Institut für Informatik
der Universität München
Dipl. Inf. Andreas Abel

WS 2006/07

22. Oktober 2008

Übungen zur Vorlesung
Lambda-Kalkül

Blatt 1

Hinweis: Statt SML können Sie auch eine andere Programmiersprache verwenden (Scheme, Haskell, ...).

Aufgabe P-1 (Terme als Bäume): Stellen Sie die Terme $\lambda x.\, x\, \lambda y.\, x\, y\, y$ und $(\lambda x.\, z\, x)\, \lambda z z$ als Bäume dar.

Aufgabe P-2 (Lokal-namenlose Repräsentation von Termen):

Geben Sie lokal-namenlose 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 P-3 (Implementierung lokal-namenloser Terme): In SML kann man λ-Terme in lokal-nameloser 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 `equal : Tm * Tm -> bool`, die zwei Terme auf (α-)Gleichheit testet.

Aufgabe P-4 (Namensverlust): Benamste Terme werden durch den folgenden Typen repräsentiert.

```
datatype NTm =
    Var of string       (* variable           *)
  | Abs of string * NTm (* lambda-abstraction *)
  | App of NTm * NTm    (*                    *)
```

Schreiben Sie eine Funktion `toNameless : NTm -> Tm`, die eine benamsten Term in die lokal namenlose Form überführt.

Aufgabe P-5 (Syntaxgerichtete Definition der α-Äquivalenz): Wiederholung: $=_\alpha$ ist die kleinste kompatible Äquivalenzrelation über dem Axiomenschema:

$$\text{EQ}_\alpha\text{-AX}\ \frac{}{\lambda x t =_\alpha \lambda y.\, t[y/x]}\ y \notin \mathsf{FV}(t)$$

Wir definieren nun $=^\alpha$ als die kleinste Relation, die unter den folgenden Regeln abgeschlossen ist.

$$\text{EQ}_\alpha\text{-VAR}\ \frac{}{x =^\alpha x} \qquad \text{EQ}_\alpha\text{-APP}\ \frac{r =^\alpha r' \qquad s =^\alpha s'}{r\, s =^\alpha r'\, s'}$$

$$\text{EQ}_\alpha\text{-ABS}\ \frac{t[y/x] =^\alpha t'[y/x']}{\lambda x t =^\alpha \lambda x' t'}\ y \notin \mathsf{FV}(t, t')$$

Zu zeigen ist nun, dass die beiden Definitionen der α-Äquivalenz gleichwertig sind.

Zeigen Sie die Korrektheit von $=^\alpha$, also die Aussage $t =^\alpha t' \implies t =_\alpha t'$, durch (Regel-)Induktion über die Herleitung von $t =^\alpha t'$.

Aufgabe H-1 (Drucken von Termen):

a) Schreiben Sie eine Funktion `toNamed : Tm -> NTm`, die einen wohlgeformten lokal-namenlosen Term in die benamste Form überführt. Bemerkung: Es muss natürlich `equal(toNameless(toNamed(t)),t).` für alle l.-n. Terme `t` gelten.

b) Schreiben Sie eine Funktion `toString : NTm -> string`, die einen benamsten Term für die Ausgabe linearisiert.

Aufgabe H-2 (Syntaxgerichtete Definition der α-Äquivalenz): Zu zeigen ist, dass die beiden Definitionen der α-Äquivalenz gleichwertig sind.

a) In der Präsenzaufgabe wurde schon die Korrektheit von $=^\alpha$ gezeigt, also die Aussage $t =^\alpha t' \implies t =_\alpha t'$.

b) Zeigen Sie, dass $=^\alpha$ eine Äquivalenzrelation (reflexiv, transitiv, symmetrisch) ist, jeweils durch eine geeignete Induktion.

c) Zeigen Sie schliesslich die Vollständigkeit, also die Aussage $t =_\alpha t' \implies t =^\alpha t'$, durch Induktion über die Herleitung von $t =_\alpha t'$.

Weiterführende Frage: Inwiefern ergibt sich aus $=^\alpha$ ein Algorithmus zur Entscheidung der α-Äquivalenz?

Abgabe der bearbeiteten Übungen (H-1 bis H-3): Mittwoch, 29. Oktober 2008, zu Beginn der Vorlesung. Wurden die Aufgaben in einem Zweierteam bearbeitet, können die Lösungen auch gemeinsam eingereicht werden.