VIII.Übungsblatt zur Vorlesung
"Semantik von Programmiersprachen" SS 97

Prof. Gert Smolka und Martin Müller

Ausgabe: 12.6.1997 Abgabe: 19.6.1997

Die Definition der in der Vorlesung vorgestellten Sprache F ist auf dem Web unter

`http://www.ps.uni-sb.de/courses/sem-prog97/material.html`

bei den Übungsblättern erhältlich. Sie wird im folgenden vorausgesetzt. Auf derselben Seite finden Sie auch Material zu funktionaler Programmierung und SML.

Aufgabe VIII.1(Freie Variablen, 2 Punkte) Definieren Sie die Menge der freien Variablen für die Terme **Ter** von F. Geben Sie dazu durch strukturelle Induktion über **Ter** eine Funktion $\mathit{frei} : \mathbf{Ter} \to \mathcal{P}(\mathbf{Var})$ an, die jeden Term $t \in \mathbf{Ter}$ auf seine freien Variablen abbildet.

Aufgabe VIII.2(Einschrittsemantik, 4 Punkte) Geben Sie eine Einschrittsemantik für F an, so daß deren reflexiv-transitiver Abschluß mit der Ergebnissemantik von F zusammenfällt (d.h. so daß $\forall t \in \mathbf{Ter}, \forall c \in \mathbf{CTer} : t \to c$ gdw. $t \to_1^* c$). Übernehmen Sie dazu die für $\to_1 \subseteq \mathbf{GTer}_0 \times \mathbf{CTer}_0$ angegebenen Regeln und ergänzen Sie Regeln für die übrigen Terme in **Ter**. NB: Es ist kein Korrektheitsbeweis verlangt!

Aufgabe VIII.3(Induktive Definitionen, 3 Punkte) *Diese Aufgabe ist auf eines der folgenden Übungsblätter verschoben. Der Setzer.*

Aufgabe VIII.4(SML zum Warmlaufen, 5 Punkte) Implementieren sie folgende Funktionen in SML.

1. `fac: int -> int` (Fakultätsfunktion)
2. `euclid: int -> int -> int` (größter gemeinsamer Teiler)
3. `rev: 'a list -> 'a list` (Listeninvertierung)
4. `curry: (('a * 'b) -> 'c) -> ('a->'b->'c)` (Currying)
5. `uncurry` (Umkehrfunktion von `curry`)

Aufgabe VIII.5(SML, 6 Punkte) Schreiben Sie einen Interpretierer für **Imp** in SML.

1. Definieren Sie die Mengen **Aexp**, **Bexp**, **Com** und Σ durch SML-Datentypen `Aexp`, `BExp`, `Com` und `Store`.

2. Schreiben Sie Auswertungsfunktionen `evala: (Aexp * Store) -> int` und `evalb: (Bexp * Store) -> bool` für arithmetische und boolesche Ausdrücke.

3. Schreiben Sie die Auswertungsfunktion `eval: (Com * Store) -> Store` für **Imp**-Anweisungen.

4. Stellen Sie die Anweisung **while** $X < Y$ **do** $X := X * 2$ und einen Speicher σ mit $\sigma(X) = 1$ und $\sigma(Y) = 10$ als SML-Werte `com1 : Com` und `store1 : Store` dar, und werten Sie `eval (com1,store1)` aus.

Zur Vereinfachung können Sie davon ausgehen, daß die Menge der Lokationen endlich ist, z.B. $\mathbf{Loc} = \{X, Y\}$.

Die SML-Programme schicken Sie bitte per Email an den entsprechenden Übungsgruppenleiter, Tim Priesnitz tim@ps.uni-sb.de oder Martin Müller mmueller@ps.uni-sb.de.