Institut für Informatik
der Universität München
Dr. Andreas Abel

SS 2007

20. April 2007

Übungen zur Vorlesung
Typsysteme

Blatt 1

Präsenzaufgaben (P-XX) werden in der Übung gemeinsam gelöst, die Hausaufgaben (H-XX) dann alleine oder in 2er-Teams bis zur nächsten Übungsstunde.

Aufgabe P-1 (Kongruenzregeln für Mehr-Schritt-Auswertung): Beweisen Sie für die Small-Step-Semantik $\longrightarrow$ aus der Vorlesung:

a) Wenn $r \longrightarrow^* r'$, dann $r\,s \longrightarrow^* r'\,s$.

b) Wenn $s \longrightarrow^* s'$, dann $v\,s \longrightarrow^* v\,s'$.

Den Beweis können Sie durch Induktion über die Länge der Reduktionsfolge $\longrightarrow^*$ führen.

Aufgabe P-2 (Big-Step-Semantik und deren Korrektheit): Ein Interpreter wird üblicherweise so implementiert, dass er ein Programm *in einem großen Schritt* auswertet. Einen Interpreter für den ungetypten λ-Kalkül mit Ganzzahlen und Addition können wir mit folgender induktiver Relation $t \searrow v$ modellieren:

$$\text{E-VAL}\ \frac{}{v \searrow v} \qquad \text{E-APP-BETA}\ \frac{r \searrow \lambda x t \qquad s \searrow v \qquad t[v/x] \searrow w}{r\,s \searrow w}$$

$$\text{E-APP-PLUS}\ \frac{r \searrow + \qquad s \searrow n}{r\,s \searrow +n} \qquad \text{E-APP-PLUS-N}\ \frac{r \searrow +n \qquad s \searrow m}{r\,s \searrow (n+m)}$$

Dabei rangieren n, m über Ganzzahlkonstanten, und $v, w ::= \lambda x t \mid n \mid + \mid +n$ über Werte.

Zeigen Sie: $t \searrow v$ impliziert $t \longrightarrow^* v$, durch Induktion über $t \searrow v$.

Aufgabe P-3 (Repräsentation von gebundenen Variablen durch de Bruijn Indizes): Um das Problem des Einfangens von Variablen bei der Substitution zu umgehen, ohne ständig Variablen umbenennen zu müssen, kann man gebundene Variablen durch natürliche Zahlen (sog. de Bruijn Indizes) darstellen. Dabei wird eine gebundene Variable durch die Zahl i repräsentiert, falls zwischen ihr und dem sie bindenden λ noch i weitere λs stehen. Zum Beispiel wir der Term $\lambda x.\, x\,(\lambda y.\, y\,x)$ repräsentiert durch $\lambda.\, 0\,(\lambda.\, 0\,1)$. Die λs müssen nun

natürlich den Namen der gebundenen Variable nicht mehr tragen. Beachten Sie, dass die Variable x einmal den Index 0 erhält, da sie dort die zuletzt gebundene Variable ist, und einandermal den Index 1, da sich ein λ (der Binder für y) zwischen ihr und dem Ort ihrer Bindung befindet. Freie Variablen behalten ihren Namen, also wird $\lambda x \lambda y.\, z\, y\, x$ repräsentiert durch $\lambda\lambda.\, z\, 0\, 1$.

Geben Sie Repräsentationen von folgenden Termen an:

a) $(\lambda y \lambda x.\ y\ (\lambda y\ x))\ (\lambda x\ x)\ (\lambda y.\ y\ y)$

b) $(\lambda x.\, y\, x\, (\lambda y.\, z\, x\, y\, (\lambda z.\, z\, y\, x)))$

Aufgabe H-1 (Vollständigkeit der Big-Step-Semantik [6 Punkte]): Zeigen Sie $t \longrightarrow t'$ und $t' \searrow v$ implizieren $t \searrow v$ durch Induktion über $t' \searrow v$. Folgern Sie daraus die Vollständigkeit der Big-Step-Semantik in Bezug auf die Small-Step-Semantik: Wenn $t \longrightarrow^* v$, dann $t \searrow v$.

Aufgabe H-2 (Implementierung des ungetypten λ-Kalküls [8 Punkte]): In SML kann man λ-Terme in de Bruijn Repräsentation wie folgt durch einen algebraischen Datentypen implementieren.

```
datatype Tm =
    BV  of int           (* bound variable     *)
  | FV  of string        (* free  variable     *)
  | Abs of string * Tm   (* lambda-abstraction *)
  | App of Tm * Tm       (* application        *)
```

Dabei ist die Komponente `string` am Konstruktor `Abs` nur ein *Vorschlag* für den Namen der dort gebundenen Variable, und dient *nur* zum Drucken des Terms.

Schreiben Sie eine Funktion `print : Tm -> string` die eine Term ausdruckt. Achten Sie darauf, dass die in dieser Funktion erzeugten Namen für gebundene Variablen nicht miteinander und mit den Namen der freien Variablen in Konflikt treten!

Aufgabe H-3 (Interpreter [6 Punkte]): In de Bruijn Repräsentation lautet die β-Reduktionsregel $(\lambda t)\, v \longrightarrow t[v/0]$, der bislang durch das λ gebundene Index 0 in t wird durch den Term v ersetzt. Substitution $t[s/i]$ wird wie folgt implementiert:

```
(* Substitution subst(s,i,t) of s for index i in t *)
fun subst (s, i, BV j) = if i=j then s else BV j
  | subst (s, i, FV x) = FV x
  | subst (s, i, Abs(x,t)) = Abs (x, subst(s,i+1,t))
  | subst (s, i, App(t,u)) = App (subst(s,i,t), subst(s,i,u))
```

Die Verallgemeinerung auf beliebiges i (statt nur für Index 0) ist nötig, da sich der Index der zu ersetzenden Variable um eins erhöht, wenn man unter ein λ schreitet.

Implementieren Sie einen Interpreter für den λ-Kalkül basierend auf der Big-Step-Semantik und testen Sie ihn an einigen Termen.
Z.B., $(\lambda x \lambda y \lambda z.\ (x\ z)\ (y\ z))\ (\lambda x \lambda y\ x)\ (\lambda x \lambda y\ x)\ (\lambda x x)$.

Abgabe bis Freitag, 27.04. zum Beginn der Übungsstunde. Die Programme bitte per email an `abel tcs ifi lmu de`, die anderen Aufgaben kann man handschriftlich einreichen. Statt SML darf auch Ocaml oder Haskell verwendet werden.