Contents

Home Page

Title Page

Contents

◀◀ ▶▶

◀ ▶

Page 2 of 43

Go Back

Full Screen

Close

Quit

1. λ-calculus with end-of-scope

Observations

- λx is like opening an ‘x-bracket’
- no corresponding closing bracket!

Proposal

- Adjoin closing ‘x-brackets’ $\reflectbox{$\lambda$}x$
  (adbmal, unbind, end-of-scope)
- $\reflectbox{$\lambda$}x.M$ closes matching λx: x is ‘free’ in $\lambda x.\reflectbox{$\lambda$}x.x$
- Can be nested $\underbrace{\lambda x.\overbrace{\lambda x.\reflectbox{λ}x}.\reflectbox{λ}x}.x$ (x free again)
- Proper nesting: $\lambda x.\lambda y.\reflectbox{λ}x.\reflectbox{λ}y.M$ not allowed
  (better: $\reflectbox{$\lambda$}x$ implictly closes λy)

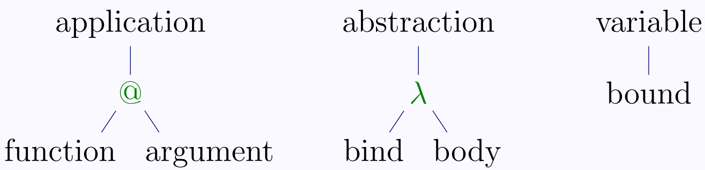
2. α-equivalence via Bourbaki

Represent bound variables as pointers to binder (λ)

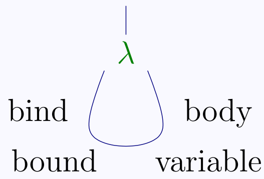
What makes this a good representation?

Thm 1 *λ-terms α-equivalent iff same Bourbaki-graph.*

2.1. Bourbaki λ-graphs

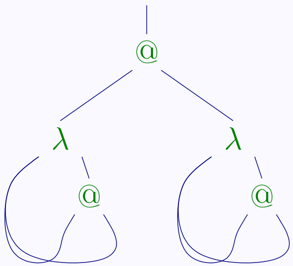


$\lambda x.x =$



$= \lambda y.y$

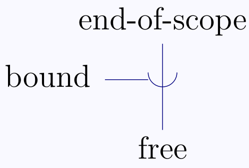
$(\lambda x.xx)(\lambda x.xx) =$



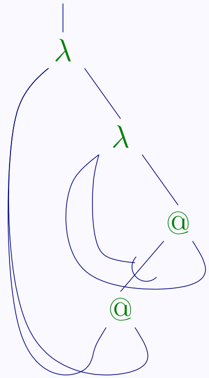
$= \Omega$

2.2. Bourbaki end-of-scope graphs

Represent end-of-scopes as pointers to binder



$\lambda xy.(\not\lambda y.xx)y$



3. β-reduction

Observation: $\reflectbox{$\lambda$}$'s in between $@$ and λ

- $(\lambda x.\reflectbox{$\lambda$}x.M)N$ should reduce to M
- $(\reflectbox{$\lambda$}y.\lambda x.x)y$ should not reduce to $\reflectbox{$\lambda$}y.y$ (but y)
- $(\reflectbox{$\lambda$}y.\lambda x.z)y$ should not reduce to z (but $\reflectbox{$\lambda$}y.z$)

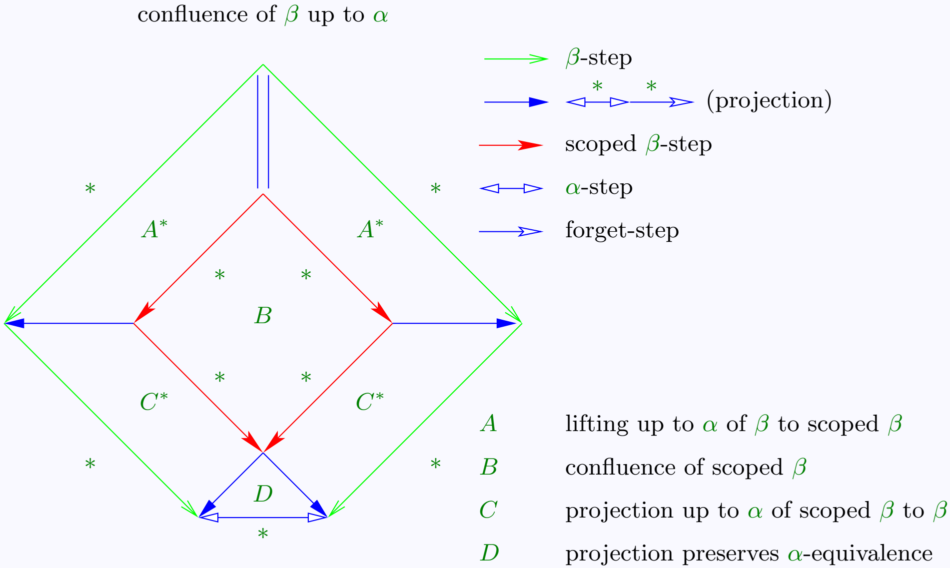
Where should end-of-scopes go?

- search for matching x
- in case of x: remove end-of-scopes, put argument
- in case of $\reflectbox{$\lambda$}x$: put end-of-scopes, remove argument

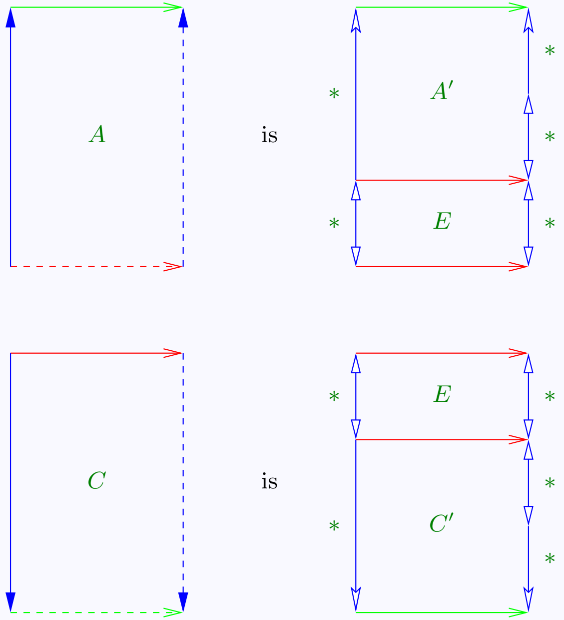
$(\reflectbox{$\lambda$}X.\lambda x.M)N \to M[X, x{:=}N, \Box]$:

- X remembers the end-of-scopes
- third argument: stack used for matching

3.1. Proof of confluence: outline



3.2. Proof of confluence: lifting and projection



A' lifting β to scoped β

E commutation of α and scoped β

C' projecting scoped β to β

4. Formalization in Coq

Axiom 1 *Assume a parameter set $\mathcal{V}$ of infinitely many variable names for which equality is decidable.*

- $x = y \vee x \neq y$ *for all* $x, y : \mathcal{V}$
- $\exists x : \mathcal{V}.\, x \notin X$ *for all* $X : \mathit{list}(\mathcal{V})$

Def 2 *The set Λ of $\lambdabar$-terms is defined by:*

$$\Lambda ::= \mathcal{V} \mid \lambdabar x.\Lambda \mid \lambda x.\Lambda \mid \Lambda\Lambda$$

4.1. Substitution with end-of-scope

Def 3 *Substitution* $M[X, x{:=}N, Y]$ *is defined by:*

$$
\begin{aligned}
y[X, x{:=}N, Y] &= y, \text{ if } y \in Y \\
y[X, x{:=}N, Y] &= \reflectbox{λ} Y.N, \text{ if } y \notin Y, x = y \\
y[X, x{:=}N, Y] &= \reflectbox{λ} Y.\reflectbox{λ} X.y, \text{ if } y \notin Y, x \neq y \\
(\lambda y.M)[X, x{:=}N, Y] &= \lambda y.M[X, x{:=}N, yY] \\
(\reflectbox{λ} y.M)[X, x{:=}N, \Box] &= \reflectbox{λ} X.M, \text{ if } x = y \\
(\reflectbox{λ} y.M)[X, x{:=}N, \Box] &= \reflectbox{λ} X.\reflectbox{λ} y.M, \text{ if } x \neq y \\
(\reflectbox{λ} y.M)[X, x{:=}N, zY] &= \reflectbox{λ} y.M[X, x{:=}N, Y], \text{ if } y = z \\
(\reflectbox{λ} y.M)[X, x{:=}N, zY] &= (\reflectbox{λ} y.M)[X, x{:=}N, Y], \text{ if } y \neq z \\
(M_1 M_2)[X, x{:=}N, Y] &= M_1[X, x{:=}N, Y] M_2[X, x{:=}N, Y]
\end{aligned}
$$

Home Page

Title Page

Contents

Page 12 of 43

Go Back

Full Screen

Close

Quit

4.2. β-reduction with end-of-scope

Def 4 *The relation $\rightarrow_\beta$ is defined as the compatible closure of the β-rule.*

$$\frac{}{(\reflectbox{λ}X.\lambda x.M)N \rightarrow_\beta M[X, x{:=}N, \Box]}$$

$$\frac{M \rightarrow_\beta N}{\lambda x.M \rightarrow_\beta \lambda x.N} \qquad \frac{M \rightarrow_\beta N}{\reflectbox{λ}x.M \rightarrow_\beta \reflectbox{λ}x.N}$$

$$\frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N} \qquad \frac{N \rightarrow_\beta N'}{MN \rightarrow_\beta MN'}$$

4.3. Confluence without α

Thm 5 $\rightarrow_\beta$ *is confluent on* Λ.

Proof strategy (Tait, Martin-Löf):

- define (inductively) multi-steps $\multimap$;
- show that multi-steps have the diamond property (substitution lemma(ta));
- then $\multimap$ is confluent;
- show $\rightarrow_\beta \subseteq \multimap \subseteq \rightarrow_\beta^*$;
- conclude $\rightarrow_\beta^* \subseteq \multimap^* \subseteq \rightarrow_\beta^*$.

4.4. Multi-step

Def 6 *Multi-steps* $\multimap\!\!\rightarrow$ *are defined by:*

$$\frac{M_1 \multimap\!\!\rightarrow N_1 \quad M_2 \multimap\!\!\rightarrow N_2}{(\lambda\!\!\!\lambda X.\lambda x.M_1)M_2 \multimap\!\!\rightarrow N_1[X, x{:=}N_2, \Box]}$$

$$\frac{}{x \multimap\!\!\rightarrow x} \qquad \frac{M_1 \multimap\!\!\rightarrow N_1 \quad M_2 \multimap\!\!\rightarrow N_2}{M_1 M_2 \multimap\!\!\rightarrow N_1 N_2}$$

$$\frac{M \multimap\!\!\rightarrow N}{\lambda x.M \multimap\!\!\rightarrow \lambda x.N} \qquad \frac{M \multimap\!\!\rightarrow N}{\lambda\!\!\!\lambda x.M \multimap\!\!\rightarrow \lambda\!\!\!\lambda x.N}$$

Home Page
Title Page
Contents
◀◀ ▶▶
◀ ▶
Page 15 of 43
Go Back
Full Screen
Close
Quit

Def 7 *A term* M *is* *scope-balanced* *if* $\langle \square \rangle M$, *where* $\langle X \rangle M$ *is defined by:*

$$\frac{}{\langle X \rangle x} \qquad \frac{\langle Xx \rangle M}{\langle X \rangle \lambda x.M} \qquad \frac{\langle X \rangle M}{\langle Xx \rangle \text{ʎ} x.M} \qquad \frac{\langle X \rangle M \quad \langle X \rangle N}{\langle X \rangle MN}$$

Balancedness is defined as scope-balancedness restricting the first clause to

$$\frac{}{\langle Xx \rangle x}$$

Here $\square$ *is the empty stack and* Xx *is the result of pushing* x *on the stack* X.

4.5. Substitution Lemma

Lem 1 *Multi-step substitution lemma:*
if

$$\langle ZxY\rangle M_1 \qquad \langle ZX\rangle M_2$$

and

$$M_1 \multimap\!\!\!\rightarrow N_1 \qquad M_2 \multimap\!\!\!\rightarrow N_2$$

then:

$$M_1[X, x{:=}M_2, Y] \multimap\!\!\!\rightarrow N_1[X, x{:=}N_2, Y]$$

Compute the critical pair from the term P

$$(\lambda y.(\lambda x.M)N)L$$

Inner redex first:

$$\begin{aligned} P &\to_\beta (\lambda y.M[\Box, x{:=}N, \Box])L \\ &\to_\beta M[\Box, x{:=}N, \Box][\Box, y{:=}L, \Box] \end{aligned}$$

Outer redex first:

$$\begin{aligned} P &\to_\beta ((\lambda x.M)N)[\Box, y{:=}L, \Box] \\ &= (\lambda x.M)[\Box, y{:=}L, \Box]N[\Box, y{:=}L, \Box] \\ &= (\lambda x.M[\Box, y{:=}L, x])N[\Box, y{:=}L, \Box] \\ &\to_\beta M[\Box, y{:=}L, x][\Box, x{:=}N[\Box, y{:=}L, \Box], \Box] \end{aligned}$$

Compute the critical pair from the term Q

$$(\lambda y.(\not\lambda y.\lambda x.M)N)L$$

Inner redex first:

$$\begin{aligned} Q &\to_\beta (\lambda y.M[y, x{:=}N, \Box])L \\ &\to_\beta M[y, x{:=}N, \Box][\Box, y{:=}L, \Box] \end{aligned}$$

Outer redex first:

$$\begin{aligned} Q &\to_\beta ((\not\lambda y.\lambda x.M)N)[\Box, y{:=}L, \Box] \\ &= (\not\lambda y.\lambda x.M)[\Box, y{:=}L, \Box]N[\Box, y{:=}L, \Box] \\ &= (\lambda x.M)N[\Box, y{:=}L, \Box] \\ &\to_\beta M[\Box, x{:=}N[\Box, y{:=}L, \Box], \Box] \end{aligned}$$

Lem 2 *Closed substitution lemma: if* $\langle X'xZ\rangle s$, $\langle Y'yZ'Z\rangle t$ *and* $\langle YZ'Z\rangle u$, *then:*

$$s[Y'yZ', x{:=}t, X'][Y, y{:=}u, X'Y'] \\ = s[Y'YZ', x{:=}t[Y, y{:=}u, Y'], X']$$

```
Lemma closed_subst_bal :
 (s,t,u:term;X',Y,Y',Z,Z':(list name);x,y:name)
 (bal (conc X' (cons x Z)) s)
  -> (bal (conc Y' (conc (cons y Z') Z)) t)
  -> (bal (conc Y (conc Z' Z)) u)
   -> (subst Y (conc X' Y') (subst (conc Y'
         (cons y Z')) X' s x t) y u)
      = (subst (conc Y' (conc Y Z')) X' s x
          (subst Y Y' t y u)).
```

Lem 3 *Open substitution lemma: if $\langle X'xY'yZ\rangle s$, $\langle XY'yZ\rangle t$ and $\langle YZ\rangle u$, then:*

$$s[X,x{:=}t,X'][Y,y{:=}u,X'XY'] = s[Y,y{:=}u,X'xY'][X,x{:=}t[Y,y{:=}u,XY'],X']$$

```
Lemma open_subst_bal :
 (s,t,u:term;X,X',Y,Y',Z:(list name);x,y:name)
  (bal (conc X' (conc (cons x Y')(cons y Z))) s)
   -> (bal (conc X (conc Y' (cons y Z))) t)
    -> (bal (conc Y Z) u)
     -> (subst Y (conc X' (conc X Y'))
          (subst X X' s x t) y u)
         = (subst X X' (subst Y
          (conc X' (cons x Y')) s y u) x
          (subst Y (conc X Y') t y u)).
```

Def 8 *α-equality à la Kahrs.* *We define $M =_\alpha N$, if $\langle\Box\rangle M =_\alpha \langle\Box\rangle N$, where for vectors of variables X and Y, $\langle X\rangle M =_\alpha \langle Y\rangle N$ is inductively defined as follows. By $|X|$ we denote the length of vector X.*

$$
\begin{aligned}
\langle\Box\rangle x &=_\alpha \langle\Box\rangle x \\
\langle Xx\rangle x &=_\alpha \langle Yy\rangle y, \text{ if } |X| = |Y| \\
\langle Xx'\rangle x &=_\alpha \langle Yy'\rangle y, \text{ if } \langle X\rangle x =_\alpha \langle Y\rangle y, \\
&\qquad\qquad\qquad x' \neq x,\ y' \neq y \\
\langle X\rangle \lambda x.M &=_\alpha \langle Y\rangle \lambda y.N, \text{ if } \langle Xx\rangle M =_\alpha \langle Yy\rangle N \\
\langle X\rangle M_1 M_2 &=_\alpha \langle Y\rangle N_1 N_2, \text{ if } \langle X\rangle M_1 =_\alpha \langle Y\rangle N_1 \\
&\qquad\qquad\qquad \langle X\rangle M_2 =_\alpha \langle Y\rangle N_2
\end{aligned}
$$

Home Page
Title Page
Contents
Page 22 of 43
Go Back
Full Screen
Close
Quit

For λ-terms we add the following clauses.

$$\begin{aligned}
\langle\Box\rangle\lambda x.M &=_\alpha \langle\Box\rangle\lambda x.N, \text{ if } \langle\Box\rangle M =_\alpha \langle\Box\rangle N\\
\langle Xx\rangle\lambda x.M &=_\alpha \langle Yy\rangle\lambda y.N, \text{ if } \langle X\rangle M =_\alpha \langle Y\rangle N\\
\langle Xx'\rangle\lambda x.M &=_\alpha \langle Yy'\rangle\lambda y.N, \text{ if } \langle X\rangle\lambda x.M =_\alpha \langle Y\rangle\lambda y.N\\
&\qquad\qquad\qquad x' \neq x,\ y' \neq y
\end{aligned}$$

4.6. Operational α-equivalence

Def 9 *α-conversion $\leftrightarrow^*_\alpha$ is defined as the reflexive, symmetric, transitive closure of single-step α-renaming $\longrightarrow_\alpha$, which is defined as the compatible closure of the α-rule:*

```
alpha_rule :
 (M:sterm;x,y:name)
  ~(In y (names M))
   ->(alpha_conv (abs x M)(abs y (rename M x y Nil)))
```

Def 10 *α-equality à la Schroer.*

```
Definition alpha_eq2
 := [M,N:sterm](EX Z:(list name)|(alpha_eq2' M N Z)).
```

makes use of an auxiliary stack Z which records the variables chosen thusfar for renaming.

```
alpha_eq2_rule :
 (M,N:sterm;x,y,z:name;Z:(list name))
  ~(In z (names M))
   ->~(In z (names N))
    ->~(In z Z)
     ->(alpha_eq2' (rename M x z Nil)(rename N y z Nil) Z)
      ->(alpha_eq2' (abs x M)(abs y N)(cons z Z))
```

The clause dealing with λ is just a compatibility clause.

Thm 11 *All three notions of α-equivalence are equivalent.*

Note that to prove that λ-terms which are α-equivalent à la Kahrs are α-equivalent according to the other two definitions, one essentially uses the Fresh variable axiom. (It is not needed in the other direction.)

Thm 12 *α-equivalence is a congruent equivalence relation.*

Lem 4 *Lifting of* $\rightarrow_{\lambda\beta}$ *to* $\rightarrow_{⅄\beta}$ *(schema* A *in Figure). If* $M \rightarrow_{\lambda\beta} N$, $\langle X\rangle M' \rightarrow_{\omega} M$, *then there are* N_1, N_2 *such that:*

$$\langle X\rangle N_1 =_{\alpha} \langle X\rangle N_2 \rightarrow_{\omega} N$$

and

$$M' \rightarrow_{⅄\beta} N_1$$

Lem 5 *Projection of* λ*-substitution to* λ*-substitution. If*

- $\langle ZxY_1\rangle M_1 =_\alpha \langle ZxY_2\rangle M_2 \to_\omega M$
- $\langle ZX\rangle N' \to_\omega N$,
- $X \cap \mathsf{FV}(M_1, Y_1x) = \emptyset$,
- $Y_2 \cap (\{x\} \cup \mathsf{FV}(N', \Box)) = \emptyset$,

then there exists a P *such that:*

$$\begin{aligned}\langle ZXY_1\rangle M_1[X, x{:=}N', Y_1] &=_\alpha \langle ZXY_2\rangle P[X, x{:=}N', Y_2] \\ &\to_\omega M[x{:=}N]\end{aligned}$$

5. Results

– Confluence of β without α

– α needed to remove $\lambdabar$'s:
$\lambda x.\lambdabar x.x \rightarrow_{\alpha} \lambda y.\lambdabar y.x \rightarrow_{\text{forget}} \lambda y.x$

– Confluence of ordinary β modulo α

– Proofs in Coq

Home Page
Title Page
Contents
Page 28 of 43
Go Back
Full Screen
Close
Quit

6. Related/current work

Related work

– refines Guillaume, David, Bird, Paterson: de Bruijn ($\reflectbox{$\lambda$}$ = Successor, variable = zero)

– Di Cosmo et al.: with labels (commutativity, α)

– Explicit weakening

Current and further research

– Push $\reflectbox{$\lambda$}$'s locally : $\reflectbox{$\lambda$} x.\lambda x.M \to \lambda_1 x.\reflectbox{λ} x.M$ (reopen, reclose scope)

– optimal implementation (scopes, no croissants/brackets)

– explicit substitution (lemmas as rules) (CR, PSN)

7. Explicit Substitution

Explicit substitution lemma ($\_[\_{:=}\_]$ as syntax):

$$P = (\lambda y.(\lambda x.M)N)L$$

$$P \;\rightarrow_{\text{inner}}\; M[x{:=}N][y{:=}L]$$
$$P \;\rightarrow_{\text{outer}}\; M[y{:=}L][x{:=}N[y{:=}L]]$$

How to orient this critical pair?

7.1. left-orientation: non-confluence

Orientation from right to left:

$$M[x{:=}N][y{:=}L] \leftarrow M[y{:=}\underline{L}][x{:=}N[y{:=}\underline{L}]]$$

Problem: repeated variable ($\underline{L}$) in LHS: non-confluence

7.2. right-orientation: loop

Orientation from left to right:

$$M[x{:=}N][y{:=}L] \\ \rightarrow\ M[y{:=}L][x{:=}N[y{:=}L]]$$

Problem: LHS embedded in RHS: non-termination (loop)

Home Page
Title Page
Contents
Page 33 of 43
Go Back
Full Screen
Close
Quit

7.3. right-orientation: breaking the loop

Explicit substitution lemma in λ-calculus:

$$(\lambda y.(\lambda x.M)N)L$$

$$\begin{array}{lll} P & \rightarrow_{\text{inner}} & M[\Box, x{:=}N, \Box][\Box, y{:=}L, \Box] \\ P & \rightarrow_{\text{outer}} & M[\Box, y{:=}L, \underline{x}][\Box, \underline{x}{:=}N[\Box, y{:=}L, \Box], \Box] \end{array}$$

How can we break the loop?

7.3. right-orientation: breaking the loop

Explicit substitution lemma in λ-calculus:

$$(\lambda y.(\lambda x.M)N)L$$

$$P \rightarrow_{\text{inner}} M[\square, x{:=}N, \square][\square, y{:=}L, \square]$$
$$P \rightarrow_{\text{outer}} M[\square, y{:=}L, \underline{x}][\square, \underline{x}{:=}N[\square, y{:=}L, \square], \square]$$

How can we break the loop?
Idea: recognise outside-in order (by the $\underline{x}$)
(substitution over λx leaves x on stack)

7.4. Explicit substitution calculus with names

Explicit substitution lemma rules of $\lambda\!\!\!\lambda_{\mathsf{ws}}$:

$$s[Y'yZ', x{:=}t, X'][Y, y{:=}u, X'Y']$$
$$\to_{\mathsf{closed}}\ s[Y'YZ', x{:=}t[Y, y{:=}u, Y'], X']$$
$$s[X, x{:=}t, X'][Y, y{:=}u, X'XY']$$
$$\to_{\mathsf{open}}\ s[Y, y{:=}u, X'xY'][X, x{:=}t[Y, y{:=}u, XY'], X']$$

with side-conditions to break the loop

Conjecture 1 *preservation of strong normalisation*

SN in λ implies SN in $\lambda\!\!\!\lambda$ implies SN in $\lambda\!\!\!\lambda_{\mathsf{ws}}$

Why?

7.4. Explicit substitution calculus with names

Explicit substitution lemma rules of $ƛ_{\mathsf{ws}}$:

$$s[Y'yZ', x{:=}t, X'][Y, y{:=}u, X'Y']$$
$$\quad\rightarrow_{\mathsf{closed}}\ s[Y'YZ', x{:=}t[Y, y{:=}u, Y'], X']$$
$$s[X, x{:=}t, X'][Y, y{:=}u, X'XY']$$
$$\quad\rightarrow_{\mathsf{open}}\ s[Y, y{:=}u, X'xY'][X, x{:=}t[Y, y{:=}u, XY'], X']$$

with side-conditions to break the loop

Conjecture 1 *preservation of strong normalisation*

SN in λ implies SN in $ƛ$ implies SN in $ƛ_{\mathsf{ws}}$

Why?

$ƛ_{\mathsf{ws}}$ is named version of David and Guillaume's λ_{ws}

length bounded by length of outside-in reduction

8. Optimal reduction

Avoid useless work

$$\begin{aligned}(\lambda x.y)\Omega &\to (\lambda x.y)\Omega \\ &\to y\end{aligned}$$

Avoid duplicate work

$$\begin{aligned}(\lambda x.xx)(IK) &\to (IK)(IK) \\ &\to K(IK) \\ &\to KK\end{aligned}$$

How to avoid useless work?

8. Optimal reduction

Avoid useless work

$$\begin{aligned}(\lambda x.y)\Omega &\rightarrow (\lambda x.y)\Omega \\ &\rightarrow y\end{aligned}$$

Avoid duplicate work

$$\begin{aligned}(\lambda x.xx)(IK) &\rightarrow (IK)(IK) \\ &\rightarrow K(IK) \\ &\rightarrow KK\end{aligned}$$

How to avoid useless work?
Leftmost-outermost strategy

8. Optimal reduction

Avoid useless work

$$
\begin{aligned}
(\lambda x.y)\Omega &\rightarrow (\lambda x.y)\Omega \\
&\rightarrow y
\end{aligned}
$$

Avoid duplicate work

$$
\begin{aligned}
(\lambda x.xx)(IK) &\rightarrow (IK)(IK) \\
&\rightarrow K(IK) \\
&\rightarrow KK
\end{aligned}
$$

How to avoid useless work?
Leftmost-outermost strategy
How to avoid duplicate work?

8. Optimal reduction

Avoid useless work

$$\begin{aligned}(\lambda x.y)\Omega &\rightarrow (\lambda x.y)\Omega \\ &\rightarrow y\end{aligned}$$

Avoid duplicate work

$$\begin{aligned}(\lambda x.xx)(IK) &\rightarrow (IK)(IK) \\ &\rightarrow K(IK) \\ &\rightarrow KK\end{aligned}$$

How to avoid useless work?
Leftmost-outermost strategy
How to avoid duplicate work?
Avoid duplication by making it explicit

Home Page

Title Page

Contents

Page 36 of 43

Go Back

Full Screen

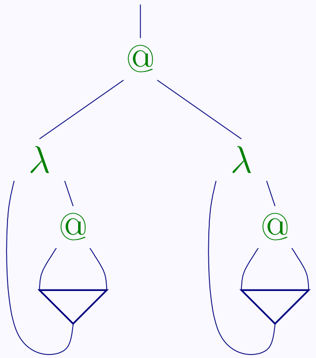
Close

Quit

8.1. Explicit duplication

Explicit duplication node: fan

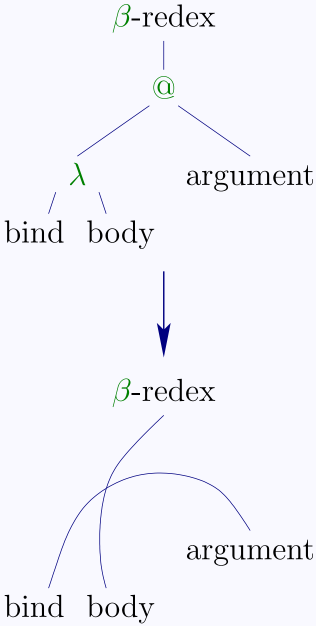
$(\lambda x.xx)(\lambda x.xx) =$

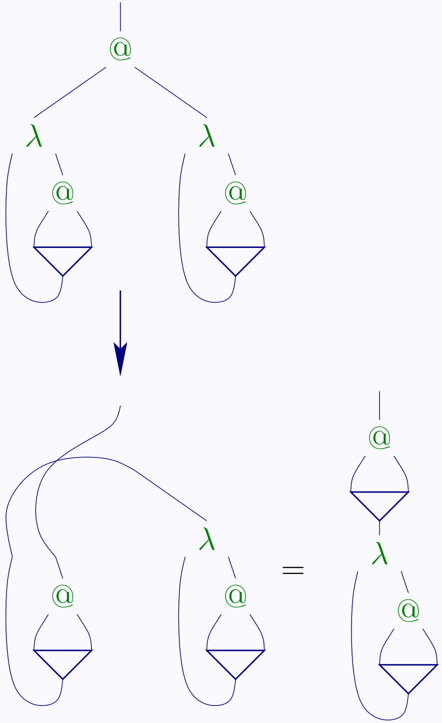


$= \Omega$

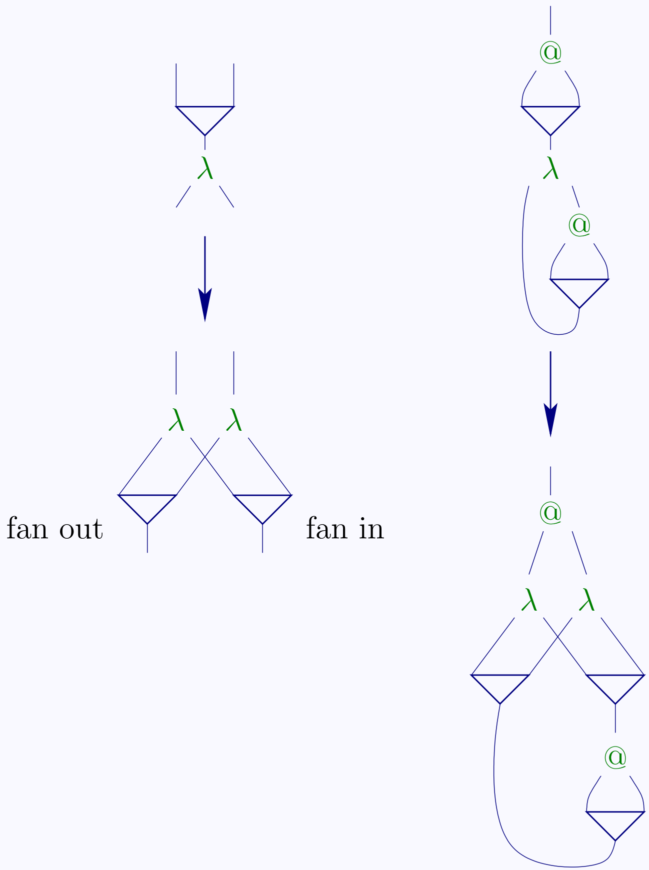
explicite duplicator $=$ ... $=$ fan

8.2. β on graphs

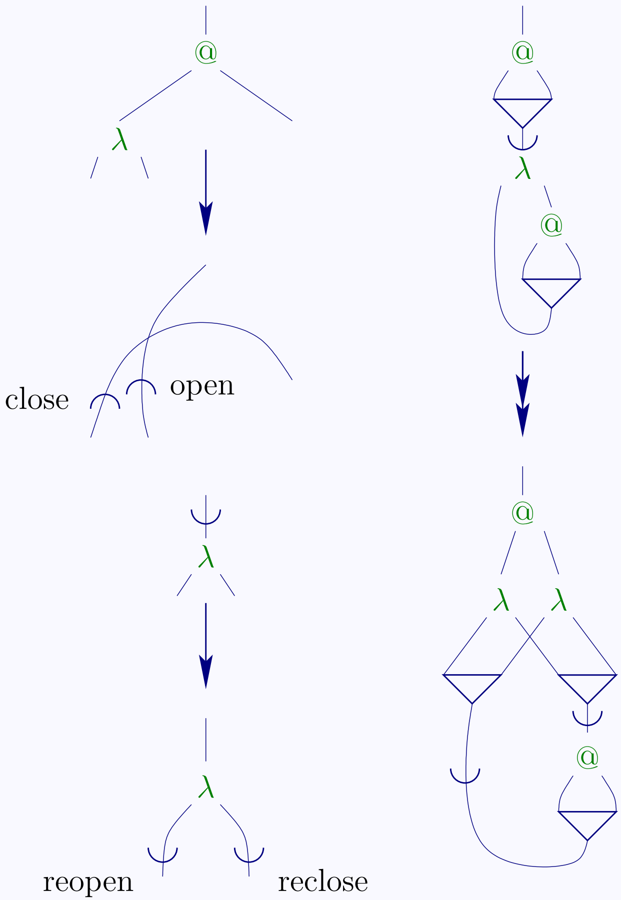




8.3. Localising duplication



8.4. Localising scoping



8.5. Benchmark

$c_{10}c_2c_2II$ where c_n is nth Church-numeral
recall: $c_m c_n \twoheadrightarrow c_{n^m}$

- Caml Light, Haskell, BOHM 1.0 explode
- BOHM 1.1 (Bolgona Optimal Higher-order Machine) 531706 steps (56 β)
- lambdascope : 1781260 steps (56 β)

lambdascope vs BOHM 1.1:

- solves explosion of brackets/croissants, hence
- no heuristics needed (always works)
- atomic steps (no compound nodes, S vs $+n$)

Home Page

Title Page

Contents

Page 43 of 43

Go Back

Full Screen

Close

Quit

8.6. Demo of lambdascope