
Concurrent Programming

<http://www-pu.informatik.uni-tuebingen.de/cp-2006>

Blatt 1

Abgabe: XXXXXXXX

1. Programmiere eine Funktion `eval-infix`, die zweistellige arithmetische Infix-Ausdrücke auswertet, welche die Operatoren `+`, `-`, `*`, `^` (bildet den Exponenten) und `/` enthalten können. Die Ausdrücke seien dabei durch Listen repräsentiert und vollständig geklammert. Der Ausdruck

```
(eval-infix '((4 * (5 + 8)) / 2))
```

soll also 26 ergeben.

2. Die Funktion `foldr` ist folgendermaßen definiert:

$$(\text{foldr } h \ u \ '(l_1 \dots l_n)) \equiv (h \ l_1 \ (h \ l_2 \ \dots \ (h \ l_n \ u) \ \dots))$$

$$(\text{foldr } h \ u \ '()) \equiv u$$

Die Funktion h ist eine Funktion mit zwei Parametern. Programmiere `foldr` in Scheme!

3. Programmiere mit Hilfe von `foldr` aus Aufgabe 2 Funktionen, welche

- die Länge einer Liste berechnet,
- $\bigvee_{i=0}^n P(l_i)$ für ein Prädikat P berechnet,
- die kleinste und die größte Zahl findet und als Paar zurückgibt,
- alle doppelten Elemente aus der Liste filtert.

4. Programmiere die Funktionen

- $((\text{compose } f \ g) \ x) \equiv (f \circ g)(x)$
- $((\text{iterate } f \ n) \ x) \equiv f^n(x)$

in Scheme. Die Funktionen f und g haben einen Parameter.

5. Implementiere einen abstrakten Datentyp für binäre Bäume! Ein Knoten des Baumes sollte neben linken und rechten Teilbaum noch eine Markierung `label` speichern können. Repräsentiere die Knoten des Baumes durch Listen. Implementiere die folgenden Operationen auf binären Bäumen:

Konstruktoren

`make-empty-tree`: $\rightarrow \text{Tree}$

`make-node`: $\text{Value} \times \text{Tree} \times \text{Tree} \rightarrow \text{Tree}$

Prädikat

`empty-tree?`: $\text{Tree} \rightarrow \text{Bool}$

Operationen

`left`: $\text{Tree} \rightarrow \text{Tree}$

`right`: $\text{Tree} \rightarrow \text{Tree}$

`label`: $\text{Tree} \rightarrow \text{Value}$

;; Höhe des Baumes

`depth`: $\text{Tree} \rightarrow \text{Integer}$

;; Anzahl Knoten

`size`: $\text{Tree} \rightarrow \text{Integer}$

;; Prüft ob ein Baum ein bestimmtes label enthält

`tree-member?`: $\text{Tree} \times \text{Value} \rightarrow \text{Bool}$

6. Programmieren Sie eine Funktion `tree-map`, die als Argument einen binären Baum t und eine Funktion f nimmt. `tree-map` wendet die Funktion f auf alle Markierungen des Baumes an und gibt diese in einem neuen Baum mit derselben Struktur wie t zurück.
7. Betrachten Sie die Funktion `foldr` aus Aufgabe 2. Die Funktion `foldr` ersetzt die Konstrukteure einer Liste durch eine Funktion. Dieses Prinzip kann auch auf die binären Bäume aus Aufgabe 5 übertragen werden. Programmieren Sie eine Funktion `tree-foldr`, die die Konstruktoreersetzung für einen binären Baum realisiert! Implementieren Sie `tree-map` durch `tree-foldr`!
8. Schreiben Sie ein Scheme-Programm, das zwei Threads startet, von denen einer den Buchstaben `a` und der andere den Buchstaben `b` wiederholt ausdrückt. In der Ausgabe sollen die beiden Buchstaben immer genau abwechselnd vorkommen.