6. Übungsblatt zu Programmierung 1, WS 2008/09

Prof. Dr. Gert Smolka, Mark Kaminski, M.Sc.
www.ps.uni-sb.de/courses/prog-ws08/

Lesen Sie im Buch: Kapitel 6

Aufgabe 6.1 Deklarieren Sie eine Prozedur *variant* : *shape* → *int*, die die Variantennummer eines geometrischen Objekts liefert. Beispielsweise soll *variant*(*Square* 3.0) = 2 gelten.

Aufgabe 6.2 Deklarieren Sie eine Prozedur *scale* : *real* → *shape* → *shape*, die ein Objekt gemäß einem Faktor skaliert (d.h. vergrößert oder verkleinert). Beispielsweise soll *scale* 0.5 (*Square* 3.0) = *Square* 1.5 gelten.

Aufgabe 6.5 Deklarieren Sie eine Prozedur *vars* : *exp* → *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* → *exp* → *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* → *exp* → *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* → *exp* → *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 $A(V\,\text{"}x\text{"},\ V\,\text{"}y\text{"})$ 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 = O | S of nat
```

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

a) Deklarieren Sie eine Prozedur *code* : *int* → *nat*, die die Darstellung einer natürlichen Zahl liefert.

b) Deklarieren Sie eine Prozedur *decode* : *nat* → *int*, sodass *decode*(*code* n) = n für alle $n \in \mathbb{N}$ gilt.

c) 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* → *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* : α *list* → α *option*, die das letzte Element einer Liste liefert.

Aufgabe 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           u^n
```

a) 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.

b) Schreiben Sie eine Prozedur *derive* : *exp* → *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* → *exp*, die versucht, einen Ausdruck auf oberster Ebene durch die Anwendung einer der folgenden Regeln zu vereinfachen:

$$\begin{array}{rcl@{\qquad}rcl} 0+u & \to & u & u+0 & \to & u \\ 0 \cdot u & \to & 0 & u \cdot 0 & \to & 0 \\ 1 \cdot u & \to & u & u \cdot 1 & \to & u \\ u^0 & \to & 1 & u^1 & \to & u \end{array}$$

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

d) Schreiben Sie eine Prozedur *simplify* : *exp* → *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*.