# 5. ИНЖЕНЕРИЯ ЗНАНИЙ

***ЦЕЛЬ***: Познакомиться с реализацией экспертной системы на языке пролог и научиться писать продукционную, фреймовую и экспертную системы

Экспертная система базируется на двух важных положениях: представление знаний (их формализация) и извлечение этих знаний, т.е. реализации процедуры вывода. Следовательно, предполагается формализация рассуждений.

Преимущество логического подхода в построении экспертных систем связано с адекватным представлением фактов и правил базы знаний в виде аксиом логической теории. Процесс вывода заключения интерпретируется как дедуктивный процесс доказательства теоремы.

С другой стороны, процессор вывода на базе других формализмов может извлекать заключения, которые иногда не могут интерпретироваться логикой. Такие случаи характерны для продукционных систем, особенно при наличии в правилах отрицания, когда процессор вывода может порождать необоснованные с точки зрения логики результаты.

Знания эксперта представляют совокупность фактов узкой предметной области, правил вывода из этих фактов заключений, рассуждений, выраженных в форме фраз естественного языка, которые формализуются и становятся содержанием базы знаний. С другой стороны, экспертные знания востребуются пользователем в форме вопросов на естественном языке. Под естественным языком следует понимать некоторое, возможно, ограниченное подмножество разговорного или профессионального языка.

Для того чтобы пользователь мог эффективно взаимодействовать с экспертной системой, ее интерфейс должен выполнять две функции: давать советы и объяснения пользователю и управлять приобретением знаний. Взаимодействие эксперта, пользователя и структурных частей системы можно представить в виде следующей базовой структуры экспертной системы (рис.1).



Объем знаний, необходимый для любого значительного приложения экспертных систем, обычно велик и поэтому для выборки, обновления и поддержания таких объемов знаний экспертным системам необходимы "хорошие" модели представления знаний. Кроме того, удачная модель представления может значительно облегчить построение логического вывода. В экспертных системах возможны модели представления знаний:

•в форме продукционных правил;

•логическая;

•семантические сети;

•фреймы;

•комбинированные.

Продукционные модели. Продукционные системы представляются в форме правил-продукций "если...то". Продукции с одной стороны, близки к логическим моделям, что позволяет организовывать на них эффективные процедуры вывода, а с другой стороны более наглядно отражают знания, чем классические логические модели. Недостаток продукция состоит в том, что при их большом количестве становится трудоемкой проверка непротиворечивости системы продукций, например при добавлении новых правил. Такие случаи характерны для продукционных систем, особенно при наличии в правилах отрицания, когда процессор вывода может порождать необоснованные с точки зрения логики результаты.

Логические модели позволяют хранить в базе лишь те знания, которые образуют множество аксиом, а все остальные знания – получать из них по правилам вывода. Логика обладает точной, простой и декларативной семантикой, поскольку она не является системой программирования. Более того, как показывает пример силлогизмов, логические рассуждения относительно близки к обычным рассуждениям человека. К недостаткам данной модели можно отнести ее неструктурируемость, а также невозможность манипулирования понятиями, истинность которых не определена.

Семантические сети обладают полнотой описания предметной области, определяют смысловые отношения между объектами. Семантика и свойства отношений может быть различной: принадлежности, каузальности, наследования, коррелятивности и т.д. В отличие от логических моделей представления знаний семантические сети позволяют успешно структурировать информацию и наглядно представить круг решаемых проблем в большей степени, чем множество правил, относящихся к ним. Связанная информация в семантических сетях хранится вместе, однако построение выводов в ней чрезвычайно затруднено.

В основе теории фреймов лежит восприятие фактов посредством полученной извне информации о некотором явлении с уже имеющимися данными, накопленными опытным путем или полученными в результате вычислений. Когда человек попадает в новую ситуацию, он вызывает из своей памяти основную структуру, называемую фреймом. Фрейм (рамка – это единица представления знаний, запомненная в прошлом, детали которой могут быть изменены согласно текущей ситуации. Класс некоторых объектов (процессов) может определяться одним типичным (базовым) объектом, который включает наиболее существенные характеристики объектов данного класса.

Фреймовое представление называют также и объектным, и это роднит с объектно-ориентированным представлением. Поэтому свойства объектно-ориентированного представления присущи и фреймовому представлению: наследование, полиморфизм и т.д. В связи с этим она имеет все свойства, присущие языку представления знаний, и одновременно она являет собой новый способ обработки информации. В настоящее время во фреймовые системы привнесены также свойства языка программирования, что позволило на базе теории фреймов программировать задачи. Представление совокупности фреймов, связанных отношениями, сближает ее с семантическими сетями. Логические выводы в фреймовых моделях также затруднены. Однако отображение фрейма в форму предикатов первого порядка позволяют модифицировать методы доказательства, развитые в теории предикатов первого порядка. Модификация не тривиальна, однако в последнее время наметились успехи в этой области.

Настоящие методические указания предназначены для самостоятельного изучения в лабораторном практикуме составных частей архитектуры экспертной системы, представленной на рис. 1. В частности, создание интерфейса пользователя, представление знаний в форме фреймов – создание, конкретизация слотов, удаление и т.д., и наконец, создание экспертных систем, основанных на фреймовом представлении.

**5.1 ПРЕДСТАВЛЕНИЕ ЗНАНИЙ С ПОМОЩЬЮ ПРОДУКЦИОННЫХ СИСТЕМ**

***ЦЕЛЬ***: знакомство с продукционной системой представления знаний.

Система продукций образуется множеством правил продукций. Эти правила формулируют определенные действия при выполнении некоторых заданных условий. В самом простом виде правила продукций близки по смыслу импликации "если-то", поэтому для правил продукций можно принять обозначение

A → B,

где A - условия применимости, а B - заключение или действие, которое имеет место при истинности A.

Рассмотрим пример продукционной экспертной системы, предназначенной для различения животных. Ниже представлен граф И/ИЛИ для правил вывода этой системы.



рис 5.1

**run:-asserta((bd\_negative('a1','b'))),asserta((bd\_positive('a1','b'))),animal\_is(X),!,nl,write('Я думаю, что это '),write(X),nl,clr.**

**run:- write('Я не знаю такого животного'),nl,clr.**

**clr:-abolish(bd\_positive/2),abolish(bd\_negative/2).**

**ask(X,Y):-write(X),write(Y),write('? '),read(Rep),nl,**

**remember(X,Y,Rep).**

**remember(X,Y,yes):-asserta((bd\_positive(X,Y))).**

**remember(X,Y,no):-asserta((bd\_negative(X,Y))),fail.**

**positive(X,Y):-bd\_positive(X,Y),!.**

**positive(X,Y):-not(negative(X,Y)),!,ask(X,Y).**

**negative(X,Y):-bd\_negative(X,Y).**

**animal\_is('гепаpд'):-it\_is('млекопитающее'),**

**it\_is('хищник'),**

**positive('имеет ','рыжекоричневый\_цвет'),**

**positive('имеет ','темные\_пятна'),!.**

**animal\_is('тигр'):-it\_is('млекопитающее'),**

**it\_is('хищник'),**

**positive('имеет ','рыжекоричневый\_цвет'),**

**positive('имеет ','черные\_полосы'),!.**

**animal\_is('жираф'):-it\_is('копытное'),**

**positive('имеет ','длинные\_ноги'),**

**positive('имеет ','длинную\_шею'),**

**positive('имеет ','темные\_пятна'),!.**

**animal\_is('зебра'):-it\_is('копытное'),**

**positive('имеет ','черные\_полосы'),!.**

**animal\_is('страус'):-it\_is('птица'),**

**positive('не\_умеет ','летать'),**

**positive('имеет ','длинные\_ноги'),**

**positive('имеет ','длинную\_шею'),**

**positive('имеет ','черно-белый\_цвет'),!.**

**animal\_is('пингвин'):-it\_is('птица'),**

**positive('не\_умеет ','летать'),**

**positive('умеет ','плавать'),**

**positive('имеет ','черно-белый\_цвет'),!.**

**animal\_is('альбатрос'):-it\_is('птица'),it\_is('хищник'),!.**

**animal\_is('воробей'):-it\_is('птица').**

**it\_is('млекопитающее'):-positive('имеет ','шеpсть');**

**positive('может ','давать\_молоко'),!.**

**it\_is('птица'):-positive('имеет ','перья'),**

**positive('может ','откладывать\_яйца'),**

**!.**

**it\_is('хищник'):-positive('может ','есть\_мясо');**

**(positive('имеет ','острые\_зубы'),**

**positive('имеет ','когти'),**

**positive('имеет ','вперед\_смотрящие\_глаза'));**

**(positive('имеет ','мощный\_клюв')),!.**

**it\_is('копытное'):-it\_is('млекопитающее'),(positive('имеет ','копыта'); positive('может ','жевать\_жвачку')),!.**

***ЗАДАНИЕ 5.1***

Запустить приведенную программу и задать несколько вопросов для определения животного. При ответах на вопросы программы следует отвечать yes/no. После ответа необходимо ставить точку.

***ЗАДАНИЕ 5.2***

Выбрать произвольную предметную область и реализовать для неё продукционную систему на языке пролог.

**5.2 ФРЕЙМЫ**

***ЦЕЛЬ:*** знакомство с фреймовой системой представления знаний.

Характеристику объекта можно пред­ставить тройкой (Объект,атрибут\_j,значение\_j).

Собрав все тройки, касающиеся данного объекта, получим объектное представление области рассуждений, относительно данного объекта. Общая форма этого представления такова:

Объект(атрибут\_j,значение\_j), j=1,...,m

Таким образом, вместо построения различных независимых формул строим более крупную структуру полной информации об объекте, которую называют фреймом. Если требуется информация о некотором объекте, то обращаются к соответствующему фрейму, внутри которого находятся свойства и факты относительно рассматриваемого объекта. Заметим, что объектное представление можно получить как из логического, так и из других представлений знаний.

Каждая пара (атрибут, значение) является слотом. Нетрудно заметить, что фрейм содержит различные (наиболее важные) слоты, характеризующие данный объект. Кроме того фрейм содержит как информационные, так и процедурные элементы (демоны), которые обеспечивают преобразование информации внутри фрейма и связи его с другими фреймами. Элементы фрейма – слоты могут быть пустыми и заполняться в процессе активизации фрейма в соответствии с определенными условиями. Таким образом, фреймы представляют собой декларативно–процедурные структуры, т.е. совокупность описаний и (возможно) связанных с ними процедур-демонов, доступ к которым выполняется прямо из фрейма.

Фрейм состоит из произвольного числа слотов, среди которых имеются системные слоты и слоты, определяемые пользователем. Каждый слот характеризуется определенной структурой и уникальным именем внутри данного фрейма.

Базовые фреймы используются для указания наиболее важных объектов, позволяют добиться лучшего понимания сущности данного объекта. На основании базового фрейма строятся фреймы для новых состояний. При этом каждый фрейм содержит слот-указатель подструктуры, что позволяет различным фреймам совместно использовать одинаковые части.

Понятие "наследование свойств" позволяет фреймам заимствовать свойства, которые имеют другие фреймы. Указатели наследования касаются только фреймовых систем иерархического типа, основанных на отношениях "абстрактное – конкретное". Они показывают, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты с такими же именами во фрейме нижнего уровня.

Различают: фрейм-образец (прототип) и фрейм-экземпляр (пример).

•Фрейм-прототип (протофрейм) содержит знания, общие для всех частных случаев, т.е. примеров.

•Фрейм-пример (фрейм-экземпляр) содержит знания, отличающие частные случай от общего.

Фрейм представляется в следующем виде:

F=(<I>,< r1 f1 >,< r2 f2>,…,< rn fn>),

где I – имя фрейма, ri -имя слота, fi - значение слота (i = 1,…,n). В качестве имен фреймов могут выступать имена объектов, событий, процессов и т.п. Слотами выступают характерные свойства или атрибуты описываемых объектов с именем I. В качестве значений слотов могут выступать:

•имена других фреймов,

•имена процедур,

•конкретные значения слотов.

1. Имя фрейма – уникальный идентификатор. Для каждого фрейма определен слот ISA– родительский данного фрейма, и слот дочерних фреймов, задаваемых списком имен (указателей). В список слотов включается ряд слотов, необходимых для организации интерфейса с пользователем, для организации внутрисистемных процессов и т.п. В их число входят, например, слоты для определения имени пользователя, даты определения, даты изменения текста, комментарии и т.п.

2. Имя слота также служит уникальным идентификатором, т.е. уникальным именем во фрейме, к которому он принадлежит. Могут вводится специальные идентификаторы, отождествляемые с отношениями ISA, PART OF, причина-следствие и другими. В качестве слота может выступать слот «условие», а его значением может быть продукционное правило. Таким образом реализуется возможность использования продукций для организации вывода.

Основными процессами обработки фреймовых структур являются:

1. создание нового экземпляра фрейма,

2. активизация фрейма,

3. формализация вывода путем последовательного поиска и активизации сети фреймов до нахождения наиболее соответствующего и построения на его основе экземпляра фрейма.

Другой, чрезвычайно перспективный подход создания межфреймовых связей и представления совокупности фреймов в виде сети состоит в представлении перечисленных связей ISA, PART OF, A KIND OF в виде фреймов. Семантика этих связей конкретизирует отношение и может быть детерминирована синтаксисом, а слот конкретизирует адресность связи.

Пример иерархической фреймовой структуры, в которой иллюстрируется наследование свойств фреймов показано на рис. 5.2. Этот пример позволит понять представление фреймовых струтур в Прологе и организацию инструментальных средств обработки фреймов. Таким образом, можно сформировать библиотеку полезных средств работы с фреймами.

|  |  |  |
| --- | --- | --- |
| mammal | VELUE | DEFAULT |
| birth | live |  |
| skin |  | fur |
| legs |  | 4 |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| monkey | VELUE | DEFAULT |  | rabbit | VELUE | DEFAULT |
| ako | live |  |  | ako | mammel |  |
| tail |  | curly |  | ears |  | long |
| legs | 2 |  |  | moves | jump |  |
|  |  |  |  | tail | short |  |



|  |  |  |
| --- | --- | --- |
| gorilla | VELUE | DEFAULT |
| ako | monkey | mammel |
| character |  | harm |
| age | 8 |  |

Рис 5.2.

Во фрейме mammal стандартные атрибуты животных: наружный покров (skin) - мех (fur) и живородящий (live). Они наследуются фреймами monkey и rabbit и не имеют таких спецификаций в этих фреймах. Слоты могут иметь значения по умолчанию, которые определены в структуре фрейма; эти значения могут быть изменены в подфреймах. Так, все млекопитающие имеют по умолчанию четыре ноги, но у обезьян их только две.

Структура данных, определенная как фрейм, включает в себя имя предиката frame с четырьмя аргументами: первый – имя фрейма, второй – фрейм-родитель, третий – фpейм-pодитель по умолчанию и последний – список слотов. Формат структуры данных фрейма :

**frame(name,[ako\_val],[ako\_def],**

**[slot\_name1,**

**[facet\_val1],**

**[facet\_def1],**

**slot\_name2,**

**[facet\_val2],**

**[facet\_def2],**

**. . . , ]).**

где : **slot\_name(j)**  – имя слота,.

**facet\_val(j)** – значение грани VALUE,.

**facet\_def(j)** – значение грани DEFAULT,.

**ako\_val** – имя фpейма-pодителя,.

**ako\_def** – имя фpейма-pодителя по умолчанию.

Напpимеp, фрейм monkey описывается так:

frame(monkey,[mammal],[],[tail,[],[curly],legs,[2],[] ]).

Замечание: все записанные в базу данных предикаты могут быть выведены с помощью предиката

**listing(frame).**

Доступ к данным в системе фреймов может быть осуществлен тремя основными предикатами :

**get\_frame** - получить данные о фрейме.

**add\_frame** – добавить или модифицировать фрейм.

**del\_frame** – удалить фрейм или атрибут фрейма.

Для удобства пользователя, эти операции будут применены к структурам, которые очень похожи на записи в базе данных: фрейм ассоциируется с записью базы данных, а слоты в фрейме – с полями в записи.

Специфика фреймовой системы – наследование, умалчиваемые значения– автоматически доступна для пользователя.

формат предикатов:

**get\_frame(Frame,Slot,Value),**

где: Frame – имя фрейма,.

Slot – имя слота, .

Value – требуемое значение..

Замечание: каждый из аргументов может быть переменной или термом.

Напpимеp, возможен следующий запрос:

**?- get\_frame(rabbit, X, Y).**

Система выдаст все атрибуты фрейма rabbit, и все, наследуемые им от mammal.

add\_frame(...), где:

а) аргументы, такие же как и у предиката frame/4,если вводится новый фрейм,

б) если добавляются или изменяются слоты в уже имеющемся фрейме, то первые три аргумента те же, а в последнем – могут стоять параметры только одного слота.

Важное замечание к пункту б):

Следует точно указывать второй и третий аргументы add\_frame/4, так как в данном случае фрейм определяется по первым трем аргументам.

Примеры:

• ввод нового фрейма cat.

**?- add(cat,[mammal],[],[painting,[],[gray],skin,[],[fluffy\_fur]]).**

• изменение слота age существующего фрейма gorilla

**?- add(gorilla,[monkey],[mammal],[age,[],[4]]).**

/\*\*\*\*Удаление фрейма \*\*\*\*/

**del\_frame(Name, SlotName),**

где Name – имя фрейма, .

SlotName – а) имя удаляемого слота, заключенное в скобки: [ ] (удаляется слот).

б) пустой список: [], если удаляется весь фрейм.

Пример:

**?- del\_frame(rabbit,[]).** % удаление фрейма rabbit.

**?- del\_frame(monkey,[legs])** % удаление слота legs из фрейма monkey.

**%Создание базы.**

**old:-**

**assertz(frame(mammal,[],[],[birth,[live],[],skin,[],[fur],**

**legs,[],[4]])),**

**assertz(frame(monkey,[mammal],[],[tail,[],[curly],**

**legs,[2],[]])),**

**assertz(frame(gorilla,[monkey],[mammal],[character,[],[harm],**

**age,[8],[]])),**

**assertz(frame(rabbit,[mammal],[],[ears,[],[long],**

**moves,[jump],[],tail,[short],[]])).**

**% new : Удаление из БД предикатов frame/4**

**new:- abolish(frame/4).**

**% add\_frame: Добавление нового фрейма в БД с проверкой наличия в базе**

**add\_frame(Name,Val,Def,[SlotName|Rest]):-**

**not(frame(Name,Val,Def,\_)),**

**assertz(frame(Name,Val,Def,[SlotName|Rest])),!.**

**% add\_frame : Добавление нового слота в фрейм или изменение старого**

**add\_frame(Name,Val,Def,[SlotName|Rest]):-**

**frame(Name,Val,Def,Slots),**

**ifthen(**

**member(SlotName,Slots),**

**del\_frame(Name,SlotName)**

**),**

**frame(Name,Val,Def,DSlots),**

**add([SlotName|Rest],DSlots,SLnew),**

**retract(frame(Name,\_,\_,\_)),**

**assertz(frame(Name,Val,Def,SLnew)).**

**% del\_frame : Удаление фрейма из БД**

**% a) - Такого фрейма в БД нет**

**del\_frame(Name, \_):-**

**not(frame(Name,\_,\_,\_)),**

**write('Такого фрейма в системе нет'),**

**!.**

**% б) - второй аргумент - [] -пустой список => удаляется весь фрейм**

**del\_frame(Name, Slot):-**

**Slot == [],**

**retract(frame(Name,\_,\_,\_)),!.**

**% в) - Второй аргумент(имя слота) не пуст => удаляется слот**

**del\_frame(Name, Slot):-**

**frame(Name,Val,Def,SlotList),**

**del3( [Slot,\_,\_], SlotList, Lnew),**

**retract(frame(Name,\_,\_,\_)),**

**assert(frame(Name,Val,Def,Lnew)).**

**% get\_frame : Получить значение из фрейма**

**% a) имя фрейма – переменная**

**get\_frame(Frame,Slot,Value):-**

**retractall(req(\_,\_,\_)),**

**var(Frame),**

**frame(Frame,\_,\_,\_),**

**help\_frame(Frame,Slot,Value),**

**req(\_,Slot,Value).**

**% б) имя фрейма – терм**

**get\_frame(Frame,Slot,Value):-**

**help\_frame(Frame,Slot,Value),**

**req(\_,Slot,Value).**

**% help\_frame : вспомогательный предикат;**

**help\_frame(Frame,\_,\_):-**

**retractall(req(\_,\_,\_)),**

**nonvar(Frame),**

**frame(Frame,\_,\_,DataList),**

**prep\_req(Frame,DataList),**

**legacy(Frame).**

**% legacy : реализация алгоритма наследования.**

**% legacy(имя\_фpейма) возвращает имя фpейма-pодителя**

**legacy(no).**

**legacy(Frame):-**

**prep\_frame(Frame,Value),**

**ifthenelse(**

**Value \== no,**

**body(Value),**

**!**

**).**

**% prep\_frame: вспомогательный предикат, возвращает имя**

**% фpейма-pодителя**

**% prep\_frame(+имя\_фpейма, -имя\_фpейма-pодителя)**

**%**

**prep\_frame(Frame,Value):-**

**frame(Frame,Val,Def,\_),**

**ifthenelse(**

**Val \== [],**

**Value =.. Val,**

**ifthenelse(**

**Def \== [],**

**Value =.. Def,**

**Value =.. [no]**

**)**

**).**

**% body : вспомогательный предикат**

**body(Value):-**

**frame(Value,\_,\_,DataList),**

**prep\_req(Value,DataList),**

**legacy(Value).**

**% prep\_req : Подготовка записей для фрейма:**

**% вспом.пpедикат, результат действия -**

**% запись в БД предикатов req/3.**

**% req(имя\_фpейма, имя\_слота, значение),**

**prep\_req(\_,[]).**

**prep\_req(F,[S1,V1,D1|Rest]):-**

**ifthen((not(req(\_,S1,\_))),**

**(ifthenelse(**

**V1 \== [],**

**assertz(req(F,S1,V1)),**

**ifthen(**

**D1 \== [],**

**assertz(req(F,S1,D1))**

**)**

**))**

**),!,**

**prep\_req(F,Rest).**

**% member : проверка принадлежности элемента списку:**

**% member(элемент, список).**

**member(X,[X|\_]):-!.**

**member(X,[\_|L]):-member(X,L).**

**% Конкатенация списков :**

**% add : add(список1, список2, список1+список2)**

**add([],L,L).**

**add([X|L1],L2,[X|L3]):-**

**add(L1,L2,L3).**

**% Добавление элемента к списку :**

**% add\_item : add\_item(элемент,список,элемент+список)**

**add\_item(X,L,[X|L]).**

**% del3 : Удаление трехэлементного подсписка из списка**

**% del3(+тpехэл.подсписок, +исх.список, -получ.список)**

**%**

**del3([X1,X2,X3], [X1,X2,X3|Rest], Rest ).**

**del3( [X1,X2,X3], [Y1,Y2,Y3|Rest], [Y1,Y2,Y3|Rest1] ):-**

**del3( [X1,X2,X3], Rest, Rest1).**

**ifthen(Cond,Act):-call(Cond),!,call(Act).**

**ifthen(\_,\_).**

**ifthenelse(Cond,Act1,\_):-call(Cond),!,call(Act1).**

**ifthenelse(\_,\_,Act2):-call(Act2).**

***ЗАДАНИЕ 5.3***

Реализовать для выбранной в задании 5.2 предметной области продукционную систему на языке пролог.

5.3 ЭКСПЕРТНАЯ СИСТЕМА, ОСНОВАННАЯ НА ФРЕЙМОВОМ ПРЕДСТАВЛЕНИИ ЗНАНИЙ

Рассмотрим фрейм-ориентированную систему, которая играет роль советчика в финансовых инвестициях. Этот советчик «Advisor» в состоянии «подсказать».

1. Будет ли расти, убывать или останется неизменной процентная ставка прибыли в течение 12-месячного периода.
2. Будет ли увеличиваться или уменьшаться годовой доход за тот же период.
3. Будут ли расходы больше или меньше бюджета.
4. Насколько велик риск пользователя.

Фрейм содержит слоты interest, revenues, expenses, risk, term и investment (рис. 3.).

Результаты обработки информации передаются в помеченные звездочкой слоты. Остальные слоты заполняются пользователем.



В Prolog’е фрейм может быть определен как структура, содержащая список слотов. Каждый слот имеет имя и значение, разделенные двоеточием. Обращаются к паре Name:Value как единому аргументу, но разные части аргумента могут быть доступны по необходимости.

**fa\_frame([interest:Interest,**

**revenues:Revenues,**

**expenses:Expenses,**

**risk:Risk,**

**term:Term,**

**investment:Investment]).**

(Имя предиката fa\_frame образовано из начальных букв **f**inansial **a**dvisor.)

Фреймы могут хранить не только фактографическую информацию, как fa\_frame, но и информацию о внутренних процедурах–демонах. Демоны могут, например, формулировать вопрос или определять список допустимых значений, которые возможны в слоте. Система использует эту дополнительную информацию, чтобы задать соответствующий вопрос об обновлении информации в слоте или проверке необходимости передачи пользователем информации в конкретный слот. Одна из процедурных связей позволяет задать вопрос в момент ожидания нового значения для слота. Например, когда системе необходимо заполнить значением слот interest, будет задан вопрос: «Какова ожидаемая ставка прибыли в следующем году?» При вводе ответа система проверит на допустимость введенных значений из заданного списка (+, -, unchanged).

Следующие структуры valdef содержат продукционную связь определения допустимых значений для конкретного слота. Например, определение expenses устанавливает два состоятельных значения для этого слота. Знак плюс означает, что расходы превысят бюджет на 10%; знак минус – расходы не превысят бюджет на 10%.

**valdef(expenses, 'Exceed budget (by more than 10%)',+ ).**

**valdef(expenses, 'Ниже уровня бюджета на 10%',- ).**

**valdef(expenses, 'Не изменится', =).**

**valdef(revenues, 'Небольшое снижение (более 10%)',- ).**

**valdef(revenues, 'Превысят ожидания (более 10%)',+ ).**

**valdef(revenues, 'Ia eciaiyony',= ).**

**valdef(interest, 'Повысятся',up).**

**valdef(interest, 'Не изменится',unchanged).**

**valdef(interest, 'Снизится',down).**

**valdef(risk,'Низкий',low).**

**valdef(risk,'средне-низкий',medium\_low).**

**valdef(risk,'средне-высокий',medium\_high).**

**valdef(risk,'высокий',high).**

**valdef(investment,'Правительственные', gov).**

**valdef(investment, 'Банков свободной зоны или евродоллары',offshore).**

**valdef(investment,'Беспроцентные залоговые ценные бумаги',unrated).**

**valdef(investment,'Облигации банков или корпораций',obligs).**

**valdef(term,'Короткий (1–3 мес)',short).**

**valdef(term,'Средний (3–6 мес)',medium).**

**valdef(term,'Длинный (6–12 мес)',long).**

Структура question содержит вопросы, которые ассоциированы с каждым слотом в фрейме.

**question(interest,'Какую ставку дохода ожидаете в следующем году?').**

**question(expenses,'Насколько могут реальные расходы отличаться (отклоняться) от ожидаемых?').**

**question(revenues,'Насколько могут отличаться (отклоняться) реальные доходы от ожидаемых?').**

**question(risk,'Каков допустимый риск?').**

**question(term,'На какой срок?').**

Сроки капитальных вложений (investment) определяются из значений interest, revenues и expenses. Каждое из следующих правил определяет сроки вложений, которые соответствуют конкретной комбинации вклада, годового дохода и расходов. Правила записаны для фреймов, которые в целом их содержат. Например, рекомендуемый срок вклада – short, если interest down (низкий), если revenue (годовой доход) «–» (минус) и expenses (расходы) “+” или “=”.

**rule(term:short, [interest:[down], revenues:[-], expenses:[+,=]]).**

**rule(term:medium, [interest:[down], revenues:[-], expenses:[-]]).**

**rule(term:medium, [interest:[down], revenues:[=], expenses:[+]]).**

**rule(term:long, [interest:[down], revenues:[=], expenses:[–,=]]).**

**rule(term:long, [interest:[down], revenues:[+], expenses:[+,=,-]]).**

**rule(term:short, [interest:[up], revenues:[+,=,-], expenses:[+,=,-]]).**

**rule(term:medium, [interest:[unchanged], revenues:[=], expenses:[=,-]]).**

**rule(term:short, [interest:[unchanged], revenues:[=], expenses:[+ ]]).**

**rule(term:long, [interest:[unchanged], revenues:[+], expenses:[=,-]]).**

**rule(term:medium, [interest:[unchanged], revenues:[+], expenses:[+]]).**

Слот investment определяется значениями слотов risk и term. Таким образом, если risk – low и term – medium or long, то рекомендуемое вложение –gov. Это правило имеет вид:

**rule(investment:gov, [risk:[low], term:[medium,long]]).**

**rule(investment:obligs, [risk:[low], term:[short]]).**

**rule(investment:obligs, [risk:[low-medium],term:[medium,long]]).**

**rule(investment:offshore, [risk:[low-medium],term:[short]]).**

**rule(investment:offshore, [risk:[medium-high],term:[short,medium,long]]).**

**rule(investment:unrated, [risk:[high],term:[short,medium,long]]).**

В системе fa\_frame определяется текущее множество условий, которые должны анализироваться.

Структуры valdef и question определяют связанные процедуры для фрейма. Наконец, правила для определения рекомендуемых вложений также представляются фреймами.

***Пользовательские консультации***

Рассматриваемая система по-разному использует статическую и динамическую информацию. Определение фрейма, его процедурные элементы и эвристики составляют статическую информацию системы. Они остаются неизменными от одного до другого выполнения программы. Очевидно, каждый раз запуская программу, пользователь заполняет копию образца фрейма. Это составляет динамическую информацию системы.

Фрейм-ориентированные системы различают прототип фрейма и его экземпляр. Все экземпляры имеют одинаковое число и тип слотов. Все процедурные связи и эвристики применяются ко всем экземплярам. Очевидно, каждый экземпляр содержит разные значения слотов.

Когда экземпляр вызывается впервые, система должна возвращать копию fa\_frame из базы:

fa\_frame([Interest,Revenues,Expenses,Risk,Term,Investment])

Когда фрейм загружен, система задает вопросы, ассоциированные со слотами investment, revenues, risk, expenses и проверяет ответ пользователя на допустимость значений для этих слотов. Эти шаги являются частью рекурсивного цикла (bind\_all), выполняемого над списком слотов, которые требуют ввода пользователем информации.

**bind\_all([],\_).**

**bind\_all([S:V|Slots],I):-**

**first\_unbound\_slot([S:V|Slots]),**

**user\_choose\_value([S:V|Slots]),**

**bind\_all(Slots,Index).**

Предикат first\_unbound\_slot рекурсивно ищет список для тех элементов, которые еще не заполнены. Предикат успешен, если аргумент Val является переменной, иначе пропускает конкретизированные переменные и проверяет, является ли следующий аргумент переменной.

**first\_unbound\_slot([Slot:Val|Slots]):-**

**var(Val),!.**

**first\_unbound\_slot([\_|Slots]):-**

**first\_unbound\_slot(Slots).**

Когда будет найден несвязанный элемент, предикат bind\_all ia?aaaco nienie i?aaeeaoo user\_choose\_value. Этот предикат задает вопрос, ассоциированный с несвязанным слотом, и вызывает предикат get\_ans, чтобы заполнить слот ответом пользователя.

**user\_choose\_value([Slot:Val|Slots]):-**

**question(Slot,Q),**

**write(Q),**

**get\_ans(Slot,Val,Q),!.**

Предикат get\_ans считывает ответ пользователя и проверяет его значимость. Если ответ незначим, get\_ans выводит сообщение, содержащее допустимые значения для слота. Поэтому запрос будет повторен для нового ввода ответа пользователем .

**get\_ans(Slot, Ans, Q):-**

**read(Ans),%get\_ans(Ans),**

**valdef(Slot,\_,Ans).%,!.**

**get\_ans(Slot, Ans, Q):-**

**write('Допустимое значение слота':Slot),nl,**

**get\_ans\_aux(Slot,Q),**

**get\_ans(Slot,Ans,Q).**

**get\_ans\_aux(Slot,Q):-**

**valdef(Slot,Prompt,Def),**

**tab(4),write(Prompt:Def),nl,fail.**

**get\_ans\_aux(Slot,Q):-**

**nl,write(Q).**

***Сокращение пространства решений***

В некоторых относительно несложных проблемах возникает необходимость сокращения пространства решений. Поскольку число переменных и число допустимых значений, которые эти переменные могут принимать возрастает, число возможных комбинаций растет настолько быстро, что вскоре мы имеем неуправляемое пространство решений. Этот феномен известен как «комбинаторный взрыв». Этот рост нелинеен. Все возможные значения всех переменных комбинируют. В рассматриваемом примере мы имеем три возможных значения для interest, expenses, term и revenues, четыре для risk и investment. Размер пространства решений принимает значение

3×3×3×3×4×4=1296.

Из этих 1296 комбинаций лишь 108 значимы. Для логического вывода удаляются все лишние правила. Таким образом 108 возможных решений могут быль выведены только семнадцатью правилами. Они определяют допустимые комбинации значений, которые фрейм может содержать. Получение решения, таким образом, требует только ту текущую конкретизацию фрейма, которая соответствует одному из определенных в правиле образцов фрейма.

Выбор текущего правила процессором логического вывода осуществляется в соответствии с обратным либо прямым методом вывода. Применение правил к экземпляру фрейма связано с унификацией. Поскольку правило может определить список значений слота, и какое-то одно значение является значимым, то процессор логического вывода не может полностью полагаться на схему унификации Prolog’а. Он активизирует предикат unify для проверки принадлежности значения слота списку значений в правиле.

**apply\_rules(Trole:Tval, Froles):-**

**rule(Trole:Tval,Frole\_list),**

**unify(Froles,Frole\_list).**

**unify([],[]).**

**unify([H:Val|T],[H1:Val1|T1]):-**

**member(Val,Val1),**

**unify(T,T1).**

Система в начале применяет правила для определения значения term. Затем использует правила для определения investment. Это достигается добавлением двух подцелей apply\_rules в запускающем предикате fa:

**fa:-**

**fa\_frame([Interest,Revenues,Expenses,Risk,Term,Investment]),**

**bind\_all([Interest,Revenues,Expenses,Risk],1),**

**apply\_rules(Term,[Interest,Revenues,Expenses]),**

**apply\_rules(Investment,[Risk,Term]),**

**get\_explain([Interest,Revenues,Expenses,Risk,Term,Investment]).**

***Объяснения результатов консультаций***

Естественным желанием пользователя является получение объяснения по выведенному системой ответу. Часто упростить объяснение гораздо сложнее, чем просто описать систему. Например, в большинстве систем в процессе извлечения информации можно задать вопрос «Почему?» Система в этом случае представляет объяснение достижения цели, которую она пыталась вывести по текущему запросу. Когда ответ найден, пользователь может задать вопрос «Как?» Система просматривает рассуждения в обратном направлении, через цепочку целей, которые были выведены в порядке их появления в решении. Очевидно, построение более простого понятного объяснения не намного сложнее, чем описать систему. Иными совами, текст объяснения фрагментируется таким образом, что при соответствующем соединении фрагментов был получен осмысленный вывод системы.

Текст объяснения является некоторой процедурной связью в слоте. Если части текста объяснения построены аккуратно, в совокупности они должны соответствовать состоянию системы.

**explain(interest:down,'Поскольку вы ожидаете снижение ставки прибыли, то было бы естественным сделать долговременные инвестиции, чтобы сохранить высокую текущую прибыль').**

**explain(interest:up,'Поскольку вы ожидаете повышение ставки прибыли, было бы естественным сделать короткие инвестиции, что позволит вам позднее реинвестировать высокие прибыли').**

**explain(interest:unchanged,'Поскольку вы ожидаете неизменной ставку прибыли, окажутся неизменными и сроки выплаты ваших инвестиций').**

**explain(revenues:'-','Так как годовой доход может быть меньше ожидаемого').**

**explain(revenues:'+','Так как годовой доход должен быть больше ожидаемого').**

**explain(revenues:'=','Отклонение годового дохода от бюджета маловероятно').**

**explain(expenses:'-','и расходы могут быть ниже ожидаемых').**

**explain(expenses:'+','и расходы должны быть выше ожидаемых').**

**explain(expenses:'=','и расходы вряд ли сильно отклонятся от бюджета').**

**explain(term:short,'целесообразны краткосрочные инвестиции. Поскольку рекомендованный срок короткий').**

**explain(term:medium,'действуйте осторожно и инвестируйте на средний срок. Так как рекомендуемый срок средний').**

**explain(term:long,'долгосрочные инвестиции наиболее целесообразны. Так ка рекомендуемый срок длинный').**

**explain(risk:low,'и вы предпочитаете небольшой риск, и хорошим инвестиционным средством для вас').**

**explain(risk:medium\_low,'и вы предпочитаете риск от низкого до среднего, и хорошим инвестиционным средством для вас').**

**explain(risk:medium\_high,'и вы предпочитаете риск от среднего до высокого, хорошим инвестиционным средством для вас').**

**explain(risk:high,'и вы предпочитаете высокий риск, и хорошим инвестиционным средством для вас').**

**explain(investment:usgov,'являются (фактически без риска) правительственные инвестиции').**

**explain(investment:obligs,'является невысокий риск по облигациям крупных банков и корпораций').**

**explain(investment:offshore,'являются банки свободной зоны или евродоллоры. (Низкий риск – короткий срок; и средний риск для больших сроков').**

**explain(investment:unrated,'является высокий риск, высокая доходность облигаций или ценных бумаг, выпущенных сберегательными и кредитными организациями').**

Объяснение строится из соответствующих фрагментов текстов. Затем эти фрагменты передаются предикату, который их выводит в определенном порядке.

**get\_explain([Interest,Revenues,Expenses,Risk,Term,Investment]):-**

**nl,**

**explain(Interest,I\_Exp),**

**explain(Revenues,Revs\_Exp),**

**explain(Expenses,Exp\_Exp),**

**explain(Risk,Risk\_Exp),**

**explain(Term,T\_Exp),**

**explain(Investment,Inv\_Exp),**

**show\_explanation([I\_Exp,Revs\_Exp,Exp\_Exp,T\_Exp,Risk\_Exp,Inv\_Exp]).**

**show\_explanation([]).**

**show\_explanation(H):-**

**write(H),nl,**

**show\_explanation(T).**

Обладая минимальными возможностями объяснений, программа fa может дать совет следующим образом:

?-fa.

Какой ожидается изменение ставки прибыли в течение следующего года? up.

Насколько могут отклоняться годовые доходы от ожидаемых? up.

Допустимые значения для слота revenues:

Превысят ожидания (более 10%): +

Небольшое снижение (более 10%): –

На сколько текущий годовой доход может отклоняться от ожидаемого? +.

На сколько текущие расходы могут отклоняться от ожидаемых? +.

Каков допустимый риск? low.

Поскольку вы ожидаете повышение ставки прибыли, было бы естественным сделать короткие инвестиции, что позволит вам позднее реинвестировать высокие прибыли.

Так как годовой доход должен быть больше ожидаемого,

и расходы должны быть выше ожидаемых,

целесообразны краткосрочные инвестиции.

Поскольку рекомендованный срок короткий

и вы предпочитаете небольшой риск,

хорошим инвестиционным средством для вас

является небольшой риск по облигациям крупных банков и корпораций.

***ЗАДАНИЕ 5.4***

Отладить экспертную систему, описанную выше. В диалоговом режиме ответить на вопросы для выбора стратегии инвестиций.

***ЗАДАНИЕ 5.5***

Реализовать для выбранной в задании 5.2 предметной области экспертную систему на языке пролог.