Programming Paradigms Fall 2023 — Problem Sets

by Nikolai Kudasov and Khaled Ismaeel

November 19, 2023

1 Problem set №12

1. Implement the following relations on lists:

   (a) Implement relation `subseqo` that checks whether the first list is a subsequence of the second list:

   ```
   (run* (q) (subseqo '(2 4 5) '(1 2 3 4 5 6)))
   ; '(_.0)

   (run* (xs) (subseqo xs '(1 2 3)))
   ; '(() (1) (2) (1 2) (3) (1 3) (2 3) (1 2 3))

   (run* (xs)
         (fresh (a b c)
                (== xs `(,a ,b ,c))
                (subseqo '(1 2) xs)))
   ; '((1 2 _.0) (1 _.0 2) (_.0 1 2))
   ```

   (b) Implement predicate `searcho`, such that `(searcho needle haystack position)` is true when `needle` occurs as a sublist in `haystack` exactly at position `position`:

   ```
   (run* (pos) (searcho '(a b a) '(c a b a b a d) pos))
   ; '((1) (1 1))           == '(1 3)

   (run* (needle) (searcho needle '(c a b a b a d) (build-num 5)))
   ; '(() (a) (a d))

   (run* (needle pos)
         (fresh (x) (== needle `(a ,x a)))
         (searcho needle '(c a b a b a d) pos))
   ; '(((a b a) (1)) ((a b a) (1 1)))
   ```

(c) Implement predicate `replaceo`, such that `(replaceo old new old-whole new-whole)` is true when `new-whole` can be produced from `old-whole` by replacing zero or more occurrences of `old` with `new`:

```
(run* (new-whole) (replaceo '(a b) '(N E W) '(a b r a b a) new-whole))
; '((a b r a b a)
;   (N E W r a b a)
;   (N E W r N E W a)
;   (a b r N E W a))

(run* (new-whole) (replaceo '(a a) '(x y) '(a a a a) new-whole))
; '((a a a a)
;   (x y a a)
;   (a x y a)
;   (x y x y)
;   (a a x y))
```

(d) Implement a predicate `not-prefixo` that checks whether one list is not a prefix of another list:

```
(run* (xs) (not-prefixo '(a b) '(a b r a b a)))
; '()

(run* (xs) (not-prefixo '(a b a) '(a b r a b a)))
; '(_.0)
```

(e) Implement a predicate `not-sublisto` to check if one list is not a contiguous sublist of another list:

```
(run* (xs) (not-sublisto '(a b a) '(a b r a b a)))
; '()

(run* (xs) (not-sublisto '(a b c) '(a b r a b a)))
; '(_.0)
```

(f) Implement a predicate `replace-allo` that is similar to `replaceo`, but replaces **all** occurrences of `old` with `new` in `old-whole`:

```
(run* (new-whole) (replace-allo '(a b) '(N E W) '(a b r a b a) new-whole))
; '((N E W r N E W a))

(run* (new-whole) (replace-allo '(a a) '(x y) '(a a a a) new-whole))
; '((x y x y) (a x y a))
```

Note: this still may produce multiple results, since occurrences of `old` *may be overlapping.*