Friedrich-Alexander-Universität
Technische Fakultät

FAU

Theorie der Programmierung

Übung 06 - der (ungetypte) λ-Kalkül II

Leon Vatthauer

5. Juni 2023

Church-Numerale

Wir betrachten erneut die Church-Kodierung natürlicher Zahlen vom vorigen Übungsblatt:

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

mit einheitlicher Kodierung

```
zero = λ f a. a
succ n = λ f a. f (n f a)
```

```
one = succ zero
two = succ one
three = succ two
four = succ three
```

sowie den Operationen `add` und `mult` mit der entsprechenden Semantik.

Notation: Von nun an schreiben wir `s + t` und `s * t` anstelle von `add s t` und `mult s t` und verwenden die $\beta\delta$-Regeln

$$\lceil n \rceil + \lceil m \rceil \to^*_{\beta\delta} \lceil n + m \rceil \qquad \lceil n \rceil * \lceil m \rceil \to^*_{\beta\delta} \lceil n \cdot m \rceil$$

Aufgabe 1.1

Church-Kodierung von Booleans

Boolesche Wahrheitswerte werden als λ-Terme wie folgt definiert:

$$\begin{aligned} \mathit{true} &= \lambda x\, y.\, x \\ \mathit{false} &= \lambda x\, y.\, y \\ \mathit{ite} &= \lambda b\, x\, y.\, b\, x\, y \end{aligned}$$

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

$$\mathit{ite}\ \mathit{true}\ s\ t \to^*_{\beta\delta} s \qquad \mathit{ite}\ \mathit{false}\ s\ t \to^*_{\beta\delta} t$$

Aufgabe 1.2

Church-Kodierung von Booleans

Boolesche Wahrheitswerte werden als λ-Terme wie folgt definiert:

$$\begin{aligned} \mathit{true} &= \lambda x\, y.\, x \\ \mathit{false} &= \lambda x\, y.\, y \\ \mathit{ite} &= \lambda b\, x\, y.\, b\, x\, y \end{aligned}$$

Vervollständigen Sie die folgenden Funktionsdefinitionen so, dass sie (unter normaler Reduktion) boolesche Negation, exklusives Oder und Implikation berechnen:

$$\begin{aligned} \mathit{not}\ b &= \ldots \\ \mathit{xor}\ b1\ b2 &= \ldots \\ \mathit{imp}\ b1\ b2 &= \ldots \end{aligned}$$

Notation: Von nun an schreiben wir „**if** *s* **then** *t* **else** *u*“ anstelle von „*ite s t u*“

Aufgabe 2

Rekursive Definitionen

In den meisten funktionalen Programmiersprachen sind *rekursive* Funktionsdefinitionen zulässig, das heißt, die definierte Funktion darf auf der rechten Seite einer solchen Funktionsdefinition vorkommen. Rekursive Funktionsdefinitionen entsprechen - wie in Übung 2, Blatt 5 - δ-Reduktionen.

Hinweis. Nehmen Sie an, dass die Subtraktion von natürlichen Zahlen (in Form von Church-Numeralen) im λ-Kalkül darstellbar ist, d.h. für $n \geq 1$ gilt $\lceil n \rceil - \lceil 1 \rceil \rightarrow^*_{\beta\delta} \lceil n - 1 \rceil$, und dass ebenso die üblichen Vergleichsoperationen möglich sind, d.h. $\lceil n \rceil \leq \lceil 1 \rceil \leftrightarrow^*_{\beta\delta} \mathit{true}$, wenn $n \leq 1$ usw. Siehe dazu Aufgabe 4.

Aufgabe 2.1

Rekursive Definitionen

Wir betrachten die folgende rekursive Funktion:

```
fact n = if n ≤ ⌈1⌉ then ⌈1⌉ else n * (fact (n - ⌈1⌉))
```

Zeigen Sie, dass $\mathit{fact}\,\lceil 3\rceil \to^*_{\beta\delta} \lceil 6\rceil$.

Hinweis

$$\lceil n\rceil - \lceil 1\rceil \to^*_{\beta\delta} \lceil n-1\rceil \text{ für } n \geq 1$$

$$\lceil n\rceil \leq \lceil m\rceil \leftrightarrow^*_{\beta\delta} \begin{cases} \mathit{true} & \text{falls } n \leq m \\ \mathit{false} & \text{sonst} \end{cases}$$

$$\lceil n\rceil == \lceil m\rceil \leftrightarrow^*_{\beta\delta} \begin{cases} \mathit{true} & \text{falls } n = m \\ \mathit{false} & \text{sonst} \end{cases}$$

Aufgabe 2.2

Rekursive Definitionen

Schreiben Sie eine rekursive Funktion *odd*, sodass:

$$odd\ \lceil n \rceil = \begin{cases} true & \text{falls } n \text{ ungerade} \\ false & \text{sonst} \end{cases}$$

Hinweis

$$\lceil n \rceil - \lceil 1 \rceil \rightarrow^*_{\beta\delta} \lceil n - 1 \rceil \text{ für } n \geq 1$$

$$\lceil n \rceil \leq \lceil m \rceil \leftrightarrow^*_{\beta\delta} \begin{cases} true & \text{falls } n \leq m \\ false & \text{sonst} \end{cases}$$

$$\lceil n \rceil == \lceil m \rceil \leftrightarrow^*_{\beta\delta} \begin{cases} true & \text{falls } n = m \\ false & \text{sonst} \end{cases}$$

Aufgabe 2.3

Rekursive Definitionen

Schreiben Sie eine rekursive Funktion *halve*, sodass:

$$(\lceil 2 \rceil \texttt{ * halve } \lceil n \rceil) \texttt{ + (if odd } \lceil n \rceil \texttt{ then } \lceil 1 \rceil \texttt{ else } \lceil 0 \rceil) \to^*_{\beta\delta} \lceil n \rceil$$

Hinweis

$$\lceil n \rceil - \lceil 1 \rceil \to^*_{\beta\delta} \lceil n - 1 \rceil \text{ für } n \geq 1$$

$$\lceil n \rceil \leq \lceil m \rceil \leftrightarrow^*_{\beta\delta} \begin{cases} \mathit{true} & \text{falls } n \leq m \\ \mathit{false} & \text{sonst} \end{cases}$$

$$\lceil n \rceil == \lceil m \rceil \leftrightarrow^*_{\beta\delta} \begin{cases} \mathit{true} & \text{falls } n = m \\ \mathit{false} & \text{sonst} \end{cases}$$

Aufgabe 3

Auswertungsstrategien

In der Vorlesung haben Sie verschiedene Reduktionsstrategien für den ungetypten λ-Kalkül kennengelernt. Diese unterscheiden sich hauptsächlich in den Zeitpunkten, zu denen β-Redexe *kontrahiert* werden, also wann in einem Term die β-Regel angewandt wird.

Applikative Reduktion $\to_a$

- $(\lambda x.t)\, s \to_a t[s/x]$, wenn t und s normal
- $\lambda x.t \to_a \lambda x.t'$, wenn $t \to_a t'$
- $t\, s \to_a t'\, s$, wenn $t \to_a t'$
- $t\, s \to_a t\, s'$, wenn $s \to_a s'$ und t normal

Normale Reduktion $\to_n$

- $(\lambda x.t)\, s \to_n t[s/x]$
- $\lambda x.t \to_n \lambda x.t'$, wenn $t \to_n t'$
- $t\, s \to_n t'\, s$, wenn $t \to_n t'$ und t keine λ-Abstraktion
- $t\, s \to_n t\, s'$, wenn $s \to_n s'$ und t normal und keine λ-Abstraktion

Aufgabe 3.1

Auswertungsstrategien

Welcher Redex im λ-Term

(a) $(\lambda x.\lambda y.\, y\,(\lambda z.\, x))\,(u\,u)\,(\lambda v.\, v\,((\lambda w.\, w)\,(\lambda w.\, w)))$

(b) $(\lambda u.\, u\,(\lambda y.\, z))\,(\lambda x.\, x\,((\lambda v.\, v)\,w))$

muss nicht kontrahiert werden, um die Normalform zu erreichen? Reduzieren Sie den Term durch $\beta\delta$-Reduktion zur Normalform, ohne diesen Redex zu kontrahieren.

Applikative Reduktion $\to_a$

- $(\lambda x.t)\,s \to_a t[s/x]$, wenn t und s normal
- $\lambda x.t \to_a \lambda x.t'$, wenn $t \to_a t'$
- $t\,s \to_a t'\,s$, wenn $t \to_a t'$
- $t\,s \to_a t\,s'$, wenn $s \to_a s'$ und t normal

Normale Reduktion $\to_n$

- $(\lambda x.t)\,s \to_n t[s/x]$
- $\lambda x.t \to_n \lambda x.t'$, wenn $t \to_n t'$
- $t\,s \to_n t'\,s$, wenn $t \to_n t'$ und t keine λ-Abstraktion
- $t\,s \to_n t\,s'$, wenn $s \to_n s'$ und t normal und keine λ-Abstraktion

Aufgabe 3.2

Auswertungsstrategien

Wir schreiben wie aus der Vorlesung bekannt $I = (\lambda x.\, x)$ und $\Omega = (\lambda x.\, x\ x)$.
Reduzieren Sie den Term $(\lambda f.\, f\ I\ (\Omega\ \Omega))(\lambda x\ y.\, x\ x)$ mittels

(a) applikativer Reduktion,

(b) normaler Reduktion.

Unterstreichen Sie in jedem Schritt den zu reduzierenden Redex. Betrachten Sie in dieser Aufgabe δ-Reduktion als explizite Schritte!

Applikative Reduktion $\rightarrow_a$

- $(\lambda x.t)\ s \rightarrow_a t[s/x]$, wenn t und s normal
- $\lambda x.t \rightarrow_a \lambda x.t'$, wenn $t \rightarrow_a t'$
- $t\ s \rightarrow_a t'\ s$, wenn $t \rightarrow_a t'$
- $t\ s \rightarrow_a t\ s'$, wenn $s \rightarrow_a s'$ und t normal

Normale Reduktion $\rightarrow_n$

- $(\lambda x.t)\ s \rightarrow_n t[s/x]$
- $\lambda x.t \rightarrow_n \lambda x.t'$, wenn $t \rightarrow_n t'$
- $t\ s \rightarrow_n t'\ s$, wenn $t \rightarrow_n t'$ und t keine λ-Abstraktion
- $t\ s \rightarrow_n t\ s'$, wenn $s \rightarrow_n s'$ und t normal und keine λ-Abstraktion

Aufgabe 3.3 a)

Auswertungsstrategien

Man erinnere sich an folgende auf Church-Kodierungen definierten Funktionen:

```
-- Allgemein
twice = λf x.
  f (f x)
```

```
-- Church-Booleans
true = λ x y. x
false = λ x y. y
```

```
-- Church-Paare
pair = λ a b select. select a b
fst = λ p. p (λ x y. x)
snd = λ p. p (λ x y. y)
```

Geben Sie die ersten fünf $\beta\delta$-Reduktionsschritte des Terms

```
twice fst (pair (pair true false) true)
```

unter **a) normaler** und b) applikativer Reduktion an. Markieren Sie (durch Unterstreichen) in jedem Schritt den zu reduzierenden Redex.

Normale Reduktion $\rightarrow_n$

- $(\lambda x.t)\, s \rightarrow_n t[s/x]$
- $\lambda x.t \rightarrow_n \lambda x.t'$, wenn $t \rightarrow_n t'$
- $t\, s \rightarrow_n t'\, s$, wenn $t \rightarrow_n t'$ und t keine λ-Abstraktion
- $t\, s \rightarrow_n t\, s'$, wenn $s \rightarrow_n s'$ und t normal und keine λ-Abstraktion

Aufgabe 3.3 b)

Auswertungsstrategien

Man erinnere sich an folgende auf Church-Kodierungen definierten Funktionen:

```
-- Allgemein
twice = λf x.
  f (f x)
```

```
-- Church-Booleans
true = λ x y. x
false = λ x y. y
```

```
-- Church-Paare
pair = λ a b select. select a b
fst = λ p. p (λ x y. x)
snd = λ p. p (λ x y. y)
```

Geben Sie die ersten fünf $\beta\delta$-Reduktionsschritte des Terms

```
twice fst (pair (pair true false) true)
```

unter a) normaler und **b) applikativer** Reduktion an. Markieren Sie (durch Unterstreichen) in jedem Schritt den zu reduzierenden Redex.

Applikative Reduktion $\rightarrow_a$

- $(\lambda x.t)\, s \rightarrow_a t[s/x]$, wenn t und s normal
- $\lambda x.t \rightarrow_a \lambda x.t'$, wenn $t \rightarrow_a t'$
- $t\, s \rightarrow_a t'\, s$, wenn $t \rightarrow_a t'$
- $t\, s \rightarrow_a t\, s'$, wenn $s \rightarrow_a s'$ und t normal