Untyped Normalization-by-Evaluation

Klaus Aehlig Felix Joachimski
Mathematisches Institut
LMU München
{aehlig,joachski}@mathematik.uni-muenchen.de

Tallinn, 17.4.4

Klaus Aehlig and Felix Joachimski. Operational aspects of untyped normalization by evaluation.
Accepted by *Mathematical Structures in Computer Science*, 2003.
http://www.mathematik.uni-muenchen.de/~aehlig

Motivation

$[\![\cdot]\!]_\xi$	:	Terms	$\longrightarrow$	Semantics	(evaluation function)
$\downarrow$	:	Semantics	$\longrightarrow$	Normal forms	(inverse)
$\uparrow$	:	Variables	$\longrightarrow$	Semantics	(valuation)

- ▷ **Soundness**: $r \to s \;\Rightarrow\; [\![r]\!]_\xi = [\![s]\!]_\xi$
- ▷ **Reproduction**: r in normal form $\;\Rightarrow\; \downarrow[\![r]\!]_\uparrow = r$

Motivation

$[\![\cdot]\!]_\xi$	:	Terms	$\longrightarrow$	Semantics	(evaluation function)
$\downarrow$	:	Semantics	$\longrightarrow$	Normal forms	(inverse)
$\uparrow$	:	Variables	$\longrightarrow$	Semantics	(valuation)

- **Soundness**: $r \to s \;\Rightarrow\; [\![r]\!]_\xi = [\![s]\!]_\xi$
- **Reproduction**: r in normal form $\Rightarrow\; \downarrow[\![r]\!]_\uparrow = r$

$$r =_\beta s \in \mathrm{NF}_\beta$$

Motivation

$[\![\cdot]\!]_\xi$	:	Terms	$\longrightarrow$	Semantics	(evaluation function)
$\downarrow$	:	Semantics	$\longrightarrow$	Normal forms	(inverse)
$\uparrow$	:	Variables	$\longrightarrow$	Semantics	(valuation)

- **Soundness**: $r \to s \;\Rightarrow\; [\![r]\!]_\xi = [\![s]\!]_\xi$
- **Reproduction**: r in normal form $\Rightarrow\; \downarrow[\![r]\!]_\uparrow = r$

$$r =_\beta s \in \mathrm{NF}_\beta \implies r \to^* s$$

Motivation

$[\![\cdot]\!]_\xi$	:	Terms	$\longrightarrow$	Semantics	(evaluation function)
$\downarrow$	:	Semantics	$\longrightarrow$	Normal forms	(inverse)
$\uparrow$	:	Variables	$\longrightarrow$	Semantics	(valuation)

- ▷ **Soundness**: $r \to s \;\Rightarrow\; [\![r]\!]_\xi = [\![s]\!]_\xi$
- ▷ **Reproduction**: r in normal form $\Rightarrow\; \downarrow[\![r]\!]_\uparrow = r$

$$r =_\beta s \in \mathrm{NF}_\beta \implies r \to^* s \implies \downarrow[\![r]\!]_\uparrow$$

Motivation

$[\![\cdot]\!]_\xi$	:	Terms	$\longrightarrow$	Semantics	(evaluation function)
$\downarrow$	:	Semantics	$\longrightarrow$	Normal forms	(inverse)
$\uparrow$	:	Variables	$\longrightarrow$	Semantics	(valuation)

- ▷ **Soundness**: $r \to s \;\Rightarrow\; [\![r]\!]_\xi = [\![s]\!]_\xi$
- ▷ **Reproduction**: r in normal form $\Rightarrow\; \downarrow[\![r]\!]_\uparrow = r$

$$r =_\beta s \in \mathrm{NF}_\beta \implies r \to^* s \implies \downarrow[\![r]\!]_\uparrow = \downarrow[\![s]\!]_\uparrow$$

Motivation

$[\![\cdot]\!]_\xi$	:	Terms	$\longrightarrow$	Semantics	(evaluation function)
$\downarrow$	:	Semantics	$\longrightarrow$	Normal forms	(inverse)
$\uparrow$	:	Variables	$\longrightarrow$	Semantics	(valuation)

- ▷ **Soundness**: $r \to s \;\Rightarrow\; [\![r]\!]_\xi = [\![s]\!]_\xi$
- ▷ **Reproduction**: r in normal form $\Rightarrow\; \downarrow[\![r]\!]_\uparrow = r$

$$r =_\beta s \in \mathrm{NF}_\beta \implies r \to^* s \implies \downarrow[\![r]\!]_\uparrow = \downarrow[\![s]\!]_\uparrow = s$$

Example — The typed λ-calculus

- **Terms:** Λ with simple types

Example — The typed λ-calculus

- **Terms:** Λ with simple types
- **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$

Example — The typed λ-calculus

- ▷ **Terms:** Λ with simple types
- ▷ **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- ▷ **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$

Example — The typed λ-calculus

- ▷ **Terms:** Λ with simple types
- ▷ **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- ▷ **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- ▷ $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

Example — The typed λ-calculus

- **Terms:** Λ with simple types
- **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- **Properties:** $\downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R$

Example — The typed λ-calculus

- **Terms:** Λ with simple types
- **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- **Properties:** $\downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R = \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R) \uparrow_\rho x)$

Example — The typed λ-calculus

- ▷ **Terms:** Λ with simple types
- ▷ **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- ▷ **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- ▷ $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- ▷ **Properties:** $\downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R = \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R) \uparrow_\rho x) = \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]).$

Example — The typed λ-calculus

- ▷ **Terms:** Λ with simple types
- ▷ **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- ▷ **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- ▷ $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- ▷ **Properties:** $\downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R = \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R) \uparrow_\rho x) = \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x])$.
  $(\uparrow_{\rho\to\sigma} r) S$

Example — The typed λ-calculus

- **Terms:** Λ with simple types
- **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- **Properties:** $\begin{array}{rcl} \downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R & = & \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R)\uparrow_\rho x) \quad = \quad \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]). \\ (\uparrow_{\rho\to\sigma} r) S & = & (\lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X)) S \end{array}$

Example — The typed λ-calculus

- **Terms:** Λ with simple types
- **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- **Properties:**

$$\begin{array}{rcccl} \downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R & = & \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R) \uparrow_\rho x) & = & \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]). \\ (\uparrow_{\rho\to\sigma} r) S & = & (\lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X)) S & = & \uparrow_\sigma (r \downarrow_\rho S). \end{array}$$

Example — The typed λ-calculus

- ▷ **Terms:** Λ with simple types
- ▷ **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- ▷ **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- ▷ $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- ▷ **Properties:** $\downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R = \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R) \uparrow_\rho x) = \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]).$
  $(\uparrow_{\rho\to\sigma} r) S = (\lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X)) S = \uparrow_\sigma (r \downarrow_\rho S).$
  $\downarrow_\iota \uparrow_\iota r$

Example — The typed λ-calculus

- ▷ **Terms:** Λ with simple types
- ▷ **Semantics:** $[\![\iota]\!] := \mathrm{NF}_\iota, \quad [\![\rho \to \sigma]\!] := [\![\rho]\!] \to [\![\sigma]\!]$
- ▷ **Evaluation:** $[\![x]\!]_\xi := \xi x, \quad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \quad [\![\lambda x r]\!]_\xi := \lambda\!\!\lambda X [\![r]\!]_{\xi, x:=X}$
- ▷ $\uparrow_\rho : \{x\vec{r} \mid \vec{r} \text{ normal}\} \to [\![\rho]\!]$ and $\downarrow_\rho : [\![\rho]\!] \to \mathrm{NF}_\rho$

$$\begin{array}{llll} \uparrow_\iota r := r, & \uparrow_{\rho\to\sigma} r := \lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X), & \\ \downarrow_\iota r := r, & \downarrow_{\rho\to\sigma} R := \lambda x \downarrow_\sigma (R \uparrow_\rho x), & x \text{ new} \end{array}$$

- ▷ **Properties:**

$$\begin{array}{rcccl} \downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R & = & \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R) \uparrow_\rho x) & = & \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]). \\ (\uparrow_{\rho\to\sigma} r) S & = & (\lambda\!\!\lambda X \uparrow_\sigma (r \downarrow_\rho X)) S & = & \uparrow_\sigma (r \downarrow_\rho S). \\ \downarrow_\iota \uparrow_\iota r & = & r. & & \end{array}$$

From properties to axioms

▷ **Properties.**

$$\begin{array}{rcccl} \downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R & = & \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R)\uparrow_\rho x) & = & \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]). \\ (\uparrow_{\rho\to\sigma} r) S & = & (\lambda\!\!\lambda X \uparrow_\sigma (r\downarrow_\rho X)) S & = & \uparrow_\sigma (r\downarrow_\rho S). \\ \downarrow_\iota \uparrow_\iota r & = & r. & & \end{array}$$

From properties to axioms

▷ **Properties.**

$$\begin{array}{rcccl} \downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R & = & \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R)\uparrow_\rho x) & = & \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]). \\ (\uparrow_{\rho\to\sigma} r) S & = & (\lambda\!\!\lambda X \uparrow_\sigma (r\downarrow_\rho X)) S & = & \uparrow_\sigma (r\downarrow_\rho S). \\ \downarrow_\iota \uparrow_\iota r & = & r. & & \end{array}$$

Discard types — we only need to know whether function or not

From properties to axioms

▷ **Properties.**

$$\begin{array}{rcccl} \downarrow_{\rho\to\sigma} \lambda\!\!\lambda X R & = & \lambda x \downarrow_\sigma ((\lambda\!\!\lambda X R)\uparrow_\rho x) & = & \lambda x \downarrow_\sigma (R[X := \uparrow_\rho x]). \\ (\uparrow_{\rho\to\sigma} r) S & = & (\lambda\!\!\lambda X \uparrow_\sigma (r\downarrow_\rho X)) S & = & \uparrow_\sigma (r\downarrow_\rho S). \\ \downarrow_\iota \uparrow_\iota r & = & r. & & \end{array}$$

Discard types — we only need to know whether function or not

▷ **Axioms.**

$$\begin{array}{rcl} \downarrow \lambda\!\!\lambda X R & = & \lambda x \downarrow (R[X := \uparrow x]). \\ (\uparrow r) S & = & \uparrow (r\downarrow S). \\ \downarrow\uparrow r & = & r. \end{array}$$

(Just another) 2-level-calculus

▷ **Terms.** $\Lambda \quad \ni \quad r, s, t \quad ::= \quad x \mid rs \mid \lambda x r,$

(Just another) 2-level-calculus

▷ **Terms.**
$$\begin{array}{llllllll} \Lambda & \ni & r,s,t & ::= & x & \mid & rs & \mid & \lambda x r, \\ \Lambda_1 & \ni & r,s,t & ::= & x & \mid & rs & \mid & \lambda x r \end{array}$$

(Just another) 2-level-calculus

▷ **Terms.**
$$\begin{array}{llllllll} \Lambda & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r, & \\ \Lambda_1 & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r & \mid \downarrow R \end{array}$$

(Just another) 2-level-calculus

▷ **Terms.**

$$\begin{array}{llllllll} \Lambda & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r, & \\ \Lambda_1 & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r & \mid \downarrow R \\ \Lambda_2 & \ni & R,S,T & ::= & X & \mid RS & \mid \lambda X R & \end{array}$$

(Just another) 2-level-calculus

▷ **Terms.**

$$\begin{array}{llllllll} \Lambda & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r, & \\ \Lambda_1 & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r & \mid \downarrow R \\ \Lambda_2 & \ni & R,S,T & ::= & X & \mid RS & \mid \lambda X R & \mid \uparrow r \end{array}$$

(Just another) 2-level-calculus

▷ **Terms.**

$$\begin{array}{lcllllll} \Lambda & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r, & \\ \Lambda_1 & \ni & r,s,t & ::= & x & \mid rs & \mid \lambda x r & \mid \downarrow R \\ \Lambda_2 & \ni & R,S,T & ::= & X & \mid RS & \mid \lambda X R & \mid \uparrow r \end{array}$$

▷ **Reductions.**

$$\begin{array}{rcll} (\lambda X R)S & \mapsto_\beta & R[X := S] & \\ (\uparrow r)S & \mapsto_a & \uparrow(r \downarrow S) & \\ \downarrow \lambda X R & \mapsto_d & \lambda x \downarrow (R[X := \uparrow x]), & x \text{ new} \\ \downarrow \uparrow r & \mapsto_d & r & \end{array}$$

(Just another) 2-level-calculus

▷ **Terms.**

$$\begin{array}{lllllllll} \Lambda & \ni & r,s,t & ::= & x & \mid & rs & \mid & \lambda x r, \\ \Lambda_1 & \ni & r,s,t & ::= & x & \mid & rs & \mid & \lambda x r \quad \mid \quad \downarrow R \\ \Lambda_2 & \ni & R,S,T & ::= & X & \mid & RS & \mid & \lambda X R \quad \mid \quad \uparrow r \end{array}$$

▷ **Reductions.**

$$\begin{array}{rcll} (\lambda X R)S & \mapsto_\beta & R[X := S] & \\ (\uparrow r)S & \mapsto_a & \uparrow(r \downarrow S) & \\ \downarrow \lambda X R & \mapsto_d & \lambda x \downarrow (R[X := \uparrow x]), & x \text{ new} \\ \downarrow \uparrow r & \mapsto_d & r & \end{array}$$

▷ This yields an orthogonal higher order rewrite system ⇒ confluence.

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi, x:=X}.$

Evaluation

- ▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi, x:=X}.$
- ▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$[\![(\lambda x r)s]\!]_\xi \quad = \quad (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$[\![(\lambda x r)s]\!]_\xi \quad = \quad (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi]$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{aligned} [\![(\lambda x r)s]\!]_\xi &= (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ &= [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} \end{aligned}$$

Evaluation

- **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$
- **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{aligned} [\![(\lambda x r)s]\!]_\xi &= (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ &= [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{aligned}$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$\downarrow[\![x\vec{r}]\!]_\uparrow$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow)$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow)$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow \to_s^* x\vec{r}.$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow \to_s^* x\vec{r}.$

$\downarrow[\![\lambda x r]\!]_\uparrow$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow \to_s^* x\vec{r}.$$

$$\downarrow[\![\lambda x r]\!]_\uparrow = \downarrow\lambda X.[\![r]\!]_{\uparrow,x:=X}$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow \to_s^* x\vec{r}.$$

$$\downarrow[\![\lambda x r]\!]_\uparrow = \downarrow\lambda X.[\![r]\!]_{\uparrow,x:=X} \to_d \lambda x\downarrow[\![r]\!]_{\uparrow,x:=X}[X := \uparrow x]$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow \to_s^* x\vec{r}.$$

$$\downarrow[\![\lambda x r]\!]_\uparrow = \downarrow\lambda X.[\![r]\!]_{\uparrow,x:=X} \to_d \lambda x\downarrow[\![r]\!]_{\uparrow,x:=X}[X := \uparrow x] = \lambda x\downarrow[\![r]\!]_\uparrow$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow \to_s^* x\vec{r}.$$

$$\downarrow[\![\lambda x r]\!]_\uparrow = \downarrow\lambda X.[\![r]\!]_{\uparrow,x:=X} \to_d \lambda x\downarrow[\![r]\!]_{\uparrow,x:=X}[X := \uparrow x] = \lambda x\downarrow[\![r]\!]_\uparrow \to_s^* \lambda x r.$$

Evaluation

▷ **Definition.** $[\![x]\!]_\xi := \xi x, \qquad [\![rs]\!]_\xi := [\![r]\!]_\xi [\![s]\!]_\xi, \qquad [\![\lambda x r]\!]_\xi := \lambda X.[\![r]\!]_{\xi,x:=X}.$

▷ **Simulation.** $r \to_\beta s \implies [\![r]\!]_\xi \to_\beta [\![s]\!]_\xi.$

$$\begin{array}{rcl} [\![(\lambda x r)s]\!]_\xi & = & (\lambda X [\![r]\!]_{\xi,x:=X})[\![s]\!]_\xi \to_\beta [\![r]\!]_{\xi,x:=X}[X := [\![s]\!]_\xi] \\ & = & [\![r]\!]_{\xi,x:=[\![s]\!]_\xi} = [\![r[x := s]]\!]_\xi \end{array}$$

▷ **Reproduction.** $\downarrow[\![r]\!]_\uparrow = r$ for normal forms r.

$$\downarrow[\![x\vec{r}]\!]_\uparrow = \downarrow((\uparrow x)[\![\vec{r}]\!]_\uparrow) \to_a^* \downarrow\uparrow(x\downarrow[\![\vec{r}]\!]_\uparrow) \to_d x\downarrow[\![\vec{r}]\!]_\uparrow \to_s^* x\vec{r}.$$

$$\downarrow[\![\lambda x r]\!]_\uparrow = \downarrow\lambda X.[\![r]\!]_{\uparrow,x:=X} \to_d \lambda x\downarrow[\![r]\!]_{\uparrow,x:=X}[X := \uparrow x] = \lambda x\downarrow[\![r]\!]_\uparrow \to_s^* \lambda x r.$$

▷ **Corollary.** $r =_\beta s \in \mathrm{NF}_\beta \implies \downarrow[\![r]\!]_\uparrow \to^* s.$

Implementation: Object calculus

Implementation: Object calculus

Choose de Bruijn terms for simplicity.

```
data Term = Var Integer | App Term Term | Abs Term   deriving Show
```

Implementation: Object calculus

Choose de Bruijn terms for simplicity.

```
data Term = Var Integer | App Term Term | Abs Term   deriving Show
```

An interface to make lifting easy:

```
type TERM = Integer -> Term

inspect :: TERM -> Term
inspect f = f 0

freevar  k = \ n -> Var (n + k)
boundvar k = \ n -> Var (n - k - 1)
apply r s  = \ n -> App (r n) (s n)
bind f     = \ n -> Abs (f (boundvar n) (n + 1))
```

Implementation — Metalevel calculus

Terms. $\uparrow r \quad | \quad \lambda X R \quad | \quad RS \quad | \quad X$

Implementation — Metalevel calculus

Terms. $\uparrow r \quad | \quad \lambda X R \quad | \quad RS \quad | \quad X$

```
data LAM = Up TERM | ABS (LAM -> LAM)
```

Implementation — Metalevel calculus

Terms. $\uparrow r \quad | \quad \lambda X R \quad | \quad RS \quad | \quad X$

```
data LAM = Up TERM | ABS (LAM -> LAM)
```

Reductions for $\downarrow$:

$$\begin{array}{rcl} \downarrow\uparrow r & \mapsto_d & r \\ \downarrow\lambda X R & \mapsto_d & \lambda x \downarrow (R[X := \uparrow x]) \end{array}$$

Implementation — Metalevel calculus

Terms. $\uparrow r \quad | \quad \lambda X R \quad | \quad RS \quad | \quad X$

```
data LAM = Up TERM | ABS (LAM -> LAM)
```

Reductions for $\downarrow$:

$$\begin{array}{rcl} \downarrow\uparrow r & \mapsto_d & r \\ \downarrow\lambda X R & \mapsto_d & \lambda x \downarrow (R[X := \uparrow x]) \end{array}$$

```
down :: LAM -> TERM                        -- the d-rules
down (ABS f)  = bind (\ x -> (down (f (Up x))))
down (Up r)   = r
```

Implementation — Metalevel calculus

Terms. $\uparrow r \quad | \quad \lambda X R \quad | \quad RS \quad | \quad X$

```
data LAM = Up TERM | ABS (LAM -> LAM)
```

Reductions for $\downarrow$:

$$\begin{array}{rcl} \downarrow\uparrow r & \mapsto_d & r \\ \downarrow\lambda X R & \mapsto_d & \lambda x \downarrow(R[X := \uparrow x]) \end{array}$$

```
down :: LAM -> TERM                        -- the d-rules
down (ABS f)  = bind (\ x -> (down (f (Up x))))
down (Up r)   = r
```

Reductions for application:

$$\begin{array}{rcl} (\lambda X R)S & \mapsto_\beta & R[X := S] \\ (\uparrow r)S & \mapsto_a & \uparrow(r \downarrow S) \end{array}$$

Implementation — Metalevel calculus

Terms. $\uparrow r \quad | \quad \lambda X R \quad | \quad RS \quad | \quad X$

```
data LAM = Up TERM | ABS (LAM -> LAM)
```

Reductions for $\downarrow$:

$$\begin{array}{rcl} \downarrow\uparrow r & \mapsto_d & r \\ \downarrow\lambda X R & \mapsto_d & \lambda x \downarrow (R[X := \uparrow x]) \end{array}$$

```
down :: LAM -> TERM                        -- the d-rules
down (ABS f)  = bind (\ x -> (down (f (Up x))))
down (Up r)   = r
```

Reductions for application:

$$\begin{array}{rcl} (\lambda X R)S & \mapsto_\beta & R[X := S] \\ (\uparrow r)S & \mapsto_a & \uparrow (r \downarrow S) \end{array}$$

```
app :: LAM -> LAM -> LAM
app (ABS f) r = f r                        -- the beta-rule
app (Up r)  s = Up (apply r (down s))      -- the a-rule
```

Implementation — Evaluation

```
type Valuation = Integer -> LAM
comma :: Valuation -> LAM -> Valuation
comma xi r n = if n == 0 then r else xi (n - 1)          ξ, x := R

eval :: Term -> Valuation -> LAM
eval (Var n)   xi = xi n                                 ξx
eval (Abs r)   xi = ABS (\ a -> eval r (xi `comma` a))   λX.[[r]]_{ξ,x:=X}
eval (App r s) xi = (eval r xi) `app` (eval s xi)        [[r]]_ξ [[s]]_ξ

nbe r = inspect (down (eval r (Up . freevar)))           ↓[[r]]_↑
      =         (down (eval r (Up . freevar))) 0
```

Implementation — Examples

```
k = Abs (Abs (Var 1))                                    k = λλ1
s = Abs (Abs (Abs (Var 2 `App` (Var 0) `App` (Var 1 `App` (Var 0)))))
i = s `App` k `App` k                                    s = λλλ.20(10)
omega  = Abs (Var 0 `App` (Var 0))                       ω = λ.00
oomega = omega `App` omega                               Ω = ωω
```

Implementation — Examples

```
k = Abs (Abs (Var 1))                                    k = λλ1
s = Abs (Abs (Abs (Var 2 `App` (Var 0) `App` (Var 1 `App` (Var 0)))))
i = s `App` k `App` k                                    s = λλλ.20(10)
omega  = Abs (Var 0 `App` (Var 0))                       ω = λ.00
oomega = omega `App` omega                               Ω = ωω
```

```
Main> nbe k
   Abs (Abs (Var 1))
```

Implementation — Examples

```
k = Abs (Abs (Var 1))                                    k = λλ1
s = Abs (Abs (Abs (Var 2 `App` (Var 0) `App` (Var 1 `App` (Var 0)))))
i = s `App` k `App` k                                    s = λλλ.20(10)
omega  = Abs (Var 0 `App` (Var 0))                       ω = λ.00
oomega = omega `App` omega                               Ω = ωω
```

```
Main> nbe k
   Abs (Abs (Var 1))


Main> nbe i
   Abs (Var 0)
```

Implementation — Examples

```
k = Abs (Abs (Var 1))
s = Abs (Abs (Abs (Var 2 `App` (Var 0) `App` (Var 1 `App` (Var 0)))))
i = s `App` k `App` k
omega  = Abs (Var 0 `App` (Var 0))
oomega = omega `App` omega
```

$k = \lambda\lambda 1$

$s = \lambda\lambda\lambda.20(10)$

$\omega = \lambda.00$

$\Omega = \omega\omega$

```
Main> nbe k
   Abs (Abs (Var 1))

Main> nbe i
   Abs (Var 0)

Main> nbe (k `App` i `App` oomega)
   Abs (Var 0)
```

Standardization

▷ **Standard reduction.** Head redexes are executed first or never.

Standardization

- ▷ **Standard reduction.** Head redexes are executed first or never.
- ▷ **Theorem.** Reduction sequences can be standardized.

Standardization

- ▷ **Standard reduction.** Head redexes are executed first or never.
- ▷ **Theorem.** Reduction sequences can be standardized.
- ▷ **Corollary.** Normalization (= reduction to normal form)
  - need not reduce under meta-abstractions,
  - may follow a call-by-name strategy.

Standardization

- ▷ **Standard reduction.** Head redexes are executed first or never.
- ▷ **Theorem.** Reduction sequences can be standardized.
- ▷ **Corollary.** Normalization (= reduction to normal form)
  - need not reduce under meta-abstractions,
  - may follow a call-by-name strategy.
- ▷ **Corollary.** We can compute (the defined parts of) Böhm trees — normalization for diverging terms.

Böhm trees — Examples

```
pair = Abs (Abs (Abs (Var 0 `App` Var 2 `App` Var 1)))
swap = Abs (Abs (Abs (Var 0 `App` Var 1 `App` Var 2)))
```

Böhm trees — Examples

```
pair = Abs (Abs (Abs (Var 0 `App` Var 2 `App` Var 1)))
swap = Abs (Abs (Abs (Var 0 `App` Var 1 `App` Var 2)))
```

```
Main> nbe (pair `App` i `App` oomega)
Abs (App (App (Var 0) (Abs (Var 0))) {Interrupted!}
```

Böhm trees — Examples

```
pair = Abs (Abs (Abs (Var 0 `App` Var 2 `App` Var 1)))
swap = Abs (Abs (Abs (Var 0 `App` Var 1 `App` Var 2)))


Main> nbe (pair `App` i `App` oomega)
Abs (App (App (Var 0) (Abs (Var 0))) {Interrupted!}

Main> cut (nbe (pair `App` i `App` oomega)) ["01"]
Abs (App (App (Var 0) (Abs (Var 0))) (Var (-1)))
```

Böhm trees — Examples

```
pair = Abs (Abs (Abs (Var 0 `App` Var 2 `App` Var 1)))
swap = Abs (Abs (Abs (Var 0 `App` Var 1 `App` Var 2)))
```

```
Main> nbe (pair `App` i `App` oomega)
Abs (App (App (Var 0) (Abs (Var 0))) {Interrupted!}
```

```
Main> cut (nbe (pair `App` i `App` oomega)) ["01"]
Abs (App (App (Var 0) (Abs (Var 0))) (Var (-1)))
```

```
Main> nbe (pair `App` i `App` oomega `App` swap)
Abs (App (App (Var 0) {Interrupted!}
```

Böhm trees — Examples

```
pair = Abs (Abs (Abs (Var 0 `App` Var 2 `App` Var 1)))
swap = Abs (Abs (Abs (Var 0 `App` Var 1 `App` Var 2)))


Main> nbe (pair `App` i `App` oomega)
Abs (App (App (Var 0) (Abs (Var 0))) {Interrupted!}

Main> cut (nbe (pair `App` i `App` oomega)) ["01"]
Abs (App (App (Var 0) (Abs (Var 0))) (Var (-1)))

Main> nbe (pair `App` i `App` oomega `App` swap)
Abs (App (App (Var 0) {Interrupted!}

Main> cut (nbe (pair `App` i `App` oomega `App` swap)) ["001"]
Abs (App (App (Var 0) (Var (-1))) (Abs (Var 0)))
```

Types — Computing η-long normal forms

Types — Computing η-long normal forms

▷ **Head expansion.** Instead of $\downarrow\uparrow r \mapsto_d r$ define

$$\downarrow\uparrow r^{\rho} \mapsto_d \eta_\rho r \qquad \text{with} \quad \eta_\iota r := r \quad \text{and} \quad \eta_{\rho\to\sigma} r := \lambda x.\eta_\sigma(r\eta_\rho x).$$

Types — Computing η-long normal forms

▷ **Head expansion.** Instead of $\downarrow\uparrow r \mapsto_d r$ define

$$\downarrow\uparrow r^{\rho} \mapsto_d \eta_\rho r \qquad \text{with} \quad \eta_\iota r := r \quad \text{and} \quad \eta_{\rho\to\sigma} r := \lambda x.\eta_\sigma(r\eta_\rho x).$$

▷ **Theorem.** On typed terms $\to$ is SN and computes η-long normal forms.

▷ **Corollary.** Typed NBE (as defined in the beginning) is a model.

Conclusions

- Normalization by Evaluation needs no types —
  the information of whether a function or not suffices.
- Non-termination: (partial) normal forms are found whenever they exist.
- Use Haskell for normalization (= doing the substitutions).
- Type-directed NBE as an example.

Technical details

Theorem. $r =_\beta \lambda\vec{x}(x\vec{r})$ & $s \to^* \downarrow[\![r]\!]$
$\implies \exists\vec{t},\vec{s}.s \twoheadrightarrow \lambda\vec{x}(x\vec{s})$ & $\vec{r} =_\beta \vec{t}$ & $\vec{s} \to^* \downarrow[\![\vec{t}]\!]$.

Proof.

$r \to^*_\beta \lambda\vec{x}(x\vec{t}) \;{}^*_\beta\!\leftarrow\; \lambda\vec{x}(x\vec{r})$	by the Church-Rosser prop. of $\to_\beta$,
$s \to^* \downarrow[\![r]\!] \to^* \lambda\vec{x}(x\downarrow[\![\vec{t}]\!])$	
$s \rightsquigarrow \lambda\vec{x}(x\downarrow[\![\vec{t}]\!])$	by standardization,
$s \twoheadrightarrow \lambda\vec{x}(x\vec{s}) \to^* \lambda\vec{x}(x\downarrow[\![\vec{t}]\!])$	property of $\twoheadrightarrow$.

Theorem. $\downarrow[\![t]\!] \to^* \lambda\vec{x}(x\vec{R}) \implies t \to^* \lambda\vec{x}(x\vec{t})$.