Функциональное и логическое программирование

Лекция 6

2.11 Списки

Список – упорядоченный набор объектов (термов). Список может содержать объекты разных типов, в том числе и списки. Элементы списка разделяются запятыми и заключаются в квадратные скобки.

Пример 1:

[a, b, 1, 4] , [[], a, [3, 4]]

2.11.1 Голова и хвост списка

Голова списка – первый элемент.
Хвост списка – часть списка без первого элемента.

Пример 2 (деление списка на хвост и голову):

Список	Голова	Хвост
[1,2,3,4]	1	[2,3,4]
[a]	a	[]
[ ]	не опр	не опр

Деление на голову и хвост осуществляется с помощью специальной формы представления списка: [Head|Tail].

Пример 3 (Сопоставление списков):

Список 1	**Список 2**	**Результаты сопоставления**
[X,Y,Z]	[1,2,3]	X=1, Y=2, Z=3
[5]	[X|Y]	X=5, Y=[]
[1,2,3,4]	[X,Y|Z]	X=1, Y=2, [3,4]
[1,2,3]	[X,Y]	false
[a,X|Y]	[Z,a]	Z=a, X=a, Y=[]
[a,b,c,d,e,f]	[_,_,X|_]	X=c

2.11.2 Операции со списками
2.11.2.1 Принадлежность элемента списку

Пример 4:

```
member1(X,[X|_]).
member1(X,[_|T]):-member1(X,T).
```

При использовании можно задавать значения одного, или двух аргументов, или не задавать их вообще.

```
?- member1(1,[2,3,1,1,4,1,9]).
true .

?- member1(1,X).
X = [1|_]
Unknown action: " (h for help)
Action? ;
X = [_, 1|_] ;
X = [_, _, 1|_] ;
X = [_, _, _, 1|_] .
```

```
?- member1(X,[1,2,3,4,5]).
X = 1 ;
X = 2 ;
X = 3 ;
X = 4 ;
X = 5 ;
false.

?- member1(X,Y).
Y = [X|_] ;
Y = [_, X|_] ;
Y = [_, _, X|_] .
```

Есть встроенный предикат member.

2.11.2.2 Соединение двух списков (аналог append)

Пример 5:

```
append1([],L2,L2).
append1([X|T],L2,[X|T1]):-append1(T,L2,T1).
```

```
?- append1([1,2,3],[4,5],L).
L = [1, 2, 3, 4, 5].
```

Есть встроенный предикат append.

Можно использовать предикат append1 для следующих целей:

- слияние двух списков;
- получение всех возможных разбиений списка;
- поиск подсписков до и после определенного элемента;
- поиск элементов списка, стоящих перед и после определенного элемента;
- удаление части списка, начиная с некоторого элемента;
- удаление части списка, предшествующей некоторому элементу.

Есть встроенный предикат append.

Вопрос в Пролог-системе	Ответ Пролог-системы
append1([1,2],[3],L).	L = [1,2,3]
append1(L1,L2,[1,2,3]).	L1 = [], L2 = [1,2,3]; L1 = [1], L2 = [2,3]; L1 = [1,2], L2 = [3]; L1 = [1,2,3], L2 = []; false
append1(Before,[3|After],[1,2,3,4,5]).	Before = [1,2], After = [4,5]
append1(_,[Before,3,After|_],[1,2,3,4,5]).	Before = 2, After = 4
append1(L1,[3|_],[1,2,3,4,5]).	L1 = [1,2]
append1(_,[3|L2],[1,2,3,4,5]).	L2 = [4,5]

2.11.2.3 Добавление и удаление элемента из списка

Пример 6 (добавление в начало списка, удаление первого вхождения заданного элемента):

```
insert(X,L,[X|L]).
select1(_,[],[]).
select1(X,[X|T],T).
select1(X,[Y|T],[Y|T1]):-select1(X,T,T1).
```

```
?- select1(a,[d,a,g,a,a,h,a],L).
L = [d, g, a, a, h, a] ;
L = [d, a, g, a, h, a] ;
L = [d, a, g, a, h, a] ;
L = [d, a, g, a, a, h] ;
L = [d, a, g, a, a, h, a] ;
```

Предикат select1 можно использовать также для добавления элемента в список. Есть встроенный предикат select.

Пример 7 (удаление всех вхождений заданного элемента):

```
delete1([],_,[]):-!.
delete1([X|T],X,L):-delete1(T,X,L),!.
delete1([Y|T],X,[Y|T1]):-delete1(T,X,T1).
```

```
?- delete1([d,a,g,a,a,h,a],a,L).
L = [d, g, h].
```

Есть встроенный предикат delete.

2.11.2.4 Деление списка на два списка по разделителю

Пример 8:

Деление списка на две части, используя разделитель M (если элемент исходного списка меньше разделителя, то он помещается в первый результирующий список, иначе – во второй результирующий список).

```
plit(_,[],[],[]).
split(M,[H|T],[H|T1],L2):-H@<M,!,split(M,T,T1,L2).
split(M,[H|T],L1,[H|T2]):-split(M,T,L1,T2).
```

```
?- split(4,[1,9,3,8,2,0,3],L1,L2).
L1 = [1, 3, 2, 0, 3],
L2 = [9, 8]
```

2.11.2.5 Подсчет количества элементов в списке

Пример 9:

```
count([],0).
count([_|T],N):- count(T,N1), N is N1+1.
```

Есть встроенный предикат length(L,N) – подсчет количества элементов N в списке L.

reverse(L1,L2) – обращение любого из списков-аргументов.

2.11.3 Сортировка списков (по неубыванию)
2.11.3.1 Сортировка вставкой

Пример 10:

Добавляем голову списка в нужное место отсортированного хвоста.

```
in_sort([],[]).
in_sort([H|T], Sort_list):- in_sort(T, Sort_T),
                            add(H, Sort_T, Sort_list).

add(X, [], [X]).
add(X, [Y|T], [Y|T1]):- X @> Y, add(X, T, T1).
add(X, L, [X|L]).
```

2.11.3.2 Пузырьковая сортировка

Пример 11:

Меняем местами соседние элементы до тех пор, пока есть неверно упорядоченные пары.

```
pu_sort(L, Sort_list):- swap(L,L1), !, pu_sort(L1, Sort_list).
pu_sort(L, L).
swap([X,Y|T],[Y,X|T]):- X@>Y.
swap([X|T],[X|T1]):- swap(T,T1).
```

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([],[]):-!.
q_sort([H|T],Sort_list):-split(H,T,Less,More),q_sort(Less,Sort_less),
                         q_sort(More,Sort_more),
                         append(Sort_less,[H|Sort_more],Sort_list).


?- q_sort([4,9,1,7,2,a,r,t,6],L).
L = [1, 2, 4, 6, 7, 9, a, r, t] .
```

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(я,гласная).
буква(л,согласная).
буква(а,гласная).
буква(и,гласная).
буква(б,согласная).
буква(ф,согласная).
```

Список согласных:

```
?- bagof(X,буква(X,согласная),L).
L = [л, б, ф].
```

Списки гласных и согласных:

```
?- bagof(X,буква(X,Y),L).
Y = гласная,
L = [я, а, и] ;
Y = согласная,
L = [л, б, ф].
```

```
?- bagof(X,буква(X,со),L).
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, б/согласная, и/гласная, л/согласная, ф/согласная, я/гласная].

?- bagof(X/Y,буква(X,Y),L).
L = [я/гласная, л/согласная, а/гласная, и/гласная, б/согласная, ф/согласная].
```

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.

```
child(a,6).
child(b,7).
child(c,10).
child(d,16).
child(e,3).
child(f,2).
```

```
?- findall(X,(child(X,Y),Y>5),L).
L = [a, b, c, d].
```

Пример 16:

Найти средний возраст всех детей.

```
goal:-findall(X,child(_,X),L),sum(L,S,N),Sr is S/N,writeln(Sr).
sum([],0,0).
sum([H|T],S,N):-sum(T,S1,N1),S is S1+H,N is N1+1.
```

```
?- goal.
7.333333333333333
true.
```