Technische Universität München
Fakultät für Informatik

A. Berlea
berlea@in.tum.de
Raum: MI 02.07.053

Programmiersprachen

Wintersemester 2006/2007

5. Übungsblatt *30. November 2006*

Aufgabe 1:

Schreibe die folgenden Programme im Continuation-Passing-Style (CPS). Addition, Multiplikation und Substraktion sind dabei auch als Funktionen im CPS darzustellen.

a)
```
fun compute (a,b,c,d,e) = (a*b + c*d)-e
```

b)
```
let
  fun f x = 2*x+1
in
  f(a+b)*f(c+d)
end
```

c)
```
fun map f l =
  case l of
    nil => nil
  | h::r => f h::map f r
```

Aufgabe 2:

Die aktuelle Continuations des Aufrufs `mult [2,3]` sei k, wobei:

```
fun mult l = case l of nil => 1
                     | h::r => h * (mult r)
```

Bestimme die impliziten Continuations, die bei der Auswertung des Aufrufs auftreten.

Aufgabe 3:

Gegeben seien der polymorphische Typ `'a cont` und die folgenden Funktionssignaturen `callcc : ('a cont -> 'a) -> 'a` und `throw : 'a cont -> 'a -> 'b` und den Datentyp `datatype state = S of state cont`. Leite den Typ der folgendenen Funktion her:

a)
```
fun prod cons = prod (callcc (fn k => throw cons k))
```

b) `fun prod (S cons) = prod (callcc (fn k => throw cons (S k)))`

Aufgabe 4:

Betrachte die folgenden Funktionen:

- ```
  fun sum l = case l of nil => 0
                      | h::r => h + sum r
  ```

- ```
  fun sum l =
    let
      fun sumAcc l a = case l of nil => a
                               | h::r => sumAcc r (a + h)
    in
      sumAcc l 0
    end
  ```

- ```
  fun sum s =
    let
      fun sumCps l k =
        case l of nil => k 0
                | x::xs => sumCps xs (fn a => k (x + a))
    in
      sumCps s (fn x => x)
    end
  ```

a) Was berechnen diese Funktionen?

b) Welche Funktion hat das beste Speicherverhalten?

c) Verallgemeinere die Funktionen, so dass statt Addition, eine beliebige binäre Funktion verwendet wird. Welche dieser Funktionen sind äquivalent?
  ```