

## Theorie der Programmierung (WS 2003/04)

### Übungsblatt 2

#### Aufgabe 1

Beweisen Sie die Wohlgetyptheit des Ausdrucks

**let val** *sum\_of\_squares* =  $\lambda x : \mathbf{int}. \lambda y : \mathbf{int}. x * x + y * y$  **in** *sum\_of\_squares* 3 **end**

mit Hilfe der Typregeln. Welcher Typ ergibt sich für den Ausdruck?

#### Aufgabe 2

Welche der folgenden Ausdrücke sind wohlgetypt?

- a. (**if** *true* **then**  $\lambda x : \mathbf{int}. x$  **else**  $\lambda x : \mathbf{int}. \tilde{x}$ ) 5
- b.  $1 + \mathbf{let\ val\ } x = 2 \mathbf{\ in\ } x + 1 \mathbf{\ end}$
- c.  $(\lambda x : \mathbf{int}. \lambda y : \mathbf{int}. x + y) (2, 3)$
- d.  $\lambda x : \mathbf{int}. (x \geq 0) = (x \leq 0)$

#### Aufgabe 3

Zeigen Sie, dass ein Ausdruck der Form **let val** *id* = *e* **in** *id id* **end** niemals wohlgetypt sein kann. (vgl. Blatt 1, Aufgabe 2 b, c, h, i).

#### Aufgabe 4

Formulieren Sie den in der Vorlesung angedeuteten rekursiven Algorithmus zur Typüberprüfung der eingeschränkten Programmiersprache präzise.

#### Aufgabe 5

Zeigen Sie, dass die in der Vorlesung eingeführte Programmiersprache (d.h. die Menge der wohlgetypten Ausdrücke) tatsächlich nicht kontextfrei ist. Hinweis: Betrachten Sie Ausdrücke der Form

$$\mathbf{let\ val\ } f = \underbrace{\lambda x : \mathbf{int}. \dots \lambda x : \mathbf{int}. 0}_m \mathbf{\ in\ } f \underbrace{1 \dots 1}_n < f \underbrace{1 \dots 1}_p \mathbf{\ end}$$

und wenden Sie Ihre Kenntnisse aus GTI an. Es genügt die Beweisidee, keine Details!

### Aufgabe 6

Welche der folgenden Ausdrücke sind wohlgetypt? Bestimmen Sie jeweils die *Menge* aller möglichen Typen.

a. **if** *true* **then** #1 **else** #3

b. **if** *true* **then** #1 **else** =

c. #1 (1,2) + #1 (1,2,3)

d. **let val** *first* = #1 **in** *first* (1,2) + *first* (1,2,3) **end**