Списъци

С помощта на термовете в ПРОЛОГ могат да се дефинират списъци с двойни кутии, подобно на езиците за процедурно програмиране по следния начин:

```
node(1,node(2,node(3,node(4,null))))
```

Този синтаксис не е удобен, затова е въведен по-удобния запис са представяне на списъци:

```
[1,2,3,4]
```

Списъците в пролог се представят с двойни кутии със служебен функтор '.', като за край се използва специалната константа за празен списък []. Така горният запис се възприема от ПРОЛОГ така:

```
'.'(1,'.'(2,'.'(3,'.'(4,[]))))
```

Съдържанието на двойната кутия може да се разглежда като глава (информационна част) и опашка (указателна част) на списъка. За разбиване на списък на глава и опашка се използва символът |. Така горният списък може да бъде записан и по следните еквивалентни начини:

```
[1|[2,3,4]]
[1|[2|[3,4]]]
[1|[2|[3|[4]]]]
[1|[2|[3|[4|[]]]]]
```

Последните три могат да се запишат съкратено по следния начин:

```
[1,2|[3,4]]
[1,2,3|[4]]
[1,2,3,4|[]]
```

Списъкът [1,[2]] се различава от списъка [1|[2]]. Първият списък се състои от два елемента - 1 и [2], докато вторият списък се състои от двете числа 1 и 2. Празният списък не може да се разбие на глава и опашка. Обектът [1|3] не е списък. Така можем да дефинираме списък индуктивно по следния начин:

- [] е списък
- ако T е списък, H е произволен терм, то [H|T] е списък.

Следвайки тази дефиниция можем да напишем рекурсивни предикати за принадлежност на елемент към списък и слепване на два списъка:

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

```
append([],X,X).
append([H|T],X,[H|Y]) :- append(T,X,Y).
```

И за двата предиката разбиваме изчислението на два случая:

- базов (лесен) случай
- рекурсивен случай, в който свеждаме списъка до неговата опашка.

Тази схема обикновено се прилага при повечето предикати за списъци.