Übungsblatt 7

Institut Intelligente Systeme
Leiter: Prof. Dr. E. Lehmann
Dietmar Lippold

Besprechungstermin: 06.12.04 - 10.12.04

Aufgabe 1

Es seien die drei nachfolgenden Programme gegeben.

```
Programm 1

(define x 2)
(define y 3)
(let ((x (+ y 1))
      (y (+ x 2)) )
  (* x y) )
```

```
Programm 2

(define x 2)
(define y 3)
(let* ((x (+ y 1))
       (y (+ x 2)) )
  (* x y) )
```

```
Programm 3

(define x 2)
(define y 3)
(letrec ((x (+ y 1))
         (y (+ x 2)) )
  (* x y) )
```

1. Ermitteln Sie das Ergebnis für die drei Programme und erklären Sie es. Lesen Sie dazu im $\mathrm{R^5RS}$ nach.
2. Unter welchen Umständen ist nur die Verwendung von `letrec` und nicht die Verwendung von `let` oder `let*` möglich? Läßt sich das `letrec` in dem Fall auf andere Weise ersetzen?

Aufgabe 2

In dieser Aufgabe soll analog zu den in der Vorlesung definierten rationalen Zahlen ein abstrakter Datentyp *Rechteck* definiert werden. Dabei sei ein Rechteck durch die Längen von zwei seiner senkrecht aufeinander stehenden Seiten (seiner Breite und Höhe) eindeutig gekennzeichnet (ohne Beachtung einer Position und Orientierung).

1. Nennen Sie die Arten von Prozeduren, die für einen zusammengesetzten abstrakten Datentyp im allgemeinen zu definieren sind.
2. Definieren Sie diese Prozeduren für den Datentyp *Rechteck*, wobei ein Rechteck durch die Längen zweier senkrecht aufeinander stehender Seiten gebildet wird.
3. Erstellen Sie Prozeduren zur Berechnung des Umfangs und des Flächeninhalts eines Rechtecks.

Aufgabe 3

Für jeden Datentyp in Scheme gibt es ein Typprädikat, das zurückgibt, ob das Argument diesen Typ hat (boolescher Wert). Testen Sie `number?`, `list?` und `symbol?`.

1. Definieren Sie eine Prozedur `(benenne objekt)`, die für Listen `'liste`, für Zahlen `'zahl` für Symbole `'symbol` und für andere Objekte `'sonstiges` zurückgibt.
2. Die Prozedur `(map f liste)` wendet `f` auf jedes Listenelement an und gibt eine Liste der Ergebnisse zurück. Definieren Sie unter Verwendung von `map` eine Prozedur `(benenne-alle objekte)`, die eine Liste der Typangaben der Objekte ausgibt.

   Es soll sich folgendermaßen verhalten:

   ```
   > (benenne-alle '(1 anna (2 3 eva)))
   (zahl symbol liste)
   ```

3. Definieren Sie ohne Verwendung von `map` eine rekursive Prozedur `benenne-rek`, die dasselbe wie `bennene-alle` tut.

   Hinweis: Die Benennung der leere Liste ist die leere Liste, andernfalls kombinieren Sie die Typangabe des ersten Elements mit den Typangaben des Rests der Liste.

Aufgabe 4

Eine geschachtelte Liste kann man als Baum auffassen, wobei die Elemente der Liste die Unterbäume sind. Die Blätter sind die Atome der Liste.

1. Definieren Sie (analog zu vorhergehenden Aufgabe) eine Prozdur `benenne-tief`, die jedes *Atom* (Blatt) durch den Namen seines Typs ersetzt und die Listenstruktur erhält.

   Hinweis: Der Name `'liste` für ein Element wird dabei natürlich nicht mehr vergeben. Wenn ein Element eine Liste ist, müssen Sie stattdessen deren Elemente benennen. Führen Sie einfach einen entsprechenden Fall ein.

   ```
   > (benenne-tief '(1 anna (2 3 eva)))
   (zahl symbol (zahl zahl symbol))
   ```

2. Kann man die tiefen-rekursive Prozedur auf einfache Art durch ein durch einen map-Ausdruck realisieren (d.h. durch `(map prozedur liste)`)? Kann man sie auf einfache Art durch eine Prozedur ersetzen, die einen iterativen Prozeß erzeugt?

Aufgabe 5 (schriftlich)

Die *Position* eines Teilbaums in einem Baum (in einer geschachtelten Liste) kann man durch die Liste der Nummern der Nachfolgerknoten (als Elemente der umgebenden Liste) auf dem Weg von der Wurzel zum Teilbaum angeben, wobei das erste Element einer Liste die Nummer 0 (Null) und die Wurzel als Position die leere Liste hat. So hat der Teilbaum *'(b c)* im Baum *'(a ((b c) d) e)* die Position *'(1 0)*. Ein Symbol ist als Blatt (Atom der Liste) ein spezieller Teilbaum.

1. Definieren Sie eine Prozedur `(gib-teilbaum position baum)`, die zu einem Baum und einer Position den entsprechenden Teilbaum liefert. Wenn die Position im Baum nicht vorhanden ist, soll `'fehlt` geliefert werden.

2. Definieren Sie eine Prozedur `(baum-eq? baum1 baum2)`, die vergleicht, ob zwei Bäume in Bezug auf ihre Struktur und ihre Blätter gleich sind. Benutzen Sie zum Vergleich die Scheme-Prozedur `eqv?`.

3. Definieren Sie eine Prozedur `(struktur-eq? baum1 baum2)`, die nur die Struktur der beiden Bäume vergleicht (nicht den Inhalt der Blätter). Ist diese Prozedur einfacher oder komplizierter als `baum-eq?`?

4. Definieren Sie eine Prozedur `(positionen teilbaum baum)`, die eine Liste der Positionen liefert, an denen `teilbaum` in `baum` vorkommt, wobei Sie zum Test der Gleichheit von zwei Bäumen `baum-eq?` benutzen. Wenn Sie Hilfsprozeduren definieren, überlegen und dokumentieren Sie, welche Semantik deren Parameter haben (d.h. welche Art von Werten den Parametern übergeben wird).