

Тема 14. Упражнения

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

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

```
1 (run* (x xs) (insert° x xs '(1 2 3) ))
2 ; '((1 (2 3)) (2 (1 3)) (3 (1 2)))
3
4 (run* (x xs) (insert° 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 `anagram°`, определяющее перестановки элементов списка:

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

3. Отношение `subseq°`, определяющее подпоследовательности:

```
1 (run* (q) (subseq° '(2 4 5) '(1 2 3 4 5 6)))
2 ; '(_ .0)
3
4 (run* (xs) (subseq° 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    (subseq° '(1 2) xs)))
11 ; '((1 2 _ .0) (1 _ .0 2) (_ .0 1 2))
```

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

```
1 (run* (pos) (search° '(a b a) '(c a b a b a d) pos))
2 ; '((1) (1 1))      == '(1 3)
3
4 (run* (needle) (search° 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   (search° 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-prefix°`, определяющее отрицание префикса:

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

2. Отношение `not-sublist°`, определяющее отрицание вхождения подстроки (подписки):

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

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

1. Отношение `replace°`, такое что `(replace° 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) (replace° '(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) (replace° '(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-all°`, аналогичное `replace°`, но заменяющее **все** вхождения `old` на `new` в `old-whole`:

```
1 (run* (new-whole) (replace-all° '(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-all° '(a a) '(x y) '(a a a a) new-whole))
5 ; '((x y x y) (a x y a))
```

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