ЛАБОРАТОРНАЯ РАБОТА № 3
«РАЗРАБОТКА И ИССЛЕДОВАНИЕ ЭКСПЕРТНОЙ СИСТЕМЫ»

Цель работы

Разработка экспертной системы продукционного типа на Прологе, исследование базовых принципов организации экспертных систем.

Постановка задачи

1. Изучить модели представления знаний по лекционному материалу и учебным пособиям [1,5,8].

2. Детально изучить организацию продукционной системы, механизмы прямого и обратного вывода, основы построения подсистемы объяснения, примеры реализации ЭС на Прологе [1,2,4,11].

3. Выполнить анализ предметной области ЭС в соответствии с вариантом задания:

а) выбрать задачу, решаемую ЭС, и определить цели системы: конечные, промежуточные и вспомогательные;

б) выделить подзадачи, которые следует решить для достижения цели;

в) разработать продукционную базу правил для решения выделенных подзадач.

4. Ознакомиться с примерами программных кодов, приведенных в приложении В, и, по аналогии, разработать ЭС продукционного типа для решения задачи в заданной предметной области.

5. Создать в среде программирования Пролог проект ЭС и выполнить его отладку.

6. Исследовать свойства разработанной системы. Получить протоколы работы системы при доказательстве различных целевых утверждений.

7. Зафиксировать результаты работы программы в виде экранных копий.

Вариант 3

Предметная область – Обучение

Ход работы

База знаний классифицирующей ЭС (файл nb.pl):

```
info:-
    nl,
    write('*************************************'),nl,
    write('*          Экспертная система         *'),nl,
    write('*       для определения будущей       *'),nl,
    write('*              профессии              *'),nl,
    write('*-----------------------------------*'),nl,
    write('*        Отвечайте на вопросы:        *'),nl,
    write('*            да, нет, почему          *'),nl,
    write('*        Для объяснения решения       *'),nl,
    write('*            введите цель             *'),nl,
    write('*************************************'),nl,
    write('Введите любой символ'),nl,
    %Ожидание ввода литеры
    get0(_).

    % тестовая база продукционных правил для определения будущей профессии
    правило1       ::       если       [нравится(работать_за_компьютером),
склад_ума(аналитический)] то программист.
    правило2       ::       если       [нравится(работать_за_компьютером),
нравится(творческая_работа), нравится(рисовать)] то цифровой_художник.
    правило3 :: если [нравится(творческая_работа), нравится(рисовать)] то
художник.
    правило4 :: если [нравится(творческая_работа), склад_ума(аналитический),
нравится(работать_за_компьютером), нравится(музыка)] то цифровой_музыкант.
    правило5 :: если [нравится(творческая_работа), нравится(музыка)] то
музыкант.
    правило6 :: если [склад_ума(гуманитарный), предпочитаете(изучать_историю)]
то историк.
    правило7 :: если [склад_ума(гуманитарный), предпочитаете(изучать_право)]
то юрист.
    правило8 :: если [нравится(работать_за_компьютером), нравится(роботы),
нравится(искусственный_интеллект)] то робототехник.
    правило9 :: если [склад_ума(аналитический), нравится(работать_с_железом)]
то робототехник.
    правило10        ::        если        [нравится(работать_с_детьми),
нравится(оказывать_моральную_поддержку)] то педагог_психолог.
    правило11 :: если [нравится(оказывать_медицинскую_помощь)] то врач.
```

```
правило12 :: если [нравится(работать_с_детьми), нравится(оказывать_медицинскую_помощь)] то врач_педиатр.
правило13 :: если [нравится(работать_с_детьми), нравится(заниматься_спортом)] то детский_тренер.
правило14 :: если [нравится(заниматься_спортом)] то спортсмен.

% гипотезы (цели)
h1 :: гипотеза(юрист).
h2 :: гипотеза(программист).
h3 :: гипотеза(педагог_психолог).
h4 :: гипотеза(цифровой_художник).
h5 :: гипотеза(цифровой_музыкант).
h6 :: гипотеза(историк).
h7 :: гипотеза(врач).
h8 :: гипотеза(спортсмен).
h9 :: гипотеза(художник).
h10 :: гипотеза(музыкант).
h11 :: гипотеза(робототехник).
h12 :: гипотеза(врач_педиатр).
h13 :: гипотеза(детский_тренер).

% признаки, истинность которых можно выяснить у пользователя
q1 :: признак(склад_ума(гуманитарный)).
q2 :: признак(склад_ума(аналитический)).
q3 :: признак(нравится(решать_логические_задачи)).
q4 :: признак(нравится(работать_за_компьютером)).
q5 :: признак(нравится(творческая_работа)).
q6 :: признак(нравится(рисовать)).
q7 :: признак(нравится(музыка)).
q8 :: признак(нравится(заниматься_спортом)).
q9 :: признак(предпочитаете(изучать_историю)).
q10 :: признак(предпочитаете(изучать_право)).
q11 :: признак(нравится(работать_с_детьми)).
q12 :: признак(нравится(оказывать_моральную_поддержку)).
q13 :: признак(нравится(оказывать_медицинскую_помощь)).
q14 :: признак(нравится(роботы)).
q15 :: признак(нравится(искусственный_интеллект)).
q16 :: признак(нравится(работать_с_железом)).
```

Интерпретатор классифицирующей ЭС (файл interp.pl):

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Интерпретатор (машина вывода) для ЭС продукционного типа
% Метод вывода: обратный вывод
% Вариант 2: интерпретатор обрабатывает правила, в которых
% предпосылки задаются в виде списка условий.
% Это позволяет в условной части правила, задавать произвольное
% количество условий.
% --------------------------------------------------------------------------
% Примеры правил см. в загружаемой тестовой базе знаний - new_anim.pl
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

:-dynamic
сообщено/2.
```

```
      определить_операторы:-
      op(950, xfx, то),
      op(960, fx, если),
      op(970, xfx, '::').
      :-определить_операторы.

      %============обратный
вывод=================================================
      % реализуется предикатом найти(S,Стек,Д),где S - список проверяемых гипотез,
      % Стек - стек из имен доказываемых гипотез и правил (используется при ответе
на
      % вопросы "почему), Д - дерево вывода целевого утверждения (используется при
отве-
      % те на вопросы "как"). Предикат получает на вход список [H] и Стек=[H] и в
про-
      % цессе обратного вывода строит дерево вывода Д.
      % Предикат "найти" для доказательства отдельных гипотез из списка S
      % использует предикат найти1(H,Стек,Дерево).
      %-------------------------------------------------------------------------
-----

      % случай1:если цель H была подтверждена пользователем,
      % то дерево вывода Д=сообщено(H).
      найти1(H,Стек,сообщено(H)):-сообщено(H,да).
      найти1(H,Стек,сообщено(H)):-запрашиваемая(H),
      not(сообщено(H,_)),спроси(H,Стек).

      % случай2:если цель H подтверждается фактом, уже известным системе,
      % то дерево вывода Д=Факт :: H
      найти1(H,Стек,Факт :: H):-Факт :: H.

      % случай3: если цель H соответствует следствию одного из
      % правил -> Правило :: если H1 то H
      % и если Д1 дерево вывода для подцели H1,
      % то Д= Правило :: если Д1 то H и добавить № правила в Стек
      найти1(H,Стек,Правило :: если Д1 то H):-
      Правило :: если H1 то H,
      найти(H1,[Правило | Стек],Д1).

      % случай4: если доказывается конъюнкция гипотез, заданная списком гипотез,
      % то найти доказательство первой гипотезы H1 из списка
      % с помощью найти1(H1,Стек,Дерево1), а затем найти доказательство оставшихся
      % гипотез T с помощью найти(T,Стек,Дерево) и
      % объединить деревья вывода в общий список [Дерево1 | Дерево].
      найти([],Стек,Дерево):-Дерево=[].
      найти([H1|T],Стек,[Дерево1 | Дерево]):-
      найти1(H1,Стек,Дерево1),найти(T,Стек,Дерево).

      % проверка: является ли гипотеза признаком, значение которого можно спросить
      запрашиваемая(H):-Факт :: признак(H).

      %=========вывод вопросов и обработка ответов "да, нет, почему"
==================
      %вывод вопроса и ввод ответа
      спроси(H,Стек):-write(H),write('?'),nl,
      read(O),ответ(H,O,Стек).

      %обработка ответов: да, нет
```

```
ответ(Н,да,Стек):-assert(сообщено(Н,да)),!.
ответ(Н,нет,Стек):-assert(сообщено(Н,нет)),!,fail.

%обработка ответов - "почему"
% случай1: стек целей пустой
ответ(Н,почему,[]):-!,write(' Вы задаете слишком много вопросов'),nl,
спроси(Н,[]).

%случай2: в стеке осталась только первая введенная цель, т.е доказываемая
гипотеза
ответ(Р,почему,[Н]):-!,write('моя гипотеза: '),
write(Н),nl,спроси(Р,[]).

%случай3: вывод заключения и номера правила для доказываемой текущей подцели
Н
ответ(Н,почему,[Правило | Стек]):-!,
Правило :: если Н1 то Н2,
write('пытаюсь доказать '),
write(Н2),nl,
write('с помощью правила: '),
write(Правило),nl,
спроси(Н,Стек).

%неправильный ответ: повторяем вопрос
ответ(Н,_,Стек):-write(' правильный ответ: да, нет, почему'),nl,
спроси(Н,Стек).

%==============обработка ответов на вопросы
"как?"==============================
% предикат как(Н,Д)- выполняет поиск подцели Н в построенном
% с помощью предиката "найти" дереве вывода Д и отображает соответствующий
% фрагмент дерева вывода, объясняя, как было получено доказательство Н.
% Дерево вывода Д представляет собой последовательность вложенных правил
% в виде списка, например:
% [правило5::если[правило1::если[сообщено(имеет(шерсть))]то млекопитающее,
%                            сообщено(ест_мясо)]то хищник,...]
%-------------------------------------------------------------------------
-----
% поиск целевого утверждения Н в дереве
как(Н,Дерево):-как1(Н,Дерево),!.

% вывод сообщения, если Н не найдено
как(Н,_):-write(Н),tab(2),write('не доказано'),nl.

% случай1: если Н сообщено пользователем,
% то вывести "Н было введено"
как1(Н,_):-сообщено(Н,_),!,
write(Н),write('было введено'),nl.

% случай2: если дерево вывода Д представлено фактом, подтверждающим Н
как1(Н,Факт :: Н):-!,
write(Н), write( 'является фактом'), write(Факт),nl.

% случай3: если дерево вывода Д - правило в заключение, которого есть Н,
% то отобразить это правило
как1(Н,[Правило :: если _ то Н]):-!,
write(Н),write(' было доказано с помощью'),nl,
Правило :: если Н1 то Н,
```

```
отобрази_правило(Правило :: если Н1 то Н).

% случай4: если в дереве Д нет правила с заключением Н,
%то поиск Н надо выполнять в дереве вывода предпосылок, т.е. в Дерево
как1(Н,[Правило :: если Дерево то _]):-как(Н,Дерево).

% случай5: если дерево вывода - список поддеревьев вывода
% каждой конъюнктивной подцели правила из БЗ,
% то поиск Н следует выполнять в каждом из поддеревьев;
% поиск Н следует выполнять сначала в поддереве [Д1], а
% если Н не найдено, то продолжить поиск в оставшихся поддеревьях
как1(Н,[]):-!.
как1(Н,[Д1|Д2]):-как(Н,[Д1]),!;
как1(Н,Д2).

%вывод правила на экран
отобрази_правило(Правило :: если Н1 то Н):-
write(Правило), write( ':'),nl,
write('если '), write(Н1), nl,
write('то '), write(Н),nl.

/* Вызов интерпретатора*/
инициализация:-retractall(сообщено(_,_)).
start:-

/* Загрузка базы знаний из файла*/
reconsult('./nb.pl'),
info,
%отображение информации о базе знаний*
go_exp_sys.
go_exp_sys:-
    инициализация,
    Факт :: гипотеза(Н),
    найти([Н],[Н],Дерево),
    write('решение:'),write(Н),nl,
    объясни(Дерево),
    возврат.
%объяснение вывода утверждения
объясни(Дерево):-write( 'объяснить ? [цель/нет]:'), nl,read(Н),
(Н\=нет,!,как(Н,Дерево),объясни(Дерево));!.
%поиск следующих решений
возврат:-write('Искать ещё решение [да/нет] ?: '),nl, read(нет).
```

Пример выполнения программы:

```
?- start.

**************************************
*          Экспертная система        *
*       для определения будущей      *
*              профессии             *
*------------------------------------*
*        Отвечайте на вопросы:       *
*            да, нет, почему         *
*        Для объяснения решения      *
*            введите цель            *
**************************************
Введите любой символ
```

```
|:
склад_ума(гуманитарный)?
|: нет.
нравится(работать_за_компьютером)?
|: да.
склад_ума(аналитический)?
|: почему.
пытаюсь доказать программист
с помощью правила: правило1
склад_ума(аналитический)?
|: да.
решение:программист
объяснить ? [цель/нет]:
|: нет.
Искать ещё решение [да/нет] ?:
|: да.
нравится(работать_с_детьми)?
|: почему.
пытаюсь доказать педагог_психолог
с помощью правила: правило10
нравится(работать_с_детьми)?
|: нет.
нравится(творческая_работа)?
|: почему.
пытаюсь доказать цифровой_художник
с помощью правила: правило2
нравится(творческая_работа)?
|: да.
нравится(рисовать)?
|: нет.
нравится(музыка)?
|: почему.
пытаюсь доказать цифровой_музыкант
с помощью правила: правило4
нравится(музыка)?
|: да.
решение:цифровой_музыкант
объяснить ? [цель/нет]:
|: цифровой_музыкант.
цифровой_музыкант было доказано с помощью
правило4:
если
[нравится(творческая_работа),склад_ума(аналитический),нравится(работать_за_компьют
ером),нравится(музыка)]
то цифровой_музыкант
```

Выводы

В ходе выполнения лабораторной работы была разработана экспертная система продукционного типа на Прологе, исследованы базовые принципы организации экспертных систем.