TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

WS 10/11

Einführung in die Informatik II **Übungsblatt 4**

Univ.-Prof. Dr. Andrey Rybalchenko, A. Herz, K. Apinis 12.11.2010

Dieses Blatt behandelt Kapitel 2.9 - 3.3.0 aus dem Buch zur Vorlesung. Lesen Sie diese Kapitel!

Aufgabe 3.1 Deklarieren Sie eine Prozedur `mul:int→int→int→int`, die das Produkt dreier Zahlen liefert. Deklarieren Sie mul auf 3 Arten: Mit einer kaskadierten Prozedurdeklaration, mit einer Prozedurdeklaration und zwei Abstraktionen, und mit einer Deklaration mit val und drei Abstraktionen.

Aufgabe 3.3 Geben Sie die Baumdarstellung des Ausdrucks

```
mul x y + mul x (y + 2) * 5
```

an.

Aufgabe 3.5 Geben Sie zu den folgenden Abstraktionen semantisch äquivalente Ausdrücke an, die ohne die Verwendung von Abstraktionen gebildet sind.

a) `fn (x : int) => x*x`

b) `fn (x : int) => fn (y : int) => x+y`

Hilfe: Verwenden Sie Let-Ausdrücke und Prozedurdeklarationen.

Aufgabe 3.9 Deklarieren Sie eine Prozedur `prod : (int→int)→int→int`, die für $n \geq 0$ die Gleichung prod $f\ n = 1 \cdot (f\ 1) \ldots \cdot (f\ n)$ erfüllt. Deklarieren Sie außerdem mithilfe von prod eine Prozedur `fac : int →int`, die für $n \geq 0$ die Fakultät $n!$ berechnet (siehe Aufgabe 1.26 auf S. 21). Die Prozedur `fac` soll nicht rekursiv sein.

Aufgabe 3.10 Deklarieren Sie mithilfe der höherstufigen Prozedur `sum` eine Prozedur `sum' : (int→int)→ int→int→int`, die für $k \geq 0$ die Gleichung sum$'$ $f\ m\ k = 0 + f(m+1) \ldots + f(m+k)$ erfüllt. Die Prozedur `sum'` soll nicht rekursiv sein.

Aufgabe 3.11 Geben Sie die Baumdarstellungen der folgenden Typen an:

a) `(int→int)→int`

b) `int→(int*bool→int)→int`

c) `(int→bool)→(bool→real)→int→real`

Aufgabe 3.13 Schreiben Sie zwei Prozeduren

a) `cas : (int*int→int)→int→int→int`

b) `car : (int→int→int)→int*int→int`

sodass `cas` zur kartesischen Darstellung einer zweistelligen Operation die kaskadierte Darstellung und `car` zur kaskadierten Darstellung die kartesische Darstellung liefert. Erproben Sie `cas` und `car` mit Prozeduren, die das Maximum zweier Zahlen liefern:

```
fun maxCas (x:int) (y:int) = if x<y then y else x
fun maxCar (x:int, y:int) = if x<y then y else x
val maxCas' = cas maxCar
val maxCar' = car maxCas
```

Wenn Sie `cas` und `car` richtig geschrieben haben, verhält sich `maxCas'` genauso wie `maxCas` und `maxCar'` genausowie `maxCar`. Hinweis:Die Aufgabe hat eine sehr einfache Lösung.