

Theorie der Programmierung (WS 2003/04)

Übungsblatt 10 (korrigierte Version)

Aufgabe 1

Welche der folgenden Ausdrücke sind wohlgetypt? Bestimmen Sie jeweils den Typ.

- a. $ref\ 0$
- b. $ref\ (ref\ 0)$
- c. $ref\ ref$
- d. $ref\ ref\ 0$
- e. $ref\ 0 := 1$
- f. $ref\ 0 := ref\ 1$
- g. $ref\ 0 := !(ref\ 1)$
- h. **let val** $x = ref\ x$ **in** x **end**
- i. **let val rec** $x : \tau = ref\ x$ **in** x **end** mit passendem Typ τ
- j. **let val rec** $x : \tau = ref\ (!x)$ **in** x **end** mit passendem Typ τ
- k. **let val** $x = e$ **in** $x := (\lambda y : \mathbf{int}. !x); x$ **end** mit passendem Ausdruck e
- l. **let val** $x = e$ **in** $x := (\lambda y : \mathbf{int}. !xy); x$ **end** mit passendem Ausdruck e

Aufgabe 2

Zeigen Sie, dass das folgende Programm wohlgetypt ist und bestimmen Sie seine small step Semantik.

```
let val  $c = \mathbf{let\ val}\ x = ref\ 0$   
      in  $(\lambda u : \mathbf{unit}. x := !x + 1,$   
         $\lambda u : \mathbf{unit}. !x)$   
      end  
      val  $inc = \#1$   
      val  $get = \#2$   
in  $inc\ c\ ();\ inc\ c\ ();\ get\ c\ ()$   
end
```

Aufgabe 3

Leiten Sie die Typregeln (SEQ), (COND-1) und (WHILE) her.

Aufgabe 4

Alternativ zum Vorgehen in der Vorlesung könnte man $e_1; e_2$ als syntaktischen Zucker für

- a. $\#2(e_1, e_2)$ oder für
- b. **let val** $x = e_1$ **in** e_2 **end** mit $x \notin \text{free}(e_1) \cup \text{free}(e_2)$

eingeführen. Wie würden dann jeweils die abgeleitete Typregel und die abgeleiteten small step Regeln für “;” aussehen?

Aufgabe 5

Überlegen Sie sich, wie man **for**-Schleifen als syntaktischen Zucker einführen könnte. Suchen Sie nach einer möglichst sinnvollen Lösung, d.h. orientieren Sie sich *nicht* an den üblichen **for**-Schleifen in imperativen Sprachen.