Übungsblatt 5

Abgabe der Lösungen: Do, 06.06., 12:00 im Briefkasten am blauen Hochhaus

Präsenzaufgaben

Zur Notation

Wir schreiben λ-Terme den folgenden Regeln entsprechend abgekürzt:

- *Applikation ist links-assoziativ.*
  Beispielsweise kürzen wir $((x(yz))u)v$ zu $x(yz)uv$.
- *Abstraktion reicht so weit wie möglich.*
  Beispielsweise kürzen wir $\lambda x.(x(\lambda y.(yx)))$ zu $\lambda x.x(\lambda y.yx)$.
- *Aufeinanderfolgende Abstraktionen werden zusammengefasst.*
  Beispielsweise kürzen wir $\lambda x.\lambda y.\lambda z.yx$ zu $\lambda xyz.yx$.

Übung 1 α-Äquivalenz und β-Reduktion

1. Entscheiden Sie für jedes der folgenden Paare von λ-Termen, ob die Terme jeweils α-äquivalent zueinander sind:

   (a) $\lambda xy.xy =^?_\alpha \lambda uv.uv$

   (b) $\lambda xy.xy =^?_\alpha \lambda uv.vu$

   (c) $(\lambda x.xx)(\lambda y.yy) =^?_\alpha (\lambda x.xx)(\lambda x.xx)$

2. Entscheiden Sie in jedem der folgenden Fälle, ob der jeweilige Reduktionsschritt eine zulässige β-Reduktion darstellt:

   (a) $(\lambda xyz.xyz)(\lambda y.yy) \to^?_\beta \lambda yz.(\lambda y.yy)yz$

   (b) $(\lambda xyz.xyz)(yy) \to^?_\beta \lambda yz.(yy)yz$

   (c) $(\lambda xyz.xyz)(yy) \to^?_\beta \lambda yz.(uu)yz$

   (d) $(\lambda xyz.xy((\lambda u.ux)(yy)))uv \to^?_\beta (\lambda xyz.xy((yy)x))uv$

Übung 2 We Are Not Anonymous (Functions)

Funktionale Programmiersprachen wie zum Beispiel HASKELL oder ML erweitern den λ-Kalkül (bzw. getypte Fragmente des λ-Kalküls) um verschiedene Konstrukte. Insbesondere werden *Definitionen* verwendet, um Funktionen zu *benennen*. Beispielsweise können wir die folgenden drei Gleichungen angeben:

$$\begin{aligned} &\mathit{flip} = \lambda f\, x\, y.\, f\, y\, x \\ &\mathit{const} = \lambda x\, y.\, x \\ &\mathit{twice} = \lambda f\, x.\, f\, (f\, x) \end{aligned}$$

und sie als – von links nach rechts zu lesende – Reduktionsregeln betrachten, die bei der Auswertung eines λ-Terms angewendet werden können. Um sie von β-Reduktionen zu unterscheiden, werden solche Reduktionen als δ-Reduktionen ("δ" wie "Definition") bezeichnet. Beispielsweise ist mit den obigen Definitionen die folgende Reduktion möglich:

$$(\lambda f.\, f\, u)\ \mathit{const} \to_\beta \mathit{const}\ u \to_\delta (\lambda x\, y.\, x)\ u \to_\beta \lambda y.\, u$$

Hinweis: Eine derartige Reduktion, bei der sowohl β- als auch δ-Schritte vorkommen können, bezeichnen wir ab sofort mals $\beta\delta$-Reduktion.

1. Ermitteln Sie die $\beta\delta$-Normalformen der folgenden Terme:

   (a) *flip const twice*

   (b) *twice flip*

2. Tatsächlich ist es in den angegebenen Programmiersprachen praktischerweise auch möglich, Funktionsargumente auf der *linken Seite* einer Funktionsdefinition anzugeben. Die obigen Definitionen würden dann beispielsweise wie folgt geschrieben werden:

   $$\begin{aligned} &\mathit{flip}\ f = \lambda x\, y.\, f\, y\, x \\ &\mathit{const}\ x\, y = x \\ &\mathit{twice}\ f\, x = f\ (f\, x) \end{aligned}$$

   Die mit diesen Definitionen verbundenen Reduktionsregeln sind nun nicht mehr offensichtlich; deshalb verwenden wir Großbuchstaben F, X, Y, um Variablen, die in einem Term vorkommen, von Variablen, die durch eine Abstraktion gebunden werden können, zu unterscheiden. Außerdem verwenden wir zur Verdeutlichung explizite Klammerung:

   $$\begin{aligned} \mathit{flip}\ F &\to_\delta (\lambda x\, y.\, f\, y\, x)[F/f] \\ (\mathit{const}\ X)\ Y &\to_\delta x[X/x,\ Y/y] \\ (\mathit{twice}\ F)\ X &\to_\delta (f\ (f\, x))[X/x,\ F/f] \end{aligned}$$

   Entscheiden Sie, welche der folgenden λ-Terme bezüglich dieser neuen Definitionen $\beta\delta$-Normalformen sind:

   (a) $\lambda x.\, \mathit{flip}\ x$

   (b) $\lambda x.\, \mathit{const}\ x$

   (c) *const flip twice*

Übung 3 Church-Numerale

Funktionale Sprachen fügen weiterhin *eingebaute* Typen (*built-in types*) zum λ-Kalkül hinzu und erlauben auch die Definition von benutzerdefinierten Typen (*user-defined types*). Diese Typen können jedoch prinzipiell allesamt unter Einsatz der sogenannten *Church-Kodierung* direkt als λ-Terme kodiert werden.

Eine natürliche Zahl n wird durch einen λ-Term $\lceil n \rceil$ kodiert, der eine gegebene Funktion f wiederholt n-mal auf einen Startwert a anwendet, was wir informell als n "Iterationen" von f startend bei a ansehen können:

$$\lceil n \rceil := \lambda f a.\, \underbrace{f(f(f(\ldots f}_{n}\, a))) \tag{1}$$

Wir kodieren dies einheitlich wie folgt:

$$
\begin{aligned}
&\mathit{zero} &&= \lambda f\, a\, .\, a\\
&\mathit{succ}\ n &&= \lambda f\, a\, .\, f\ (n\ f\ a)\\
\\
&\mathit{one} &&= \mathit{succ}\ \mathit{zero}\\
&\mathit{two} &&= \mathit{succ}\ \mathit{one}\\
&\mathit{three} &&= \mathit{succ}\ \mathit{two}\\
&\mathit{four} &&= \mathit{succ}\ \mathit{three}
\end{aligned}
$$

1. Zeigen Sie, dass für jede natürliche Zahl n gilt: $\mathit{succ}\ \lceil n \rceil \to^*_{\beta\delta} \lceil n+1 \rceil$.

2. Definieren Sie die Addition und Multiplikation natürlicher Zahlen bezüglich dieser Kodierung. Vervollständigen Sie also die folgenden Definitionen:

   $$
   \begin{aligned}
   &\mathit{add}\ \ n\ m &&= \ldots\\
   &\mathit{mult}\ n\ m &&= \ldots
   \end{aligned}
   $$

   derart, dass für alle x und y gilt:

   $$\mathit{add}\ \lceil x \rceil \lceil y \rceil \to^*_{\beta\delta} \lceil x + y \rceil \qquad\qquad \mathit{mult}\ \lceil x \rceil \lceil y \rceil \to^*_{\beta\delta} \lceil x \cdot y \rceil$$

HAUSAUFGABEN

Übung 4 α-Äquivalenz und β-Reduktion II (8 Punkte)

1. Entscheiden Sie für jedes der folgenden Paare von λ-Termen, ob die Terme jeweils α-äquivalent zueinander sind und begründen Sie Ihre Antwort:

   (a) $\lambda f.f(\lambda xy.yx) =^?_\alpha \lambda y.y(\lambda fx.xf)$

   (b) $(\lambda fgx.f(\lambda y.gxy)) =^?_\alpha (\lambda gfy.g(\lambda y.fyy))$

   (c) $(\lambda xy.(\lambda x.xx)yx) =^?_\alpha (\lambda xy.(\lambda y.yy)yx)$

2. Entscheiden Sie in jedem der folgenden Fälle, ob der jeweilige Reduktionsschritt eine zulässige β-Reduktion darstellt. *Falls nein*, geben Sie stattdessen eine gültige β-Reduktion an. *Falls ja*, begründen Sie Ihre Antwort.

   (a) $(\lambda fx.fxf)(\lambda y.xx) \to^?_\beta \lambda v.(\lambda y.xx)v(\lambda y.xx)$

   (b) $(\lambda fgh.ghf)(vv) \to^?_\beta \lambda fg.g(vv)f$

   (c) $(\lambda xyz.xy((\lambda u.ux)(yy)))uv \to^?_\beta (\lambda xuz.xu((yy)x))uv$

3. Die folgenden λ-Terme haben eindeutige β-Normalformen. Finden Sie diese und geben Sie jeweils eine vollständige Herleitung der Normalform an.
   Hinweis: Die Normalformen sind "eindeutig", wenn wir die Namen gebundener Variablen ignorieren (d.h. *eindeutig modulo α-Äquivalenz*).

   (a) $(\lambda fxy.f(fy)(xx))(\lambda uv.u)$

   (b) $(\lambda fgx.(\lambda z.f(xz))(gx))(\lambda x.gx)(\lambda z.z)zz$

Übung 5 Church-Kodierung (4 Punkte)

Um alternativ einen Wert vom Typ A oder vom Typ B zurückzugeben, werden in funktionalen Sprachen häufig *Summentypen* $A + B$ verwendet. Beispielsweise ist ein Element vom Typ $A + B$ entweder ein Element vom Typ A oder ein Element vom Typ B, wobei eindeutig gekennzeichnet ist, welche Alternative vorliegt. Die Primitiven für solche Typen können im λ-Kalkül folgendermaßen kodiert werden:

$$\begin{aligned} &\mathit{inl}\ a = \lambda\, f\, g\, .\, f\, a \\ &\mathit{inr}\ b = \lambda\, f\, g\, .\, g\, b \\ &\mathit{case} = \lambda\, f\, g\, s\, .\, s\, f\, g \end{aligned}$$

Zeigen Sie, dass für alle λ-Terme s, t und u gilt:

$$\mathit{case}\ s\ t\ (\mathit{inl}\ u) \to^*_{\beta\delta} s\ u \qquad\qquad \mathit{case}\ s\ t\ (\mathit{inr}\ u) \to^*_{\beta\delta} t\ u$$

Übung 6 η-Reduktion (8 Punkte)

Zusätzlich zur β-Reduktion, die das Einsetzen von Termen für gebundene Variablen in λ-Termen beschreibt, kann man noch die Ansicht vertreten, dass ein Term t dieselbe Funktion beschreibt wie der Term $\lambda x.\, tx$, also eine Äquivalenz

$$\lambda x.\, tx =_\eta t \qquad (x \notin FV(t))$$

einführen (ähnlich wie bei α-Äquivalenz). Zugehörig ist eine Reduktionsregel

$$\lambda x.\, tx \to_\eta t \qquad (x \notin FV(t)).$$

1. Nach einem Theorem von Böhm gibt es für zwei in Bezug auf $\beta\eta$-Reduktion unterschiedliche Terme ohne freie Variablen s und t ein $n \in \mathbb{N}$ und Terme $u_1, \ldots, u_n$ (ebenfalls ohne freie Variablen), sodass

   $$\begin{aligned} s\, u_1 \ldots u_n\, x\, y &\leftrightarrow^*_{\beta\eta} x \\ t\, u_1 \ldots u_n\, x\, y &\leftrightarrow^*_{\beta\eta} y \end{aligned}$$

   Finden Sie ein n sowie passende u_i, $i = 1 \ldots n$, um $\lceil 0 \rceil$ und $\lceil 1 \rceil$ auf diesem Wege zu unterscheiden. **Hinweis:** η-Reduktion könnte bei der Suche nach geeigneten Termen hilfreich sein.

2. Sei nun $I = \lambda x.x$. Zeigen Sie, dass wir auch mit römischen Zahlen rechnen können:

   $$\mathit{add}\ I\ I \leftrightarrow^*_{\beta\eta} \lceil 2 \rceil.$$

3. Gegeben seien die λ-Terme (sogenannte Kombinatoren) $B = \lambda fgx.f(gx)$ und $C = \lambda fxy.fyx$. Zeigen Sie, dass $BCC \leftrightarrow^*_{\beta\eta} I$.