2. Übungsblatt „Programmiersprachen“ SS 99

Leif Kornstaedt, Prof. Gert Smolka

Ausgabe: 23. April 1999 **Abgabe:** 30. April 1999

Aufgabe II.1: Lexikalische Syntax ($2+2+2+2$ Punkte)

In dieser Aufgabe betrachten wir eine Variante von F1, die neben den alphanumerischen Bezeichnern lexikalisch noch *symbolische* Bezeichner zulässt. Aus der „*Definition of Standard ML (Revised)*“, Abschnitt 2.4:

> An identifier is either *alphanumeric*: any sequence of letters, digits, primes (') and underbars (_) starting with a letter or prime, or *symbolic*: any non-empty sequence of the following *symbols*
>
> ```
> ! % & $ # + - / : < = > ? @ \ ~ ` ^ | *
> ```
>
> In either case, however, reserved words are excluded.

Geben Sie für jede der folgenden Zeichenfolgen an, ob sie zulässig sind und wie sie gegebenenfalls in Wortfolgen übersetzt werden. Schreiben Sie zudem jede zulässige Zeichenfolge mit einer minimalen Anzahl von Trennzeichen, ohne dass die Übersetzung in Wörter beeinflusst wird.

a. `val % = ~1`

b. `fun f : = fn . => plus . :`

c. `fun a_'#*@! = #*@! plus 1`

d.
```
fn /- => fn a' => fn ` => fn a => fn neht => fn -\ => fn fi =>
    if /- then a' else `a neht -\ fi
```

Aufgabe II.2: Statische Semantik von F1 ($1+2+6$ Punkte)

Betrachten Sie folgendes F1-Programm p:

```
fun f x = plus x
```

a. Wieviele Typumgebungen T' mit $T_0 \vdash p \Rightarrow T'$ gibt es?

b. Geben Sie ein solches T' an.

c. Geben Sie den Ableitungsbaum an, der für p die Typumgebung T' aus Teilaufgabe b herleitet (entsprechend den Regeln der statischen Semantik).

Aufgabe II.3: Dynamische Semantik von F1 ($2+1+1+1+2+1+1$ Punkte)

a. Modifizieren Sie die Regeln A.17 und A.20 der dynamischen Semantik so, dass die Wertumgebung, die in Prozeduren gespeichert wird, auf die freien Bezeichner der Prozedur eingeschränkt wird. Sie können hierbei die Funktion $\mathrm{FB} : \mathrm{Aus} \to \mathrm{Menge(Bez)}$ aus der Vorlesung benutzen.

Bearbeiten Sie dann folgende Teilaufgaben unter Verwendung der modifizierten dynamischen Semantik:

b. Sei $a_1 =$ `plus ((fn x => x) 1)`.
   Wie lautet w in $W_0 \vdash a_1 \Rightarrow w$?

c. Sei $a_2 =$ `fn x => plus 1 x`.
   Wie lautet w in $W_0 \vdash a_2 \Rightarrow w$?

d. Sei $a_3 =$ `(fn x => fn y => plus x y) 1`.
   Wie lautet w in $W_0 \vdash a_3 \Rightarrow w$?

e. Sei $a_4 =$ `let fun f x = x in f end`.
   Wie lautet w in $W_0 \vdash a_4 \Rightarrow w$, wie lautet rek w?

f. Sei $d =$ `fun f x = plus 1 x`.
   Wie lautet W in $W_0 \vdash d \Rightarrow W$?

g. Sei $p =$ `fun f x = if less x 1 then 1 else f (minus x 1)`.
   Wie lautet W in $W_0 \vdash p \Rightarrow W$?

Aufgabe II.4: Semantik von F1 mit Konjunktion ($2+3$ Punkte)

In dieser Aufgabe erweitern wir die Ausdrücke der abstrakten Syntax von F1 um ein Konstrukt a_1**andalso** a_2, das äquivalent zu folgendem Ausdruck sein soll:

if a_1 **then** a_2 **else** false

a. Formulieren Sie die Regel(n) der statischen Semantik für dieses Konstrukt.

b. Formulieren Sie die Regel(n) der dynamischen Semantik für dieses Konstrukt.

Aufgabe II.5: Monomorphie versus Polymorphie (2 Punkte)

Ein Ausdruck heißt *geschlossen* genau dann, wenn er keine freien Bezeichner hat. Geben Sie einen geschlossenen Ausdruck mit F1-Syntax an, der laut der Definition der statischen Semantik in F1 unzulässig ist, in SML aber zulässig ist. Verwenden Sie dabei nicht die Identitätsfunktion.