# X.Übungsblatt zur Vorlesung "Semantik von Programmiersprachen" SS 97

Prof. Gert Smolka und Martin Müller

Ausgabe: 26.6.1997 Abgabe: 3.7.1997

**Aufgabe X.1(Kanonische Terme, 2 Punkte)** Geben Sie zwei kanonische $F_0$-Terme $t_1, t_2$ an, die verschieden aber äquivalent sind. Begründen Sie informell, warum $t_1{\sim}t_2$ gilt.

**Aufgabe X.2(Rekursive Typen, 6 Punkte)** Geben Sie geschlossene TF-Typen an, die den folgenden Datenstrukturen entsprechen. Geben Sie außerdem kanonische TF-Terme des entsprechenden Typs zu den angegebenen Werten an.

1. Natürliche Zahlen: 0, 3
2. Binäre Bäume, deren Blätter natürliche Zahlen sind: 0, ((0,1), 0)
3. Listen von Bäumen wie in (2): [0,(0,1)]
4. Bäume mit beliebigem Verzweigungsgrad, deren Blätter Zahlen sind: node(1,0,0)

**Aufgabe X.3(SML-Datentypen, 4 Punkte)** Definieren Sie in SML die TF-Typen, die Sie in Aufgabe 2 angegeben haben, zusammen mit jeweils einer Operation:

1. Den Datentyp `NAT` der natürlichen Zahlen mit der Additionsoperation `add: NAT -> NAT -> NAT`.
2. Den Datentyp `TREE` der binären Bäume, deren Blätter Elemente von `NAT` sind, und eine Funktion `sumup: TREE -> NAT`, die die Summe der Blätter eines Baums bestimmt.
3. Den Datentyp `MTREE` der Bäume mit beliebigem Verzweigungsgrad, deren Blätter Elemente von `NAT` sind, sowie die Operation `sumupm : MTREE -> NAT` entsprechend (3). (Hinweis: Verschränkt rekursive Datentypdefinitionen benötigen das SML-Schlüsselwort `and`).

**Aufgabe X.4(Typherleitung, 4 Punkte)** Es sei $\tau_0 \in \mathbf{GTyp}$ ein beliebiger geschlossener Typ, $typ(f) = \tau_0 \to \tau_0$ und $typ(x) = \mu X.X \to \tau_0$. Zeigen Sie

1. $\lambda x.(f\ (x\ x))\ :\ \mu X.X \to \tau_0$
2. $\lambda f.(\lambda x.(f\ (x\ x))\ \lambda x.(f\ (x\ x)))\ :\ (\tau_0 \to \tau_0) \to \tau_0$

**Aufgabe X.5(Divergenz, 4 Punkte)** Geben Sie TF-Terme an, deren Auswertung divergiert. Wählen Sie $t_1, t_2, t_3 \in$ **GTer** und $\tau_1, \tau_2, \tau_3 \in$ **GTyp** so, daß $t_1{:}\tau_1$, $t_2{:}\tau_2$, $t_3{:}\tau_3$ und

1. $t_1$ enthält **rec**, aber $\tau_1$ enthält kein $\mu$
2. $t_2$ enthält kein **rec**, aber $\tau_2$ enthält $\mu$
3. $t_3$ enthält kein **rec**, und $\tau_3$ enthält kein $\mu$

**Aufgabe X.6(F0-Interpretierer, 10 Zusatzpunkte)** Schreiben Sie einen Interpretierer für (leicht erweitertes) $F_0$ in SML. Stellen Sie eine geschlossenen $F_0$-Term als Paar $(E, t)$ aus einem beliebigen $F_0$-Term und einer Bindungsumgebung $E$ für seine freien Variablen $FV(t)$ dar. Solche Paare heißen üblicherweise *Closures*. Ein Vorteil dieser Darstellung ist es, daß man zur Evaluierung der Applikation keine explizite Termsubstitution wie in $(\lambda x.t)\ c \to t[c/x]$ benötigt, sondern nur eine Erweiterung der Bindungsumgebungen. Zum Beispiel, wenn $C = (\{z \mapsto t_z\}, \lambda x'.(x'\ z))$, dann ist eine typische Auswertung von der folgenden Form.

$$(\{y \mapsto 21\}, \lambda x.y + y)\ C \quad \to_1 \quad (\{y \mapsto 21, x \mapsto C\}, y + y) \quad \to_1 \quad 21 + 21 \quad \to_1 \quad 42$$

1. Implementieren Sie den $F_0$-Interpretierer als eine Funktion `evalf0: Env -> Lambda -> Closure` und verwenden Sie dafür die folgenden Definitionen.

```
  infixr 6 ^; infix 7 @;
  type Var = int; datatype Binder = \ of Var;
(* F0-Terme *)
  datatype Lambda = X of int | ^ of Binder * Lambda
    | @ of Lambda * Lambda | Int of int | Plus of Lambda * Lambda;
(* Bindungsumgebungen *)
  datatype Env = Nil | Cons of (Var * Closure) * Env
       and Closure = Term of Lambda | Clos of (Env * Closure);
(* Ausnahmen *)
  exception variableUndefined;
  exception applyingNonFunction;
  exception addingNonIntegers;
```

2. Testen Sie Ihren Interpretierer an der in der Vorlesung vorgestellten Kodierung von Booleschen Werten, Paaren, natürlichen Zahlen und dem Rekursionsoperator. Nach geeigneter Definition von `zero, true, false, inc, eq: Lambda` sollte die Auswertung von

   - `evalf0 Nil (eq @ (inc @ zero) @ (inc @ zero))`
     das Ergebnis `Clos(e, (Term true))` und

- `evalf0 Nil (eq @ zero @ (inc @ zero))`
  das Ergebnis `Clos(e, (Term false))` liefern,

wobei `e` die entsprechende Bindungsumgebung ist.