Построение продукционной экспертной системы

Традиционно под экспертной системой понимают компьютерную программу, предназначенную для выдачи ответов по ограниченному ряду вопросов в рамках определенной предметной области знаний. Работа экспертной системы основана на модели знаний, как правило, продукционной или фреймовой.

Структурно экспертная система может быть представлена тремя самостоятельными блоками: базой знаний, машиной вывода и интерфейсом пользователя. Наименее технологичным является этап создания базы знаний ввиду сложности формализации знаний. Знания являются слабо структурированной или вовсе неформализованной информацией. Для того чтобы представить знания в продукционной форме и далее в форме предикатов, которые являются структурными единицами языка Пролог, знания нужно сначала хотя бы частично формализовать. На этом этапе часто приходится идти на компромисс, в чём-то упрощая и снижая достоверность модели предметной области. Преимуществом проектирования экспертной системы в среде Пролог является наличие встроенной в язык машины вывода, основанной на рассмотренных ранее механизмах унификации и бектрекинга. Таким образом, работа программиста в среде Пролог сводится к формированию базы знаний и, при необходимости, разработке интерфейса пользователя.

Рассмотрим подробнее процесс разработки продукционной экспертной системы на языке Пролог. База знаний продукционной экспертной системы содержит факты и правила. Правило с точки зрения продукционной модели знаний содержит посылку и следствие. Постановка вопроса к продукционной экспертной системе инициирует процесс последовательного перебора фактов и правил из базы знаний и построения из них цепочки вывода. Промежуточные результаты сохраняются в оперативной памяти в фактах динамической базы данных.

В качестве предметной области знаний выберем парк комиссионных автомобилей. Пользователь экспертной системы будет иметь возможность получить информацию о наличии подходящего автомобиля по ряду его характеристик (атрибутов). В качестве опознавательных атрибутов выберем такие: страна производитель, тип привода, тип коробки передач.

Опишем конспективно структуру экспертной системы.

Начнем с инициации процесса поиска. Раздел внутренней цели должен содержать очистку динамической базы данных, чтобы результаты преды-

дущего анализа не накладывались на текущие, и вызов предиката поиска решения (отбор):

```
цель
    очистить Динамическую БД,
    отбор
```

В свою очередь предикат отбор будет иметь два пути для унификации. Первый путь инициализирует (предикат `авто`) поиск названия автомобиля по его атрибутам и, при успешной унификации, выводит найденное решение на экран. Второй – обеспечивает информирование пользователя при неудачном поиске:

```
отбор:-
  авто(Name),печать(Name);
  печать(нет вариантов).
```

Предикатом авто в базе данных описаны выставленные на продажу автомобили посредством декларации их атрибутов:

```
авто(Нива):-
   атрибут(отечественный,1),
   атрибут(повыш.прох.,1).
авто(Mazda 2):-
   атрибут(отечественный,2),
   атрибут(повыш.прох.,2),
   атрибут(спортивный,1).
```

Предикат для описания атрибутов содержит два аргумента: наименование атрибута и его значение, соответствующее автомобилю. Таким образом, уникальная совокупность атрибутов будет однозначно указывать на конкретный автомобиль. Совокупность предложений с предикатом авто в голове формирует базу данных об автомобилях, основанную на продукционной модели знаний. Успешная унификация предиката `авто(Name)` на любом из предложений такой базы приведет к возврату значения (название автомобиля) в вызывающий предикат, то есть в `авто(Name)` из предложения `отбор` с последующим выводом на экран этого значения.

Успешная унификация предиката `авто(A)` возможна только при успешной унификации всех входящих подцелей, то есть всех предикатов `атрибут`. Данный предикат обеспечивает непосредственный диалог с пользователем, задавая вопросы про атрибуты автомобиля и обрабатывая полученные ответы:

```
атрибут(Title,значение из описания авто):-
  взять значение из Динамической БД и сравнить,!;
  спросить про значение атрибута и сравнить.
```

Обратите внимание, что работа предиката `атрибут` основана на анализе фактов из динамической базы данных, где мы будем хранить ответы пользователя по поводу предпочитаемых им атрибутов автомобиля. Унификация предиката `атрибут` возможна по одному из двух путей. Если ранее пользователю уже задавался вопрос про значение текущего атрибута, то из динамической базы извлекается факт с соответствующим значением. В обратном случае инициализируется диалог с пользователем через предикат `спросить`. В любом случае после получения значения атрибута от пользователя оно сравнивается с аналогичным значением из описания автомобиля и, при их совпадении, предикат `атрибут` унифицируется успешно. Успешная унификация позволяет перейти к рассмотрению следующего атрибута данного автомобиля. При неуспешной унификации осуществляется переход к следующему автомобилю из базы.

Непосредственный диалог по поводу атрибута реализуем через считывание значения с клавиатуры и сохранения его в динамическую базу данных:

```
спросить(Title,значение):-
      печать(Title),
      считать(значение),
      добавить в Динамическую БД.
```

Второй аргумент в предикате `спросить` предназначен для того, чтобы возвращать значение атрибута, введенное с клавиатуры пользователем в вызывающий предикат `атрибут`.

После схематичного рассмотрения структуры экспертной системы можно перейти к её реализации на языке Пролог:

```
domains
  i=integer
  s=string
predicates
  otbor
  avto(s)
  at(s,i)
  ask(s,i)
database
  zn_at(s,i)
include "dbd.txt"
clauses
  at(A,B):-
    zn_at(A,K),!,B=K;
    ask(A,K),B=K.

  ask(A,B):-
```

```
    write(A,"? (1-да,2-нет) - "),
    readint(B),
    assert(zn_at(A,B)).

  otbor:-
    avto(Name),
    write(Name),nl,!.
  otbor:-
    write("нет вариантов").

goal
  clearwindow,
  retractall(_),
  otbor
```

Предикат `zn_at(s,i)` объявлен как факт динамической базы данных для того, чтобы во время диалога с пользователем была возможность сохранять сведения о предпочтениях пользователя.

Обратите внимание, что текст программы не содержит информации об автомобилях. Для удобства данная информация вынесена в отдельный файл `dbd.txt` и подключается в программу директивой `include`. Содержимое файла `dbd.txt` формально эквивалентно продукционной модели знаний про парк автомобилей:

```
constants
  v1="отечественный"
  v2="привод полный"
  v3="КПП автомат"
  v4="спортивный"
clauses
  avto("ВАЗ 2114"):-
    at(v1,1),at(v2,2).
  avto("Нива"):-
    at(v1,1),at(v2,1).
  avto("Santa Fe"):-
    at(v1,2),at(v2,1),at(v3,1).
  avto("Audi A6"):-
    at(v1,2),at(v2,2),at(v4,1).
```

Данную базу можно редактировать и пополнять независимо от текста программы экспертной системы, при этом программный код можно откомпилировать в исполняемый файл и запускать самостоятельно. Основной файл программы и файл с базой данных должны располагаться в одной папке. В меню среды программирования в разделе `Setup / Directories / Include directory` следует указать путь размещения файла `dbd.txt`.

Для удобства пользования экспертной системой интерфейс можно доработать следующим образом. Во-первых, имеет смысл использовать стандартный предикат makewindow. Во-вторых, необходимо усовершенствовать диалог таким образом, чтобы программа предлагала пользователю после поиска подходящего автомобиля выбор: выходить из программы или повторить поиск. Реализацию такого дополнения можно также вынести в отдельный модуль – файл `escape.txt`:

```
predicates
  repeat
  escape

clauses
  repeat.
  repeat:-repeat.

  escape:-
    write("Esc - выход, Enter - повторить"),
    readchar(D),
    char_int(D,27).
```

При унификации безаргументного предиката `escape` происходит считывание введенного с клавиатуры символа, который сохраняется в переменной D. Если индекс символа в таблице символов равен 27, что соответствует нажатой клавише Esc, то унификация предиката `escape` завершается успешно. Обратите внимание на тело предложения `start` в приведенной ниже итоговой программе:

```
domains
  i=integer
  s=string
predicates
  otbor
  avto(s)
  at(s,i)
  ask(s,i)
  start
database
  zn_at(s,i)

include "dbd.txt"
include "escape.txt"
```

```
clauses
  at(A,B):-
    zn_at(A,K),!,B=K;
    ask(A,K),B=K.

  ask(A,B):-
    write(A,"? (1-да,2-нет) - "),
    readint(B),
    assert(zn_at(A,B)).

  otbor:-
    avto(Name),
    write(Name),nl,!.
  otbor:-
    write("нет вариантов").

  start:-
    repeat,
    clearwindow(),
    retractall(_),
    otbor,nl,nl,
    escape.

goal
  makewindow(1,30,30," Диалог ",3,3,16,40),
  start
```

В предложении `start` подцель `escape` расположена на последнем месте. Таким образом, при успешной унификации подцели `escape` (при нажатии клавиши Esc) происходит завершение программы, а иначе срабатывает механизм бектрекинга и происходит откат на ближайшую точку возврата. В данном предложении она сформирована искусственно предикатом `repeat`. Дело в том, что для его унификации возможны два пути:

```
repeat.
repeat:-repeat.
```

Первый путь – безусловная успешная унификация, не содержащая никакого функционала, кроме собственно организации первого пути. Второй путь – рекурсия, необходимая для генерации альтернативного пути, унификация которого опять порождает два пути. И так происходит до тех пор, пока пользователь не решит, что больше не желает работать с программой.

Приложение №1.

ГОРЯЧИЕ КЛАВИШИ.

Показать горячие клавиши	Alt-H
Помощь по системе команд Prolog	Shift-F1
Выход из Prolog	Alt-X
Загрузить файл	F3
Загрузить файл из списка последних	Alt-F3
Сохранить файл	F2
Запустить программу на исполнение	Alt-R
Активировать меню ФАЙЛ	Alt-F
Активировать редактор программ	Alt-E
Активировать меню ОПЦИИ	Alt-O
Активировать меню УСТАНОВКИ	Alt-S
Активировать меню ФАЙЛ	Alt-F
Показать информацию о системе Prolog	Alt-V
Текущее окно в полный экран/обратно	F5
Перейти в следующее окно	F6
Открыть вспомогательный файл	F8
Вернуться в основной	Esc
Редактировать содержимое буфера	Ctrl-F8
Режим редактора – “запись поверх”	Ins
Авто отступ при нажатии Enter	Alt-I
Текущее слово в верхний регистр	Ctrl-B + Ctrl-U
Текущее слово в нижний регистр	Ctrl-B + Ctrl-L
Сменить регистр текущего слова	Ctrl-B + Ctrl-R
Сменить текущую директорию	Ctrl-K + Ctrl-L
Копировать блок из файла	
.....сначала выбираем файл:	F7 …
.....затем выделяем блок:	… Ctrl-K + Ctrl-R …
.....затем вставляем с позиции курсора:	… Ctrl-K + Ctrl-R …
Поиск фразы	Ctrl-F3 … фраза … Ctrl-F3
Поиск следующей фразы	Shift-F3
Заменить	Ctrl-F4
Заменить снова	Shift-F4
Прервать выполнение программы	Ctrl-Break

Cursor movement		
Line up	↑	Ctrl-E
Line down	↓	Ctrl-X
Left	←	Ctrl-S
Right	→	Ctrl-D
Word left	Ctrl-←	Ctrl-A
Word right	Ctrl-→	Ctrl-F
Start of line	Home	Ctrl-Q Ctrl-S
End of line	End	Ctrl-Q Ctrl-D
Start of page	Ctrl-Home	Ctrl-Q Ctrl-E
End of page	Ctrl-End	Ctrl-Q Ctrl-X
Scroll up		Ctrl-W
Scroll down		Ctrl-Z
Page up	PgUp	Ctrl-R
Page down	PgDn	Ctrl-C
Start of text	Ctrl-PgUp	Ctrl-Q Ctrl-R
End of text	Ctrl-PgDn	Ctrl-Q Ctrl-C
Previous position		Ctrl-Q Ctrl-P
Goto line	Ctrl-F2	
Goto position	Shift-F2	
Goto blockstart		Ctrl-Q Ctrl-B
Goto blockend		Ctrl-Q Ctrl-K

Insert & Delete		
Insert new line		Ctrl-N
Backspace	Ctrl-H	Ctrl-H
Delete character	Del	Ctrl-G
Delete word		Ctrl-T
Delete to start of line		Ctrl-Q Ctrl-T
Delete to end of line		Ctrl-Q Ctrl-Y
Delete line	Ctrl-BackSpace	Ctrl-Y

Block functions		
Block select		Ctrl-K Ctrl-M
Copy block to pastebuffer		Ctrl-K Ctrl-I
Move block to pastebuffer		Ctrl-K Ctrl-Y
Paste	Ctrl-F7	Ctrl-U
Mark blockstart		Ctrl-K Ctrl-B
Mark blockend		Ctrl-K Ctrl-K
WordStar show/hide block		Ctrl-K Ctrl-H
WordStar copy block		Ctrl-K Ctrl-C
WordStar move block		Ctrl-K Ctrl-V
MultiMate copy block	Ctrl-F5	
MultiMate move block	Alt-F6	
MultiMate delete block	Alt-F7	