Приложение В
(справочное)
Примеры реализации ЭС продукционного типа

В.1. Пример диагностической ЭС

В.1. 1. База знаний диагностической ЭС (файл nb.pl.)

```
info:-
      nl,
      write('*******************************'),nl,
      write('*     Экспертная система      *'),nl,
      write('*      для диагностики        *'),nl,
      write('*       неисправностей        *'),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 диагностики электрической плиты
правило1 :: если лампа(светится) и плита(холодная)
            то нагреватель(неисправен).
правило2 :: если тока(нет)
            то выключатель(не_включен).
правило3 :: если тока(нет)
            то напряжения(нет).
правило4 :: если плита(холодная) и лампа(не_светится)
            то тока(нет).
правило5 :: если лампа(не_светится) и плита(горячая)
            то лампа(неисправна).

% гипотезы неисправности (цели)
h1 :: гипотеза(нагреватель(неисправен)).
h2 :: гипотеза(выключатель(не_включен)).
h3 :: гипотеза(напряжения(нет)).
h4 :: гипотеза(лампа(неисправна)).

% признаки, истинность которых можно выяснить у пользователя
q1 :: признак(лампа(светится)).
q2 :: признак(плита(холодная)).
q3 :: признак(лампа(не_светится)).
q4 :: признак(плита(горячая)).
```

В.1.2. Интерпретатор ЭС для диагностической базы знаний

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Интерпретатор (машина вывода) для ЭС продукционного типа
% Метод вывода: обратный вывод
% Вариант 1: интерпретатор обрабатывает правила, в  которых
% предпосылки задаются в виде условий (не более 2-х), соединенных оператором "и".
% --------------------------------------------------------------------------------
% Примеры правил см. в загружаемой тестовой  базе знаний - nb.pl
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
:-dynamic
сообщено/2.
% объявление операторов
определить_операторы:-
           op(920, xfy, и),
           op(950, xfx, то),
           op(960, fx, если),
           op(970, xfx, '::').
:-определить_операторы.

%===========обратный вывод=========================================================
% реализуется предикатом найти(Н,Стек,Д), где Н - проверяемая гипотеза (цель),
% 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 Н и Стек=[Н] и в процессе обрат-
% ного вывода строит дерево вывода Д.
%---------------------------------------------------------------------------------

% случай1:если цель Н была подтверждена пользователем,
% то дерево вывода Д=сообщено(Н).
найти(Н,Стек,сообщено(Н)):-сообщено(Н,да).
% если цель - это признак, то спроси его значение
найти(Н,Стек,сообщено(Н)):-запрашиваемая(Н),
                           not(сообщено(Н,_)),спроси(Н,Стек).

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

% случай3: если цель Н соответствует следствию одного из
% правил -> Правило :: если Н1 то Н
% и если Д1 дерево вывода для подцели Н1,
% то Д= Правило :: если Д1 то Н и номер правила добавить в Стек
найти(Н,Стек,Правило :: если Д1 то Н):-
                    Правило :: если Н1 то Н,
                    найти(Н1,[Правило | Стек],Д1).

% случай4: если доказывается конъюнкция гипотез Н=Н1 и Н2,
% то дерево вывода Д=Д1 и Д2, где Д1,Д2 - деревья вывода гипотез Н1 и Н2
найти(Н1 и Н2,Стек,Д1 и Д2):-
            найти(Н1,Стек,Д1),найти(Н2,Стек,Д2).

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

%=========вывод вопросов и обработка ответов "да, нет, почему" =================
% вывод вопроса и ввод ответа
спроси(Н,Стек):-write(Н),write('?'),nl,
                read(О),ответ(Н,О,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 только первая введенная цель, т.е. доказываемая гипотеза
```

```
ответ(H,почему,[H1]):-!,write('моя гипотеза: '),
                       write(H1),nl,спроси(H,[]).
% случай3: если в стеке несколько элементов, то вывод заключения (т.е. подцели)
% и номера текущего применяемого правила
ответ(H,почему,[Правило | Стек]):-!,
            Правило :: если H1 то H2,
            write('пытаюсь доказать '),
            write(H2),nl,
            write('с помощью правила: '),
            write(Правило),nl,
            спроси(H,Стек).

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

%==============обработка ответов на вопросы "как?"==============================
% предикат как(H,Д)- выполняет поиск подцели H в построенном
% с помощью предиката "найти" дереве вывода Д и отображает соответствующий
% фрагмент дерева вывода, объясняя, как было получено доказательство H.
% Дерево вывода Д представляет собой последовательность вложенных правил, напри-
% мер:
% правило2::если (правило4::если сообщено(плита(холодная))и
%                                 сообщено(лампа(не_светится))
%                              то тока(нет))
%           то выключатель(не_включен)
% ------------------------------------------------------------------------------

% поиск целевого утверждения H в дереве
как(H,Дерево):-как1(H,Дерево),!.

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

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

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

% случай3: если дерево вывода Д - правило в заключение, которого есть H,
% то отобразить это правило
как1(H,Правило :: если _ то H):-!,
         write(H),write('было доказано с помощью'),nl,
         Правило :: если H1 то H,
         отобрази_правило(Правило :: если H1 то H).

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

% случай5: если предпосылки образуют конъюнкцию,
% то выполнить поиск в поддеревьях Дерево1 и Дерево2
как1(H,Правило :: если Дерево1 и Дерево2 то _):-как(H,Дерево1),как(H,Дерево2).

%вывод правила на экран
отобрази_правило(Правило :: если H1 то H):-
         write(Правило), write( ':'),nl,
         write('если '), write(H1), nl,
         write('то '), write(H),nl.

/* ====================Вызов интерпретатора=================================*/
инициализация:-retractall(сообщено(_,_)).
start:-
      /* Загрузка базы знаний из файла */
      reconsult('D:/Eclipse_prolog_w/exp_sys/src/nb.pl'),
      info,                    %отображение информации о базе знаний*
```

```
      go_exp_sys.

go_exp_sys:-  инициализация,
              Факт :: гипотеза(H),
              найти(H,[H],Дерево),
              write('решение:'),write(H),nl,
              объясни(Дерево),
              возврат.


%объяснение вывода утверждения
объясни(Дерево):-write( 'объяснить ? [цель/нет]:'), nl,read(H),
                  (H\=нет,!,как(H,Дерево),объясни(Дерево));!.

%поиск следующих решений
возврат:-nl,write('Искать ещё решение [да/нет]?: '),nl, read(нет).
```

В.1.3. Протокол работы диагностической ЭС

```
:-start.
*****************************
*     Экспертная система    *
*      для диагностики      *
*       неисправностей      *
*-------------------------- *
*   Отвечайте на вопросы:   *
*    да, нет, почему        *
*   Для объяснения решения  *
*     введите цель          *
*****************************
Введите любой символ

лампа(светится)?
:-нет.
плита(холодная)?
:-да.
лампа(не_светится)?
:-да.
решение:выключатель(не_включен)
объяснить ? [цель/нет]:
:-выключатель(не_включен).
выключатель(не_включен)было доказано с помощью
правило2:
если тока(нет)
то выключатель(не_включен)
объяснить ? [цель/нет]:
:-тока(нет).
тока(нет)было доказано с помощью
правило4:
если плита(холодная)и лампа(не_светится)
то тока(нет)
объяснить ? [цель/нет]:
:-плита(холодная).
плита(холодная)было введено
объяснить ? [цель/нет]:
:-нет.
Искать ещё решение [да/нет]?:
:-да.
решение:напряжения(нет)
объяснить ? [цель/нет]:
:-нет.
Искать ещё решение [да/нет]?:
:-да.
плита(горячая)?
:-почему.
пытаюсь доказать лампа(неисправна)
с помощью правила: правило5
плита(горячая)?
:-почему.
моя гипотеза: лампа(неисправна)
плита(горячая)?
:-почему.
 Вы задаете слишком много вопросов
плита(горячая)?
:-нет.
false.
```

В.2 . Пример классифицирующей ЭС

В.2.1. База знаний классифицирующей ЭС (файл new_anim.pl)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% База знаний ЭС игры "Отгадай животное".
% Суть игры состоит в том, чтобы пользователь задумал какое-либо животное,
% а ЭС, задавая вопросы, пытается отгадать его. ЭС используя введенные признаки
% животного, по сути, выполняет классификацию и относит описываемое животное к
% соответствующему виду.
% База знаний, необходимая для проведения игры, содержит сведения о признаках
% различных видов животных
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
info:-
      nl,
      write('******************************'),nl,
      write('*     Экспертная система      *'),nl,
      write('*   Игра "Отгадай животное    *'),nl,
      write('*                             *'),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(_).

% база продукционных правил игры "отгадай животное"
% предпосылки правил задаются в виде списка условий
% [условие1,условие2,...] означает - условие1 и условие2 и...
правило1 ::  если  [имеет(шерсть)]
             то   млекопитающее.
правило2 ::  если [кормит_детенышей(молоком)]
             то   млекопитающее.
правило3 ::  если [имеет(перья)]
             то  птица.
правило4 ::  если  [летает, откладывает_яйца]
             то птица.
правило5 ::  если [млекопитающее,ест_мясо]
             то хищник.
правило6 ::  если [млекопитающее, имеет(острые_зубы),
                   имеет(острые_когти), имеет(прямо_посаженные_глаза)]
             то  хищник.
правило7 ::  если [млекопитающее,имеет(копыта)]
             то  жвачное.
правило8 ::  если [млекопитающее, жует(жвачку)]
             то   жвачное.
правило9 ::  если [хищник, желто-коричневое, имеет(темные_пятна)]
             то гепард.
правило10 :: если [хищник,желто-коричневое,полосатое,имеет(черные_полосы)]
             то тигр.
правило11 :: если [жвачное, длинношеее, длинноногое,
                   желто-коричневое, имеет(темные_пятна)]
             то   жираф.
правило12 :: если [жвачное,полосатое,имеет(черные_полосы)]
             то  зебра.
правило13 :: если [птица,не_летает,длинношеее,длинноногое,черно-белое]
             то  страус.
правило14 :: если [птица,не_летает,плавает,черно-белое]
             то  пингвин.
правило15 :: если  [птица, хорошо_летает]
             то   альбатрос.

% гипотезы
h1 :: гипотеза(гепард).
h2 :: гипотеза(тигр).
```

```
h3 :: гипотеза(жираф).
h4 :: гипотеза(зебра).
h5 :: гипотеза(страус).
h6 :: гипотеза(пингвин).
h7 :: гипотеза(альбатрос).

% признаки животных, истинность которых можно выяснить у пользователя
q1 :: признак(имеет(шерсть)).
q2 :: признак(кормит_детенышей(молоком)).
q3 :: признак(имеет(перья)).
q4 :: признак(летает).
q5 :: признак(откладывает_яйца).
q6 :: признак(ест_мясо).
q7 :: признак(имеет(острые_зубы)).
q8 :: признак(имеет(острые_когти)).
q9 :: признак(имеет(прямо_посаженные_глаза)).
q11 :: признак(имеет(копыта)).
q12 :: признак(жует(жвачку)).
q13 :: признак(желто-коричневое).
q14 :: признак(имеет(темные_пятна)).
q15 :: признак(полосатое).
q16 :: признак(имеет(черные_полосы)).
q17 :: признак(длинношеее).
q18 :: признак(длинноногое).
q19 :: признак(не_летает).
q20 :: признак(плавает).
q21 :: признак(черно-белое).
q22 :: признак(хорошо_летает).
```

В.2.2. Интерпретатор классифицирующей ЭС

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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(H,да,Стек):-assert(сообщено(H,да)),!.
ответ(H,нет,Стек):-assert(сообщено(H,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('D:/Eclipse_prolog_w/exp_sys/src/new_anim.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).
```

В.2.2. Протокол работы классифицирующей ЭС

```
:-start.
*****************************
*     Экспертная система    *
*   Игра "Отгадай животное  *
*                           *
*-------------------------- *
*   Отвечайте на вопросы:   *
*    да, нет, почему        *
*   Для объяснения решения  *
*     введите цель          *
*****************************
Введите любой символ
имеет(шерсть)?
:-да.
ест_мясо?
:-да.
желто-коричневое?
:-почему.
пытаюсь доказать гепард
с помощью правила: правило9
желто-коричневое?
:-да.
имеет(темные_пятна)?
:-да.
решение:гепард
объяснить ? [цель/нет]:
:-гепард.
гепард было доказано с помощью
правило9:
если [хищник,желто-коричневое,имеет(темные_пятна)]
то гепард
объяснить ? [цель/нет]:
:-хищник.
хищник было доказано с помощью
правило5:
если [млекопитающее,ест_мясо]
то хищник
объяснить ? [цель/нет]:
:-млекопитающее.
млекопитающее было доказано с помощью
правило1:
если [имеет(шерсть)]
то млекопитающее
объяснить ? [цель/нет]:
:-ест_мясо.
ест_мясобыло введено
объяснить ? [цель/нет]:
:-нет.
Искать ещё решение [да/нет] ?:
:-да.
кормит_детенышей(молоком)?
:-да.
решение:гепард
объяснить ? [цель/нет]:
:-нет.
Искать ещё решение [да/нет] ?:
:-да.
имеет(острые_зубы)?
:-да.
имеет(острые_когти)?
:-да.
имеет(прямо_посаженные_глаза)?
:-да.
решение:гепард
объяснить ? [цель/нет]:
:-нет.
Искать ещё решение [да/нет] ?:
:-да.
```

```
решение:гепард
объяснить ? [цель/нет]:
:-гепард.
гепард было доказано с помощью
правило9:
если [хищник,желто-коричневое,имеет(темные_пятна)]
то гепард
объяснить ? [цель/нет]:
:-хищник.
хищник было доказано с помощью
правило6:
если [млекопитающее,имеет(острые_зубы), имеет(острые_когти),
      имеет(прямо_посаженные_глаза)]
то хищник
объяснить ? [цель/нет]:
:-нет.
Искать ещё решение [да/нет] ?:
:-нет.
true
```

Заказ № _____ от « _____ » _____202 г. Тираж _____экз.
Изд-во СевГУ