Логическое программирование

Кевролетин В.В. группа c8403a(246)

17 May 2012

Содержание

1 Задание 20

1.1 Условие

Написать программы для отношений occurrences/3 (9.2i) и position/3 (9.2ii).

1.2 occurrences/3

1.2.1 Исходный код

Если 2 терма равны, то число вхождений равно 1. Если терм, в котором происходит поиск составной, то следует рекурсивно применить поиск для всех его аргументовв. В реализации введены 2 вспомогательных терма ocurences/4 - накапливает текущий результат в аргументе CurrnN и вызывает проверку для каждого аргумента составного терма. ocurences/5 итеративно применяет проверку для каждого аргумента.

```
ocurences(Sub, Term, N):-
        ocurences(Sub, Term, 0, N).

ocurences(Sub, Term, CurrnN, N):-
        compound(Term),
        functor(Term, _, ArgsCnt),
        ocurences(Sub, Term, ArgsCnt, CurrnN, N).

ocurences(Term, Term, CurrN, NewCurrN) :-
        NewCurrN is CurrN + 1.

ocurences(_, _, CurrN, CurrN).
```

```
ocurences(_, _, 0, CurrN, CurrN).

ocurences(Sub, Term, ArgNum, CurrN, N) :-
        ArgNum > 0,
        compare(=, Sub, Term),
        NewCurrN is CurrN + 1,
        NextArg is ArgNum - 1,
        ocurences(Sub, Term, NextArg, NewCurrN, N).

ocurences(Sub, Term, ArgNum, CurrN, N) :-
        ArgNum > 0,
        \+ compare(=, Sub, Term),
        arg(ArgNum, Term, Arg),
        ocurences(Sub, Arg, SubInArg),
        NewCurrN is CurrN + SubInArg,
        NextArg is ArgNum - 1,
        ocurences(Sub, Term, NextArg, NewCurrN, N).
```

1.2.2 Тесты

Для написания тестов использовалось расширение языка от SWI-Prolog:

```
:- begin_tests(ocurences).

test(t01) :-
        ocurences(a, a, 1), !.
test(t02) :-
        ocurences(a, b, 0), !.
test(t03) :-
        ocurences(a, [a, a], 2), !.
test(t04) :-
        ocurences(a, node(node(a, a), node(a, null)), 3), !.
test(t05) :-
        ocurences(a, node(null, null), 0).
test(t06) :-
        ocurences(node(a, b), node(node(a, b), node(a, node(a, b))), 2).

:- end_tests(ocurences).
```

1.3 position/3

1.3.1 Исходный код

Будем восстанавливать порядок обхода снизу вверх:

```
position(Term, Term, []).

position(Sub, Term, Result) :-
        compound(Term),
        functor(Term, _, N),
```

```
        position(Sub, Term, N, Result).

position(Sub, Term, N, Result) :-
        N > 1,
        N1 is N - 1,
        position(Sub, Term, N1, Result).

position(Sub, Term, N, [N | Result]) :-
        arg(N, Term, Arg),
        position(Sub, Arg, Result).
```

1.3.2 Тесты

```
?- position(null, null, []).
true

?- position(a, node(a), [1]).
true

?- position(a, node(b,node(a,a)), X).
X = [2, 1]
X = [2, 2]
false.

?- position(a, node(b, c), X).
false.

?-
```