

Тема 14. Упражнения

Упражнение 14.1 (Простые отношения). Реализуйте следующие отношения для списков:

1. Отношение `inserto`, такое что `(inserto x before after)` верно, если `after` получено из `before` вставкой `x` в произвольном месте:

```
1 (run* (x xs) (inserto x xs '(1 2 3)))
2 ; '((1 (2 3)) (2 (1 3)) (3 (1 2)))
3
4 (run* (x xs) (inserto x xs '(a b a c)))
5 ; '((a (b a c)) (b (a a c)) (a (a b c)) (c (a b a)))
```

2. Отношение `anagramo`, определяющее перестановки элементов списка:

```
1 (run 1 (q) (anagramo '(d o r m i t o r y) '(d i r t y r o o m)))
2 ; '(_.0)
```

3. Отношение `subseqo`, определяющее подпоследовательности:

```
1 (run* (q) (subseqo '(2 4 5) '(1 2 3 4 5 6)))
2 ; '(_.0)
3
4 (run* (xs) (subseqo xs '(1 2 3)))
5 ; '((() (1) (2) (1 2) (3) (1 3) (2 3) (1 2 3))
6
7 (run* (xs)
8     (fresh (a b c)
9         (== xs `(,a ,b ,c))
10        (subseqo '(1 2) xs)))
11 ; '(((1 2 _.) (1 _.) 2) (_.) 1 2))
```

4. Отношение `searcho`, такое что `(searcho needle haystack position)` верно при вхождении `needle` как подстроки в `haystack` точно в позиции `position`:

```
1 (run* (pos) (searcho '(a b a) '(c a b a b a d) pos))
2 ; '((1) (1 1)) == '(1 3)
3
4 (run* (needle) (searcho needle '(c a b a b a d) (build-num 5)))
5 ; '(() (a) (a d))
6
7 (run* (needle pos)
8     (fresh (x) (== needle `(a ,x a))))
9     (searcho needle '(c a b a b a d) pos))
10 ; '(((a b a) (1)) ((a b a) (1 1)))
```

Упражнение 14.2 (Отрицание). Реализуйте следующие отношения с отрицаниями:

1. Отношение `not-prefixo`, определяющее отрицание префикса:

```
1 (run* (xs) (not-prefixo '(a b) '(a b r a b a)))
2 ; '()
3
4 (run* (xs) (not-prefixo '(a b a) '(a b r a b a)))
5 ; '(_.0)
```

2. Отношение `not-sublisto`, определяющее отрицание вхождения подстроки (подсписка):

```
1 (run* (xs) (not-sublisto '(a b a) '(a b r a b a)))
2 ; '()
3
4 (run* (xs) (not-sublisto '(a b c) '(a b r a b a)))
5 ; '(_.0)
```

Упражнение 14.3 (Замена). Реализуйте следующие отношения для замены подстрок (подсписков):

1. Отношение `replaceo`, такое что (`replaceo old new old-whole new-whole`) верно, если `new-whole` получено из `old-whole` заменой нуля или более вхождений `old` на `new`:

```
1 (run* (new-whole) (replaceo '(a b) '(N E W) '(a b r a b a) new-whole))
2 ; '((a b r a b a)
3 ;     (N E W r a b a)
4 ;     (N E W r N E W a)
5 ;     (a b r N E W a))
6
7 (run* (new-whole) (replaceo '(a a) '(x y) '(a a a a) new-whole))
8 ; '((a a a a)
9 ;     (x y a a)
10 ;     (a x y a)
11 ;     (x y x y)
12 ;     (a a x y))
```

2. Отношение `replace-allo`, аналогичное `replaceo`, но заменяющее **все** вхождения `old` на `new` в `old-whole`:

```
1 (run* (new-whole) (replace-allo '(a b) '(N E W) '(a b r a b a) new-whole))
2 ; '((N E W r N E W a))
3
4 (run* (new-whole) (replace-allo '(a a) '(x y) '(a a a a) new-whole))
5 ; '((x y x y) (a x y a))
```

Это отношение всё ещё может пораждать неоднозначные результаты, поскольку вхождения `old` могут накладываться.