Функциональное и логическое программирование

Лекция 7

2.11.3.3 Быстрая сортировка

Пример 12:

Разбиваем список на два списка по разделителю – голове списка, сортируем эти списки и соединяем их.

```
q_sort([Head|Tail],Sort_list):-split(Head,Tail,Less,More),
                               q_sort(Less,Sort_less),
                               q_sort(More,Sort_more),
                               append(Sort_less,[Head|Sort_more],Sort_list).
q_sort([],[]).
```

Временная сложность алгоритма быстрой сортировки примерно $n \cdot \log n$.

Есть встроенные предикаты сортировки по неубыванию: sort (с удалением дубликатов), msort.

2.11.4 Компоновка данных в список

bagof(X,P,L)

- порождает список L всех объектов X, удовлетворяющих цели P.

(X и P содержат общие переменные). Если таких объектов нет, то bagof неуспешен.

Если один и тот же X найден многократно, то все его экземпляры будут занесены в L, что приведет к появлению в L повторяющихся элементов.

Пример 13:

Опишем предикат буква для разбиения букв из некоторого множества на гласные и согласные.

```
буква(д,согласная).
буква(я,гласная).
буква(а,гласная).
буква(ц,согласная).
буква(о,гласная).
буква(з,согласная).
буква(л,согласная).
```

Список согласных:

```
?- bagof(X,буква(X,согласная),L).
L = [д, ц, з, л] ;
false.
```

Списки гласных и согласных:

```
?- bagof(X,буква(X,Y),L).
Y = гласная,
L = [я, а, о] ;
Y = согласная,
L = [д, ц, з, л] ;
false.
```

setof(X,P,L)

- аналогичен bagof.

Отличие от bagof: список L упорядочен по отношению ‘@<', и не содержит повторяющихся элементов.

Пример 14:

Список пар вида <буква>/<вид>.

?- setof(X/Y,буква(X,Y),L).

L = [а/гласная, д/согласная, з/согласная, л/согласная, о/гласная, ц/согласная, я/гласная] ;

false.

findall(X,P,L)

- аналогичен предикату bagof.

Отличия от bagof:

- Собирает в список все объекты X, не обращая внимание на возможно отличающиеся для них конкретизации тех переменных из P, которых нет в X.
- Если объекты не найдены, то предикат успешен, а список L – пустой.

Пример 15:

Имеется предикат ребенок, связывающий имя ребенка и его возраст. Сформировать список детей старше 5 лет.

```
ребенок(ира,5).
ребенок(марина,4).
ребенок(паша,8).
ребенок(игорь,10).
ребенок(леша,3).
ребенок(максим,6).
ребенок(оля,9).

?- findall(X,(ребенок(X,Y),Y>5),L).
L = [паша, игорь, максим, оля] ;
false.
```

Пример 16:

Найти средний возраст всех детей.

```
goal:-findall(X,ребенок(_,X),L),sr(L,S,N),Sred is S/N, writeln(Sred).
sr([],0,0).
sr([Head|Tail],S,N):-sr(Tail,S1,N1),S is S1+Head,N is N1+1.

?- goal.
6.428571428571429
```

2.12 Решение логических задач с использованием списков

2.12.1 Задача о фермере, волке, козе и капусте

Фермер (**farmer**), волк (**wolf**) , коза (**goat**) и капуста (**cabbidge**) находятся на одном берегу. Всем надо перебраться на другой берег на лодке. Лодка перевозит только двоих. Нельзя оставлять на одном берегу козу и капусту (коза съест капусту), козу и волка (волк съест козу).

Процесс перевозки - последовательность состояний state с четырьмя аргументами, каждый из которых может принимать 2 значения: left или right и отражает соответственно нахождение фермера, волка, козы и капусты.

Например, state(left,right,left,right) отражает, что фермер и коза находятся на левом берегу, а волк и капуста – на правом.

Фермер может перевести на другой берег, кроме себя, либо волка, либо козу, либо капусту.

Для описания перевозки определим предикат move, описывающий переходы из одного состояния в другое. Для переправления с одного берега на другой введем предикат opposite, который определяет сторону берега, противоположную исходной.

Например, перевозка волка на другой берег:

```
move(state(X,X,G,C),state(Y,Y,G,C)):-opposite(X,Y).
```

С помощью предиката danger определим опасные состояния (волк и коза на одном берегу, а фермер на другом или коза и капуста на одном берегу, а фермер на другом).

Предикат path формирует список из последовательности состояний, решающих поставленную задачу.

Будем добавлять в результирующий список новое состояние, если в него возможен переход из текущего состояния, оно не опасное и не содержится в сформированной части списка (для избегания зацикливания).

```
goal:-S=state(left,left,left,left),
      G=state(right,right,right,right),
      path(S,G,[S],L),reverse(L,L1),
      writeln('Результат='),writeln(L1),fail.
goal.
move(state(X,X,G,C),state(Y,Y,G,C)):-opposite(X,Y).
move(state(X,W,X,C),state(Y,W,Y,C)):-opposite(X,Y).
move(state(X,W,G,X),state(Y,W,G,Y)):-opposite(X,Y).
move(state(X,W,G,C),state(Y,W,G,C)):-opposite(X,Y).
opposite(left,right).
opposite(right,left).
danger(state(F,_,X,X)):-opposite(F,X).
danger(state(F,X,X,_)):-opposite(F,X).
path(G,G,L,L).
path(S,G,L,L1):-move(S,S1),
                not(danger(S1)),not(member(S1,L)),
                path(S1,G,[S1|L],L1).
```

?- goal.

Результат=

[state(left,left,left,left),state(right,left,right,left),state(left,left,right,left),
state(right,right,right,left),state(left,right,left,left),
state(right,right,left,right),state(left,right,left,right),
state(right,right,right,right)]

Результат=

[state(left,left,left,left),state(right,left,right,left),state(left,left,right,left),
state(right,left,right,right),state(left,left,left,right),state(right,right,left,right),
state(left,right,left,right),state(right,right,right,right)]

2.12.2 Решение числовых ребусов

$$\begin{array}{r} \text{APAPA} \\ +\ \text{SLEEPY} \\ \hline \text{ETUDES} \end{array}$$

Задача состоит в том, чтобы заменить все буквы цифрами. Одинаковым буквам должны соответствовать одинаковые цифры, а разным буквам – разные цифры.

Определим предикат sum(N1,N2,N), который истинен, если существует такая замена букв цифрами в словах N1, N2, N, что N1+N2=N.

Числа будем представлять списком цифр, из которых оно состоит. Будем считать, что все списки имеют одинаковую длину.

При выполнении сложения столбиком суммирование цифр чисел выполняется справа налево с учетом переноса из предыдущего разряда. Следовательно, нужно хранить информацию о переносе из предыдущего разряда и о переносе в следующий разряд.

Разным буквам должны соответствовать разные цифры, значит при суммировании цифр нужна информация о цифрах, доступных до и после сложения (эта информация будет представляться двумя списками). Заведем вспомогательный предикат sum1:

sum(N1,N2,N, P_before,P_after, Cifri_before,Cifri_after)

Определим предикат sum1:

- Если у всех трех чисел имеется хотя бы одна цифра, то суммируем числа без самой левой цифры и, используя список оставшихся цифр и перенос из предыдущего разряда, суммируем самые левые цифры и добавляем полученную сумму в результат суммирования без самой левой цифры. Для суммирования цифр напишем предикат sumc.

- Если все три списка пустые, то переносов разрядов нет, и списки цифр до и после сложения совпадают. Это соответствует правилу остановки рекурсии.

Для суммирования цифр определим предикат sumc.

sumc(D1,D2,D,B, A,LB,LA)

D1,D2 – цифры, которые складываются

D – цифра результата в этом разряде

B – перенос из предыдущего разряда

A – перенос в следующий разряд

LB – список цифр для выбора до сложения

LA - список цифр для выбора после сложения

Первые три аргумента предиката должны быть цифрами. Если какая-нибудь из этих переменных не конкретизирована, то ее необходимо конкретизировать какой-нибудь цифрой из списка LB. После конкретизации цифру следует удалить из списка доступных цифр. Если переменная уже имела значение, то удалять из списка доступных цифр ничего не надо.

Для получения значения неозначенной переменной будем использовать предикат delete1.

Если цифра не конкретизирована, то ее надо конкретизировать цифрой из списка цифр, которые еще не использовались, и удалить эту цифру из списка доступных цифр. Если цифра конкретизирована (проверка предикатом nonvar), то список доступных цифр не изменятся.

```
sum(N1,N2,N):-sum1(N1,N2,N,0,0,[0,1,2,3,4,5,6,7,8,9],_).
sum1([],[],[],0,0,L,L).
sum1([D1|N1],[D2|N2],[D|N],C1,C,L1,L):-
                                        sum1(N1,N2,N,C1,C2,L1,L2),
                                        sumc(D1,D2,D,C2,C,L2,L).
sumc(D1,D2,D,C2,C,L2,L):- delete1(D1,L2,L3),
                          delete1(D2,L3,L4),
                          delete1(D,L4,L),
                          S is D1+D2+C2,
                          D is S mod 10,
                          C is S//10.
delete1(X,L,L):-nonvar(X),!.
delete1(X,[X|L],L).
delete1(X,[Y|L],[Y|L1]):- delete1(X,L,L1).
```

```
?- sum([0,A,P,A,P,A], [S,L,E,E,P,Y], [E,T,U,D,E,S]).
A = 2
P = 8
S = 5
L = 7
E = 6
Y = 3
T = 0
U = 4
D = 9 ;
false
```

```
  БАЙТ
+ БАЙТ
-------
 СЛОВО
```

?- sum([0,Б,А,Й,Т], [0,Б,А,Й,Т], [С,Л,О,В,О]).

Б = 3

А = 6

Й = 4

Т = 1

С = 0

Л = 7

О = 2

В = 8;

…….

false

Этот ребус имеет 14 решений.