6. Übungsblatt zu Programmierung (WS 1999/2000)

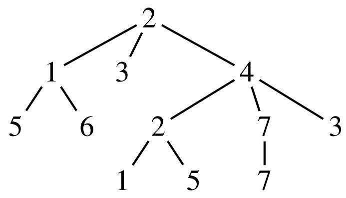
Andreas Rossberg, Prof. Gert Smolka

Abgabe: 3. Dezember 1999 vor der Vorlesung

Hinweis: Die Aufgaben 6.1 bis 6.4 brauchen nur schriftlich abgegeben zu werden.

Aufgabe 6.1: Baumdarstellungen (8 Punkte)

(a) Formalisieren Sie die Baumdarstellung



als Term, als geordneten Baum und als Graphen. Da einige Marken mehrfach vorkommen, müssen Sie bei der Graphdarstellung darauf achten, dass Sie wirklich einen Baum bekommen.

(b) Zeichnen Sie den geordneten Baum

$$\{\, [] \mapsto 5,\ [1] \mapsto 2,\ [1,1] \mapsto 3,\ [1,2] \mapsto 1,\ [1,2,1] \mapsto 6,\ [1,2,1,1] \mapsto 3,$$
$$[1,2,1,2] \mapsto 3,\ [1,2,2] \mapsto 5,\ [1,2,3] \mapsto 2,\ [1,3] \mapsto 8,\ [1,4] \mapsto 4,\ [1,4,1] \mapsto 9 \,\}$$

Geben Sie den zugeordneten Graphen und den zugeordneten Term an.

(c) Zeichnen Sie den Term

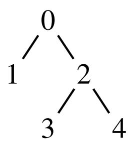
$$\begin{array}{l}\langle 6,\ \langle 1,\ \langle 2,\ \langle 5\rangle,\ \langle 4\rangle\rangle, \\ \qquad \langle 7\rangle\ \rangle, \\ \quad \langle 1,\ \langle 2,\ \langle 6,\ \langle 4\rangle, \\ \qquad\qquad \langle 3,\ \langle 1\rangle\rangle, \\ \qquad\qquad \langle 5\rangle\ \rangle\ \rangle\ \rangle\ \rangle\end{array}$$

als Baum. Geben Sie den zugeordneten geordneten Baum an.

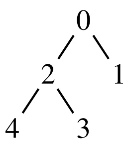
Aufgabe 6.2: Graphen vs. geordnete Bäume (5 Punkte)

Betrachten Sie die folgenden Baumdarstellungen:

A:



B:



Beschreiben Sie beide Bäume sowohl als Graph, als auch als geordneten Baum. Was fällt Ihnen auf?

Aufgabe 6.3: Länge von konkatenierten Listen (5 Punkte)

Wir betrachten Listen als mathematische Objekte gemäß

$$\mathcal{L}(X) = \{\emptyset\} \cup X \times \mathcal{L}(X)$$

Wir schreiben nil für $\emptyset$ und $x :: w$ für (x, w) und definieren induktiv zwei Funktionen:

$$\begin{aligned} l\colon\ &\mathcal{L}(X) \to \mathbb{N} \\ l(\mathrm{nil}) &= 0 \\ l(x :: w) &= 1 + l(w) \end{aligned} \qquad\qquad \begin{aligned} \_ @ \_\colon\ &\mathcal{L}(X) \times \mathcal{L}(X) \to \mathcal{L}(X) \\ \mathrm{nil} @ v &= v \\ (x :: w) @ v &= x :: (v @ w) \end{aligned}$$

Beweisen Sie durch Induktion über $l(v)$, dass

$$l(v @ w) = l(v) + l(w)$$

für alle $v, w \in \mathcal{L}(X)$ gilt.

Aufgabe 6.4: Größenverhältnisse in Bäumen (8 Punkte)

Ein *ternärer* Baum ist ein Baum, bei dem jeder Knoten, der kein Blatt ist, genau 3 Unterbäume hat.

(a) Beweisen Sie durch Induktion über die Tiefe, dass für jeden nichtleeren, balancierten ternären Baum mit b Blättern und Tiefe t gilt:

$$b = 3^t$$

(b) Beweisen Sie durch Induktion über die Tiefe, dass für jeden nichtleeren, balancierten ternären Baum der Größe g und Tiefe t gilt:

$$g = \frac{1}{2}(3^{t+1} - 1)$$

Aufgabe 6.5: Einfache Operationen auf Binärbäumen (12 Punkte)

Gegeben sei folgende Deklaration für Binärbäume über beliebigen Typen:

```
datatype 'a tree = L of 'a | N of 'a * 'a tree * 'a tree
```

(a) Schreiben Sie eine Prozedur `size: 'a tree -> int`, die die Größe eines Baumes berechnet.

(b) Schreiben Sie eine Prozedur `root: 'a tree -> 'a`, die die Marke der Baumwurzel liefert.

(c) Schreiben Sie eine Prozedur `nodes: 'a tree -> 'a list`, die alle Marken eines Baumes als Liste liefert. Die Reihenfolge der Marken sei darin wie folgt:

- zu jedem Knoten v stehen die Marken des linken Unterbaumes *vor* der Marke von v,
- zu jedem Knoten v folgen die Marken des rechten Unterbaumes *nach* der Marke von v.

Zum Beispiel soll für den Baum

```
   1
  / \
 2   3
    / \
   4   5
```

die Liste `[2,1,4,3,5]` geliefert werden.

(d) Schreiben Sie eine Prozedur `nnodes: 'a tree -> 'a list`, die die Marken aller Knoten liefert, die keine Blätter sind. Die Reihenfolge der Marken sei analog zu (c).

(e) Schreiben Sie eine Prozedur `inodes: 'a tree -> 'a list`, die die Marken aller inneren Knoten liefert. Die Reihenfolge sei ebenfalls wie unter (c). Verwenden Sie `nnodes` als Hilfsprozedur.

Aufgabe 6.6: Baumhomomorphismus (5 Punkte)

Gegeben sei erneut folgende Deklaration für Binärbäume:

```
datatype 'a tree = L of 'a | N of 'a * 'a tree * 'a tree
```

(a) Schreiben Sie eine Prozedur

```
double: int tree -> int tree
```

die in einem Baum über ganzen Zahlen jede Marke verdoppelt.

(b) Schreiben Sie eine Prozedur

```
mapTree: ('a -> 'b) -> 'a tree -> 'b tree
```

die eine Prozedur auf jede Marke eines Baumes anwendet (analog zur Prozedur `map` für Listen).