Логическое программирование

Кевролетин В.В. группа c8403a(246)

19 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 аргументов. В реализации введены 2 вспомогательных терма: compare_terms сравнивает 2 терма на равенство и возвращает 0 или 1. compare_arguments итеративно применяет проверку для каждого аргумента составного терма.

ocurences(Sub, Term, N) истина, если N - число вхождений терма Sub в терм Term

Sub произвольный тип

Term произвольный тип

N число

compare_terms(Term1, Term2, X) истина, если 2 терма равны и X = 1, либо не равны и X = 0

Term1 произвольный тип

Term2 произвольный тип

X число

compare_arguments(Sub, Term, ArgNum, CurrN, N) истина, если N - сумма числа вхождений терма Sub в первые ArgNum аргнументов терма Term и числа CurrN

Sub произвольный тип

Term произвольный тип

ArgNum число

CurrN число

N число

```
ocurences(Sub, Term, N) :-
        compound(Term),
        functor(Term, _, ArgCnt),
        compare_terms(Sub, Term, X),
        compare_arguments(Sub, Term, ArgCnt, X, N).

ocurences(Sub, Term, N) :-
        \+ compound(Term),
        compare_terms(Sub, Term, N).

compare_terms(Term, Term, 1).

compare_terms(Term1, Term2, 0) :-
        \+ compare(=, Term1, Term2).

compare_arguments(_, _, ArgNum, CurrN, CurrN) :-
        =(ArgNum, 0).

compare_arguments(Sub, Term, ArgNum, CurrN, N) :-
        >(ArgNum, 0),
        arg(ArgNum, Term, Arg),
        ocurences(Sub, Arg, ArgN),
        (NewN is ArgN + CurrN),
        (NextArgNum is ArgNum - 1),
        compare_arguments(Sub, Term, NextArgNum, NewN, N).
```

1.2.2 Тесты

- ocurences(+, +, +)

```
?- ocurences(a, a, 1).

true

?- ocurences(a, b, 0).

true.
```

```
?- ocurences(a, [a, a], 2).
```

true

```
?- ocurences(a, node(node(a, a), node(a, null)), 3).
```

true

```
?- ocurences(a, node(null, null), 0).
```

true

```
?- ocurences(node(a, b), node(node(a, b), node(a, node(a, b))), 2).
```

true

```
?- ocurences(node(a, b), node(node(a, b), node(a, node(a, b))), 3).
false.
```

- ocurences(+, +, -)

```
?- ocurences(a, [b, c, d], X).
X = 0
?- ocurences(a, [b, c, d], X).
X = 0
?- ocurences(b, [b, c, d], X).
X = 1
?- ocurences([d], [d, a, b, c, d], X).
X = 1
```

1.3 position/3

1.3.1 Исходный код

position(Sub, Term, Result) истина, если Result содержит список, описывающий положение терма Sub в терме Term

Sub произвольный тип

Term произвольный тип

Result список чисел

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(+, +, +)

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

?- position(c, [a, b, c], [2, 2, 1]).
true
```

- position(+, +, -)

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

true

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

true

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

X = [2, 1]

X = [2, 2]

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