1. Übungsblatt „Programmiersprachen“ SS 99

Leif Kornstaedt, Prof. Gert Smolka

Ausgabe: 16. April 1999 **Abgabe:** 23. April 1999

Vorbemerkung: Die Lösungen der Programmieraufgaben (I.4 bis I.6) sind per E-Mail an die Übungsgruppenleiter zu schicken; die Lösungen zu den anderen Aufgaben werden in der Vorlesung eingesammelt. Alle Programme müssen in F1 geschrieben sein. SML ist nicht zulässig.

Auf der Vorlesungsseite finden Sie eine Datei `F1.sml`, die die primitiven Operationen von F1 bereitstellt. Sie können diese in Moscow ML oder SML/NJ verwenden, indem Sie folgendes eingeben:

```
use "F1.sml";
```

Aufgabe I.1: Klammerung von Ausdrücken (6 Punkte)

Folgende BNF-Grammatik definiert eindeutig eine konkrete Syntax für F1.

dec	::=	**val** *x* = *exp*	*appexp*	::=	*atexp*
	|	**fun** x_1 x_2 = *exp*		|	*appexp atexp*
exp	::=	*appexp*	*atexp*	::=	*c*
	|	**if** exp_1 **then** exp_2 **else** exp_3		|	*x*
	|	**fn** *x* => *exp*		|	**let** *dec* **in** *exp* **end**
				|	(*exp*)

Stellen Sie für jede der folgenden Deklarationen fest, ob sie mit dieser Grammatik geparst werden kann. Ist dies der Fall, so geben Sie eine äquivalente Deklaration an, in der jedes Konditional, jede Abstraktion und jede Applikation geklammert ist und in der sonst keine Klammern vorkommen.

a. `val x = fn y => fn z => if y then f else g z`

b. `fun f x = if x y z then if a then b else c else d`

c. `val x = fn y => y fn z => plus z 1`

d.
```
fun f x = let val x = x 1 2
          in fn y => plus y x
          end (x 2 3)
```

Aufgabe I.2: Grammatik für Typausdrücke (4 Punkte)

Gegeben sei folgende Grammatik für die konkrete Syntax von F1-Typausdrücken.

ty	::=	**int** | **bool**
	|	*ty* -> *ty*
	|	(*ty*)

Mit der Zusatzbedingung, dass der Pfeiloperator rechtsassoziativ ist, wird diese Grammatik eindeutig. Geben Sie eine äquivalente Grammatik an, die ohne diese Zusatzbedingung eindeutig ist. (Zu Grammatiken siehe auch Sethi, Kapitel 2.)

Aufgabe I.3: Bindungsanalyse (6 Punkte)

Gegeben sei folgendes F1-Programm. Umkreisen Sie zunächst alle definierenden Auftreten von Identifikatoren und versehen Sie jedes davon mit einem eindeutigen Index. Versehen Sie dann jedes benutzende Auftreten mit dem Index des zugehörigen definierenden Auftretens. (Dies wurde in der Vorlesung vorgeführt.)

Bestimmen Sie außerdem zu jeder Deklaration die Menge ihrer freien Bezeichner.

```
val x = 1
val y = plus x 2
val z = fn y => (plus ((fn x => plus x 1) y) x)
val z = fn x => if equal x 0 then 0
                else z (minus x 1)
fun z x = if equal x 0 then 0
          else z (minus x 1)
fun z x = plus let val y = fn z => times z x
               in y x
               end y
```

Aufgabe I.4: Programmieren in F1 ($4 + 4$ Punkte)

a. Schreiben Sie in F1 eine Prozedur `max3: int -> int -> int -> int`, die das Maximum von drei ganzen Zahlen berechnet.

b. Schreiben Sie in F1 eine Prozedur `sqrt: int -> int`, die für nichtnegative Argumente folgende mathematische Funktion berechnet:

$$\mathrm{sqrt}(x) = \max\{n \in \mathbb{N} \mid n^2 \leq x\}$$

Aufgabe I.5: Prozeduren höherer Ordnung (3 Punkte)

Schreiben Sie in F1 eine Prozedur `sum: (int -> int) -> int -> int`, die als Argumente eine Prozedur f und eine natürliche Zahl n erhält und die Summe $\sum_{i=1}^{n} f(i)$ berechnet.

Aufgabe I.6: Typen in F1 (6 Punkte)

Geben Sie zu jedem der folgenden Typen τ einen F1-Ausdruck e an, der von diesem Typ ist. Dabei soll gelten, dass der Typ, den Moscow ML (nach `use "F1.sml";`) für e ausrechnet, exakt τ ist, also keine Typvariablen enthält (experimentieren Sie!).

a. `int -> (int -> int)`
b. `(int -> int) -> int`
c. `bool -> int`
d. `bool -> bool -> bool`
e. `bool -> (int -> bool) -> bool`
f. `(((int -> int) -> int) -> int) -> int`