Philipps Universität Marburg

Übungen zur „Praktischen Informatik III“, WS 2008/09

Prof. Dr. R. Loogen · Fachbereich Mathematik und Informatik · Hans-Meerwein-Straße, D-35032 Marburg

Nr. 8, Abgabe: 10. Dezember 2008 vor der Vorlesung

18. Countdown-Problem — 8 Punkte

Das *Countdown Problem* lautet wie folgt: Gegeben seien eine Folge positiver ganzer Zahlen, die Quellzahlen, und eine positive ganze Zahl, die Zielzahl. Gesucht wird ein arithmetischer Ausdruck, in dem jede der Quellzahlen höchstens einmal auftritt und der als Ergebnis die Zielzahl liefert. Als Operatoren können in dem Ausdruck $+$, $-$, $*$ und / verwendet werden, wobei jedes Zwischenergebnis ebenfalls eine positive ganze Zahl sein muss. Zum Beispiel löst für die Quellzahlen $[1, 3, 7, 10, 25, 50]$ und die Zielzahl 765 der Ausdruck $(1 + 50) * (25 - 10)$ das Problem.

In dieser Aufgabe soll ein Programm zur Lösung dieses Problems entwickelt werden. Das Programm soll nach entsprechenden Eingabeaufforderungen die Zahlenliste und die Zielzahl einlesen und anschließend eine erste Lösung des Countdown-Problems berechnen und ausgeben. Weitere Lösungen sollen nur nach Eingabe von `'\n'` (Returntaste) ausgegeben werden. Bei Betätigung einer anderen Eingabetaste oder, falls keine weiteren Lösungen existieren, soll das Programm stoppen.

Vervollständigen Sie die auf der Vorlesungsseite gegebenene Datei `countdown.hs`. Eine Lösung für das Countdown-Problem wird darin wie folgt ermittelt:

```
solutions      :: [Int] -> Int -> [Expr]
solutions ns n =  [ e | ns' <- subbags ns, e <- exprs ns', eval e == [n]]
```

wobei gilt:

(a) `subbags :: [a] -> [[a]]` bestimmt die Liste aller Permutationen aller Teillisten einer gegebenen Liste.

(b) `exprs :: [Int] -> [Expr]` liefert zu einer Zahlenliste `ns` die Liste aller Ausdrücke `e`, für die gilt `values e == ns`, wobei `values :: Expr -> [Int]` die Liste aller in einem Ausdruck vorkommenden ganzen Zahlen bestimmt.

(c) `eval :: Expr -> [Int]` bestimmt zu einem arithmetischen Ausdruck seinen Wert in einer einelementigen Liste. Falls der Ausdruck keine positive ganze Zahl ergibt oder ein Zwischenergebnis keine positive ganze Zahl ist, liefert die Funktion die leere Liste als Resultat.

19. Typinferenz — 4 Punkte

Führen Sie für die folgende Funktion eine Typinferenz mit dem in der Vorlesung vorgestellten Verfahren durch:

```
span p []     = ([], [])
span p (x:xs) = let (ys, zs) = span p xs
                in  if p x then (x:ys, zs) else ([], x:xs)
```