PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen, J. Berthold

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

21. Januar 2003

Übungen zu „Parallelität in funktionalen Sprachen“

Nr. 12 (letztes Blatt), Abgabe: 28. Januar 2003 in der Vorlesung

---

Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.

12.1 Abstrakte Interpretation von Listenfunktionen — 6 Punkte

Führen Sie für jede der folgenden drei Funktionen eine Striktheitsanalyse durch. Zeigen Sie außerdem die Abhängigkeit der Definiertheitsstufe des Ergebnisses von der des Arguments durch die Darstellung in einer Tabelle auf.

(a)
```
greaterThan0 l = case l of
                   []    -> 0
                   (h:t) -> if h > 0 then 1
                                     else greaterThan0 t
```

(b)
```
prodlist l = case l of
               []    -> 1
               (h:t) -> h * prodlist t
```

(c)
```
length' l = case l of
              []    -> 0
              (h:t) -> 1 + length' t
```

Schreiben Sie für jede Definiertheitsstufe $i \in \underline{4}$ eine Haskell-Funktion `evaluate_i`, die ihr Argument (eine Liste) zum gegebenen Grad i auswertet.

12.2 Evaluation Transformers — 4 Punkte

Betrachten Sie analog zum Beispiel aus der Vorlesung die drei Ausdrücke

$e_i =$ `f (zip l1 l2)`

mit `f` $\in$ `{search0, length, sum}`, wobei `zip` die folgende Bedeutung hat:

```
zip               :: [a] -> [b] -> [(a,b)]
zip (a:as) (b:bs)   = (a, b) : zip as bs
zip _      _        = []
```

Bestimmen Sie für jedes e_i die maximal sichere Auswertungsstufe in Abhängigkeit von den maximal sicheren Auswertungsstufen der Argumente `l1` und `l2`.

12.3 Striktheitsanalyse für Listenfunktionen — 10 Punkte

Drücken Sie die folgenden Funktionen in Mini-Haskell aus und bestimmen Sie deren abstrakte Interpretation und Evaluation Transformer.

a)
```
filter :: (a -> Bool) -> [a] -> [a]
filter p xs = [ x | x <- xs, p x ]
```

b)
```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl f z [] = z
foldl f z (x:xs) = foldl f (f z x) xs
```