики), по которым решаются проблемы в конкретной предметной области, хранятся в базе знаний. Проблемы ставятся перед системой в виде совокупности фактов, описывающих некоторую ситуацию, и система с помощью базы знаний пытается вывести заключение из этих фактов. Эвристики представляют собой правила вывода, которые позволяют находить решения по известным фактам.

Структура, например, экспертной системы продукционного типа может включать семь основных модулей (см. рис. 1).

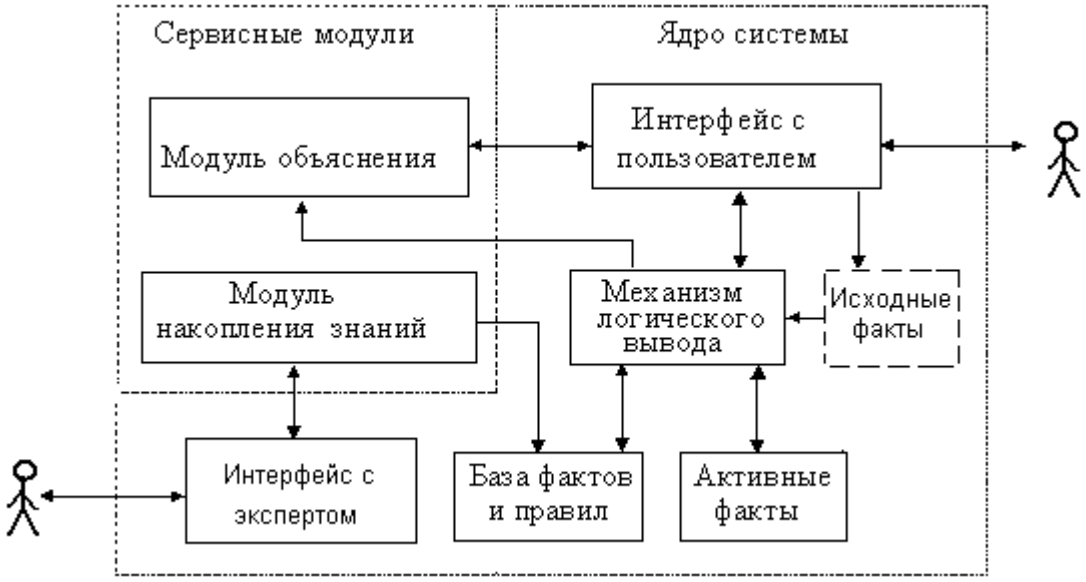


Рисунок 1 - Структура экспертной системы продукционного типа.

Этапы проектирования экспертной системы

1. Выбор предметной области. Формулировка цели. Определение задачи. Разработка алгоритма функционирования экспертной системы.

На первом шаге необходимо провести оценку предметной области на достоверность и полноту накопленных знаний. В результате данной оценки должны быть конкретизированы источники знаний.

При определении ресурсов следует уточнить, какие виды, формы представления и преобразования семантической информации будут использоваться в экспертной системе.

Определение задачи, решаемой ЭС, должно проводиться с функциональной точки зрения. Все множество задач классифицируется по следующим группам: задачи интерпретации; диагностика; контроль; прогнозирование; планирование; проектирование.

На завершающем шаге данного этапа необходимо определить структуру разработать обобщенный алгоритм функционирования ЭС.

2. Извлечение знаний.

Вначале необходимо зафиксировать на естественном языке всю предоставленную информацию из различных источников знаний и уточнить основные понятия (тезаурус). Носителями знаний могут быть: человек-эксперт; документация; магнитный носитель и др.

С целью системного подхода к процессу извлечения знаний, необходимо провести логическую декомпозицию предметной области задачи, определить количество уровней детализации, определить виды и способы представления знаний.

3. Выбор инструментальных средств проектирования.

На данном этапе необходимо сделать заключение по выбору инструментального средства, учитывая при этом результаты предыдущих этапов. В случае выбора специальной программной среды, автоматизирующей проектирование ЭС, или «оболочки» ЭС, дальнейшие этапы разработки заметно упрощаются.

4. Формализация знаний в виде машинных процедур. Построение базы знаний.

Следующим важным шагом проектирования ЭС является разработка структуры базы знаний и определение ее прикладной единицы (базового элемента или структуры). От решения этой задачи зависят: эффективность работы механизма логического вывода; дублирование и избыточность информации базы знаний; осуществление диалога между экспертной системой и пользователем; возможность

подключения блока объяснения непосредственно в ходе ведения диалога; возможность создания блока накопления знаний для пользователя не программиста и др.

В завершении данного этапа разрабатываются исходные тексты программ, реализующих базу знаний на выбранном инструментальном средстве.

5. *Разработка семантического интерфейса «ЭС - пользователь».*

Разработка интерфейса «ЭС - пользователь», основные функции которого: осуществлять преобразование сообщения из естественно-языковой формы в форму внутреннего представления и обратное преобразование; анализ и синтез сообщений пользователя и ЭС; отслеживание и запоминание пройденного пути диалога.

На выходе данного этапа должны быть разработаны возможные сценарии диалога между ЭС и пользователем в виде структурных нотаций и текстов программ.

6. Разработка механизма логического вывода.

После того, как знания представлены с помощью выбранных методов, необходимо приступить к разработке механизма логического вывода (МЛВ).

Основная задача МЛВ, например, для продукционной модели, это реализация стратегии выбора соответствующего правила, факта. В частности необходимо разработать четыре процесса: выбор активных правил и фактов; сопоставление; разрешение конфликтов; выполнение выбранного означенного правила (действие).

Основной задачей при реализации первого процесса является разработка различных схем порождения новых фактов и формирование правил, направленных на достижение целевых фактов.

После того, как было получено множество активных правил, возникает необходимость в их сопоставлении, т.е. определение какие правила максимально охватывают множество активных фактов и какие правила целесообразнее выполнять в первую очередь.

Далее необходимо разработать процесс разрешения конфликтов. Данный процесс в зависимости от способа выбора правила должен обрабатывать все возможные конфликтные ситуации.

В результате должны быть разработаны алгоритмы функционирования ядра экспертной системы.

7. *Разработка модуля объяснений.*

Назначение данного модуля - сделать ЭС «прозрачной» для пользователя, т.е. предоставить пользователю возможность понимать логику действий ЭС, дать надежную гарантию правильности полученных результатов.

8. Разработка модуля накопления знаний и манипулирования со знаниями.

Основными функциями модуля накопления знаний являются: автоматизация процесса наполнения базы знаний; актуализации базы знаний. На этом этапе разрабатываются алгоритмы функционирования модуля накоплений и экранные формы, позволяющие осуществлять операции манипулирования со знаниями применительно к выбранной структуре базы знаний.

В данной лабораторной работе применительно к заданной предметной области предлагается решить две задачи:

- Разработать модель базы знаний;
- Разработать алгоритмы обработки знаний.

При разработке базы знаний могут использоваться разные модели: продукционная, фреймовая, семантическая сеть и комбинированная. Выбор алгоритмов обработки знаний и способов их реализации зависит от поставленной цели и способа представления знаний.

Проектирование базы знаний

Процесс проектирования базы знаний зависит от выбранного способа представления знаний для конкретной предметной области задачи. При проектировании необходимо построить модели для всех видов знаний в соответствии с конкретным способом представления знаний.

Одной из важных задач на этапе проектирования систем с целью является получение полных и точных спецификаций, которые формируются на основе результатов, в первую очередь, логической декомпозиции предметной области задачи. Процесс проектирования усложняется, например из-за того, что, представляемые знания экспертов в основном эвристические, эмпирические и неопределенные.

При разработке базы знаний (БЗ) могут использоваться разные модели: продукционная, фреймовая, семантическая сеть и комбинированная.

В общем случае необходимо оценить применимость вышеуказанных моделей для конкретной задачи предметной области и впоследствии разработать формальную модель для трех видов знаний. На рисунке 2 представлена иерархия видов знаний с учетом используемых классических моделей представления знаний.

Рисунок 2 - Иерархия видов знаний.

Рассмотрим особенности видов знаний применительно к проектированию ДИЭС.

Декларативный вид знаний представляет собой факты конкретных ситуаций. Такие факты могут быть описаны заранее и включены в базу знаний на этапе создания ЭС. К декларативным знаниям можно отнести факты, которые собираются в процессе

диалога с пользователем непосредственно во время работы ДИЭС. Структура представления данного вида информации, а также способы ее обработки (считывание, модификация и т.п.), оказывает сильное влияние на организацию всех модулей ДИЭС.

Процедурный вид знаний обычно формируются на основе опроса экспертов конкретной предметной области и составляют ядро базы знаний. На их основе строится механизм логического вывода (МЛВ). Процедурные знания непосредственно связаны с декларативными знаниями, позволяют обрабатывать имеющиеся в базе знаний факты, а при необходимости генерировать новые факты. Процедурные знания показывают, как факты связаны между собой и порядок их вывода. К основным способам можно отнести прямой порядок вывода и обратный.

Управляющие знания могут представлять собой некоторый набор стратегий, чтобы можно было рассматривать альтернативные возможности получения вывода во время работы. Такие знания обеспечивают соблюдение важного принципа проектирования информационных систем как «функциональная избыточность». Соблюдение данного принципа позволит ДИЭС в случае неудачи работы МЛВ по одной стратегии, переходить к альтернативной стратегии. Управляющие знания, например в продукционных моделях, могут определять, какие из процедурных правил следует использовать для получения вывода в конкретной ситуации. К основным стратегиям при проектировании сложных ДИЭС можно отнести поиск в глубину в случае упорядоченной иерархии решений, например, в виде «дерева» и др.

Результаты процесса логической декомпозиции предметной области, которые лежат в основе проектирования ДИЭС, можно представить в виде иерархии, где в качестве компонент выступают факты, которые семантически описываю цели и подцели.

На рисунке 3 показано, как целевой факт разворачивается на множестве аспектов. В каждой конкретной ситуации целевой факт требует подтверждения или опровержения. Такая задача возложена на МЛВ. Под фактом понимают атомарную семантическую единицу, и в общем виде факт может быть представлен в виде множества связанных объектов, объединенных под одним отношением.

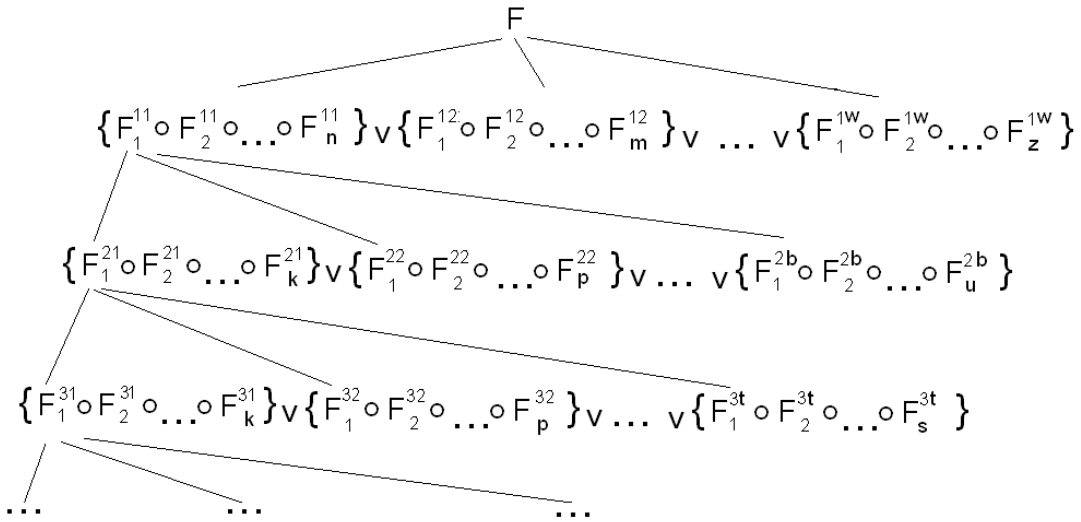


Рисунок 3 - Структура представления целевого факта.

На рисунке 3 используются следующие обозначения:

- F - целевой факт;
- $\{F_1^{11} \circ F_2^{11} \circ \ldots \circ F_n^{11}\}$ – группа (цепочка) связных фактов первого уровня;
- $\{F_1^{11} \circ F_2^{11} \circ \ldots \circ F_n^{11}\} \vee \{F_1^{12} \circ F_2^{12} \circ \ldots \circ m\} \vee \ldots \vee \{F_1^{1w} \circ F_2^{1w} \circ \ldots \circ F_z^{1w}\}$ – альтернативные группы связных фактов, где w – номер альтернативной группы фактов, n, m и z – количество фактов в группах, а ° - знак отношения (связности) между фактами.

При описании связности можно использовать импликацию, например:

$$\{F_1^{3t} \circ F_2^{3t} \circ \ldots \circ F_s^{3t}\} \rightarrow F_1^{21}$$

При разработке базы знаний необходимо определить:

- множество фактов предметной области задачи;
- основные и альтернативные цепочки связанных фактов;
- количество уровней и связей между ними;
- структуры фактов и способы объединения фактов в группы.

В общем виде факт можно представить так

$$< Q_{11}, Q_{12}, \ldots, Q_{1n} > \circ < Q_{21}, Q_{22}, \ldots, Q_{2m} >$$

где $Q_{11}, Q_{12}, \ldots, Q_{1n}, Q_{21}, Q_{22}, \ldots, Q_{2m}$ - объекты,

$\circ$ - знак отношения.

Ниже приведен пример структуры представления декларативных знаний на языке Пролог (см. листинг 1) , в частности фактов заключений разных уровней:

<Имя предиката>(<№ уровня>,<№ факта>,<Заключение на ЕЯ>).

В структуре факта предусмотрено наличие как управляющей информации (учетной), так и информации непосредственно касающейся семантики факта.

```
/* Факты заключений 1-го уровня*/
z(1,1,"Млекопитающее").
z(1,2,"Хищник").
z(1,3,"Копытное").
z(1,4,”Птица”).
/* Факты заключений 2-го уровня*/
z(2,5,"Обезьяна").
z(2,6,"Тигр").
z(2,7,"Жираф").
z(2,8,"Зебра").
z(2,9,"Страус").
z(2,10,"Пингвин").
z(2,11,"Альбатрос").
```

Листинг 1 - Пример представления фактов.

Аналогично могут быть представлены и другие виды фактов, например, исходные факты, факты различных уровней др.

При определении структур фактов для заданной предметной области задачи, которые непосредственно участвуют при выработке заключения, необходимо сделать следующее:

- из текстовой информации, представленной на естественном языке, выделить фразы, которые представляют семантические единицы-факты;
- выделить словоформы и на основе частотного принципа определить главный предикат и объекты;
- провести минимизацию количества предикатов на всем множестве выделенных фактов (это позволяет повысить универсальность и унификацию кодов системы).

Важной задачей является проектирование структур знаний, которые можно отнести к процедурному виду. На процедурные знания возложена задача проверки на срабатывание различных связанных цепочек фактов, поиск альтернативных цепочек и др.

Например, факты в продукционной модели могут быть объединены в группы с помощью правил вида:

ЕСЛИ <Цепочка связных фактов> ТО <Факт-заключение>.

Количество правил, а в больших системах и групп правил, может быть достаточно велико, следовательно, встает задача эффективности работы системы. В этом случае структура базы знаний усложняется, т.е. добавляется дополнительная управляющая информация (знания), которая связана со стратегией работы правил.

В общем виде под правилом продукции понимается выражение вида:

$$R \;\; is \;\; < i, Q, P, A \to B, N >$$

где

R - знак правила;

i - порядковый номер правила в базе знаний (имя правила), с помощью которого данное правило выделяется из множества правил;

Q - сфера применения правила (или группы правил), необходим для обеспечения эффективной работы ;

$A \to B$ - ядро продукции, где $\to$ - знак секвенции (связывает факты в логические цепочки);

P - условие применимости ядра продукции (необходимо для реализации более эффективной стратегии вывода);

N - постусловия продукции.

В итоге продукционная система может иметь большое количество групп фактов и правил разных уровней. Возникает необходимость определить стратегию (или стратегии) управления фактами и правилами, а, следовательно, определить управляющие знания (метазнания).

Другим способом объединения фактов в связанные (семантические) группы является использование фреймов.

Фрейм - это специальные информационные структуры для представления стереотипных знаний об объектах окружающего мира. Каждый фрейм имеет имя, которое представляет факт.

В наиболее общем виде формально фреймом называют структуру представления знаний следующего вида

$$F\ is \left\{ \begin{array}{l} N\,(u_1, q_1, p_1) \\ \quad (u_2, q_2, p_2) \\ \quad \cdots\cdots\cdots \\ \quad (u_k, q_k, p_k) \end{array} \right\}$$

где F - знак фрейма; is - предикат "быть", "являться";

N - имя фрейма; u_k - имя слота; q_k - значение слота; p_k - имя процедуры.

Из вышесказанного следует, что фреймы позволяют представлять связанные группы утверждений (фактов) различных уровней и создавать специальные фреймы верхнего уровня с целью управления фреймами нижнего уровня.

Во время разработки фреймовой модели могут быть использованы декларативные фреймы, которые представляют собой структуру, объединяющую связную группу фактов, которые, как правило, закладываются в БЗ на этапе проектирования. Также в модели могут

быть использованы процедурные фреймы, основным отличием которых от декларативных фреймов является то, что значения слотов определяются во время работы МЛВ.

При проектировании фреймов разных уровней и видов необходимо учитывать:

- имена слотов фрейма и значения должны быть семантически проработаны и способствовать достижению цели (подцели), которую представляет имя фрейма;
- при проектировании процедурных и управляющих фреймов необходимо обеспечить, чтобы количество слотов во фрейме не зависело от количества фактов конкретной ситуации.

Значениями слотов в сложных ДИЭС могут быть: числа, математические соотношения, тексты на естественном языке; программы, правила вывода или ссылки на другие слоты данного фрейма или других фреймов и др.

Из вышесказанного следует, что фреймы позволяют представлять связанные группы утверждений (фактов) различных уровней и создавать специальные фреймы верхнего уровня с целью управления фреймами нижнего уровня.

Кроме продукций и фреймов существует способ представления фактов и групп связанных фактов в виде семантической сети, которая в первую очередь отображает декларативные знания.

В сложных ДИЭС используют иерархические семантические сети, которые включать узлы с собственной внутренней структурой, а, следовательно, позволяют строить иерархии групп связных фактов. Такие сети можно разбить на подсети, т.е. на несколько уровней (пространств).

Семантическая сеть - это модель представления знаний, в основе которой находится понятие сети, образованной с помощью узлов (точек, вершин) и дуг (связей). Узлы представляют собой сущности, в качестве которых могут выступать: объекты, события, процессы и явления. Дуги описывают отношения между сущностями.

Иерархические семантические сети позволяют включать узлы с собственной внутренней структурой, а следовательно строить иерархии групп связных фактов. Такие сети можно разбить на подсети, т.е. на несколько уровней (пространств).

На рисунке 4 представлен фрагмент иерархической семантической сети.

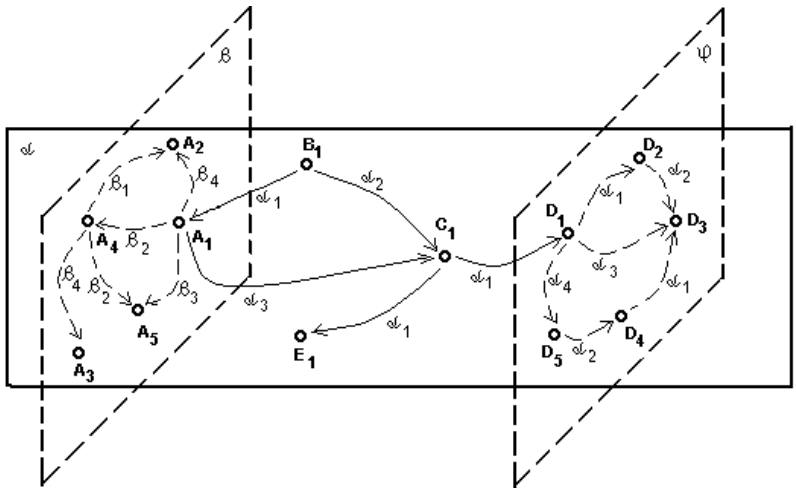


Рисунок 4 - Фрагмент иерархической семантической сети.

Элементы A_1, B_1, C_1, D_1, E_1 - являются сущностями пространства α , а $\alpha_1, \alpha_2, \alpha_3$ - отношения между сущностями пространства α . Сущности A_1 *и* D_1, являясь сущностями пространства α , имеют собственную внутреннюю структуру (подсеть), изображенную соответственно в пространствах β *и* φ .

В настоящее время существуют языки представления знаний (ЯПЗ) для реализации продукционной, фреймовой модели и семантической сети. В таких языках имеется возможность реализовать модели представления знаний, а также алгоритмы обработки знаний различных видов и уровней.

Данные языки позволяют реализовать проекты, полученные на основе следующих видов декомпозиции: логической, функциональной, продукционной, инвариантной, объектной и др.

Проектирование механизма вывода

Механизм логического вывода (МЛВ) выполняет следующие функции:

- формирование и обработка активных фактов конкретной ситуации;
- определение порядка выбора и применения фактов и правил.

Механизм вывода можно представить в виде четырех последовательных процессов:

- выбор активных правил и фактов;
- сопоставление (определяется какие правила выполнять в первую очередь);
- разрешение конфликтов;
- выполнение выбранного означенного правила (действие).

В общем случае выделяют два порядка механизма вывода: прямой и обратный. Управление прямым выводом осуществляется проще, чем управление обратным выводом. Прямой порядок вывода строится от активных фактов к заключению, т.е. по известным фактам отыскивается заключение, которое из этих фактов следует. При обратном порядке вывода заключения просматриваются последовательно до тех пор, пока не будут обнаружены факты конкретной ситуации, подтверждающие какое либо из заключений, т.е. путем подбора подходящих фактов под имеющееся заключение.

В некоторых ЭС вывод основывается на сочетании вышеприведенных подходов - обратного и ограниченного прямого. Такой комбинированный метод получил название циклического.

В ЭС, база знаний которой насчитывает сотни правил возникает необходимость использования некоторой стратегии управления выводом, позволяющей структурировать процесс вывода и минимизировать время поиска решения.

К числу таких стратегий относятся:

- поиск в глубину;
- поиск в ширину:
- разбиение на подзадачи;
- альфа-бета алгоритм и др.

Идея поиска в глубину состоит в том, что при выборе очередной подцели в пространстве состояний предпочтение стремятся отдать той, которая соответствует следующему, более детальному, уровню описания задачи. Такой механизм встроен в систему Пролог. Поиск решений в глубину наиболее адекватен рекурсивному стилю

программирования. Обрабатывая цели, пролог-система сама просматривает альтернативы именно в глубину.

В противоположность поиску в глубину стратегия поиска в ширину предусматривает переход в первую очередь к подцели того же уровня.

Например, при поиске в глубину ЭС, сделав на основе известных симптомов предположение о наличии определенного заболевания, будет продолжать запрашивать уточняющие признаки и симптомы этой болезни до тех пор, пока полностью не отвергнет выдвинутую гипотезу. При поиске в ширину, напротив, система вначале проанализирует все симптомы, находящиеся на одном уровне пространства состояний, даже если они относятся к разным заболеваниям, и лишь затем перейдет к симптомам следующего уровня детальности.

Стратегия разбиение на подзадачи состоит в том, что в исходной задаче выделяют подзадачи, решение которых рассматривается как достижение промежуточных целей на пути к конечной цели. Такая стратегия хорошо зарекомендовала себя в диагностической ЭС определения неисправности в автомобиле. Вначале выявляется отказавшая подсистема (например, система электропитания), затем на следующем уровне уточняется (например, неисправен аккумулятор) и на последнем шаге выдается причина неисправности (например, нет контакта или аккумулятор разряжен).

Альфа-бета алгоритм. Задача сводится к уменьшению пространства состояний путем удаления в нем ветвей, не перспективных для поиска успешного решения. Поэтому просматриваются только те вершины, в которые можно попасть в результате следующего шага, после чего неперспективные направления исключаются из дальнейшего рассмотрения. Например, если цвет объекта, который мы ищем не красный, то его бессмысленно искать среди красных объектов. Такой подход нашел широкое применение в играх, в шахматных системах. Данный подход используется для повышения эффективности поиска решений в продукционных системах.

Механизм логического вывода в продукционных системах

В общем виде продукционная система и МЛВ представляется:

$$S = (F , R , I)$$

где F – множество фактов, R – множество правил, I - интерпретатор.

В свою очередь интерпретатор представляется выражением:

$$I = (V , M , C , W)$$

где V - процесс выбора из множества F и R активных фактов F_a и R_a;

M - процесс сопоставления; C - процесс разрешения конфликтов;

W - процесс выполнения выбранного означенного правила (действие).

Указанный перечень процессов реализуют непосредственно механизм логического вывода.

Ниже приведен пример поиска маршрутов для графа (см. рис.5), в котором используется стратегия поиска решений в глубину и процессы сопоставления и выполнения.

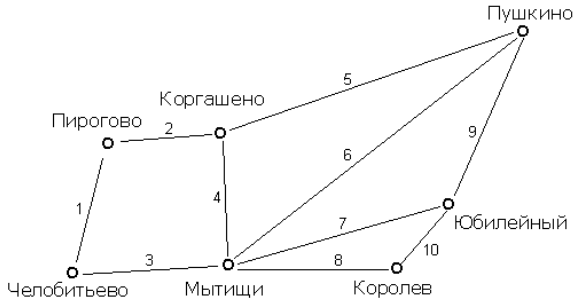


Рисунок 5 - Граф примера поиска маршрутов.

Ниже приведено два варианта реализации кода программы, которая выдает все решения, как добраться из одного населенного пункта в другой (см. листинги 2а и 2б). Отличие одной программы от другой в том, что первая программа написана в функциональном стиле, а другая в предикатном стиле и дополнительно подсчитывает длину пути. Код базы данных является общим и представлен в листинге 2в.

Функциональный стиль (без подсчета длины пути)	Предикатный стиль (с подсчетом длины пути)
implement main open core,console domains s = string. u =unsigned. class facts - graph p:(s Точка1, s Точка2, u Дуга,u Расстояние). class predicates start: (s, s) -> string* nondeterm. path: (s, s, string*) -> string* nondeterm. ed: (s, s, u, u) nondeterm (i,o,o,o). clauses ed(X, Y, N, R):- p(X, Y, N, R); p(Y, X, N, R). start(Begin, End) = list::reverse(path(Begin, End, [Begin])).	implement main open core,console domains s = string. u =unsigned. class facts - graph p:(s Точка1, s Точка2, u Дуга,u Расстояние). class predicates start: (s, s, string* [out], u [out]) nondeterm. path: (s, s, string*, string* [out], u, u [out]) nondeterm. ed: (s, s, u, u) nondeterm (i,o,o,o). clauses start(Begin, End, list::reverse(P), R):- path(Begin, End, [Begin], P, 0, R). ed(X, Y,N, R):- p(X, Y, N,R); p(Y, X, N,R).

```
    path(End, End, P) = P:- !.
    path(V, End, SumP) = path(NextV, End, [NextV |
SumP]):-
        ed(V, NextV, _,_),
        not(list::isMember(NextV, SumP)).

    run():-  file::consult("db.txt", graph),
        List = start("Юбилейный","Пирогово"),
            write(string::concatWithDelimiter(List, " -
> ")), nl,
        fail;
        _ = readLine().
end implement main

goal
    console::run(main::run).
```

```
    path(End,End, P, P, R, R):- !.
    path(V, End, SumP, P, SumD, R):-
        ed(V, NextV, _,Rd),
        not(NextV in SumP),
path(NextV, End, [NextV | SumP], P, SumD + R
d, R).
    run():-  file::consult("db.txt", graph),
        start("Пирогово", "Юбилейный", P, D),
write(string::concatWithDelimiter(P, " -
> "), " : ", D), nl,
fail;
        _ = readLine().
end implement main
goal
    console::run(main::run).
```

Листинги 2а и 2б – Коды файлов поиска маршрутов.

```
clauses
p("Челобитьево","Пирогово",1,4).
p("Пирогово","Коргашено",2,3).
p("Челобитьево","Мытищи",3,4).
p("Коргашено","Мытищи",4,3).
p("Коргашено","Пушкино",5,8).
p("Мытищи","Пушкино",6,10).
p("Мытищи","Юбилейный",7,6).
p("Мытищи","Королев",8,4).
p("Пушкино","Юбилейный",9,6).
p("Королев","Юбилейный",10,2).
```

Листинг 2в – Содержимое файла "db.txt".

```
Юбилейный -> Мытищи -> Пушкино -> Коргашено -> Пирогово
Юбилейный -> Мытищи -> Челобитьево -> Пирогово
Юбилейный -> Мытищи -> Коргашено -> Пирогово
Юбилейный -> Пушкино -> Коргашено -> Мытищи -> Челобитьево -> Пирогово
Юбилейный -> Пушкино -> Коргашено -> Пирогово
Юбилейный -> Пушкино -> Мытищи -> Челобитьево -> Пирогово
Юбилейный -> Пушкино -> Мытищи -> Коргашено -> Пирогово
Юбилейный -> Королев -> Мытищи -> Пушкино -> Коргашено -> Пирогово
Юбилейный -> Королев -> Мытищи -> Челобитьево -> Пирогово
Юбилейный -> Королев -> Мытищи -> Коргашено -> Пирогово
```

Рисунок 6 - Результаты работы программы 1а.

```
Пирогово -> Коргашено -> Мытищи -> Пушкино -> Юбилейный : 22
Пирогово -> Коргашено -> Мытищи -> Юбилейный : 12
Пирогово -> Коргашено -> Мытищи -> Королев -> Юбилейный : 12
Пирогово -> Коргашено -> Пушкино -> Юбилейный : 17
Пирогово -> Коргашено -> Пушкино -> Мытищи -> Юбилейный : 27
Пирогово -> Коргашено -> Пушкино -> Мытищи -> Королев -> Юбилейный : 27
Пирогово -> Челобитьево -> Мытищи -> Пушкино -> Юбилейный : 24
Пирогово -> Челобитьево -> Мытищи -> Юбилейный : 14
Пирогово -> Челобитьево -> Мытищи -> Королев -> Юбилейный : 14
Пирогово -> Челобитьево -> Мытищи -> Коргашено -> Пушкино -> Юбилейный : 25
```

Рисунок 7 - Результаты работы программы 1б.

Механизм логического вывода во фреймовых системах

В процессе получения вывода возникают задачи: какой фрейм, или какой слот считать означенным (конкретизированным). Слот считается конкретизированным, если его значение определено. Если в качестве значения слота выступает другой фрейм более низкого уровня, то конкретизация слота сводится к означиванию этого фрейма.

Для означивания фрейма можно выделить следующие четыре концепции.

- Фрейм считается означенным, если конкретизированы все его слоты.
- Фрейм считается означенным, если конкретизирована какая-либо группа его слотов.
- Фрейм считается означенным, если конкретизирован слот (или какой-либо набор слотов), который является определяющим.
- Фрейм считается означенным, если суммарный вес конкретизированных слотов равен или превышает заранее заданный порог.

Вывод во фреймовых системах можно осуществлять следующими способами: - с помощью присоединительных процедур (демона и служебной процедуры); - с помощью механизма наследования. Демон - это процедура, связанная со слотом фрейма и автоматически запускаемая при обращении к слоту. Отличие демона от служебной процедуры в том, что он конкретизирует слот без обращения к другим фреймам. Служебная процедура - это процедура, которая позволяет конкретизировать слот на основе означенного фрейма более низкого уровня. Демон дополняет фрейм, он может являться запросом к пользователю, т.е. непосредственно быть связанным с интерфейсной частью экспертной системы.

Управление выводом с помощью механизма наследования базируется на отношениях «абстрактное - конкретное», это наиболее удобный способ реализации отношения $IS-A$.

В данном способе осуществляется автоматический поиск и определение значений слотов фрейма верхнего уровня и служебных процедур.

Фреймы, описывающие различные объекты, называют шаблонами, а фреймы верхнего уровня, используемые для представления этих шаблонов, называют фреймами класса.

Инструментальным средством, с помощью которого целесообразнее всего строить механизм вывода на основе наследования может быть объектно-ориентированный язык.

Механизм логического вывода в семантических сетях

Отличительной особенностью семантических сетей является сложность в разграничении базы знаний и механизма логического вывода.

Интерпретация семантических сетей осуществляется с помощью использующих ее процедур. Можно выделить два способа организации процедур:

- способ сопоставления частей сетевой структуры;
- способ перекрестного поиска.

Способ сопоставления основан на построении подсети по определенному запросу и сопоставлении ее с базой данных сети. При сопоставлении с базой данных вершинам переменных подсети присваиваются гипотетические значения.

При перекрестном поиске осуществляется поиск отношений между концептуальными объектами и ответ на вопрос путем обнаружения узла, в котором пересекаются дуги, идущие от двух различных узлов. Другими словами, это обнаружение третьего узла, в котором пересекаются дуги, идущие от двух других узлов.

Проектирование интерфейса с пользователем

Проектирование интерфейса с пользователем осложняется тем, что у ЭС имеется несколько действующих лиц. Но независимо от того, является ли пользователь специалистом или нет, всех их объединяет следующее: языком общения является ограниченный естественный язык, а не формальный язык программирования.

В интерфейсном модуле могут использоваться известные формы диалога, например: директивная, табличная, фразовая и др. Фразовая форма используются во многих существующих ЭС, но достаточно на примитивном уровне. Допустимые входные сообщения пользователя ограничены набором понятий, содержащихся в базе знаний.

Актуальным направлением является разработка ЭС, которые ориентированы на пользователя не специалиста и могут общаться с системой с помощью полных предложений, включающих в себя любые части речи.

Для уменьшения времени набора фраз могут применяться сокращения, шаблоны фраз, программируемые клавиши ключевых слов и меню.

При разработке интерфейса с фразовой формой требуется реализовать ряд алгоритмов, связанных с такими видами анализа как: морфологический, синтаксический и семантический.

Применительно к фразовой форме можно выделить следующие функции интерфейса:

- осуществлять преобразование сообщения из естественно-языковой формы в форму внутреннего представления и обратное преобразование;
- анализ и синтез сообщений пользователя и системы;
- отслеживать и запоминать пройденный путь диалога.

Обобщенная схема модуля интерфейса с пользователем приведена на рисунке 8.

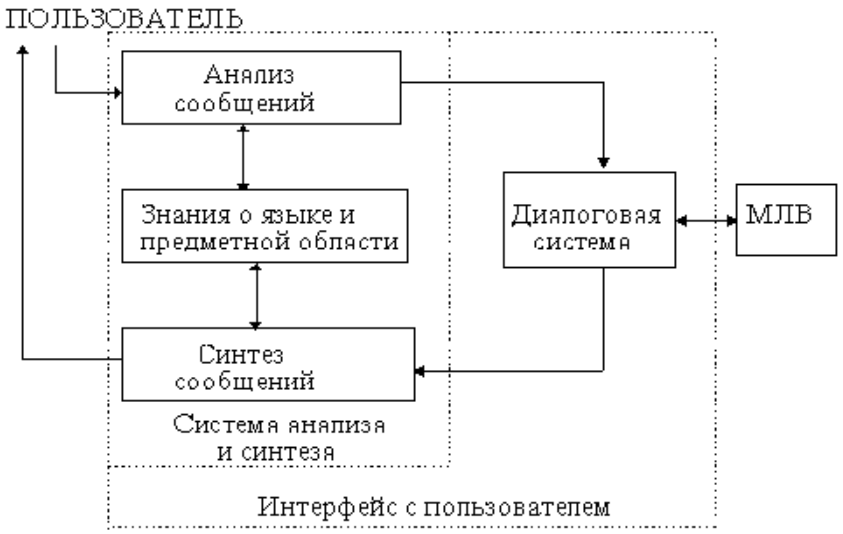


Рисунок 8 - Обобщенная схема модуля интерфейса с пользователем.

Задача системы анализа и синтеза состоит в обработке отдельных сообщений системы и пользователя. Сложность методов анализа и синтеза зависит как от языка общения, так и от языка, используемого для представления знаний.

Например, на этапе консультации язык общения может быть строго формализован фиксированным набором запросов системы и множеством возможных ответов пользователя. Здесь задача синтеза сводится к генерации подготовленных заранее вопросов, а задача анализа к обработке слов и словосочетаний с помощью морфологического анализа.

Задача синтеза на этапе приобретения знаний и объяснения в существующих системах сводится к использованию шаблонов или заранее заготовленных сообщений.

Проектирование модуля объяснения

Наряду с понятием «модуль объяснений» (МО) используются другие понятия : «блок объяснений», «подсистема объяснений» и др.

Выделяют следующие основные положения, касающиеся функций и назначения модуля объяснений.

Экспертная система должна уметь объяснять свое поведение и свои решения пользователю так же, как это делает эксперт-человек.

Без механизма объяснений пользователь не будет доверять полученным результатам, а ЭС не будет иметь спроса.

Назначение модуля объяснений - сделать ЭС «прозрачной» для пользователя, т.е. предоставить пользователю возможность понимать логику действий ЭС, дать надежную гарантию правильности полученных результатов.

При проектировании МО необходимо определить, какие типы объяснений можно применять для предметной области задачи.

В настоящее время применительно к МО выделяют пять типов объяснений:

- причинные объяснения (вскрывают причинные взаимосвязи между явлениями);
- объяснения на основе теоретических законов;
- функциональные объяснения (сводятся к установлению функций, выполняемых определенной частью системы);
- структурное объяснение (используют описание структуры, которая обеспечивает выполнение функций и поведение объясняемой системы в целом);
- историческое объяснение (раскрывает условия, причины и законы, которые привели к текущему состоянию системы).

Процессу объяснения можно дать следующее определение: “Объяснить - сделать ясным, ответить на вопросы: “Как ?”, “Почему ?”, “Что следует предпринять далее ?”, “Что будет если ?”, а также обосновать , подтвердить правильность результата.

Все возможные вопросы, возникающие у пользователя в процессе диалога с ЭС, классифицированы на два вида:

- вопросы о действиях ЭС;
- вопросы, касающиеся базы знаний.

Наибольший интерес вызывает первый вид вопросов, так как он связан непосредственно с объяснением работы ЭС. Поэтому от того , как построено объяснение для этого вида вопросов во многом зависит степень доверия к выводам, сделанным ЭС. Для этого МО должны быть средства, позволяющие отслеживать и запоминать все действия ЭС.

Второй вид вопросов касается знаний, которые имеет в своем распоряжении ЭС, в том числе знаний о самой ЭС.

Во время работы ЭС может возникнуть необходимость выдавать краткие сообщения пользователю о действиях системы независимо от того, задается вопрос или нет.

Согласно вышесказанному МО имеет в своем составе три части: активную, активно-пассивную и пассивную. На рисунке 9 приведена структурная схема процесса взаимодействия ядра ЭС и МО.

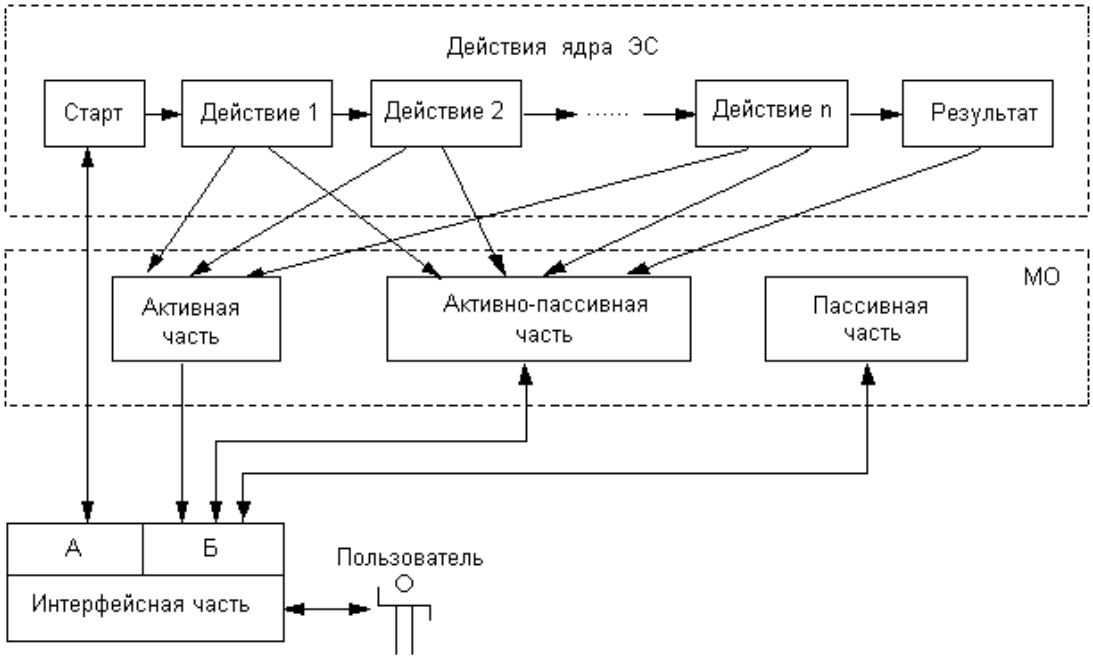


Рисунок 9 - Структурная схема процесса взаимодействия ядра ЭС и МО.

Если активная часть включена, то МО выдает постоянно краткие сообщения о действиях ЭС. Данная часть реализует формирование сообщений, которые можно отнести к функциональным объяснениям. Такие объяснения строятся по принципу : “Выполняется действие X для того, чтобы выполнить действие У ”.

Возможен упрощенный вариант функционального объяснения, который сводится к выдаче первой части сообщения о выполняемом действии в текущий момент времени.

Важно отметить, что работа МО в активном режиме будет существенно влиять на время получения результатов экспертизы.

Активно-пассивная часть - отслеживает и запоминает все действия ядра ЭС и, если есть запрос от пользователя, отвечает на вопросы о действиях системы.

Данная часть может формировать дополнительно и другие типы объяснений. Это связано с тем, что могут возникнуть вопросы, на которые невозможно ответить, используя только функциональное объяснение. К таким вопросам можно отнести:

- ✓ Какое текущее состояние логики действия ЭС?;
- ✓ Какая структура базы знаний ЭС?;
- ✓ Как система пришла к полученному выводу? и т.п.

Чтобы ответить на вопросы такого вида, необходимо использовать структурное и историческое объяснение.

Например, можно описать структуру действий системы, которая охватывает поведение ЭС в целом с помощью графа. Далее можно помечать на этом графе пройденный путь, текущее состояние логики действия ЭС, а при необходимости формировать историческое объяснение.

Пассивная часть начинает работать только в том случае, если есть запрос от пользователя. Данная часть может отвечать на вопросы, которые касаются не только знаний, но и метазнаний. В пассивной части МО могут использоваться все типы объяснений.

Например, в диагностической экспертной системе можно выделить следующие основные функции модуля объяснения:

1. Перечисление гипотез по экспертизе диагностируемого устройства, обеспечивающих идентификацию причин неисправности;
2. Стратегия поиска неисправности в диагностируемом устройстве, представленная в форме дерева поиска;
3. Анализ текущего состояния логики действия экспертной системы;
4. Объяснение пройденного ЭС пути в процессе экспертизы диагностируемого устройства;
5. Объяснение причины, сделанного экспертной системой вывода («Почему сделан вывод Nk?»);
6. Формирование списка имен еще не проверенных гипотез;
7. Демонстрация трассы, соответствующей данному выводу (Формирование объяснения на вопрос «Как ?». Например, «Как ЭС пришла к такому выводу ?»);
8. Формирование комментария пройденного ЭС пути;
9. Консультация по экспертизе (например, сформировать ответ на вопрос: «Что следует предпринять далее ?»).

Проектирование модуля накопления знаний

Модуль накопления (МН) является сервисным модулем, выполняющим различные вспомогательные функции. Как правило, добавление знаний осуществляется в

дискретные интервалы времени в процессе эксплуатации системы. Естественно, что добавление знаний предполагает добавление «новых» знаний. Постоянное пополнение новыми знаниями делают систему стабильной. Для специальных систем отсутствие новых знаний может привести ее к деградации, а комплекс, в состав которого входит ЭС, будет подвергаться большой опасности.

В процессе проектирования ЭС в роли пользователя выступает эксперт, который формирует базу знаний. Суть формирования знаний заключается в их *извлечении, структурировании и формализации*.

В процессе создания ЭС, а также во время ее функционирования возникает необходимость в изменении базы знаний, которое может быть связано с выполнением следующих действий: добавление нового правила или фрейма в БЗ, а также модифицирование, удаление правила или фрейма и др.

Выбранная модель представления знаний во многом определяет инструментальное средство. Язык программирования Пролог например, удовлетворяет требованиям продукционной системы. В данном языке имеются собственные средства редактирования для некоторых видов знаний.

Сложность заключается в том, что действия связанные с изменением БЗ должен выполнять программист, знающий структуру ЭС. Носителем же знаний является эксперт-человек, который, как правило, не знает структуры ЭС и языка программирования.

Возникает необходимость создания МН, основной задачей которого является обеспечение возможности изменения БЗ пользователем не программистом, а в некоторых случаях пользователем не являющимся экспертом.

С другой стороны, МН позволит сократить время переноса знаний от эксперта в БЗ.

Процесс переноса знаний очень сложный, он включает в себя последовательное выполнение многих функций.

- *Функция извлечение знаний*. Для выполнения этой функции МН должен в диалоге с экспертом выявлять новые правила и фреймы, а также выявлять имеющиеся в БЗ правила и фреймы, для которых необходимо модифицирование или удаление. Форма диалога должна позволять формулировать правила, фреймы на подмножестве знаков естественного языка, что позволит не требовать от эксперта знания языка программирования.

- *Функция структуризации знаний*. При выполнении данной функции извлеченные знания должны быть преобразованы из естественной формы представления, удобной для эксперта, во внутреннюю форму, принятую в ЭС.
- *Функция проверки на существование*. Прежде чем выполнить добавление, модифицирование или удаление из БЗ правила или фрейма, необходимо выяснить, существует ли это правило или фрейм. Данную функцию можно назвать вспомогательной, реализация которой позволит избежать дублирования знаний в базе.
- *Функция добавления*. В случае, если не было обнаружено в БЗ вновь введенного правила или фрейма, эксперту должна предоставляться возможность добавить знания без повторного ввода с клавиатуры. Если эксперт уверен, что данного правила или фрейма не существует в БЗ, ему должна предоставляться возможность перейти непосредственно к функции добавления. Так же должна быть возможность использования данной функции в процессе работы ЭС без специального обращения к МН. Например, когда необходимая информация в базе знаний не найдена, ЭС запрашивает ее у пользователя. В этом случае МН автоматически предоставляет возможность ввести требуемую информацию.
- *Функция модифицирования*. В случае положительного результата проверки на существование правила или фрейма, пользователю должна быть предоставлена возможность их модифицировать.
- *Функция удаления*. Необходимость данной функции очевидна. Целесообразно удалять те знания, которые либо устарели, либо противоречат вновь введенным знаниям, либо являются неправильными.
- *Функция проверки на непротиворечивость*. Вновь введенное правило или фрейм должны проверяться на непротиворечивость. Данная функция, в случае обнаружения противоречий, должна позволять в диалоге с экспертом исправлять их.

Пример проектирования ЭС

Рассмотрим пример проектирования ядра диагностической экспертной системы, целью которой является определение причины неисправности автомобиля.

После того, как была проведена оценка предметной области на достоверность и полноту накопленного опыта в области диагностики автомобиля, необходимо определить структуру (см. рис. 10) и разработать обобщенный алгоритм функционирования ЭС (см. рис.11).

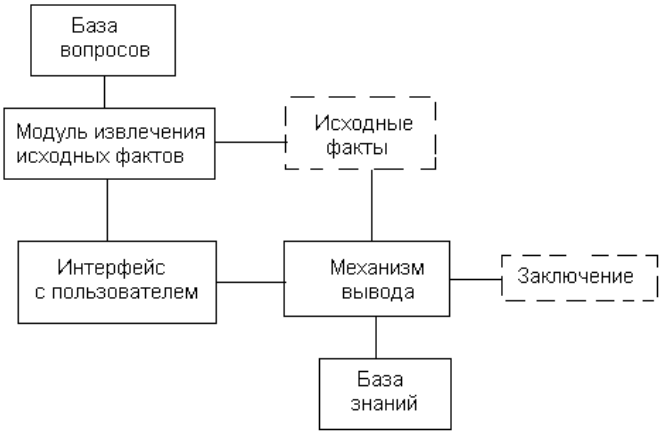


Рисунок 10 - Структура ЭС.

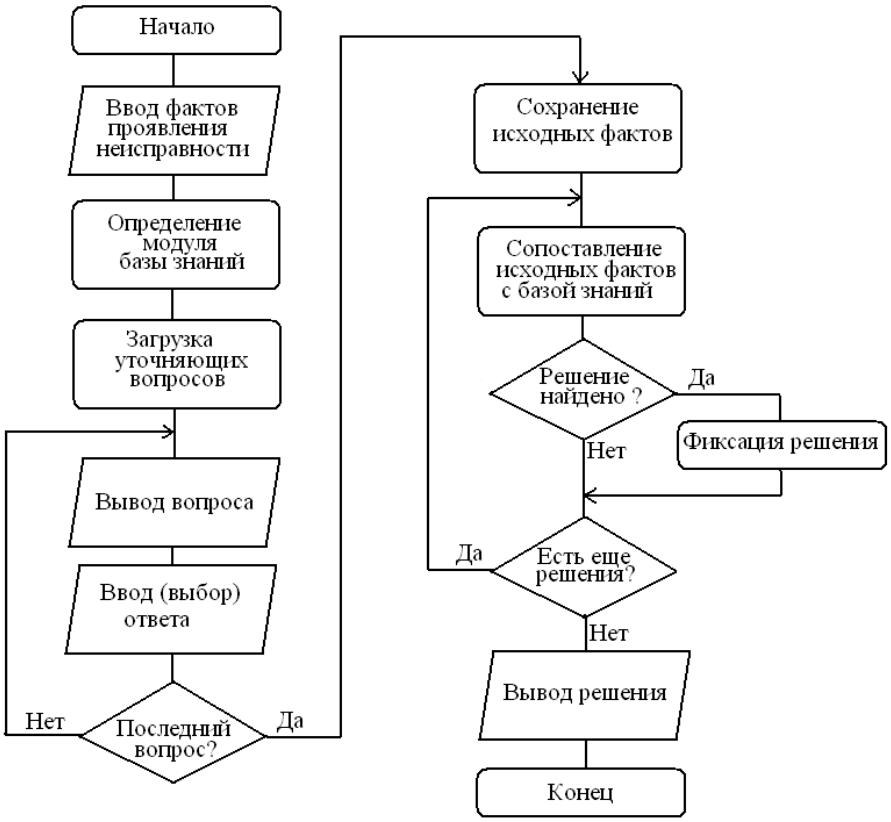


Рисунок 11 - Обобщенный алгоритм работы экспертной системы.

Основная цель диагностической ЭС, например, это определение причины неисправности автомобиля. Кроме определения причины в систему могут быть заложены дополнительные возможности (цели), например, формирование инструкции по устранению неисправности. В связи с вышесказанным, база знаний должна хранить опыт экспертов, а в ДИЭС должны быть заложены возможности модифицирования знаний, например, с целью накопления опыта получаемого во время эксплуатации системы.

При проектировании интерфейса необходимо учитывать, что диалог между пользователем не экспертом и системой должен обеспечить дополнительно возможность достижения подцели, например, получение исходных фактов конкретной ситуации. Другими словами, должно быть разработано несколько структур целевых фактов. В итоге должен быть разработан семантический интерфейс, управляемый системой, у которого имеется собственная база знаний (вопросов). Такая база вопросов может быть представлена в виде множества логически связанных вопросов и реализована с использованием нелинейных структур данных, например, в виде двоичного «дерева», в котором переход от одной вершины к другой зависит от ответа пользователя.

Важным процессом является проведение логической декомпозиции предметной области задачи. Результаты логической декомпозиции можно представить в виде иерархии, общий вид которой представлен на рисунке 12.

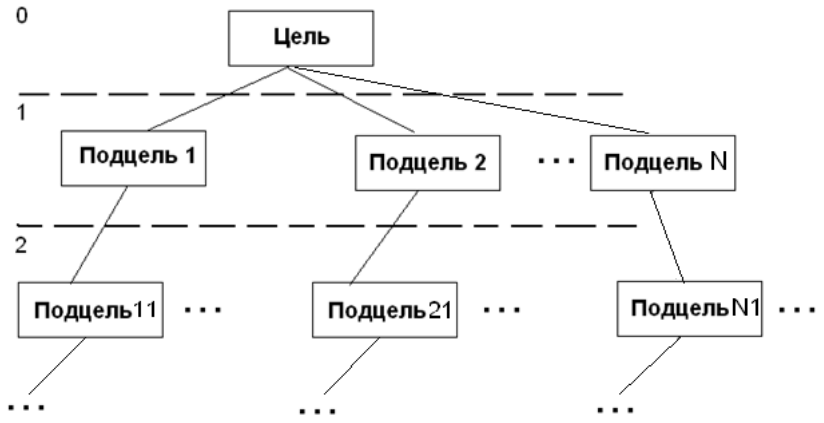


Рисунок 12 - Обобщенная схема иерархии результатов логической декомпозиции.

Результатом логической декомпозиции в рамках проектирования диагностической ЭС определения причины неисправности автомобиля может быть схема, которая представлена на рисунке 13. Процесс выделения подцелей может быть продолжен до уровня, детализация которого невозможна.

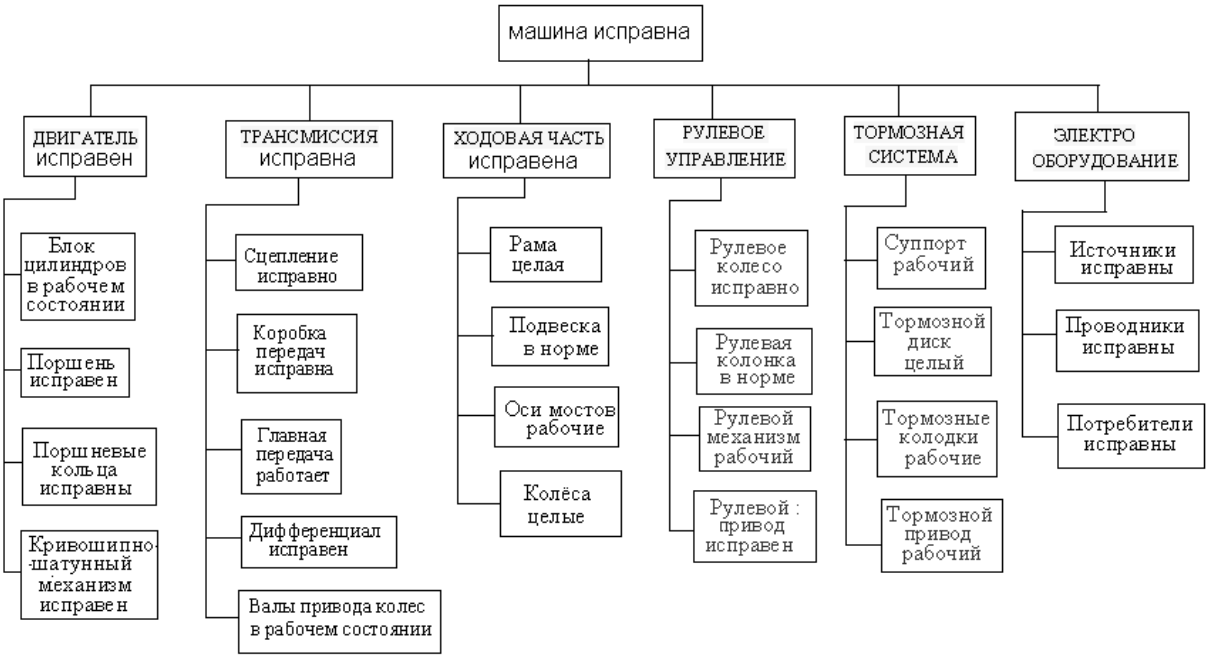


Рисунок 13 - Схема результатов логической декомпозиции.

Далее на основе результатов логической декомпозиции может быть разработана модель представления знаний.

Если выбрана продукционная модель, необходимо детально развернуть компоненты, которые были описаны в проектирование базы знаний.

Возникает вопрос, на каком уровне остановить процесс детализации. Это зависит от видов знаний, которые используются при достижении целей. Если подцель достигается на основе только декларативных знаний (фактов, которые не требуют доказательства), то декомпозицию можно завершить. Очень важным вопросом на данном этапе является обеспечение полноты и точности представления целевых фактов. Плохая проработка модели знаний может привести к неудачной реализации ДИЭС, которая может выражаться в невысокой технологичности системы, т.е. модификация такой системы может привести к перепроектированию системы в целом.

Далее на основе результатов логической декомпозиции может быть разработана модель представления знаний.

Если выбрана продукционная модель, необходимо детально развернуть компоненты, которые были описаны выше при рассмотрении материала связанного с проектированием базы знаний.

Пример фрагмента представления знаний в графической форме для продукционной модели показан на рисунке 14. Из рисунка видно, что на такой схеме можно увидеть, как факты связаны между собой, порядок вывода, а также количество уровней в модели.

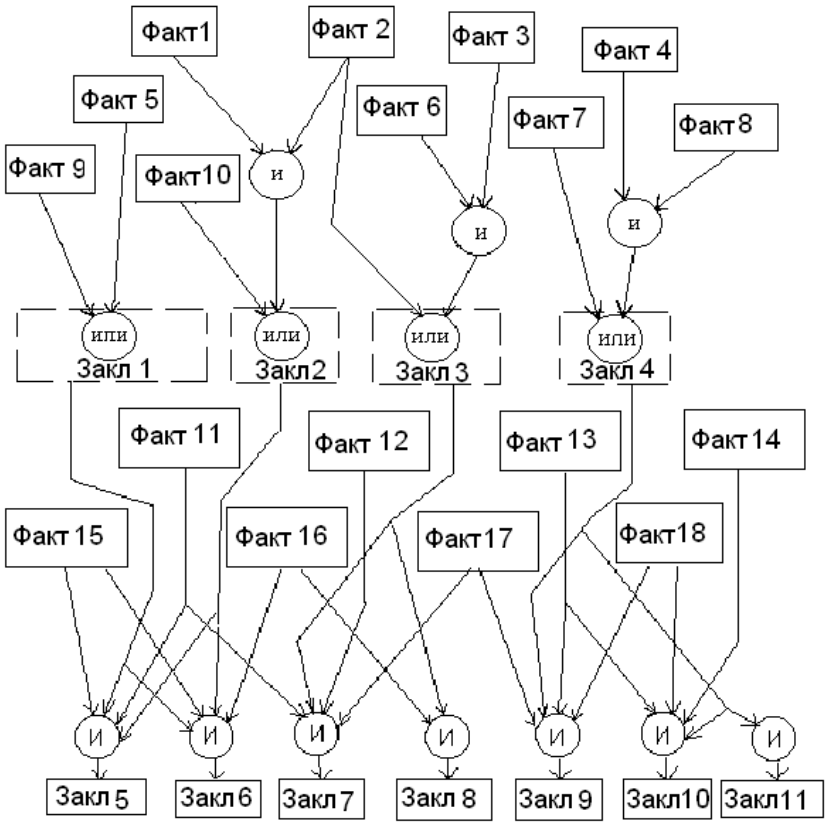


Рисунок 14 - Представления знаний в продукционной модели.

Важным шагом этапа проектирования является разработка структуры и алгоритмов работы модуля логического вывода, которые сильно зависят от конкретной модели представления знаний. Важными процессами МЛВ являются:

- процесс определения активных знаний (различных видов для конкретной ситуации);
- процесс сопоставления (включает сопоставление, как отдельных фактов, так и более сложных семантических структур);
- процесс разрешения конфликтов (необходим в сложных системах, чтобы при формировании заключения не использовались взаимоисключающие целевые факты) и др.

Если выбрана фреймовая модель, то необходимо определить слоты фреймов всех уровней, связи между фреймами, способы конкретизации слотов и фреймов.

Пример фрагмента обобщенной фреймовой модели показан на рисунке 15. На данном рисунке представлена схема, которая позволяет ответить на следующие вопросы:

- как целевые факты связаны между собой;
- какие виды фреймов используются в модели;

- какое количество уровней в модели;
- какой способ конкретизации каждого слота;
- какой способ конкретизации каждого фрейма.

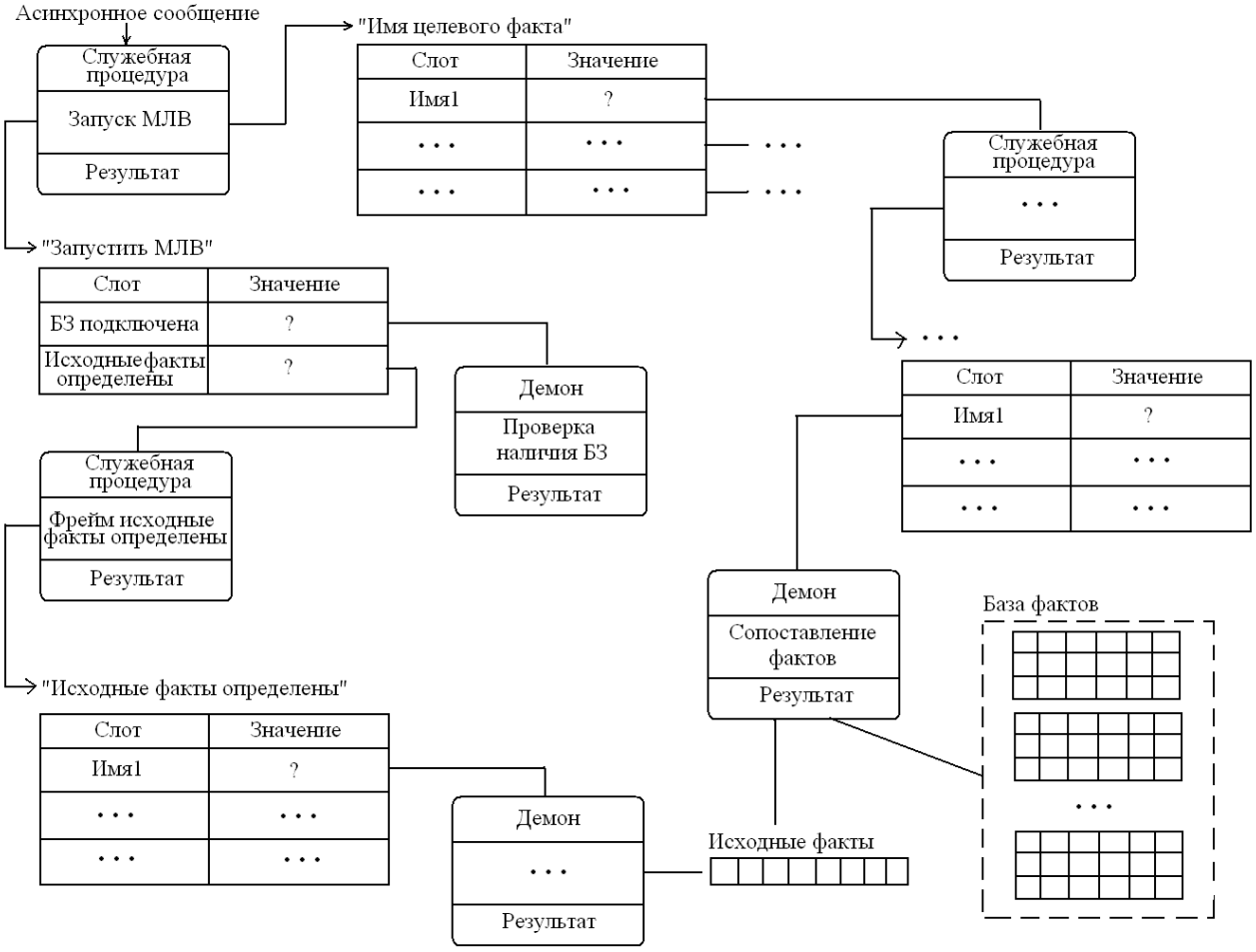


Рисунок 15 - Представления знаний во фреймовой модели.

Примеры экспертных систем

В приложении 1 приведен код программы простейшей экспертной системы продукционного типа.

Особенностями данной программы являются:

- структура базы знаний такова: что на ее основе можно разработать модуль накопления ориентированный на пользователя не программиста;

- базу фактов и базу правил можно вынести за пределы программы в отдельный текстовый файл в соответствии с форматом базы данных, а при необходимости подгружать их (или удалять из памяти);
- механизм логического вывода в своем роде универсальный (не требует коррекции при появлении новых фактов или правил);
- реализован обратный вывод с поиском в глубину;
- в интерфейсном модуле реализована фразовая форма с использованием морфологического анализа декларативным методом.

На рисунке 16 приведена модель представления знаний ЭС, которая определяет животное и принадлежность его к определенному классу.

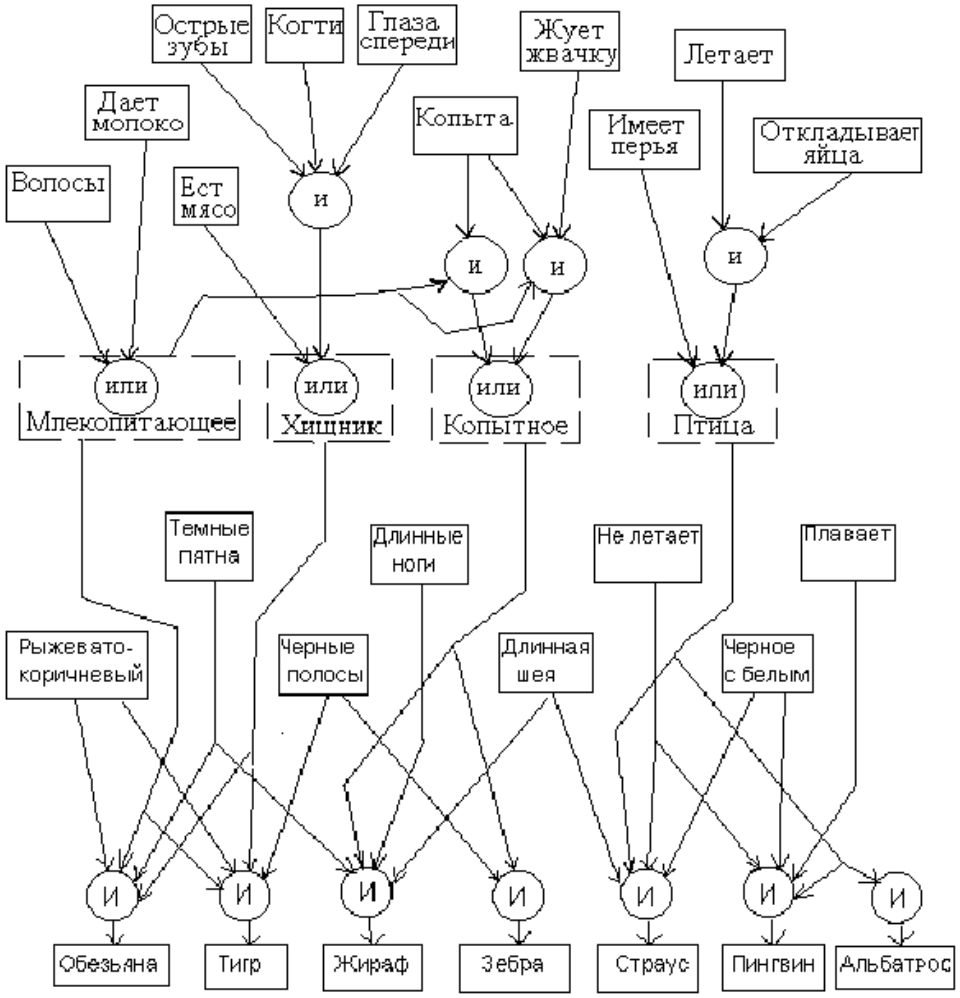


Рисунок 16 - Модель знаний ЭС определения животного.

Обобщенный алгоритм программы приложения 1 можно описать так:

- Получение активных фактов в результате диалога;

- Формирование списков групп различных фактов и правил;
- Просмотр заключений и выбор активных правил;
- Выдача результатов работы;
- Возможность уточнения (ввода дополнительных фактов);
- Работа МЛВ до тех пор, пока генерируются новые факты.

На рисунке 17 представлена схема МЛВ экспертной системы определения животного.

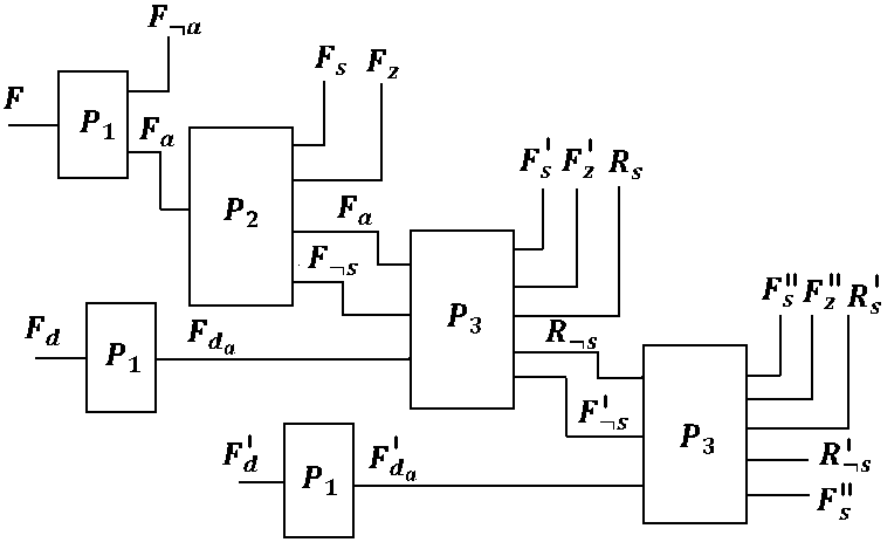


Рисунок 17 - схема МЛВ экспертной системы определения животного.

Для примера рассмотрим схему МЛВ продукционной модели, которая представлена на рисунке 17. На данном рисунке используются следующие обозначения:

$F, F_a, F_{\neg a}$ - исходные, активные и неактивные факты;

$F_s, F_{\neg s}$ - факты, которые сработали и не сработали;

F_z, F_d - факты заключения и дополнительные факты;

F_{d_a} - активные дополнительные факты;

R_a – активные правила;

$R_s, R_{\neg s}$ - правила, которые сработали и не сработали;

P_1 - процесс определения активных фактов;

P_2 - процесс определения активных правил и формирование заключений 1-го уровня;

P_3 - процесс формирования заключений 2-го уровня с возможностью дополнительных итераций.

Из рисунка 17 видно, схема носит итерационный характер, состоит из 3-х видов процессов и обеспечивает следующие возможности:

- получение активных фактов в результате диалога;
- формирование списков групп различных фактов и правил;

- просмотр заключений и выбор активных правил;
- возможность уточнения (ввода дополнительных фактов);
- работа МЛВ до тех пор, пока генерируются новые факты.

Если выбрана фреймовая модель, то необходимо определить слоты фреймов всех уровней, связи между фреймами, способы конкретизации слотов и фреймов.

Подход к проектированию на основе логической декомпозиции с использованием фреймовой модели находит применение во многих задачах [6]. Обусловлено это тем, что такую модель можно реализовать на языке общего назначения.

В приложении 2 приведена экспертная система определения вероятности инфаркта.

Особенностями данной системы являются:

- Форма модуля интерфейса табличная и директивная;
- Используется функция модуля объяснения (отслеживается пройденный путь).

Критерии оценки проекта

В целом удачный проект должен обеспечивать следующие свойства ДИЭС:

- структура базы знаний должна быть такая, чтобы на ее основе можно было разработать модуль накопления знаний ориентированный на пользователя эксперта не программиста;
- была возможность реализовывать базу знаний, в которой легко было бы разграничить виды и уровни знаний с целью реализации их в виде отдельных независимых модулей, чтобы обеспечить возможность подгружать или удалять из памяти;
- механизм логического вывода должен обладать высоким уровнем универсальности, например таким, при котором не требовалась бы модификация алгоритма вывода при появлении новых фактов и других структур знаний, а также была обеспечена возможность реализации различных способов и стратегий вывода;
- разработанная модель знаний и алгоритмы МЛВ должны давать возможность реализации модуля объяснения, задачей которого является объяснение выводов на основе различных типов объяснений.

Порядок выполнения работы

1. Изучить способы представления знаний и методы обработки знаний.
2. Изучить этапы проектирования ЭС и ее компонентов.
3. Провести логическую декомпозицию, построить модель базы знаний, реализовать модель базы знаний на выбранном языке программирования.
4. Разработать и реализовать алгоритмы работы обрабатывающих компонентов ЭС.
5. Оформит отчет о проделанной работе с включением структурных нотаций и кодом программ.

Требования к отчету

Отчет по каждому заданию должен включать:

- Номер варианта и задание;
- Схему результатов логической декомпозиции;
- Модель базы знаний;
- Структуру и алгоритмы работы модуля вывода или другого компонента в соответствии с заданием;
- Текст отлаженной программы с комментариями;
- Фрагмент базы данных (если данные в отдельном файле);
- Результаты работы реализованных компонентов ЭС;
- Заключение (перечисляются, какие стратегии, виды и способы представления знаний и были использованы).

Порядок проведения защиты:

- Ответить на вопросы, предложенные преподавателем по теоретическому материалу.
- Рассказать, как работает механизм вывода или другие компоненты ЭС.

Контрольные вопросы

1. Какие этапы проектирования ЭС существуют?
2. Назовите способы представления знаний?
3. Как работает МЛВ в продукционных системах?
4. Как работает МЛВ во фреймовых системах?
5. Как работает МЛВ в семантических сетях?
6. Особенности проектирования модуля накопления?

7. Особенности проектирования модуля объяснения?
8. Что такое стратегия поиска в глубину, привести пример?
9. Какие существуют типы объяснений?
10. Какие существую стратегии поиска решений?
11. Как работает программа, приведенная на листинге №X?

Задание 1 ко второй части лабораторных работ

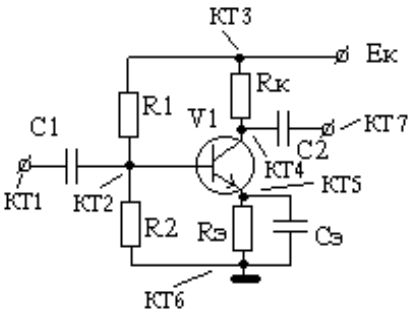
Тема 1. Консультирующая интерактивная экспертная система по определению оптимальной конфигурации ПЭВМ.

Основными входными фактами (данными) являются:

- Цель, для которой приобретается ПЭВМ;
- Пределы допустимой суммы ($ - $);
- Фирма (страна) изготовитель.

Тема 2. Диагностическая интерактивная экспертная система проверки работоспособности однокаскадного усилителя в статике.

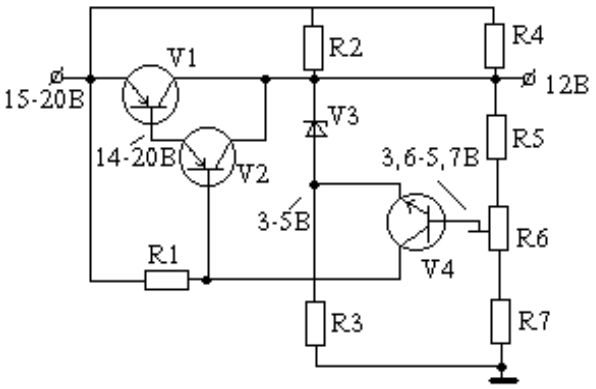
Основными входными фактами (данными) являются величины напряжений в контрольных точках.



R1 - 8,2 кОм	КТ1 - +11В
R2 - 4,7 кОм	КТ2 - +7,5В
Rк - 1,2 кОм	КТ3 - +18В
Rэ - 1 кОм	КТ4 - +15В
C1 - 220	КТ5 - +7В
C2 - 470	КТ6 - 0В
Сэ - 6800	КТ7 - +11В

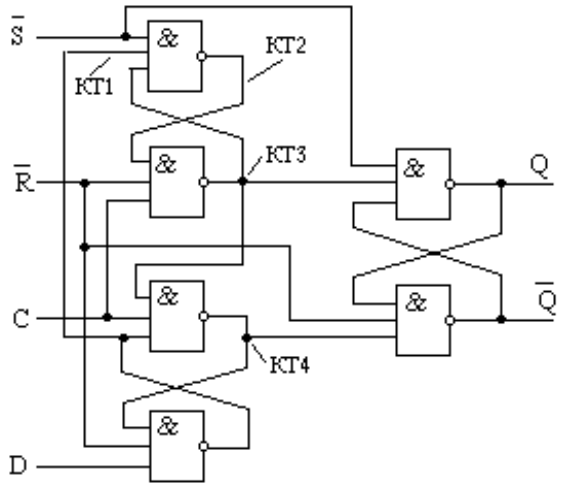
Тема 3. Диагностическая интерактивная экспертная система проверки работоспособности модуля стабилизации.

Основными входными фактами (данными) являются величины напряжений в контрольных точках.



Тема 4. Диагностическая интерактивная экспертная система проверки работоспособности D- триггера (на элементах ТТЛ) в статике.

Основными входными фактами (данными) являются величины напряжений на входах, выходах и в контрольных точках узла.



Тема 5. Консультирующая экспертная система для выбора породы собаки.

Основными входными фактами (данными) являются:

- шерсть (длинная, короткая);
- окрас (черная, белая и т.п.);
- рост;
- хвост (низкопосаженный и т.п.);
- уши (длинные, короткие и т.п.);
- характер;
- вес;
- для каких целей.

Например,

- Английский бульдог (короткая шерсть, рост меньше 22 дюймов, низкопосаженный хвост, хороший характер и т.д.);
- Гончая (короткая шерсть, рост меньше 22 дюймов, длинные уши, хороший характер и т.д.);
- Дог (короткая шерсть, рост меньше 30 дюймов, длинные уши, хороший характер, вес более 100 фунтов и т.д.);
- Американская гончая (короткая шерсть, низкопосаженный хвост, длинные уши, хороший характер и т.д.);

- Коккер-спаниэль (длинная шерсть, рост меньше 22 дюймов, низкопосаженный хвост, длинные уши, хороший характер и т.д.);
- Ирландский сеттер (длинная шерсть, рост меньше 30 дюймов, длинные уши и т.д.);
- Колли (длинная шерсть, рост меньше 22 дюймов, , низкопосаженный хвост, хороший характер и т.д.);
- Сенбернар (длинная шерсть, низкопосаженный хвост, хороший характер, вес более 100 фунтов и т.д.) и т.д.

Тема 6. Медицинская консультирующая экспертная система по выбору лекарственных трав.

- Основными входными фактами (данными) являются симптомы болезни.

Ниже приведены примеры некоторых растений:
(Имя растения, показания к применению, действие на организм)

1. Аир обыкновенный(Acorus Calamus).
   Язва желудка, гастрит с пониженной желудочной секрецией.
   Улучшает аппетит, усиливает отделение желудочного сока.
2. Аскомицеты или сумчатые грибы (Ascomicetis).
   Гипертония , повышенная раздражительность, спазмы сосудов
   Останавливает внутреннее кровотечение седативное действие на возбуждение ЦНС(Центральная Нервная Система), снижает кровяное давление.
3. Боярышник отогнуто чашелистиковый(Crategus curvisepala Lindin).
   Спонтанные боли в сердце, начальная гипертония.
   Снижает возбудимость ЦНС, тонизирует мышцу сердца, усиливает кровообращение в коронарных сосудах, снижает А/Д.
4. "Лопух большой(Arctium lappa).
   Отеки почечного и сердечного происхождения, отсутствие пота, обострение геморроя, детский диатез, злокачественные опухоли, бельмо, бородавки, хронические болезни печени.
   Способствует увеличению диуреза, снимает воспалительные процессы, при обработке волос - укрепляет волосяные сумки, помогает при экземе, улучшает функцию почек.
5. Можжевельник обыкновенный(Iunperus communis).
   Отеки почечного происхождения, гипотония, инфекция мочевых путей, кашель, неустойчивый стул.",
   Оказывает мочегонное действие, дезинфекция мочевых путей, отхаркивающее действие, улучшает пищеварение."
6. Ольха черная (Alnus glutinosa),ольха серая (Volrha sehera)",
   Энтероколиты, ожоги, дерматиты, кровотечения из носа, поносы, геморрой.
   Противовоспалительные и бактерицидные, особенно при поверхностных гнойных процессах."
7. Одуванчик лекарственный (Taraxacum officinalis Wigg).
   Нарушение аппетита, запоры, холициститы, перевозбуждение ЦНС.
   Возбуждает аппетит; как желчегонное средство, как успокаивающее средство. делает жидким стул.
8. Пастушья сумка(Capsella bursa- pastoris).
   Гипертиреоз, при капиллярном кровотечении.
   Повышает свертываемость крови, понижает давление крови, принимают при поносе, болезнях печени.
9. Сирень обыкновенная (Syrtuga Vulgaris).

"Невралгии, воспалительные процессы, язва желудка, кашель, повязки на гнойные раны, отеки почечного происхождения.",
"Снимает боль при местных воспалениях, улучшает состояние при общих простудах, воспалительных процессах, улучшает почечные отеки, мочегонное.

10. Хрен обыкновенный.
Отсутствие аппетита ,авитаминоз, понижение желудочной секреции, ухудшение пищеварения(неустойчивый стул),вирусный гепатит,ангины.
"Повышает аппетит, улучшает пищеварение, усиливает секрецию пищеварительных желез, желчегонное средство, как отвлекающее.
11. "Полынь горькая(Artemisia absintinum).
"Пониженный аппетит.
Улучшает аппетит, пищеварение ,желчегонное действие, спазмолитическое действие, аппликации при рентгеновских ожогах, экземах, бронхиальной астме.
12. Грецкий орех(Iuglands regia).
Гиповитаминоз С, поносы, неустойчивый стул, желудочное геморроидальное кровотечение, кожный туберкулез, другие кожные заболевания.
Вяжущее действие на раздраженную слизистую оболочку, кровоостанавливающее действие на десны, геморрой, язву желудка.

Тема 7. Диагностическая медицинская экспертная система.

Основными входными фактами (данными) являются ответы пациента на вопросы, задаваемые экспертной системой.
Пациент должен выразить степень согласия на ниже перечисленные утверждения.

- 'В общем я нервный';
- 'Я очень беспокоюсь о своей работе';
- 'Я часто ощущаю нервное напряжение';
- 'Моя повседневная деятельность вызывает большое напряжение';
- 'Общаясь с людьми, я часто ощущаю нервное напряжение';
- 'К концу дня я совершенно истощен физически и психически';

Степень согласия может быть выражена одним из следующих четырех вариантов:

1. "ДА, СОГЛАСЕН"
2. "СКОРЕЕ, СОГЛАСЕН"
3. "СКОРЕЕ, НЕ СОГЛАСЕН"
4. "НЕТ, НЕ СОГЛАСЕН"

Каждый вариант ответа имеет свой вес в соответствии с его порядковым номером, т.е. равен ему.
Система может выдать два решения:

1: Повышенное психоэмоциональное напряжение.
2: Психоэмоциональное напряжение в норме.

Решение выдается на основе среднего веса следующим образом:

- если пациент мужчина и (A1+A2+A3+A4+A5+A6)/6<=2, то 1 вариант решения;
- если пациент женщина и (A1+A2+A3+A4+A5+A6)/6<=1.83, то 1 вариант решения.

Во всех остальных случаях выдается второй вариант решения.

Варианты заданий:

Вариант	Тема	Модуль, функции экспертной системы

1	1	Механизм вывода с поиском в глубину
2	1	Механизм вывода с поиском в ширину
3	1	Механизм вывода для фреймовой модели
4	1	Модуль накопления знаний с проверкой на непротиворечивость
5	2	Механизм логического вывода
6	2	Модуль накопления знаний
7	2	Модуль объяснения
8	3	Механизм логического вывода
9	3	Модуль накопления знаний
10	3	Стратегия поиска в форме дерева
11	4	Механизм логического вывода для продукционной модели
12	4	Механизм логического вывода для фреймовой модели
13	4	Модуль накопления знаний
14	4	Модуль объяснения
15	5	Механизм обратного вывода
16	5	Механизм прямого вывода
17	5	Модуль объяснения
18	5	Функция извлечения и структуризации
19	6	Механизм вывода для продукционной модели
20	6	Механизм вывода для фреймовой модели
21	6	Модуль накопления знаний
22	6	Модуль накопления знаний с проверкой на непротиворечивость
23	7	Механизм обратного вывода
24	7	Механизм для продукционно–фреймовой модели
25	7	Механизм прямого вывода
26	7	Модуль накопления

Задание 2 ко второй части лабораторных работ

Выбрать актуальную задачу в рамках своего научного направления, построить модель знаний и алгоритмы основных функции обработки знаний (аналогично заданию 1 второй части).

Приложение 1. Код ЭС определения животного

```
Domains
  r = real s = string  c = char
  list=real* %  список вещественных чисел
  usp=u(r,list) % список условий срабатывания правил с учетом вопроса
  listusp=usp*

Database-f    % база фактов интерфейсного модуля
   w(r,r,s)   % вопросы
   f(r,r,s)  % возможные ответы

Database-rul    % база заключений
   rule(r,listusp,list,r) % факты заключений
   z(r,s)
          /* Общая база  */
Database
   b(s,s) quit % для вспомогательных целей
   wp_bd(r)    % для опроса
   f_act(r,r)  % для формирования активных фактов
   buff(s)     % буфер выделенных слов или фраз
   vbuff(s)    % для формирование одного слова или одной фразы
   r_act(r,r)  % активные правила
   ok(r,r)     % для проверки на срабатывае правила
   f_s(r,r)    % сработанные факты и правила
   f_time(r,r)  % для проверки сработанных правил
   old_new(r,r)  % для выхода из МЛВ

Predicates
 start
 readmy(s) interface  rmv % предикаты интерфейсного модуля
 asbuff(s,s)   % для множества выделенных слов или фраз
 prob(s,s)     % для определения лишних пробелов
 predprob(s,s) % для удаления пробела перед разделителем
 opros  wp(r,r)       % для обработки вопросника
 d1(s) pr1(s,s) pr12(r) pr2(s) v(c,c)  sumb(s)  % для обработки строк
 formact asform(s)  % для формирования активных фактов
     % предикаты для организации механизма вывода
 repeat  no_iterac   analiz(r,listusp,list,r)
 kol_u(listusp)    kol(list)
 cross_f(listusp)   cross_u(usp)  cross_r(list)
 pr_fs(r,r)  as_fs kol_r(list,r) kol_rs
 ud_f_act  ud(r,r) % удаляем сработанные факты
 new_ok(r,r) prov(r,r)
 zak  outzakl          % вывод заключений
 inform inf_fs inf_rs


  Goal start.

Clauses

start:-makewindow(1,1,7," ЭС определения животных (Выход - Esc... )",0,0,25,80),
       makewindow(2,27,47,"Перечислите возможные ответы",5,10,15,59),
       retractall(_),interface.
```

```
interface:-  retractall(_),                % удаляем все факты общей базы
      opros, not(quit),
                  /* МЛВ */
      format,    % формирование активных фактов
      assert(old_new(0,0)), % начальные сработанные правила
      repeat,     % вызов механизма вывода
      zak,        % вывод результатов работы
      fail.

interface:-  quit,cursor(5,10), write("Сеанс работы завершен ..."),
      readchar(_),rmv,exit.
interface:- interface.

      /*удаляет все пользовательские окна */
rmv:- shiftwindow(N),N<=127,removewindow,fail.
rmv:- shiftwindow(N),N>127. %  max 127- обозначений, max 34 окна
rmv:-rmv.

% ********** Интерфейс с пользователем **********
% ********** морфологический анализ декларативным методом ******

% Цикл опроса и формирование активных фактов (max 100 групп вопросов)
% Вопрос выводим как заголовок меню

opros:-not(wp_bd(_)),assert(wp_bd(1)),fail.
opros:- not(quit),  % если не Esc, то работаем дальше
      wp_bd(G),wp(G,1),
  retractall(wp_bd(_)),G<100,G1=G+1,assert(wp_bd(G1)),fail.
opros:-wp_bd(G),G=100,not(quit),!.
opros:-quit,!.
opros:-opros.

wp(K,N):-W(K,N,W),not(quit),makewindow(3,91,27,W,10,14,4,50), d1(S),N1=N+1,
       removewindow,wp(K,N1),!.
wp(K,N):-quit,removewindow,!.
wp(K,N).

% возвращаем обработанную строку (буквы после обработки только строчные)

d1(S):- readmy(Sv), pr1(Sv,S), pr2(S),!.

readmy(S):-  readln(S),!.
readmy(S):-  S="",not(quit),assert(quit),fail. % если Esc

% обработка входной строки
pr1(Sv,S1):- str_len(Sv,L),retractall(b(_,_)),
      assert(b(Sv,"")),pr12(L),b(_,S1),!.
pr12(L):- b(S,H),L<>0,frontchar(S,C,So),v(C2,C),
        str_char(Sc,C2),concat(H,Sc,H1),
        retractall(b(_,_)),L1=L-1,assert(b(So,H1)),pr12(L1).
pr12(L):-L=0,!.
pr12(L):-pr12(L).

% формируем базу фактов выделенных слов или фраз
pr2(S):-S<>"",frontstr(1,S,S1,S2),asbuff(S1,S2),pr2(S2).
```

```
pr2(S).

% формируем в vbuff фразу или слово и отправляем в buff
% при этом лишние пробелы пропускаем

asbuff(S1,S2):-not(vbuff(_)),assert(vbuff("")),fail.
asbuff(S1,S2):-not(sumb(S1)),vbuff(S),prob(S1,S2),concat(S,S1,Sz),
        retractall(vbuff(_)),assert(vbuff(Sz)),fail.
asbuff(S1,S2):-not(sumb(S1)),S2<>"",!.
asbuff(S1,S2):-vbuff(Sn), predprob(Sn,S),
        retractall(vbuff(_)),assert(buff(S)),!.
% если разделитель, а предыдущий пробел, то удаляем этот пробел
predprob(Sn,S):- str_len(Sn,L),L1=L-1,frontstr(L1,Sn,Ss,Sz),Sz=" ",S=Ss,!.
predprob(Sn,S):- S=Sn,!.

% узнаем это лишний пробел или нет
prob(S1,S2):-S1<>" ",!.
prob(S1,S2):-S1=" ",vbuff(S),S<>"",str_len(S,L),L1=L-1,
            frontstr(L1,S,Sn,Sz),Sz<>" ",!.
prob(S1,S2):-S2="".

sumb(",").sumb(".").sumb(";").    % разделители слов или фраз


 %*************** Формирование  активных фактов *****************

formact:-buff(A), asform(A),fail.
formact.
            % получаем номера активных фактов
asform(A):-f(W,N,A), not(f_act(W,N)),assert(f_act(W,N)),!.

/***** База правил  (процедурные знания) ************/
/* (перебираем все заключения и анализируем,
   получаем заключения сработанных правил fact_rule)*/

repeat:-rule(N,Split_f,Split_rule,Zakl),analiz(N,Split_f,Split_rule,Zakl),fail.
repeat:-kol_rs,no_iterac,!.  % если не нужно делать еще итерацию, то выходим
repeat:-repeat.

no_iterac:-old_new(S,N),S=N,retractall(old_new(_,_)),assert(old_new(0,0)),!.
no_iterac:-old_new(S,N),S<>N,retractall(old_new(_,_)),assert(old_new(N,0)),fail.

kol_rs:- r_act(P,_),old_new(S,N),N1=N+1,retractall(old_new(_,_)),
            assert(old_new(S,N1)),fail.
kol_rs.

% анализируем на срабатывание заключений

analiz(N,S_u,S_r,Z):-   retractall(f_time(_,_)),
   new_ok(0,0), kol_u(S_u),    % анализ списка условий
   cross_f(S_u), ok(Fu,Fa),  Fu=Fa,  as_fs,
   new_ok(0,0),  kol_r(S_r,0),
   cross_r(S_r), ok(R,Rs),   R=Rs,  % анализ сработанных правил
   prov(Fu,Rs),    % совместная проверка
   % ud_f_act,      % удаляем сработанные факты
   not(r_act(N,_)), assert(r_act(N,Z)),!.
```

```
prov(X,Y):-X>0,Y=0; X=0,Y>0; X>0,Y>0.

as_fs:-f_time(W,A),not(f_s(W,A)),assert(f_s(W,A)),fail.
as_fs.

kol_u(S):-S=[],!.
kol_u([u(W,A)|B]):- not(A=[]), kol(A), kol_u(B),!.
kol([A|B]):-A>=0,ok(N,_),N1=N+1,new_ok(N1,0),kol(B),!.
kol(S):-!.

kol_r([A|B],N):-A>=0,N1=N+1,new_ok(N1,0),kol_r(B,N1).
kol_r(S,N).

 % анализ на пересечение фактов

cross_f(S):-S=[],!.% если условий нет, то выходим
cross_f([A|B]):-  cross_u(A), cross_f(B).   %  отбираем u(W,[u1,...])

cross_u(u(W,[A|B])):- f_act(W,A), pr_fs(W,A), cross_u(u(W,B)).
cross_u(S):-!.

pr_fs(W,A):- not(f_time(W,A)), assert(f_time(W,A)),
      ok(N,K),K1=K+1, new_ok(N,K1),!.
pr_fs(W,A).

cross_r(S):-S=[],!. % анализ сработанных правил
cross_r([A|B]):-A>=0,r_act(A,Z),ok(N,K),K1=K+1,new_ok(N,K1),cross_r(B).

ud_f_act:-f_s(W,F),ud(W,F),fail. % удаляем сработанные факты
ud_f_act.

ud(W,F):-f_act(W,F),retractall(f_act(W,F)),!.

new_ok(A,B):-retractall(ok(_,_)),assert(ok(A,B)).


% Вывод сформированных заключений

zak:-r_act(_,_), % если есть заключения, то выводим
    makewindow(4,32,47," Результаты работы МЛВ ! ",3,4,17,70),
    outzakl,readchar(_),removewindow,inform,!.
zak:-not(r_act(_,_)), % если нет заключений
    makewindow(4,32,47," Просим прощения ! ",3,4,17,70),
    cursor(5,10),write("Трудно что-нибудь вывести по данным фактам!"),
    readchar(_),removewindow,!.

outzakl:-r_act(N,Z),z(Z,S),write("Это ",S,".  (Сработало правило ",N,")"),
       nl,fail.
outzakl.

inform:-r_act(_,_),
      makewindow(4,20,30,"",0,0,25,80),write("Сработанные факты: "),nl,
  inf_fs,   write("Сработанные правила: "),nl,
  inf_rs, readchar(_),removewindow,!.
inform.
```

```
inf_fs:- f_s(W,F),write(W,"-",F,"   "),nl,fail.          inf_fs.
inf_rs:- r_act(R,_),write(R,"  "),nl,fail.       inf_rs.

% Вопросник разбивается по группам
W(1,1,"Что имеет ?").
W(1,2,"Что дает ?").
W(1,3,"Чем питается ?").
W(1,4,"Как перемещается ?").
W(2,1,"Что дополнительно можно добавить ?").

% База фактов интерфейса с пользователем (декларативные знания)
f(1,1,"волосы").
f(1,2,"копыта").
f(1,3,"зубы").   f(1,3,"острые зубы"). f(1,3,"зубы острые").
f(1,4,"когти").  f(1,4,"большие когти").
f(1,5,"глаза спереди"). f(1,5,"вперед смотрящие глаза").
f(1,6,"лапы").
f(1,7,"перья").

f(2,1,"молоко").
f(2,2,"яйца").
f(3,1,"мясо").
f(3,2,"жует жвачку"). f(3,2,"трава"). f(3,2,"растения").
f(3,3,"рыба").
f(3,4,"овощи").
f(3,5,"фрукты").
f(3,6,"чем угодно").f(3,6,"всем").f(3,6,"что попадется").
f(4,1,"летает").
f(4,2,"плавает").
f(4,3,"ходит").
f(4,4,"бегает"). f(4,4,"быстро бегает"). f(4,4,"бегает быстро").
f(5,1,"рыжевато-коричневый").
f(5,2,"темные пятна").  f(5,2,"пятна темные").
f(5,3,"черные полосы"). f(5,3,"полосы черные").
f(5,4,"длинные ноги").  f(5,4,"ноги длинные").
f(5,5,"длинная шея").   f(5,5,"шея длинная").
f(5,6,"не летает").
f(5,7,"черное с белым").  f(5,7,"белое с черным").
f(5,8,"плавает").
f(5,9,"хорошо летает").  f(5,9,"летает хорошо").

/***************** База фактов заключений ****************************
 rule(N правила, списки номеров фактов и срабатываемых правил,заключение) */
% группа правил реализует 'ИЛИ'
rule(1,[u(1,[1])],[],1).
rule(1,[u(2,[1])],[],1).
rule(1,[u(1,[1]),u(2,[1])],[],1).
rule(2,[u(3,[1])],[],2).
rule(2,[u(3,[3])],[],2).
rule(2,[u(1,[3,4,5])],[],2).
rule(2,[u(1,[3,4,5]),u(2,[1])],[],2).
rule(2,[u(1,[3,4,5]),u(2,[3])],[],2).
rule(3,[u(3,[2])],[1],3).
rule(3,[u(1,[2])],[1],3).
rule(4,[u(2,[2]),u(4,[1])],[],4).
```

```
rule(4,[u(1,[7])],[],4).
rule(5,[u(5,[1,2])],[1,2],5).
rule(6,[u(5,[1,3])],[1,2],6).
rule(7,[u(5,[2,4,5])],[3],7).
rule(8,[u(5,[3])],[3,1],8).
rule(9,[u(5,[5,6,7])],[4],9).
rule(10,[u(5,[6,7,8])],[4],10).
rule(11,[u(5,[9])],[4],11).
z(1,"Млекопитающее").
z(2,"Хищник").
z(3,"Копытное").
z(4,"Птица").
z(5,"Обезьяна").
z(6,"Тигр").
z(7,"Жираф").
z(8,"Зебра").
z(9,"Страус").
z(10,"Пингвин").
z(11,"Альбатрос").

% Для преобразования входной строки из русских букв
v('й','Й').v('ш','Ш').v('в','В').v('д','Д').v('м','М').v('ц','Ц').
v('щ','Щ').v('а','А').v('ж','Ж').v('и','И').v('у','У').v('з','З').
v('п','П').v('э','Э').v('т','Т').v('к','К').v('х','Х').v('р','Р').
v('я','Я').v('ь','Ь').v('е','Е').v('ъ','Ъ').v('о','О').v('ч','Ч').
v('б','Б').v('н','Н').v('ф','Ф').v('л','Л').v('с','С').v('ю','Ю').
v('г','Г').v('ы','Ы').
v(C,C2):-C=C2,!.
```

Приложение 2. Код ЭС определения вероятности инфаркта

```
/*************** ЭКСПЕРТНАЯ СИСТЕМА **********************/

CONSTANTS
        cz0=63 % исходный цвет закраски пунктов меню
        col=18 % цвет для пометки выбранных пунктов
        c1=63  % цвет фона и символов окон 14 и 15
        c2=118 % цвет обрамления и названия окон 14 и 15

DOMAINS
% вводим с целью упрощения дальнейшего описания
  i = integer   s = string  c = char   r = real

DATABASE
wpw(s,s,s,s,s,s,s,s,s) % для хранения вариантов ответов и вероятности инфаркта
ar(i,i)  %для хранения кода нажатой клавиши и номера выбранного пункта меню
mn(i,i)  % вспомогательная база для вывода меню
tw(i)    % для хранения текущего номера вопроса или кода меню
aw(i,s)  % вспомогательная база интерфейсной части
rom(i,i) % для отдельного хранения номера вопроса и ответа
ind(i)   % для индикации: 1 - работа в главном режиме; 2 - в режиме опроса
q(i)     % вспомогательная база режима опроса

PREDICATES
        start       % предикат целевого утверждения
        menu        % для работы с меню
        wop(i,i,i)
        k(i)        % факты кодов клавиш стрелок

        read_kod(i)  char_kod(i,i) % для чтения с клавиатуры
        o(i,i,s,i,i)  d(i,i,i,i)   % для описания фактов меню

        bor          % для основного цикла программы
        an(i,i)      % для анализа и обработки нажатых клавиш
        uz(i,i,s,i) % для описания фактов-вопросов

        rw(i) ro(i,i) pw(i) po(i,i) % для разметки или пометки пунктов меню

        lop(i)       % для организации считывания цифр в режиме опроса
        let(i,s)
        cur(i,i,i)
        asrom(i,i)  % для инициализации базы с предикатом rom
        lb(i,i)     % для защиты по количеству вводимых цифр
        preobr      % для описания основных правил непосредственного
        vero(r,s)   % определения баллов и вероятности инфаркта
        dbl(i,s)    % для работы с базой rom(i,i)
        qduk        % для работы с базой q(i)
        lb1(i,i)    % факты для ограничения диапазона ввода цифр
        wiw         % для вывода информации в окно пройденного пути
        wiwex       % для вывода основного результата работы системы
```

```
      sob              % для вывода служебного сообщения
```

GOAL

```
   assert(ind(1)),assert(wpw("","","","","","","","","","")),
   makewindow(3,c1,c2," П Р О Й Д Е Н Н Ы Й   П У Т Ь ",2,40,21,37),
   makewindow(4,c1,c2," О К Н О   Д И А Л О Г А ",2,3,21,37),
   start.
```

CLAUSES

```
start:-ind(Q),Q=1,
   makewindow(1,82,111," МЕДИЦИНСКАЯ ЭКСПЕРТНАЯ СИСТЕМА ",0,0,25,80),
   makewindow(2,62,c2," ГЛАВНОЕ МЕНЮ ",5,10,8,50),
   retractall(mn(_,_)),assert(mn(10,10)),  menu,
   retractall(mn(_,_)),po(10,1),
   retractall(tw(_)),assert(tw(10)),
   retractall(ar(_,_)),assert(ar(80,1)),
   bor,!.

% считывание из базы пунктов меню
menu:-  mn(N1,N2),X2=N1+1,N1<N2+1,retractall(mn(_,_)),
          assert(mn(X2,N2)),uz(N1,_,_,Y),  wop(N1,1,Y),fail.
menu:-  mn(N1,N2),N1<N2+1,menu,!.
menu:-  mn(N1,N2),N1=N2+1,!.

% непосредственный вывод пунктов меню
wop(N,X,Y):-   o(N,X,A,D1,D2),cursor(D1,D2),
                  write(A),str_len(A,L),field_attr(D1,D2,L,cz0),
                  X2=X+1,X<Y,wop(N,X2,Y),!.
wop(N,X,Y):-X=Y,!.

% Инициализация вспомогательной базы для хранения вариантов ответов
asrom(N,M):-N<=M,assert(rom(N,0)),N1=N+1,asrom(N1,M).
asrom(N,M).

% Основные правила анализа выбранных пунктов меню и
%       чтения кода клавиш

bor:-  ar(K,R),K<>100,tw(W),W>4,W<>6,W<8,read_kod(K1),  an(K1,R),fail.
bor:-  ar(K,R),K=100,tw(W),W>4,W<>6,W<8,  an(K,R),fail.
bor:-  ar(K,R),K<>27,tw(W),W<7,W<>5,an(K,R),fail.
bor:-  ar(K,R),K<>13,tw(W),W=10,read_kod(K1),an(K1,R),fail.

% выход на исполнение пунктов главного меню
bor:-  ar(13,R),tw(10),R=1,an(300,1),fail,!.  % режим опроса
bor:-  ar(13,R),tw(10),R=2,an(500,1),fail,!.  % сведения о системе
bor:-  ar(13,R),tw(10),R=3,exit.                    % выход из системы

% Если код 27, то выходим в главное меню
bor:-  ar(K,R),K=27,retractall(ind(_)),assert(ind(1)),
          retractall(ar(_,_)),assert(ar(80,1)),  an(K,R),    fail,!.
```

```
% Если вопросы закончились (их 7, т.е. W=8), то выводим результаты экспертизы
bor:- ind(Y),Y=2,tw(W),W=8, wiwex,
          retractall(ind(_)),assert(ind(1)),
          retractall(ar(_,_)),assert(ar(80,1)),
          an(27,1),fail,!.

bor:- ind(Y),Y=2,q(X),X=1,retractall(ar(_,_)),assert(ar(80,1)),an(27,1),fail,!.

bor:-!,bor.

% Коды клавиш стрелок вверх и вниз
k(72).k(80).

% Анализ нажатия клавиши Esc, если нажата, то
%  возвращаемся в главное меню с установкой исходных параметров
an(K,R):-K=27,ind(Q),Q=1,retractall(q(_)),
   shiftwindow(1),shiftwindow(2),
   retractall(mn(_,_)),assert(mn(10,10)),    menu,
   retractall(mn(_,_)),po(10,1),
   retractall(tw(_)),assert(tw(10)),
   retractall(ar(_,_)),assert(ar(80,1)),!.

% если режим опроса,то устанавливаем параметры и в итоге активизируем окно 4
an(K,R):-K=300,  retractall(ind(_)),assert(ind(2)),
            retractall(tw(_)),assert(tw(1)),
   retractall(rom(_,_)),asrom(1,7),
   retractall(ar(_,_)),assert(ar(80,1)),
   shiftwindow(3),wiw,shiftwindow(4),clearwindow,sob,!.

an(K,R):-K=500,
   makewindow(5,30,112,"КРАТКИЕ СВЕДЕНИЯ О СИСТЕМЕ ( Выход - Esc ... )
",3,5,12,70),nl,nl,
   Sf="prim2_es.hlp",existfile(Sf),file_str(Sf,S),display(S),  removewindow,
   retractall(ind(_)),assert(ind(2)),retractall(ar(_,_)),assert(ar(80,2)),!.


an(K,R):-ind(Q),Q=0,removewindow,removewindow,removewindow,removewindow,
            removewindow,removewindow,removewindow,removewindow,exit.

an(K,R):-ind(O),O=2,
            K=100,tw(W),W>4,W<8,W<>6,
            shiftwindow(4),clearwindow,sob,
            uz(W,Y,P2,_),cursor(Y,2),write(P2),
            retractall(mn(_,_)),assert(mn(W,W)),
            menu,retractall(mn(_,_)),
            field_attr(Y,16,1,240),
            po(W,1),retractall(ar(_,_)),assert(ar(80,1)),!.

an(K,R):-ind(O),O=2,  K=13,tw(W),W>4,W<8,W<>6,
            retractall(rom(W,_)),
```

```
            Ws=W+1,Rs=R-1,assert(rom(W,Rs)),
            retractall(tw(_)),assert(tw(Ws)),
            shiftwindow(3),wiw,gotowindow(4),
            retractall(ar(_,_)),assert(ar(80,1)),!.

an(K,R):-k(K),tw(W),W>4,W<>6,W<8,uz(W,W2,S,_),
        str_len(S,Lw),
        o(W,R,A,Y,X),d(W,R,K,Rs),
        o(W,Rs,As,Ys,Xs),d(W,Rs,Ks,_),str_len(A,L),str_len(As,Ls),
        field_attr(Y,X,L,c1),field_attr(Ys,Xs,Ls,col),
        retractall(ar(_,_)),assert(ar(K,Rs)),!.

an(K,R):-k(K),tw(W),W>8,
        o(W,R,A,Y,X),d(W,R,K,Rs),
        o(W,Rs,As,Ys,Xs),d(W,Rs,Ks,_),str_len(A,L),str_len(As,Ls),
        field_attr(Y,X,L,c1),field_attr(Ys,Xs,Ls,col),
        retractall(ar(_,_)),assert(ar(K,Rs)),!.

an(K,R):-K=13,tw(W),W=10,R=1,retractall(ind(_)),assert(ind(2)),
             retractall(ar(_,_)),assert(ar(K,R)),!.

an(K,R):-K=13,tw(W),W=10,R=2,
            retractall(ind(_)),assert(ind(3)),
            retractall(ar(_,_)),assert(ar(K,R)),!.

an(K,R):-K=13,tw(W),W=10,R=3,retractall(ar(_,_)),assert(ar(K,R)),!.

an(K,R):-  ind(O),O=2,  q(U),U=1,retractall(q(_)),
              retractall(ind(_)),assert(ind(1)),
              retractall(ar(_,_)),assert(ar(K,R)),!.

an(K,R):-ind(O),tw(W),O=2,W<7,W<>5,
            shiftwindow(4),clearwindow,sob,
            uz(W,Y,P2,_),cursor(Y,2),
            write(P2),  field_attr(Y,16,1,240),
            cur(W,Y1,X1),cursor(Y1,X1),
            let(W,Lt),qduk,
            dbl(W,Lt),  retractall(tw(_)),
            W1=W+1,assert(tw(W1)),
            shiftwindow(3),wiw,gotowindow(4),
            retractall(ar(_,_)),assert(ar(100,1)),!.

an(K,R):-ind(O),O=2,  tw(W),W<7,W<>5,
            retractall(ar(_,_)),assert(ar(27,1)),!.

an(K,R):-ind(O),O=3,K=13,tw(W),W>4,W<8,W<>6,
            retractall(rom(W,_)),
            Rs=R-1,assert(rom(W,Rs)),
            retractall(ar(_,_)),assert(ar(80,R)),
            fail,  !.
```

```
an(K,R):-ind(O),O=3,K=27,tw(W),W>4,W<8,W<>6,
            retractall(tw(_)),  assert(tw(20)),
            retractall(ar(_,_)),assert(ar(80,W)),
            shiftwindow(2),  !.

an(K,R):-ind(O),O=2,K=27,tw(W),W>4,W<8,W<>6,
            retractall(q(_)),assert(q(1)),!.

an(K,R).

sob:-cursor(18,0),S="Esc  -  выход",write(S),
      str_len(S,L),field_attr(18,0,L,112),!.
sob.

qduk:-not(q(_)),!.
qduk:-q(V),V<>3,!.

% преобразуем и добавляем ответ в базу данных
dbl(R,Lt):-str_int(Lt,N),retractall(rom(R,_)),assert(rom(R,N)),!.
dbl(R,Lt).

% основное правило для подсчета количества баллов по каждому вопросу
% и обновление результирующей базы
preobr:-  rom(1,X1),rom(2,X2),rom(3,X3),
            rom(4,X4),rom(5,X5),rom(6,X6),rom(7,X7),
            M=4.2*X1+3.6*X2+4.5*X3+222*X4+68*X5-78*X6+224*X7,
            str_int(S1,X1),str_int(S2,X2),str_int(S3,X3),
            str_int(S4,X4),str_int(S5,X5),str_int(S6,X6),str_int(S7,X7),
            str_real(S8,M),  vero(M,S9),
            wpw(H1,H2,H3,H4,H5,H6,H7,H8,H9),
            retractall(wpw(_,_,_,_,_,_,_,_,_)),
            assert(wpw(S1,S2,S3,S4,S5,S6,S7,S8,S9)),!.
preobr.


cur(1,9,16).cur(2,10,16).cur(3,10,16).cur(4,10,16).cur(6,11,16).

let(W,X):-    retractall(aw(_,_)),assert(aw(1,"")),
                rom(W,U),lop(W),aw(E,X),!.


% для защиты по количеству вводимых цифр при ответе на вопросы
lb(1,2).lb(2,2).lb(3,3).lb(4,1).lb(6,2).

% факты, ограничивающие диапазон ввода цифр для всех вопросов,
% кроме 5-го и 7-го
lb1(1,100). % для количества лет, т.е. по 1-му вопросу и т.д.
lb1(2,X):-rom(1,X),!.
lb1(3,301).
lb1(4,10).
lb1(6,169).
```

```
% основное правило описывающее интерфейсную часть
% диалогового окна (анализ и обработка нажатых клавиш)
% и защита от случайного ввода

    % анализ клавиш цифр (коды 48 - 57) и формирование строки ответа
    % позволяет ввести только цифры

lop(W):-ind(G),G=2,aw(X,Q),read_kod(K),retractall(aw(_,_)),
         assert(aw(K,Q)),K>=48,K<=57,char_int(C,K),
        str_len(Q,L),lb(W,M),L<M,
        write(C),str_char(S,C),concat(Q,S,S1),
        retractall(aw(_,_)),assert(aw(K,S1)),fail.

    % обработка клавиши Esc (код 27) - выход из цикла
lop(W):-ind(G),G=2,aw(X,Q),X=27,retractall(q(_)),assert(q(1)),!.

    % обработка клавиши Enter (код 13) - выход из цикла
lop(W):-ind(G),G=2,aw(X,Q),X=13,str_len(Q,L),lb(W,M),L<=M,str_int(Q,M1),!.

    % обработка клавиши Backspase (код 8)
lop(W):-aw(X,Q),X<>13,X=8,str_len(Q,L),L>0,
              cur(W,E1,E2),  cursor(E1,E2),write("          "),
              retractall(aw(_,_)),assert(aw(1,"")),
              cursor(E1,E2),  rom(W,U),  fail.
    % зацикливаем правило
lop(W):-lop(W).


% правила для изменения цвета пунктов меню на экране
ro(W,R):-o(W,R,A,Y,X),str_len(A,L),field_attr(Y,X,L,58),!.
po(W,R):-o(W,R,A,Y,X),str_len(A,L),field_attr(Y,X,L,col),!.
po(W,R).

rw(R):-  uz(R,W,S,_),  str_len(S,L),field_attr(W,2,1,62),!.
pw(R):-  uz(R,W,S,_),  str_len(S,L),field_attr(W,2,1,240),!.

% вывод информации о пройденном пути
wiw:-  clearwindow,preobr,  wpw(H1,H2,H3,H4,H5,H6,H7,H8,H9),
           cursor(2,0),
              write("\n                    ВОЗРАСТ : ",H1,"\n                    КУРИТ(Л) : ",H2),
              write("\n        ВЕРХНЕЕ ДАВЛЕНИЕ : ",H3,"\n КОЛИЧЕСТВО
ИНФАРКТОВ : ",H4),
              write("\n        НАСЛЕДСТВЕННОСТЬ : ",H5,"\n                    СПОРТ :
",H6),
              write("\n            БОЛИ В СЕРДЦЕ : ",H7),  !.


% Вывод результатов экспертизы
wiwex:-   preobr,  wpw(H1,H2,H3,H4,H5,H6,H7,H8,H9),
      makewindow(10,0,0,"",17,22,3,38),
```

```
    makewindow(11,32,31,"ВЫВОДЫ ЭКСПЕРТНОЙ СИСТЕМЫ ",16,20,3,38),
    write("      Вероятность инфаркта - ",H9,"%"),readchar(_),
    removewindow,removewindow,  !.

% Основные правила, определяющие вероятность инфаркта
% по набранным баллам
vero(M,S):-M<570,S="0",!.
vero(M,S):-M>=570,M<=595,S="5",!.
vero(M,S):-M>595,M<=810,S="15",!.
vero(M,S):-M>810,M<=860,S="35",!.
vero(M,S):-M>860,M<=920,S="55",!.
vero(M,S):-M>920,M<=1100,S="70",!.
vero(M,S):-M>1100,M<=1800,S="90",!.
vero(M,S):-M>1800,S="100",!.
vero(M,S):-S="",!.

% База вопросов
% Имеет формат:
% uz(N вопроса; координата по У; вопрос; кол. пунктов, если вопрос по меню)

uz(1,3,"          1\n\n\n\n      Сколько вам лет ?",1).
uz(2,3,"          2\n\n\n\n      Сколько лет вы курите\n      или курили
?",1).
uz(3,3,"          3\n\n\n\n      Какое у вас верхнее\n      артериальное
давление ?",1).
uz(4,3,"          4\n\n\n\n      Сколько было у Вас\n      инфарктов
?",1).
uz(5,3,"          5\n\n\n\n      Имелись ли сердечно-\n      сосудистые
заболевания\n      у родителей ?",3).
uz(6,3,"          6\n\n\n\n      Сколько часов в неделю\n      Вы занимаетесь
спортом\n      ( физкультурой ) ?",1).
uz(7,3,"          7\n\n\n\n      Есть ли  боли в сердце\n      при физической
нагрузке ?",2).
uz(10,1,"",3).

% Описывают переход курсора при нажатии клавиш стрелок
% Факты для работы меню по 5-му вопросу
    d(5,1,80,2).d(5,1,72,3).d(5,2,80,3).d(5,2,72,1).d(5,3,80,1).d(5,3,72,2).

% Факты для работы меню по 7-му вопросу
    d(7,1,80,2).d(7,1,72,2).d(7,2,80,1).d(7,2,72,1).

% Факты для работы основного меню
    d(10,1,80,2).d(10,1,72,3).
    d(10,2,80,3).d(10,2,72,1).
    d(10,3,80,1).d(10,3,72,2).

% Нижеперечисленные факты меню имеют следующий формат:
% o(код меню, номер пункта меню, имя пункта, координата по Y и X)

% Факты-пункты меню по 5-му вопросу
```

```
o(5,1,"НЕ ИМЕЛИСЬ",11,8).
o(5,2,"У ОДНОГО РОДИТЕЛЯ",12,8).
o(5,3,"У ОБОИХ РОДИТЕЛЕЙ",13,8).

% Факты-пункты меню по 7-му вопросу
o(7,1,"НЕТ",10,15).
o(7,2,"ДА",11,15).

% Факты-пункты главного меню
o(10,1,"РЕЖИМ ОПРОСА",1,5).
o(10,2,"КРАТКИЕ СВЕДЕНИЯ О СИСТЕМЕ",2,5).
o(10,3,"ВЫХОД",3,5).

% Правила, позволяющие читать простой и расширенный коды
    read_kod(Kod):- readchar(C), char_int(C,A),char_kod(A,Kod),!.
    char_kod(A,Kod):- A<>0,Kod=A,!.
    char_kod(0,Kod):- readchar(C),char_int(C,Kod),!.
```