Основы программирования на языке Пролог.

Лектор:
доцент каф. АОИ
Салмина Нина Юрьевна

Преобразования типов

str_int (S, X)

str_real (S, X)

S – строка

X – число (целое / вещественное)

«Семейные отношения», структуры данных

```
domains
pol = man; woman
date = day (integer day, integer month, integer year)
ass = string*

predicates
Person (string fam, symbol name, pol, date birthday)
Child (string fam, ass)
Family (string fam, string name_husband, string name_wife,
                                    ass childrens)
age_husband (string fam, date data_today, integer age)
Age (date date_today, date birthday, integer age)
```

«Семейные отношения», без предупреждений и повторов

```
clauses
Family (ivanov, ivan, lena, [olga,peter]).
Family (sidorov, david, olga, [vera, igor]).
child(X,Y) :- family(X,_,_,Y).
Person (ivanov, "ivan", man, day(5,1,1960)).
Person (ivanov, lena, woman, day(21,4,1962)).
Person (ivanov, olga, woman, day(13,12,1990)).
Age (day(_,_,G), day(_,_,C),_) :- G<C, fail, !.
Age (day(_,M,G), day(_,B,C), Y)  :-  B>M, !, Y=G-C-1.
Age (day(_,M,G), day(_,B,C), Y)  :-  B<M, !, Y=G-C.
Age (day(D,_,G), day(A,_,C), Y)  :-  D>A, !, Y=G-C-1.
Age (day(_,_,G), day(_,_,C), Y)  :- Y=G-C.
age_husband (X,C,Y) :- family(X,Z,_,_), person(X,Z,_,A), age(C,A,Y).
goal
age_husband (ivanov, day(1,9,2017), Y).
```

«Семейные отношения», вычисление среднего возраста всех мужчин/женщин

```
ave_age (X, D,Y) :- sum_age (X, D, [ ], 0, S, 0, K), Y=S\K.

sum_age (X, D, L, SN, S, KN, K) :- person (N, _, X, _),
                not (member (N, L)), !,
                age (D, N, Y),
                KS = KN+1, SS = SN+Y,
                sum_age (X, D, [N| L], SS, S, KS, K).
sum_age (_, _, _, X, X, Y, Y).
goal
ave_age (man, day(1,10,2017), Y).
```

Выбор информации из набора фактов

FINDALL (X, Y, L)

X – аргумент, значение которого заносится в L

Y – предикат, из которого выбирается информация

L – список значений X, выбранных из всех найденных предикатов Y

например:

findall (X, person(_, _, man, X), L)

L – список дат рождений всех мужчин

«Семейные отношения», вычисление среднего возраста всех мужчин/женщин

```
ave_age (X, D, Z) :- findall (Y, person(_, _, X, Y), L),
                     sum_age (D, L, S, K), Z=S/K.

sum_age (D, [X | T], S, K) :- sum_age (D, T, S1, K1),
                              age (D, X, Y),
                              K = K1+1, S = S1+Y.
sum_age (_, [ ], 0, 0).
goal
ave_age (man, day(1,10,2017), Y).
```

Работа с динамической БД фактов. Хранение фактов в файле

Facts [– general / <имя секции фактов>]

% описание предикатов - фактов

<предикат (тип_арг1, …, тип_аргN)>

. . .

Goal

consult (“<имя файла>” [,< имя секции фактов >]), …

Несколько БД – имя секции обязательно!

Загружает факты из текстового файла в заданную область ОП

Работа с динамической БД фактов. Удаление фактов.

retract (<факт> [,< имя секции фактов >**])**

retractall (<факт> [,< имя секции фактов >**])**

Примеры:

retract (person (_, _, man, _)) – удалит 1-й найденный факт

retractall (person (_, _, man, _)) – удалит ВСЕХ мужчин

Работа с динамической БД фактов. Добавление фактов.

assert (<факт> [,< имя секции фактов >**])** – в конец БД

assertz (<факт> [,< имя секции фактов >**])** – в конец БД

asserta (<факт> [,< имя секции фактов >**])** – в начало БД

Изменение факта (путем замены):

retract (person (ivan, X, data(Y, Z, G))),G1=G-1,

assert (person (ivan, X, data(Y, Z, G1))).

Работа с динамической БД фактов. Сохранение фактов.

Если после работы надо СОХРАНИТЬ все изменения:

save (“<имя файла**>” [,**< имя секции фактов >**])**

!! Без данного предиката после окончания работы изменения в БД не сохраняются !!

Выезд на бал

Андрей Иванович, Федор Петрович, Валерий Сергеевич и Григорий Алексеевич сопровождают своих дочерей на бал.

Заключительный танец каждая девушка танцевала не со своим отцом. Образовались следующие пары:

- Лена с Андреем Ивановичем,
- Анна с отцом Кати,
- Таня с отцом Анны,
- Федор Петрович с дочерью Валерия Сергеевича,
- Валерий Сергеевич с дочерью Андрея Ивановича.

Кто кому приходится дочерью?

Вопрос – допустимая перестановка

```
question(L,S) :- another (L,S),
        may_be ( [ai, fp, vs, ga], S).

goal

question( [lena, any, katy, tany],Y).
```

Вспомогательные предикаты

```
% выбор из списка N-го элемента
n_elem (1, [X | _ ], X) :- !.
n_elem (N, [ _ | T], X) :- N1=N-1,
                    n_elem(N1,T,X).
% определить номер элемента в списке
douther (X, [X | _ ], 1) :- !.
douther (X, [ _ | T], N1) :- douther (X, T, N),
                      N1=N+1.
```

Работа с динамической БД фактов.

Facts

% факт, определяющий родственную пару
relatives (string douther, string father)

% факт, определяющий танцующую пару
para (string she, string he)

Определение пары танцующих

```
% пара танцующих уже определена
f_para(X,Y):-para(X,Y),!.
% девочка/отец уже танцует с другим
f_para(X,_):-para(X,_),!,fail.
f_para(_,Y):-para(_,Y),!,fail.
% пара танцующих не родственники – добавить в БД
f_para (X,Y) :- not (relatives(X,Y)),
                assert (para(X,Y)).
```

Проверка допустимости выбранной перестановки

```
may_be(F,[D1,D2,D3,D4]) :- assert (relatives(D3,vs)),
                           assert (relatives(D2,fp)),
                           assert (relatives(D1,ai)),
                           assert (relatives(D4,ga)),
          f_para(D3,fp), f_para(D1,vs), f_para(lena,ai),
          douther(any,[D1,D2,D3,D4],X2),
          douther(katy,[D1,D2,D3,D4],X3),
          n_elem(X3,F,Yk), f_para(any,Yk),
          n_elem(X2,F,Ya), f_para(tany,Ya),!.

may_be (_, _) :- retractall (relatives(_, _)),
          retractall (para(_, _)), fail.
```

Результат

Y = ["tany", "katy", "any", "lena"]

1 Solution

соответствующий список отцов

[ai, fp, vs, ga],

Задача классификации объектов

В базе данных содержатся результаты теннисных партий, сыгранных членами некоторого клуба:

Победил (Победитель, Проигравший).

Необходимо определить отношение

Класс (Игрок, Категория)

где победитель – игрок, победивший во всех сыгранных им играх;

боец – игрок, в некоторых играх победивший, в некоторых – проигравший;

спортсмен – игрок, проигравший во всех сыгранных им играх.

Информация в файле

won("ivan","petr")

won("ivan","tom")

won("ivan","jim")

won("jim","tom")

won("jim","petr")

won("petr","tom")

Описание термов и предикатов

```
domains
sp1 = string*
iss = winner; athlete; fighter
facts - winner
won (symbol, symbol)
predicates
klass (string, iss)          % определение класса
del_double (sp1,sp1)          % удаление повторных имен
member (string,sp1)
write_klass (sp1,iss)      % печать списка спортсменов
                    % заданного класса
all                    % основная функция
```

Задача классификации объектов. Формулировки правил по категориям. Замена НЕ на ИНАЧЕ

Если X победил кого-либо и X был кем-то побежден,

то X – боец,

иначе, если X победил кого-либо,

то X – победитель,

иначе, если X был кем-то побежден,

то X – спортсмен.

Классификация объектов. Программа.

```
clauses
klass (X, fighter) :- won ( X, _), won (_, X), !.
klass (X, winner) :- won (X, _), !.
klass (X, athlete) :- won (_, X).
del_double ([X | XT], [X | L]) :- not (member (X, XT)), !,
                          del_double (XT, L).
del_double ([_ | XT], L) :- del_double (XT, L), !.
del_double ([ ], [ ]).
write_klass ([L | LT], X) :- klass (L,Y), Y=X, !,
                    write (L,"  \n"), write_klass (LT, X).
write_klass ([_ | LT], X) :- write_klass (LT, X).
write_klass (_ ,_ ) :- nl.
```

Классификация объектов. Программа.

```
all :- findall (X, won (X, _), L1), del_double (L1, L),
       write ("winners:\n"), write_klass (L, winner),
       write ("fighters:\n"), write_klass (L, fighter),
       findall (Y, won (_, Y), L2), del_double (L2, L3),
       write ("athletes:\n"), write_klass (L3, athlete).

goal
consult (“competition", winner), all.
```

Результат работы программы

winners:
ivan

fighters:
jim
petr

athletes:
tom

yes