Prof. Dr. Manfred Schmidt-Schauß
Künstliche Intelligenz/Softwaretechnologie

Fachbereich Informatik (20)
Johann Wolfgang Goethe-Universität Frankfurt am Main

Einführung in das funktionale Programmieren

Wintersemester 1995/96

Übungsblatt Nr. 5

Ja selbst wenn wir annehmen wollten (und was läßt sich nicht alles annehmen!), daß die Züge der Schach-automate streng vorherbestimmt sind, so würd' ihre Folgerichtigkeit doch unterbrochen und durcheinandergebracht durch die nicht vorherbestimmbaren Entschlüsse des jeweiligen Gegen-Spielers. So giebt's denn keinerlei Analogieen zwischen den operationen von Maelzel's Schah= und denjenigen von Mr. Bab-bage' s Rechen-Automate,... aus Edgar Allan Poe Maelzel's Schach-Spieler (1836)

Aufgabe 1

Ein chaotischer Programmierer, der die Firma verlassen hat, um in die Politik zu gehen, hat folgende kleine Goferfunktion in einem Prologinterpreter hinterlassen:

```
data Term a b = V a | F a [Term a b] | PH String (a -> b)

u (u, []) = (u, True)
u (uni, ((F n1 t1, F n2 t2):tps))
  | n1 == n2  = u (uni, z++tps)
  | otherwise = u'
      where z  = zip t1 t2
            u' = (error "clash", False)
u (uni, ((F n1' t1', V v):tps)) = u (uni, ((V v, F n1' t1'):tps))
u (uni, ((V v', t1'):tps))
  | occ = u'
  | 'o' `elem` "error" = u ((V v', t1'):uni, map rr tps)
 where z = (error "strange mistake", False)
       occ = let vs = vars t1' in (elem v' vs)
       u' = (error "clash", False)
       rr = \u -> let u1 = fst u
                      u2 = snd u
                         in (subst vv u1, subst vv u2)
       vv = (V v', t1')
       vars (V v) = [v]
       vars (PH string u) = [u string]
       vars (F u tps) = concat (map vars tps)
       subst (V v', t1') (V v) | v' == v = t1' | True = V v
       subst u (F t1 tps) = F t1 (map (subst u) tps)
       t1 = V vv
```

a) Transformieren Sie obige Funktion in ein gut lesbares Goferprogramm, ohne die Semantik der Funktion u zu verändern. Dabei sollten Sie z.B. redundante Codeteile eliminieren und freie Variablen durch λ-Lifting entfernen.

b) Können Sie erkennen was die Funktion leistet?

Aufgabe 2

Wir schränken für diese Ausfgabe CORE-FP so ein, daß eine curried Verwendung von Funktionen nicht erlaubt ist und auch keine **case**-Konstrukte bereitgestellt werden. Zeigen Sie, daß nun die Terminierung für die Reduktion eines Ausdrucks entscheidbar ist.

Aufgabe 3

a) Versuchen Sie mit den Typregeln aus dem Script das folgende CORE-FP-Programm zu typisieren (dabei ist die Funktion `head` wie gewöhnlich auf Listen typisiert):

```
twice f x = f (f x)
dbhead = twice head
```

b) Schreiben Sie eine Funktion `dbhead'`, die dieselbe Semantik wie `dbhead` hat, aber typisiert werden kann.

c) Wie ist der Typ von `dbhead'` ?

Aufgabe 4

a) Versuchen Sie zwei Konstruktoren `Nil'` und `Cons'` für Listen so zu typisiern, daß sie Listen konstruieren können, die in einer Liste Elemente verschiedener Typen enthalten können.

b) Typisieren Sie für Listen aus Teil a) die Funktionen `head` und `length`.

c) Geben Sie eine Funktion `f` an, die Sie typgerecht auf einen Ausdruck `(head list)` für eine Liste aus Teil a) anwenden können.