Übungen zur "Theorie der Programmierung" - SoSe2015

Freitag 12:15-13:45, Martenstr. 3, Raum 02.133-113

Daniel Hausmann

daniel.hausmann@fau.de

Friedrich-Alexander-Universität Erlangen
Department Informatik

Lehrstuhl 8

May 15, 2015

Wiederholung aus der Vorlesung

λ-Term

Die Menge der λ-Terme $T_\lambda(V)$ über V ist induktiv definiert:

- jede Variable $x \in V$ ist λ-Term.
- Abstraktion: Ist t λ-Term und x Variable, so ist $(\lambda x.t)$ λ-Term.
- Applikation: Sind s und t λ-Terme, so ist $(s\ t)$ λ-Term.

Freie Variablen

Die Menge $FV(t)$ der freien Variablen von t ist induktiv definiert:

- $FV(x) = \{x\}$
- $FV(\lambda x.t) = FV(t) - \{x\}$
- $FV(s\ t) = FV(s) \cup FV(t)$

Wiederholung aus der Vorlesung

Kollisionsfreie Substitution

Die kollisionsfreie Substitution $s\sigma$ ist induktiv definiert:

- $x\sigma = \sigma(x)$
- $(t\ s)\sigma = (t\sigma)(s\sigma)$
- $(\lambda x.t)\sigma = \lambda x.(t\sigma)$, wenn $\sigma(x) = x$ und $x \notin FV(\sigma(y))$ für $y \in FV(t)$ mit $\sigma(y) \neq y$; andernfalls: zuerst Umbenennung gebundener Variablen per α-Äquivalenz.

α-Äquivalenz

$(\lambda x.t) =_\alpha (\lambda y.t[y/x])$, wenn $y \notin FV(t)$.

Wiederholung aus der Vorlesung

β-Reduktion

Reduktionsregel: $(\lambda x.t)\ x \rightarrow_{0,\beta} t$, erlaubt insbesondere

$$(\lambda x.s)\ t \rightarrow_{\beta} s[t/x]$$

wobei $(\lambda x.s)\ t$ als β-Redex bezeichnet wird.

η-Reduktion

Reduktionsregel: $\lambda x.y\ x \rightarrow_{\eta} y$.

δ-Reduktion

Einsetzen einer Definition eines benannten λ-Ausdrucks ($fun = \lambda x.t$)

$$\lambda y.fun \rightarrow_{\delta} \lambda y.(\lambda x.t)$$

Übung 1 - α-Äquivalenz und β-Reduktion

- Applikation ist links-assoziativ:

  $x(yz)uv$ anstelle von $((x(yz))u)v$

- λ's reichen so weit, wie möglich:

  $\lambda x.x(\lambda y.yx)$ anstelle von $\lambda x.(x(\lambda y.(yx)))$

- Aufeinanderfolgende λ's werden zusammengefasst:

  $\lambda xyz.yx$ anstelle von $\lambda x.\lambda y.\lambda z.yx$

1 Gelten die folgenden α-Äquivalenzen?

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

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

c) $\lambda xy.xy =^?_\alpha \lambda yx.yx$

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

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

f) $(\lambda x.x(\lambda y.yx)) =^?_\alpha (\lambda y.y(\lambda y.yy))$

Übung 1 - α-Äquivalenz und β-Reduktion

2. Handelt es sich um zulässige β-Reduktionen?

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

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

   c) $(\lambda xyz.xyz)(yy) \rightarrow_\beta^? \lambda uz.(yy)uz$

   d) $(\lambda xyz.xyz)(yy) \rightarrow_\beta^? \lambda yz.(uu)yz$

   e) $(\lambda xyz.xyz)(uu) \rightarrow_\beta^? \lambda xy.xy(uu)$

   f) $(\lambda xyz.xy((\lambda u.ux)(yy)))uv \rightarrow_\beta^? (\lambda xyz.xy((yy)x))uv$

   g) $(\lambda xyz.xy((\lambda u.ux)(yy)))uv \rightarrow_\beta^? (\lambda xuz.xu((yy)x))uv$

3. Geben Sie eine vollständige Herleitung der Normalform der folgenden Terme an:

   a) $(\lambda xyz.x(zz)y)(\lambda uv.v)$

   b) $(\lambda xy.(\lambda zu.y(uz)x)xx)uvu$

Übung 2 - We are not Anonymous (functions)

Benennung von Funktionen:

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

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

1. Ermitteln Sie die $\beta\delta$-Normalformen der folgenden Terme:
   a) $\mathit{flip}\ \ \mathit{const}\ \ \mathit{twice}$
   b) $\mathit{twice}\ \ \mathit{flip}$

Übung 2 - We are not Anonymous (functions)

2 Konkretere Funktionsdefinitionen:

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

Die zugehörigen δ-Reduktionsregeln:

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

Entscheiden Sie, welche der folgenden λ-Terme diesbezüglich $\beta\delta$-Normalformen sind:

a) $\lambda x\ .\ \ \mathit{flip}\ \ x$
b) $\lambda x\ .\ \ \mathit{const}\ \ x$
c) $\mathit{const}\ \ \mathit{flip}\ \ \mathit{twice}$

Übung 4 - Church-Kodierung

Kodierung von Typen durch λ-Terme:

1. Boolesche Wahrheitswerte und Funktionen:

$$\begin{aligned} \mathit{true} &= \lambda x\ y\ .\ x \\ \mathit{false} &= \lambda x\ y\ .\ y \\ \mathit{if\_then\_else} &= \lambda b\ x\ y\ .\ b\ x\ y \end{aligned}$$

1. Zeigen Sie für alle λ-Terme s und t:

$$\mathit{if\_then\_else}\ \ \mathit{true}\ \ s\ \ t \rightarrow^*_{\beta\delta} s \qquad \mathit{if\_then\_else}\ \ \mathit{false}\ \ s\ \ t \rightarrow^*_{\beta\delta} t$$

2. Vervollständigen Sie die Definitionen der folgenden Funktionen:

$$\begin{aligned} \mathit{not}\ b &= \ldots \\ \mathit{and}\ b1\ b2 &= \ldots \\ \mathit{or}\ \ b1\ b2 &= \ldots \end{aligned}$$

Übung 4 - Church-Kodierung

2 Paare:

$$\begin{aligned} pair\ a\ b &= \lambda select\ .\ \ select\ \ a\ \ b \\ fst\ \ p &= p\ (\lambda x\ y\ .\ \ x) \\ snd\ p &= p\ (\lambda x\ y\ .\ \ y) \end{aligned}$$

Schreiben Sie eine Funktion *swap*, welche die beiden Komponenten eines gegebenen Paares austauscht, sodaß für alle λ-Terme s und t:

$$fst\,(swap\ (pair\ \ s\ \ t)) \rightarrow^*_{\beta\delta} t \qquad snd(swap\ (pair\ \ s\ \ t)) \rightarrow^*_{\beta\delta} s$$

Übung 4 - Church-Kodierung

3. Church-Numeral: $\lceil n \rceil := \lambda f a.\, \underbrace{f(f(f(\ldots f}_{n}\, a))))$

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

1. Zeigen Sie für alle n: $\mathit{succ}\ \lceil n \rceil \to^*_{\beta\delta} \lceil n+1 \rceil$.
2. Definieren Sie *add*, *mult*, *exp*, so daß für alle x und y gilt:

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

$$\mathit{exp}\ \lceil x \rceil \lceil y \rceil \to^*_{\beta\delta} \lceil x^y \rceil$$

3. Definieren Sie eine Funktion *isZero*, so daß:

$$\mathit{isZero}\ \lceil 0 \rceil \to^*_{\beta\delta} \mathit{true} \qquad \mathit{isZero}\ \lceil n+1 \rceil \to^*_{\beta\delta} \mathit{false}$$