Логическое программирование

Кевролетин В.В. группа c8403a(246)

19 May 2012

Содержание

1 Задание 22

1.1 Условие

Написать программу для отношения substitute для термов с использованием univ.

1.2 Решение

substitute(Old, New, Term, Term1) истина, если Term1 получается из терма Term заменой функторов Old на New

Old произвольный тип

New произвольный тип

Term произвольный тип

Term1 произвольный тип

Вводится вспомогательный терм substitute_each для итерации по списку, полученному при помощи предиката univ:

substitute_each(Old, New, OldList, NewList) аналогично предыдущему предикату, только для списков

Old произвольный тип

New произвольный тип

OldList список

OldList список

1.2.1 Исходный код

```
substitute(Old, New, Old, New).

substitute(Old, _, Term, Term) :-
        atomic(Term),
        \+ compare(=, Old, Term).

substitute(Old, New, Term, Term1) :-
        compound(Term),
        Term =.. [F | Args],
        substitute_each(Old, New, Args, NewArgs),
        Term1 =.. [F | NewArgs].

substitute_each(Old, New, [Old | Xs], [New | Ys]) :-
        substitute_each(Old, New, Xs, Ys).

substitute_each(Old, New, [ X | Xs], [X | Ys]) :-
        substitute_each(Old, New, Xs, Ys).

substitute_each(_, _, [], []).
```

1.2.2 Тесты

Для тестирования используется расширения языка SWI-Prolog

- substitute(+, +, +, +)

```
substitute(a, b, a, a).

false.

substitute(a, b, a, b).

true

substitute(a, b, c, c).

true

substitute(a, b, node(a), node(b)).

true

substitute(a, b, node(a, a), node(b, b)).

true

substitute(node(a, b), node(c, d),
           node(node(a, b),
```

```
                node(c, d)),
        node(node(c, d),
                node(c, d))).
```

true.

- substitute(+, +, +, -)

```
?- substitute(a, b, [a, b, c], X).

X = [b, b, c]

?- substitute(a, b, [a, b, a], X).

X = [b, b, b]

?- substitute([a], [b], [[a], b, c, a], X).

X = [[b], b, c, b]

?- substitute(a, b, [b, b, b], X).

X = [b, b, b]
```

- substitute(+, -, +, +)

```
substitute(a, X, [a], [b]).

X = b
```

- substitute(-, +, +, +)

```
substitute(X, b, [a], [b]).

Y = a
```

- substitute(-, -, +, +)

```
substitute(X, Y, [a], [b]).

X = [a],
Y = [b]
```