Übungsblatt 11 zu "Programmiersprachen"

Berthold Hoffmann, Studiengang Informatik (hof@tzi.de)

Besprechung am 24. 1. 2005

Aufgabe 1: Auswahl-Funktion

Definiere in HASKELL eine Funktion

```
cond :: Bool -> alpha -> alpha
```

so dass der Ausdruck `cond` $C\,T\,E$ genau dasselbe tut wie **if** C **then** T **else** E.

Warum kann so eine Funktion in ADA oder C++ nicht geschrieben werden?

Aufgabe 2: Listenverkettung

Nimm an, die Verkettung von Listen sei in HASKELL definiert als

```
(++)  :: [alpha] -> [alpha] -> [alpha]
[]    ++ l = l
(h:t) ++ l = h: (t ++ l)
```

(Hierbei ist `[]` die leere Liste, und `(:)` der Listenkonstruktor, der ein Element vorn in eine Liste einhängt.)

Benutze die in der Vorlesung eingeführte Darstellung von Ausdrücken als *Termgraphen*, um die Auswertungsschritte für den Ausdruck `[1,2,3] ++ [4,5,6]` nachzuvollziehen.

Überlege Antworten auf die folgenden Fragen:

- Welchen Zeitaufwand hat die Funktion?
- Welchen Speicherbedarf hat die Funktion?
- Ist die Funktion *strikt*?

Aufgabe 3: Baum-Kombinator

Setze voraus, Bäume seien – wie in der Vorlesung – folgendermaßen definiert:

```
data Tree alpha = Leaf alpha
                | Node (Tree alpha) (Tree alpha)
```

Definiere einen Kombinator

```
map (alpha -> beta) -> (Tree alpha) -> (Tree beta)
```

so dass `map` $f\,t$ jedes Blatt `Leaf` v in t durch das Blatt `Leaf` $(f\,v)$ ersetzt.