

Theorie der Programmierung

Übungsblatt 2

Aufgabe 1

Beweisen Sie die Wohlgetyptheit des Ausdrucks

$\text{let } \textit{sum_of_squares} = \lambda x : \textit{int}. \lambda y : \textit{int}. x * x + y * y \text{ in } \textit{sum_of_squares} 3$

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

Aufgabe 2

Welche der folgenden Ausdrücke sind wohlgetypt?

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

Aufgabe 3

Zeigen Sie, dass ein Ausdruck der Form

$\text{let } id = e \text{ in } id \ id$

niemals wohlgetypt sein kann. (vgl. Blatt 1, Aufgabe 2 b, c, h, i).

Aufgabe 4

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

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

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

Aufgabe 5

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 first = #1 in first (1, 2) + first (1, 2, 3)`

Aufgabe 6

Denken Sie sich abgeleitete Typregeln für die folgenden Ausdrücke aus:

- a. `e1 || e2`
- b. `e1 && e2`
- c. `let (id1, ..., idk) = e1 in e2`

und weisen Sie nach, dass es sich tatsächlich um abgeleitete Typregeln handelt (in **c.** nur für $k = 2$).

ÜBUNGSBLÄTTER FINDEN SICH AUF

<http://tinyurl.com/52mf6>

BESPRECHUNG DIESES ÜBUNGSBLATTES: MI 3.11.2004 (GRUPPE I) UND MI 10.11.2004 (GRUPPE II), 14-16 IN H-F 104/5.