



6. Übungsblatt zu Programmierung 1, WS 2010/11

Prof. Dr. Gert Smolka, Christian Doczkal, M.Sc.

www.ps.uni-sb.de/courses/prog-ws10/

Lesen Sie im Buch: Kapitel 6

Aufgabe 6.5 Deklarieren Sie eine Prozedur $vars : exp \rightarrow var\ list$, die zu einem Ausdruck eine Liste liefert, die die in dem Ausdruck vorkommenden Variablen enthält. Orientieren Sie sich an der Prozedur *subexps*.

Aufgabe 6.6 Deklarieren Sie eine Prozedur $count : var \rightarrow exp \rightarrow int$, die zählt, wie oft eine Variable in einem Ausdruck auftritt. Beispielsweise tritt x in $x + x$ zweimal auf.

Aufgabe 6.7 Deklarieren Sie eine Prozedur $check : exp \rightarrow exp \rightarrow bool$, die für zwei Ausdrücke e und e' testet, ob e ein Teilausdruck von e' ist.

Aufgabe 6.8 Schreiben Sie eine Prozedur $instantiate : env \rightarrow exp \rightarrow exp$, die zu einer Umgebung V und einem Ausdruck e den Ausdruck liefert, den man aus e erhält, indem man die in e vorkommenden Variablen gemäß V durch Konstanten ersetzt. Beispielsweise soll für die in § 6.4.2 deklarierte Umgebung env und den Ausdruck $A(V\ "x", V\ "y")$ der Ausdruck $A(C\ 5, C\ 3)$ geliefert werden. Orientieren Sie sich an der Prozedur *eval*.

Aufgabe 6.10 (Konstruktordarstellung natürlicher Zahlen) In dieser Aufgabe stellen wir die natürlichen Zahlen mit den Werten des Konstruktortyps

`datatype nat = 0 | S of nat`

dar: $0 \mapsto O$, $1 \mapsto SO$, $2 \mapsto S(SO)$, $3 \mapsto S(S(SO))$, und so weiter.

- Deklarieren Sie eine Prozedur $code : int \rightarrow nat$, die die Darstellung einer natürlichen Zahl liefert.
- Deklarieren Sie eine Prozedur $decode : nat \rightarrow int$, sodass $decode(code\ n) = n$ für alle $n \in \mathbb{N}$ gilt.
- Deklarieren Sie für nat kaskadierte Prozeduren *add*, *mul* und *less*, die den Operationen $+$, $*$ und $<$ für natürliche Zahlen entsprechen. Verwenden Sie dabei keine Operationen für *int*.

Aufgabe 6.12 Schreiben Sie eine Prozedur $test : int \rightarrow bool$, die testet, ob das Quadrat einer ganzen Zahl im darstellbaren Zahlenbereich liegt.

Aufgabe 6.13 Führen Sie zweistellige Sequenzialisierungen $(e_1; e_2)$ auf Abstraktionen und Applikationen zurück.

Aufgabe 6.17 Schreiben Sie eine Prozedur $last : \alpha\ list \rightarrow \alpha\ option$, die das letzte Element einer Liste liefert.

Aufgabe 7.1 Schreiben Sie eine Prozedur $compound : tree \rightarrow bool$, die testet, ob ein Baum zusammengesetzt ist.

Aufgabe 7.3 Geben Sie Ausdrücke an, die Bäume mit den unten gezeigten grafischen Darstellungen beschreiben. Dabei können Sie die Bezeichner $t1 = T[]$ und $t2 = T[t1, t1, t1]$ verwenden.



Aufgabe 7.5 Geben Sie die Gestalt des Ausdrucks $(x + 3)(y + 7)$ an.

Aufgabe 7.6 Geben Sie einen Ausdruck mit der Gestalt $T[T[t1, t1], t1]$ an.

Aufgabe 7.10 Schreiben Sie eine Prozedur $binary : tree \rightarrow bool$, die testet, ob ein Baum binär ist.

Aufgabe 7.12 (Spiegeln) Spiegeln reversiert die Ordnung der Unterbäume der Teilbäume eines Baums:



Schreiben Sie eine Prozedur $mirror : tree \rightarrow tree$, die Bäume spiegelt.

Zusatzaufgabe 6.9 (Symbolisches Differenzieren) Sie sollen eine Prozedur schreiben, die Ausdrücke nach der Variable x ableitet. Hier ist ein Beispiel:

$$(x^3 + 3x^2 + x + 2)' = 3x^2 + 6x + 1$$

Ausdrücke sollen gemäß des folgenden Typs dargestellt werden:

```
datatype exp = C of int                c
             | X                      x
             | A of exp * exp          u + v
             | M of exp * exp          u · v
             | P of exp * int          un
```

- Schreiben Sie eine Deklaration, die den Bezeichner u an die Darstellung des Ausdrucks $x^3 + 3x^2 + x + 2$ bindet. Der Operator $+$ soll dabei links klammern.
- Schreiben Sie eine Prozedur $derive : exp \rightarrow exp$, die die Ableitung eines Ausdrucks gemäß den folgenden Regeln berechnet:

$$\begin{aligned} c' &= 0 \\ x' &= 1 \\ (u + v)' &= u' + v' \\ (u \cdot v)' &= u' \cdot v + u \cdot v' \\ (u^n)' &= n \cdot u^{n-1} \cdot u' \end{aligned}$$

Die Ableitung darf vereinfachbare Teilausdrücke enthalten (z.B. $0 \cdot u$).

- c) Schreiben Sie eine Prozedur $simplifyTop : exp \rightarrow exp$, die versucht, einen Ausdruck auf oberster Ebene durch die Anwendung einer der folgenden Regeln zu vereinfachen:

$$\begin{array}{ll} 0 + u \rightarrow u & u + 0 \rightarrow u \\ 0 \cdot u \rightarrow 0 & u \cdot 0 \rightarrow 0 \\ 1 \cdot u \rightarrow u & u \cdot 1 \rightarrow u \\ u^0 \rightarrow 1 & u^1 \rightarrow u \end{array}$$

Wenn keine der Regeln auf oberster Ebene anwendbar ist, soll der Ausdruck unverändert zurückgeliefert werden.

- d) Schreiben Sie eine Prozedur $simplify : exp \rightarrow exp$, die einen Ausdruck gemäß der obigen Regeln solange vereinfacht, bis keine Regel mehr anwendbar ist. Gehen Sie bei zusammengesetzten Ausdrücken wie folgt vor:
1. Vereinfachen Sie zuerst die Komponenten.
 2. Vereinfachen Sie dann den Ausdruck mit den vereinfachten Komponenten mithilfe von $simplifyTop$.