

# **Explizite Substitutionen**

Martin Gasbichler

# Klassischer $\lambda$ -Kalkül

Syntax:  $e ::= x \mid \lambda x.e \mid (e e)$

Wichtigste Regel:  $(\lambda x.a)b \rightarrow_{\beta} a\{b/x\}$

Beobachtungen:

- Fast alles steckt in Substitution
- Substitution ist Operation auf der Meta-Ebene
- Substitution wird auf einmal ausgeführt
- Weit weg von realer Implementierung

## Relevanz der Substitution

Substitution verwirklicht lexikalische Bindung:

...

$$(\lambda v.e)\{f/v\} = (\lambda v.e)$$

$$(\lambda x.e)\{f/v\} = (\lambda x'.e\{x'/x\}\{f/v\}) \text{ wenn } x \in \text{free}(f), x' \text{ frisch}$$

...

Alternativ: Variablenkonvention von Barendregt

## Explizite Substitutionen

Idee (Abadi et. al.): Erweitere Termsprache um Terme mit Substitutionen

Notation:  $e[s]$

Damit  $\beta$ -Reduktion:  $(\lambda x.a)b \rightarrow_{Beta} a[(b/x) \cdot id]$

Substitution  $(b/x) \cdot id$  ersetzt  $x$  durch  $b$ , Rest bleibt unverändert

## Die De-Brujin-Notation

Idee: Ersetze Variable durch Zahl die auf Bindung verweist

$$(\lambda x. (\lambda y. xy)) \rightsquigarrow (\lambda (\lambda \mathbf{21}))$$

Funktionsanwendung: **1** wird ersetzt, der Rumpf dekrementiert:

$$(\lambda a)b \rightarrow_{\beta} a\{b/\mathbf{1}, \mathbf{1}/\mathbf{2}, \mathbf{2}/\mathbf{3}, \dots\}$$

Substitutionen für Abstraktionen:

$$(\lambda c)\{b/\mathbf{1}, \mathbf{1}/\mathbf{2}, \mathbf{2}/\mathbf{3}, \dots\} = (\lambda c\{1/\mathbf{1}, b\{\mathbf{2}/\mathbf{1}, \mathbf{3}/\mathbf{2}, \dots\}/\mathbf{2}, \mathbf{2}/\mathbf{3}, \dots\})$$

## Der $\lambda\sigma$ -Kalkül

Syntax:

Terme:  $a, b ::= \mathbf{1} \mid ba \mid \lambda a \mid a[s]$

Substitutionen:  $s, t ::= id \mid a \cdot s \mid \uparrow \mid s \circ t$

Beobachtung: Keine Metavariablen,  $\mathbf{2} = \mathbf{1}[\uparrow]$

# Reduktionsregeln für Funktionsanwendung und explizite Substitutionen

$$(\lambda a)b \rightarrow_{\beta} a[b \cdot id] \quad (\text{Beta})$$

$$\mathbf{1}[id] \rightarrow_{\sigma} \mathbf{1} \quad (\text{VarId})$$

$$\mathbf{1}[a \cdot s] \rightarrow_{\sigma} a \quad (\text{VarCons})$$

$$(ab)[s] \rightarrow_{\sigma} (a[s])(b[s]) \quad (\text{App})$$

$$(\lambda a)[s] \rightarrow_{\sigma} \lambda(a[\mathbf{1} \cdot (s \circ \uparrow)]) \quad (\text{Abs})$$

$$a[s][t] \rightarrow_{\sigma} a[s \circ t] \quad (\text{Clos})$$

## Reduktionsregeln für $\circ$

$$id \circ s \rightarrow_{\sigma} s \quad (\text{IdL})$$

$$\uparrow \circ id \rightarrow_{\sigma} \uparrow \quad (\text{ShiftId})$$

$$\uparrow \circ (a \cdot s) \rightarrow_{\sigma} s \quad (\text{ShiftCons})$$

$$(a \cdot s) \circ t \rightarrow_{\sigma} a[t] \cdot (s \circ t) \quad (\text{Map})$$

$$(s_1 \circ s_2) \circ s_3 \rightarrow_{\sigma} s_1 \circ (s_2 \circ s_3) \quad (\text{Assoc})$$

## Beispiel

$(\lambda 1)(\lambda(\lambda 2 1))$

$\rightarrow$  (Beta)

$1[(\lambda(\lambda 2 1)) \cdot id]$

$\rightarrow$  (VarId)

$(\lambda(\lambda 2 1))$

## Beispiel II

$(\lambda(\lambda\mathbf{2}))\lambda\mathbf{1}$

$\rightarrow$ (Beta)

$(\lambda\mathbf{2})[\lambda\mathbf{1} \cdot id]$

$\rightarrow$ (Abs)

$\lambda(\mathbf{2}[\mathbf{1} \cdot ((\lambda\mathbf{1} \cdot id) \circ \uparrow)])$

$\rightarrow$ (Def 2)

$\lambda(\mathbf{1}[\uparrow][\mathbf{1} \cdot ((\lambda\mathbf{1} \cdot id) \circ \uparrow)])$

$\rightarrow$ (Clos)

$\lambda(\mathbf{1}[\uparrow \circ (\mathbf{1} \cdot (\lambda\mathbf{1} \cdot id) \circ \uparrow)])$

$\rightarrow$ (ShiftCons)

$\lambda(\mathbf{1}[(\lambda\mathbf{1} \cdot id) \circ \uparrow])$

$\rightarrow$ (Map,...)

$\lambda\mathbf{1}[\lambda\mathbf{1} \cdot \uparrow]$

$\rightarrow$ (VarId)

$\lambda\lambda\mathbf{1}$

## Eigenschaften von $\sigma$

- terminating
- confluent
- Normalformen:

Terme:  $a, b ::= \mathbf{1} \mid \mathbf{1}[\uparrow^n] \mid ba \mid \lambda a$

Substitutionen:  $s, t ::= id \mid \uparrow^n \mid a \cdot s$

## Bewahrung der Terminierung

Mellies, 1995: Es gibt einen einfach getypten  $\lambda$ -Term dessen Auswertung in  $\lambda\sigma$  nicht immer terminiert.

Problem: Neue  $\beta$ -Redexe in nicht mehr anwendbaren Substitutionen

Lösungen:

- Garbage-Collection:  $e[b/\mathbf{1}] \rightarrow e$  if  $\mathbf{1} \notin \text{free}(e)$
- Reduktionsreihenfolge beschränken
- Komposition von Substitutionen verbieten

# $\lambda$ -Kalkül mit Namen

$e ::= x^n \mid (\lambda x.e) \mid (e e) \mid e \downarrow x \mid e[s]$  (Expressions)

$s ::= e/x \mid s \uparrow x$  (Substitutionen)

$((\lambda x.e)f) \rightarrow e[f/x] \downarrow x^0$  ( $\beta$ -Reduktion)

$x^n \downarrow x^m \rightarrow x^{n-1}$  **if**  $n > m$

$(\lambda x.e) \downarrow x^m \rightarrow (\lambda x.e \downarrow x^{m+1})$

$(e_1 e_2) \downarrow x^m \rightarrow (e_1 \downarrow x^m e_2 \downarrow x^m)$

$x^m [e/x^m] \rightarrow e$

$x^m [e/x^n] \rightarrow x^m$  **if**  $n \neq m$

$(\lambda x.e)[s] \rightarrow (\lambda x.e[s \uparrow x])$

$(e_1 e_2)[s] \rightarrow (e_1[s] e_2[s])$

$e/x^m \uparrow x \rightarrow (e \uparrow x)/x^{m+1}$

## Vorteile expliziter Substitutionen

- Näher an abstrakten Maschinen oder realen Implementierungen
- Reduktionsstrategie für Substitutionen
- „safe-for-space“-Closures