SE3

Aufgabenblatt 12

Softwareentwicklung 3: Logikprogrammierung - WS 2019/2020 - W. Menzel

Funktionale Programmierung

Gesamtpunktzahl: 30 Abgabe der Lösungen bis zum 27.1.2020

Hinweis: Zum Testen können Sie auf den Poolrechnern der Informatik 'drracket' verwenden. Tragen Sie im (oberen) Editor-Fensterteil '`#lang scheme`' ein und starten Sie dann den Interpreter mit dem Button [Start].

Aufgabe 1: Wertesemantik 9 Punkte

maximale Bearbeitungszeit: 30 Minuten

Geben Sie für die folgenden s-Ausdrücke an, in welcher Reihenfolge und mit welchen Zwischenergebnissen ein Scheme-Interpreter ihren Wert ermittelt. Z.B. lässt sich für den Ausdruck

```
(> (car (quote (2 4))) (car (cdr (quote (1 2 3)))) )
```

die Auswertungsreihenfolge durch folgendes Ablaufprotokoll veranschaulichen:

```
(> (car (quote (2 4)) (car (cdr (quote (1 2 3)))) )
   (car (quote (2 4))
      (quote (2 4))
      ==> (2 4)
   ==> 2
   (car (cdr (quote (1 2 3))))
      (cdr (quote (1 2 3)))
         (quote (1 2 3))
         ==> (1 2 3)
      ==> (2 3)
   ==> 2
==> #f
```

1.
```
(list (car (cdr (quote (1 2 3 4))))
      (cdr (quote (1 2 3 4))) )
```

2. `(if (< (car (cdr (quote (5 -3 4 -2)))) (- 2 6)) 0 1)`

Geben Sie für die folgenden Scheme-Ausdrücke an, zu welchem Wert sie evaluieren.

3.
```
(cons (cdr (quote (1 2 3 4)))
      (car (quote (1 2 3 4))) )
```

4.
```
(map (lambda (x) (if (pair? x) (car x) x))
     (quote (lambda (x) (if (pair? x) (car x) x))) )
```

5.
```
(filter (curry > 5)
        (reverse (quote (1 3 5 7 9))) )
```

```
(filter (compose positive?
                 (lambda (x) (- x 5)) )
        (quote (1 3 5 7 9)) )
```

Aufgabe 2: Programmverstehen 15 Punkte

maximale Bearbeitungszeit: 80 Minuten

Was berechnen die folgenden Funktionen? Reimplementieren Sie jeweils ein analoges Prädikat in Prolog. Diskutieren Sie Unterschiede und Gemeinsamkeiten der beiden Implementationen.

1.
```
;; x und y sind Listen von Atomen
(define (foo1 x y)
   (if (null? x)
       #t
       (if (not (null? y) )
           (if (eq? (car x) (car y))
               (foo1 (cdr x) (cdr y) )
               #f
           )
           #f
       )
   )
)
```

2.
```
(define (foo2 x)
    (if (null? x)
        (quote ())
        (if (member (car x) (cdr x))
            (foo2 (cdr x))
            (cons (car x) (foo2 (cdr x)) ) ) ) )
```

3.
```
(define (foo3 x y)
    (if (null? x)
        (quote ())
        (if (member (car x) y)
            (cons (car x) (foo3 (cdr x) y))
            (foo3 (cdr x) y) ) ) )
```

4.
```
(define (foo4 x)
    (letrec ((foo4a
              (lambda (x y)
                 (if (null? x)
                     y
                     (foo4a (cdr x) (cons (car x) y)) ) ) ) )
      (foo4a x (quote ())) ) )
```

   Hinweis: `letrec` ist eine Variante von `let`, die auch die Verwendung rekursiver Funktionsdefinitionen unterstützt.

5.
```
(define (foo5 x)
    (if (null? x)
        (quote ())
        (if (pair? (car x))
            (append (foo5 (car x)) (foo5 (cdr x)) )
            (cons (car x) (foo5 (cdr x))) ) ) )
```

Aufgabe 3: Programmentwicklung 6 Punkte

maximale Bearbeitungszeit: 40 Minuten

Reimplementieren Sie auf der Basis der von Ihnen in der Präsenzübung gewählten Repräsentation die Prädikate `lt/2, integer2peano/2` und `add/3` für das Rechnen mit PEANO-Zahlen als Scheme-Funktionen. Diskutieren Sie Unterschiede und Gemeinsamkeiten zwischen den Prolog- und Scheme-Implementationen.