Reductions and Causality (VI)

jean-jacques.levy@inria.fr
Escuela de Ciencias Informáticas
Universidad de Buenos Aires

July 26, 2013

http://jeanjacqueslevy.net/courses/13eci

Reductions and Causality (VI)

jean-jacques.levy@inria.fr
Escuela de Ciencias Informáticas
Universidad de Buenos Aires

July 26, 2013

http://jeanjacqueslevy.net/courses/13eci

Reductions and Causality (VI)

jean-jacques.levy@inria.fr
Escuela de Ciencias Informáticas
Universidad de Buenos Aires

July 26, 2013

http://jeanjacqueslevy.net/courses/13eci

Reductions and Causality (VI)

jean-jacques.levy@inria.fr
Escuela de Ciencias Informáticas
Universidad de Buenos Aires

July 26, 2013

http://jeanjacqueslevy.net/courses/13eci

Plan

- complete reductions
- sublattice of complete reductions
- more on canonical representatives
- costs of reductions + sharing
- speculative computations
- semantics with Böhm trees

Plan

- complete reductions
- sublattice of complete reductions
- more on canonical representatives
- costs of reductions + sharing
- speculative computations
- semantics with Böhm trees

Plan

- complete reductions
- sublattice of complete reductions
- more on canonical representatives
- costs of reductions + sharing
- speculative computations
- semantics with Böhm trees

Plan

- complete reductions
- sublattice of complete reductions
- more on canonical representatives
- costs of reductions + sharing
- speculative computations
- semantics with Böhm trees

Labeled λ-calculus

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Labeled λ-calculus

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Labeled λ-calculus

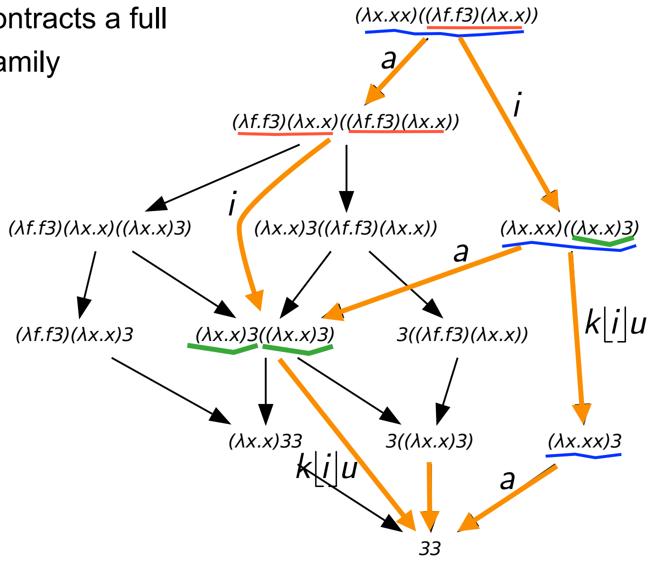
CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Labeled λ-calculus

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

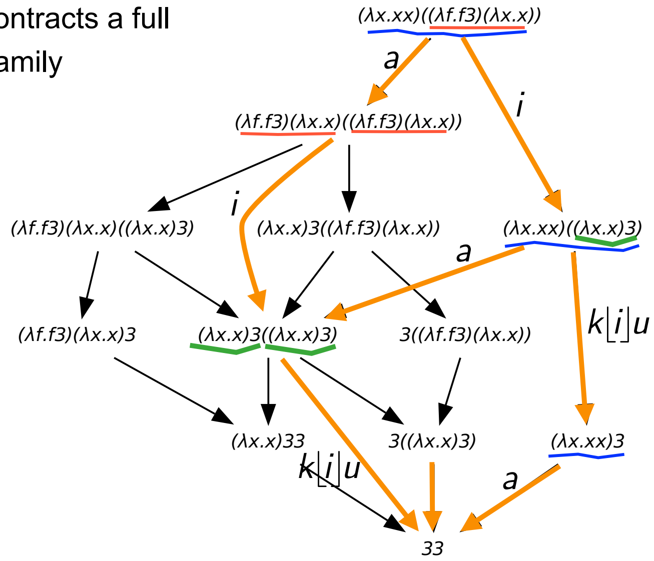
Complete reductions (1/5)

- Each step contracts a full single redex family



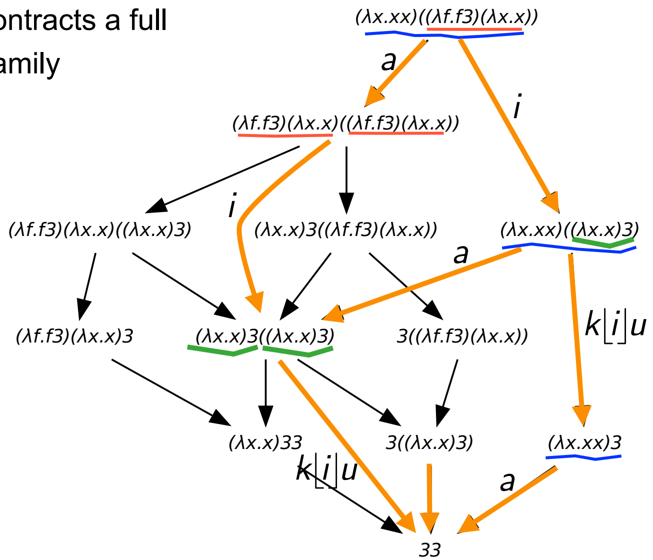
Complete reductions (1/5)

- Each step contracts a full single redex family



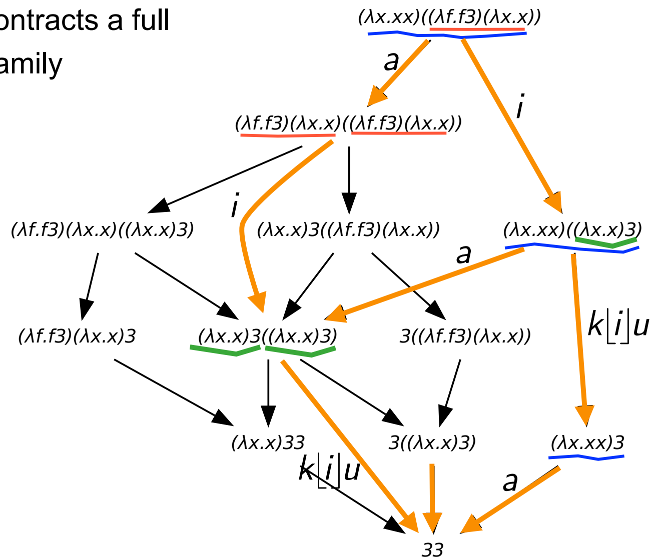
Complete reductions (1/5)

- Each step contracts a full single redex family



Complete reductions (1/5)

- Each step contracts a full single redex family



Complete reductions (2/5)

- **Definition** [complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is an historical set of redexes when $\mathcal{F}$ is a set of redexes in final term of ρ.

$\langle \rho, \mathcal{F} \rangle$ is **f-complete** when it is maximum set such that

$$R, S \in \mathcal{F} \text{ implies } \langle \rho, R \rangle \sim \langle \rho, S \rangle$$

An **f-complete reduction** contracts an f-complete set at each step.

- **Proposition** [lattice of f-complete reductions]

Complete reductions form a sub-lattice of the lattice of reductions.

Proof simple use of following lemma which implies f-complete parallel moves.

Complete reductions (2/5)

- **Definition** [complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is an historical set of redexes when $\mathcal{F}$ is a set of redexes in final term of ρ.

$\langle \rho, \mathcal{F} \rangle$ is **f-complete** when it is maximum set such that

$$R, S \in \mathcal{F} \text{ implies } \langle \rho, R \rangle \sim \langle \rho, S \rangle$$

An **f-complete reduction** contracts an f-complete set at each step.

- **Proposition** [lattice of f-complete reductions]

Complete reductions form a sub-lattice of the lattice of reductions.

Proof simple use of following lemma which implies f-complete parallel moves.

Complete reductions (2/5)

- **Definition** [complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is an historical set of redexes when $\mathcal{F}$ is a set of redexes in final term of ρ.

$\langle \rho, \mathcal{F} \rangle$ is **f-complete** when it is maximum set such that

$$R, S \in \mathcal{F} \text{ implies } \langle \rho, R \rangle \sim \langle \rho, S \rangle$$

An **f-complete reduction** contracts an f-complete set at each step.

- **Proposition** [lattice of f-complete reductions]

Complete reductions form a sub-lattice of the lattice of reductions.

Proof simple use of following lemma which implies f-complete parallel moves.

Complete reductions (2/5)

- **Definition** [complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is an historical set of redexes when $\mathcal{F}$ is a set of redexes in final term of ρ.

$\langle \rho, \mathcal{F} \rangle$ is **f-complete** when it is maximum set such that

$$R, S \in \mathcal{F} \text{ implies } \langle \rho, R \rangle \sim \langle \rho, S \rangle$$

An **f-complete reduction** contracts an f-complete set at each step.

- **Proposition** [lattice of f-complete reductions]

Complete reductions form a sub-lattice of the lattice of reductions.

Proof simple use of following lemma which implies f-complete parallel moves.

Complete reductions (3/5)

- **Notations**
  $M \overset{\alpha}{\Longrightarrow} N$ when $M \xrightarrow{\mathcal{F}} N$ and $\mathcal{F}$ is the set of redexes with name α in M.

  MaxRedNames(M) when all redexes in M have maximal names.

- **Lemma** `[complete reductions preserve max redex names]`
  $M \overset{\alpha}{\Longrightarrow} N$ and MaxRedNames(M) implies MaxRedNames(N)

Complete reductions (3/5)

- **Notations**
  $M \overset{\alpha}{\Longrightarrow} N$ when $M \xrightarrow{\mathcal{F}} N$ and $\mathcal{F}$ is the set of redexes with name α in M.

  MaxRedNames(M) when all redexes in M have maximal names.

- **Lemma** `[complete reductions preserve max redex names]`
  $M \overset{\alpha}{\Longrightarrow} N$ and MaxRedNames(M) implies MaxRedNames(N)

Complete reductions (3/5)

- **Notations**
  $M \overset{\alpha}{\Longrightarrow} N$ when $M \xrightarrow{\mathcal{F}} N$ and $\mathcal{F}$ is the set of redexes with name α in M.

  MaxRedNames(M) when all redexes in M have maximal names.

- **Lemma** `[complete reductions preserve max redex names]`
  $M \overset{\alpha}{\Longrightarrow} N$ and MaxRedNames(M) implies MaxRedNames(N)

Complete reductions (3/5)

- **Notations**
  $M \overset{\alpha}{\Longrightarrow} N$ when $M \xrightarrow{\mathcal{F}} N$ and $\mathcal{F}$ is the set of redexes with name α in M.

  MaxRedNames(M) when all redexes in M have maximal names.

- **Lemma** `[complete reductions preserve max redex names]`
  $M \overset{\alpha}{\Longrightarrow} N$ and MaxRedNames(M) implies MaxRedNames(N)

Complete reductions (4/5)

- **Definition** [d-complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is **d-complete** when it is maximum set such that

$$\langle \rho_0, R_0 \rangle \leq \langle \rho, \mathcal{F} \rangle \text{ for some } \langle \rho_0, R_0 \rangle$$

An **d-complete reduction** contracts a d-complete set at each step.

- **Proposition** [below canonical representative]
  Let $\langle \rho_0, R_0 \rangle$ be canonical representative in its family.
  Let $\rho_0 \sqsubseteq \rho$. Then $\langle \rho_0, R_0 \rangle \sim \langle \rho, R \rangle$ iff $\langle \rho_0, R_0 \rangle \leq \langle \rho, R \rangle$.

  Proof difficult.

- **Proposition** [f-complete = d-complete]
  d-complete reductions coincide with f-complete reductions.

Complete reductions (4/5)

- **Definition** [d-complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is **d-complete** when it is maximum set such that

$$\langle \rho_0, R_0 \rangle \leq \langle \rho, \mathcal{F} \rangle \text{ for some } \langle \rho_0, R_0 \rangle$$

An **d-complete reduction** contracts a d-complete set at each step.

- **Proposition** [below canonical representative]
  Let $\langle \rho_0, R_0 \rangle$ be canonical representative in its family.
  Let $\rho_0 \sqsubseteq \rho$. Then $\langle \rho_0, R_0 \rangle \sim \langle \rho, R \rangle$ iff $\langle \rho_0, R_0 \rangle \leq \langle \rho, R \rangle$.

  Proof difficult.

- **Proposition** [f-complete = d-complete]
  d-complete reductions coincide with f-complete reductions.

Complete reductions (4/5)

- **Definition** [d-complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is **d-complete** when it is maximum set such that

$$\langle \rho_0, R_0 \rangle \leq \langle \rho, \mathcal{F} \rangle \text{ for some } \langle \rho_0, R_0 \rangle$$

An **d-complete reduction** contracts a d-complete set at each step.

- **Proposition** [below canonical representative]
  Let $\langle \rho_0, R_0 \rangle$ be canonical representative in its family.
  Let $\rho_0 \sqsubseteq \rho$. Then $\langle \rho_0, R_0 \rangle \sim \langle \rho, R \rangle$ iff $\langle \rho_0, R_0 \rangle \leq \langle \rho, R \rangle$.

  Proof difficult.

- **Proposition** [f-complete = d-complete]
  d-complete reductions coincide with f-complete reductions.

Complete reductions (4/5)

- **Definition** [d-complete reductions]

$\langle \rho, \mathcal{F} \rangle$ is **d-complete** when it is maximum set such that

$$\langle \rho_0, R_0 \rangle \leq \langle \rho, \mathcal{F} \rangle \text{ for some } \langle \rho_0, R_0 \rangle$$

An **d-complete reduction** contracts a d-complete set at each step.

- **Proposition** [below canonical representative]
  Let $\langle \rho_0, R_0 \rangle$ be canonical representative in its family.
  Let $\rho_0 \sqsubseteq \rho$. Then $\langle \rho_0, R_0 \rangle \sim \langle \rho, R \rangle$ iff $\langle \rho_0, R_0 \rangle \leq \langle \rho, R \rangle$.

  Proof difficult.

- **Proposition** [f-complete = d-complete]
  d-complete reductions coincide with f-complete reductions.

Complete reductions (5/5)

- **Proposition** [length of reduction = number of families]

  In complete reductions, number of steps equals the number of contracted redex families.

  Proof application of MaxRedNames lemma.

- **Corollary** [optimal reductions]

  In complete reductions, never redex of same family is contracted twice.

- **Implementation** [optimal reductions]

  Can we implement efficiently complete reductions ?

Complete reductions (5/5)

- **Proposition** [length of reduction = number of families]

  In complete reductions, number of steps equals the number of contracted redex families.

  Proof application of MaxRedNames lemma.

- **Corollary** [optimal reductions]

  In complete reductions, never redex of same family is contracted twice.

- **Implementation** [optimal reductions]

  Can we implement efficiently complete reductions ?

Complete reductions (5/5)

- **Proposition** [length of reduction = number of families]

  In complete reductions, number of steps equals the number of contracted redex families.

  Proof application of MaxRedNames lemma.

- **Corollary** [optimal reductions]

  In complete reductions, never redex of same family is contracted twice.

- **Implementation** [optimal reductions]

  Can we implement efficiently complete reductions ?

Complete reductions (5/5)

- **Proposition** [length of reduction = number of families]

  In complete reductions, number of steps equals the number of contracted redex families.

  Proof application of MaxRedNames lemma.

- **Corollary** [optimal reductions]

  In complete reductions, never redex of same family is contracted twice.

- **Implementation** [optimal reductions]

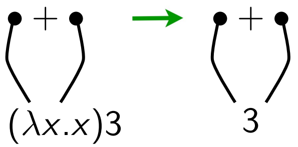
  Can we implement efficiently complete reductions ?

Implementation (1/5)

- **Implementation** [optimal reductions] algorithm [John Lamping, 90 -- Gonthier-Abadi-JJ, 91]

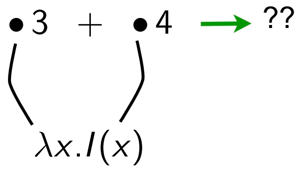
- Sharing of basic values is easy:

$(\lambda x.x + x)((\lambda x.x)3) \longrightarrow$



- Problem is sharing of functions:

$(\lambda x.x3 + x4)((\lambda x.I(x)) \longrightarrow$

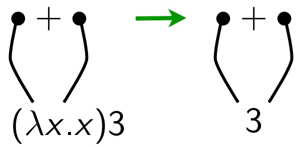


Implementation (1/5)

- **Implementation** [optimal reductions] algorithm [John Lamping, 90 -- Gonthier-Abadi-JJ, 91]

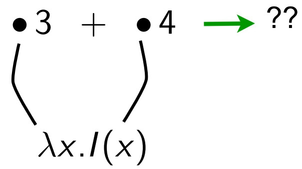
- Sharing of basic values is easy:

$(\lambda x.x + x)((\lambda x.x)3) \longrightarrow$



- Problem is sharing of functions:

$(\lambda x.x3 + x4)((\lambda x.I(x)) \longrightarrow$

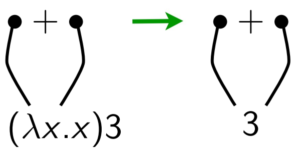


Implementation (1/5)

- **Implementation** [optimal reductions] algorithm [John Lamping, 90 -- Gonthier-Abadi-JJ, 91]

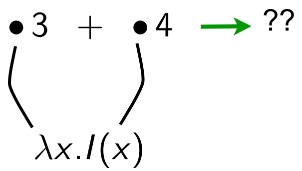
- Sharing of basic values is easy:

$(\lambda x.x + x)((\lambda x.x)3) \longrightarrow$



- Problem is sharing of functions:

$(\lambda x.x3 + x4)((\lambda x.I(x)) \longrightarrow$

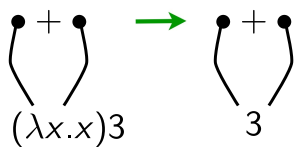


Implementation (1/5)

- **Implementation** [optimal reductions] algorithm [John Lamping, 90 -- Gonthier-Abadi-JJ, 91]

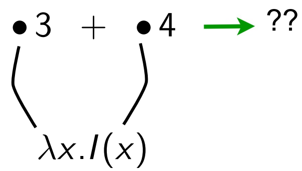
- Sharing of basic values is easy:

$(\lambda x.x + x)((\lambda x.x)3) \longrightarrow$



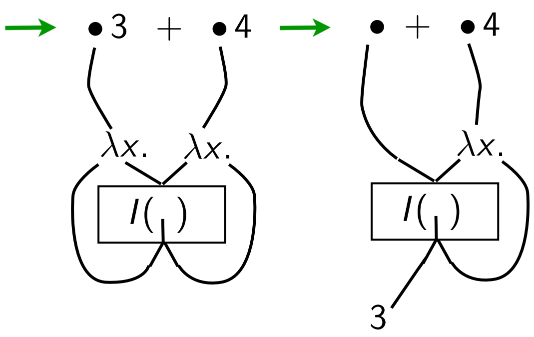
- Problem is sharing of functions:

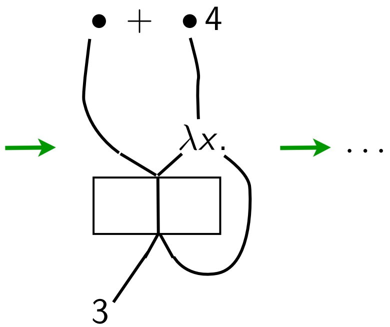
$(\lambda x.x3 + x4)((\lambda x.I(x)) \longrightarrow$



Implementation (2/5)

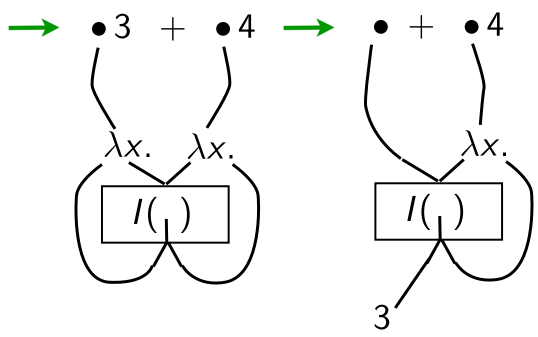
$(\lambda x.x3 + x4)((\lambda x.I(x))$ $\rightarrow$

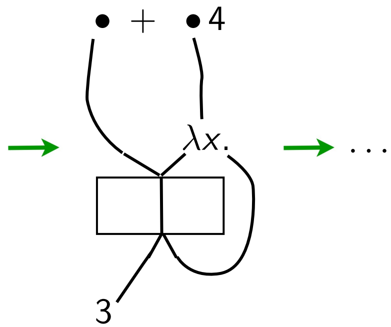




Implementation (2/5)

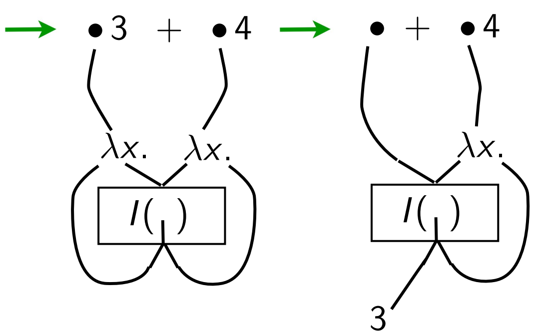
$(\lambda x.x3 + x4)((\lambda x.I(x))$ $\rightarrow$

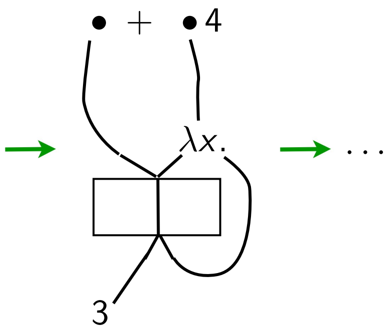




Implementation (2/5)

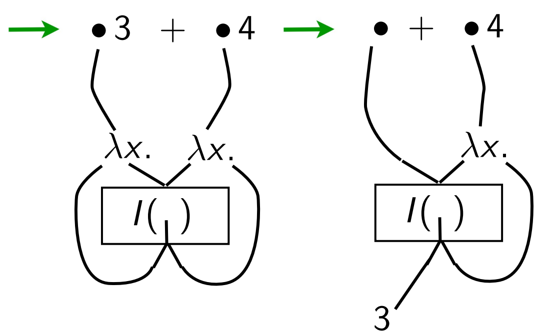
$(\lambda x.x3 + x4)((\lambda x.I(x))$ $\rightarrow$

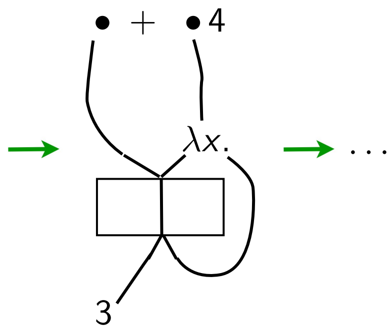




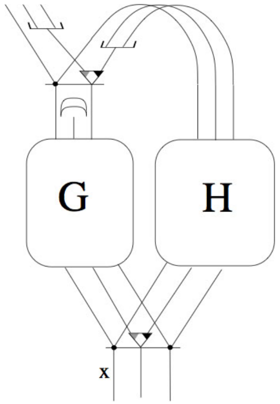
Implementation (2/5)

$(\lambda x.x3 + x4)((\lambda x.I(x))$ $\rightarrow$

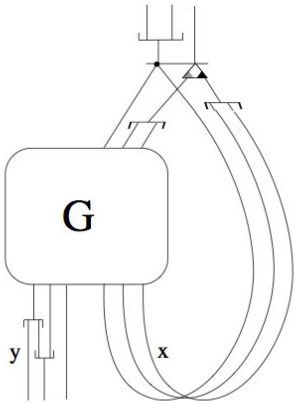




Implementation (3/5)

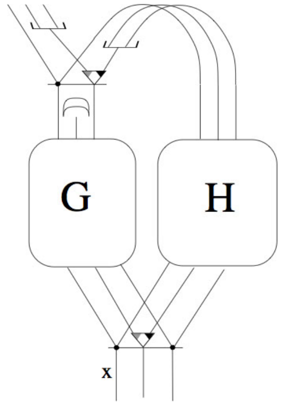


application

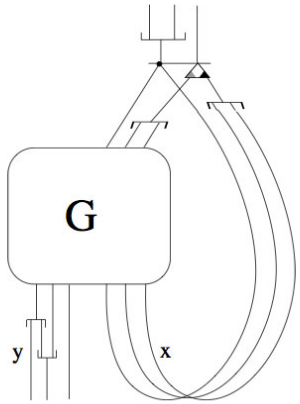


λ-abstraction

Implementation (3/5)

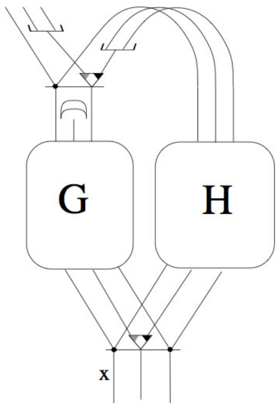


application

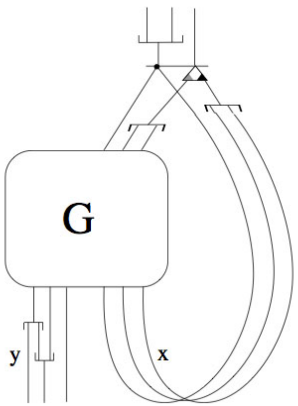


λ-abstraction

Implementation (3/5)

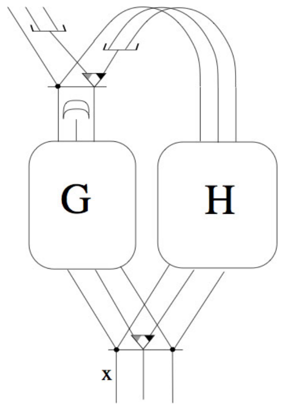


application

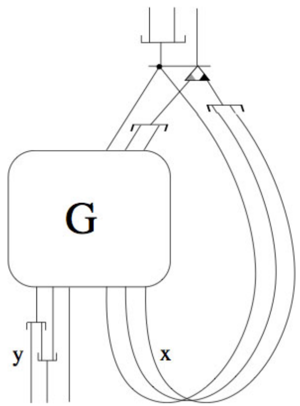


λ-abstraction

Implementation (3/5)

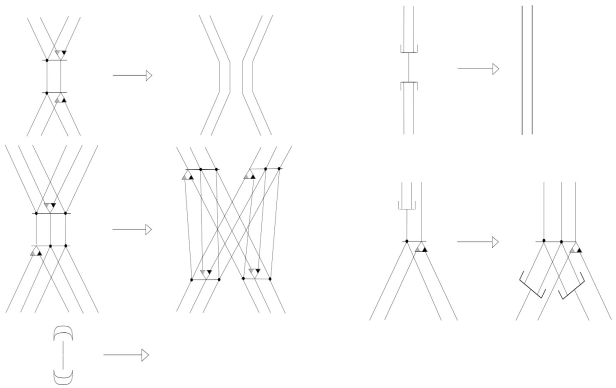


application

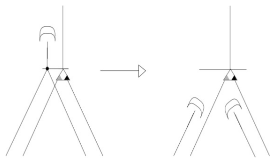


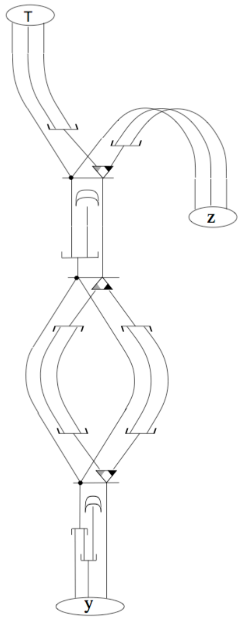
λ-abstraction

Implementation (4/5)



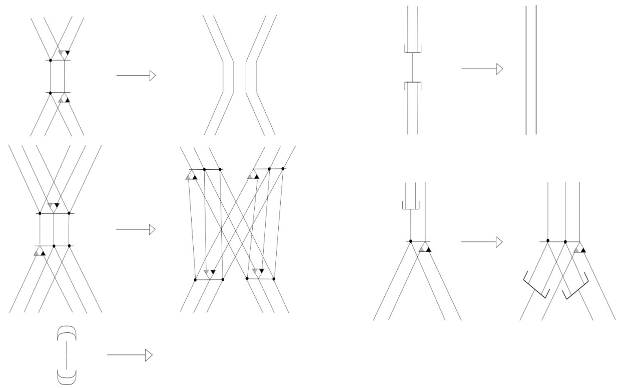
rules



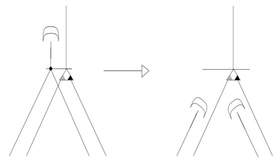


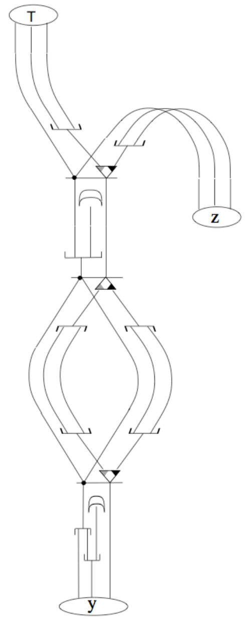
$(\lambda x.yx)z$

Implementation (4/5)



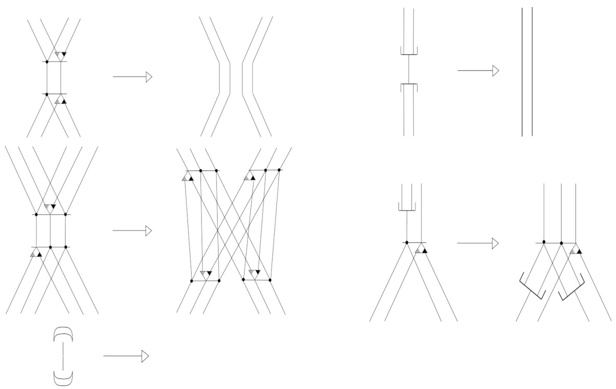
rules



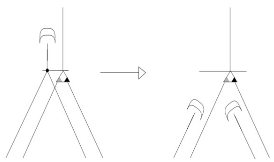


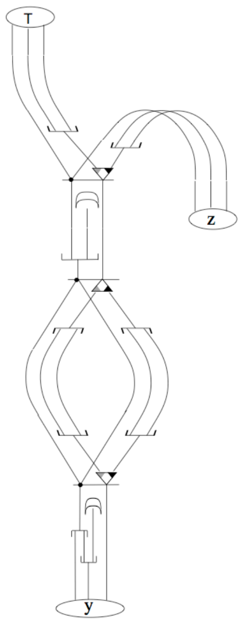
$(\lambda x.yx)z$

Implementation (4/5)



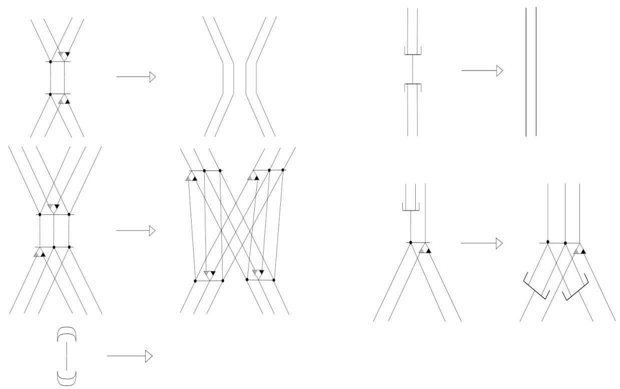
rules



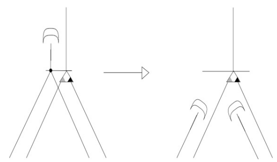


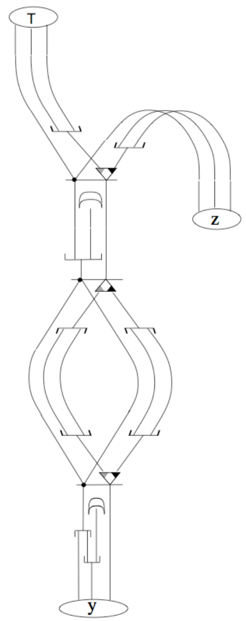
$(\lambda x.yx)z$

Implementation (4/5)



rules





$(\lambda x.yx)z$

Implementation (5/5)

- beautiful Lamping's algorithm is unpractical
- highly exponential in the handling of fans node (not elementary recursive) [Asperti, Mairson 2000]
- nice algorithms unsharing paths to bound variables [Wadsworth 92, Shivers-Wand 2010]
- Haskell, Coq, Caml ??

Implementation (5/5)

- beautiful Lamping's algorithm is unpractical
- highly exponential in the handling of fans node (not elementary recursive) [Asperti, Mairson 2000]
- nice algorithms unsharing paths to bound variables [Wadsworth 92, Shivers-Wand 2010]
- Haskell, Coq, Caml ??

Implementation (5/5)

- beautiful Lamping's algorithm is unpractical
- highly exponential in the handling of fans node (not elementary recursive) [Asperti, Mairson 2000]
- nice algorithms unsharing paths to bound variables [Wadsworth 92, Shivers-Wand 2010]
- Haskell, Coq, Caml ??

Implementation (5/5)

- beautiful Lamping's algorithm is unpractical
- highly exponential in the handling of fans node (not elementary recursive) [Asperti, Mairson 2000]
- nice algorithms unsharing paths to bound variables [Wadsworth 92, Shivers-Wand 2010]
- Haskell, Coq, Caml ??

Speculative computations

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Speculative computations

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Speculative computations

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Speculative computations

CENTRE DE RECHERCHE COMMUN
INRIA
MICROSOFT RESEARCH

Permutations in call by value

- **Definition** `[call by value]`

  A **value** remains a value if computed or substituted by a value

  $$V \ ::= \ x \ | \ \lambda x.M$$

  The call-by-value reduction strategy is defined by:

  $$(\lambda x.M)V \xrightarrow[\text{cbv}]{} M\{x := V\}$$

  $$\frac{M \xrightarrow[\text{cbv}]{} M'}{MN \xrightarrow[\text{cbv}]{} M'N} \qquad \frac{N \xrightarrow[\text{cbv}]{} N'}{MN \xrightarrow[\text{cbv}]{} MN'}$$

- **Fact** `[permutations in call by value]`

  Equivalence by permutations only permute disjoint redexes.

Permutations in call by value

- **Definition** `[call by value]`

  A **value** remains a value if computed or substituted by a value

  $$V \ ::= \ x \ | \ \lambda x.M$$

  The call-by-value reduction strategy is defined by:

  $$(\lambda x.M)V \xrightarrow[\text{cbv}]{} M\{x := V\}$$

  $$\frac{M \xrightarrow[\text{cbv}]{} M'}{MN \xrightarrow[\text{cbv}]{} M'N} \qquad \frac{N \xrightarrow[\text{cbv}]{} N'}{MN \xrightarrow[\text{cbv}]{} MN'}$$

- **Fact** `[permutations in call by value]`

  Equivalence by permutations only permute disjoint redexes.

Permutations in call by value

- **Definition** `[call by value]`

  A **value** remains a value if computed or substituted by a value

  $$V \ ::= \ x \ | \ \lambda x.M$$

  The call-by-value reduction strategy is defined by:

  $$(\lambda x.M)V \xrightarrow[\text{cbv}]{} M\{x := V\}$$

  $$\frac{M \xrightarrow[\text{cbv}]{} M'}{MN \xrightarrow[\text{cbv}]{} M'N} \qquad \frac{N \xrightarrow[\text{cbv}]{} N'}{MN \xrightarrow[\text{cbv}]{} MN'}$$

- **Fact** `[permutations in call by value]`

  Equivalence by permutations only permute disjoint redexes.

Permutations in call by value

- **Definition** `[call by value]`

  A **value** remains a value if computed or substituted by a value

  $$V \ ::= \ x \ | \ \lambda x.M$$

  The call-by-value reduction strategy is defined by:

  $$(\lambda x.M)V \xrightarrow[\text{cbv}]{} M\{x := V\}$$

  $$\frac{M \xrightarrow[\text{cbv}]{} M'}{MN \xrightarrow[\text{cbv}]{} M'N} \qquad \frac{N \xrightarrow[\text{cbv}]{} N'}{MN \xrightarrow[\text{cbv}]{} MN'}$$

- **Fact** `[permutations in call by value]`

  Equivalence by permutations only permute disjoint redexes.

Speculative reductions

- **Definition** [speculative call, Boudol-Petri 2010]

$$V ::= x \mid \lambda x.M$$

The speculative reduction strategy is defined by:

$$(\lambda x.M)V \xrightarrow[\text{spec}]{} M\{x := V\}$$

$$(\lambda x.M)N \xrightarrow[\text{spec}]{} (\lambda V?\, M\{x := V\})N$$

$$(\lambda V?\, M)V \xrightarrow[\text{spec}]{} M$$

$$\frac{M \xrightarrow[\text{spec}]{} M'}{MN \xrightarrow[\text{spec}]{} M'N} \qquad \frac{N \xrightarrow[\text{spec}]{} N'}{MN \xrightarrow[\text{spec}]{} MN'} \qquad \frac{M \xrightarrow[\text{spec}]{} M'}{\lambda V?M \xrightarrow[\text{spec}]{} \lambda V?M'}$$

Speculative reductions

- **Definition** [speculative call, Boudol-Petri 2010]

$$V ::= x \mid \lambda x.M$$

The speculative reduction strategy is defined by:

$$(\lambda x.M)V \xrightarrow[\text{spec}]{} M\{x := V\}$$

$$(\lambda x.M)N \xrightarrow[\text{spec}]{} (\lambda V?\, M\{x := V\})N$$

$$(\lambda V?\, M)V \xrightarrow[\text{spec}]{} M$$

$$\frac{M \xrightarrow[\text{spec}]{} M'}{MN \xrightarrow[\text{spec}]{} M'N} \qquad \frac{N \xrightarrow[\text{spec}]{} N'}{MN \xrightarrow[\text{spec}]{} MN'} \qquad \frac{M \xrightarrow[\text{spec}]{} M'}{\lambda V?M \xrightarrow[\text{spec}]{} \lambda V?M'}$$

Speculative reductions

- **Definition** [speculative call, Boudol-Petri 2010]

$$V ::= x \mid \lambda x.M$$

The speculative reduction strategy is defined by:

$$(\lambda x.M)V \xrightarrow[\text{spec}]{} M\{x := V\}$$

$$(\lambda x.M)N \xrightarrow[\text{spec}]{} (\lambda V?\, M\{x := V\})N$$

$$(\lambda V?\, M)V \xrightarrow[\text{spec}]{} M$$

$$\frac{M \xrightarrow[\text{spec}]{} M'}{MN \xrightarrow[\text{spec}]{} M'N} \qquad \frac{N \xrightarrow[\text{spec}]{} N'}{MN \xrightarrow[\text{spec}]{} MN'} \qquad \frac{M \xrightarrow[\text{spec}]{} M'}{\lambda V?M \xrightarrow[\text{spec}]{} \lambda V?M'}$$

Speculative reductions

- **Definition** [speculative call, Boudol-Petri 2010]

$$V ::= x \mid \lambda x.M$$

The speculative reduction strategy is defined by:

$$(\lambda x.M)V \xrightarrow[\text{spec}]{} M\{x := V\}$$

$$(\lambda x.M)N \xrightarrow[\text{spec}]{} (\lambda V?\, M\{x := V\})N$$

$$(\lambda V?\, M)V \xrightarrow[\text{spec}]{} M$$

$$\frac{M \xrightarrow[\text{spec}]{} M'}{MN \xrightarrow[\text{spec}]{} M'N} \qquad \frac{N \xrightarrow[\text{spec}]{} N'}{MN \xrightarrow[\text{spec}]{} MN'} \qquad \frac{M \xrightarrow[\text{spec}]{} M'}{\lambda V?M \xrightarrow[\text{spec}]{} \lambda V?M'}$$

Assigning meaning to λ-expressions

CENTRE DE RECHERCHE COMMUN INRIA MICROSOFT RESEARCH

Assigning meaning to λ-expressions

CENTRE DE RECHERCHE COMMUN INRIA MICROSOFT RESEARCH

Assigning meaning to λ-expressions

CENTRE DE RECHERCHE COMMUN INRIA MICROSOFT RESEARCH

Assigning meaning to λ-expressions

CENTRE DE RECHERCHE COMMUN INRIA MICROSOFT RESEARCH

Semantics

Definition A semantics of the λ-calculus is any equivalence such that:

(1) $M \xrightarrow{*} N$ implies $M \equiv N$

(2) $M \equiv N$ implies $C[M] \equiv C[N]$

- Thus β-interconvertibility $=_\beta$ is a semantics.
- Any other interesting semantics ?

Semantics

Definition A semantics of the λ-calculus is any equivalence such that:

(1) $M \xrightarrow{*} N$ implies $M \equiv N$

(2) $M \equiv N$ implies $C[M] \equiv C[N]$

- Thus β-interconvertibility $=_\beta$ is a semantics.
- Any other interesting semantics ?

Semantics

Definition A semantics of the λ-calculus is any equivalence such that:

(1) $M \xrightarrow{*} N$ implies $M \equiv N$

(2) $M \equiv N$ implies $C[M] \equiv C[N]$

- Thus β-interconvertibility $=_\beta$ is a semantics.
- Any other interesting semantics ?

Semantics

Definition A semantics of the λ-calculus is any equivalence such that:

(1) $M \xrightarrow{*} N$ implies $M \equiv N$

(2) $M \equiv N$ implies $C[M] \equiv C[N]$

- Thus β-interconvertibility $=_\beta$ is a semantics.
- Any other interesting semantics ?

Böhm's theorem

Theorem [Bohm, 68]

Let M and N be 2 distinct normal forms. Then for any x and y, there exists a context $C[\,]$ such that:

$$C[M] \xrightarrow{*} x \quad \text{and} \quad C[N] \xrightarrow{*} y$$

Corollary Any (consistent) semantics of the λ-calculus cannot identify 2 distinct normal forms.

Notice Distinct normal forms means not η-interconvertible.

Exercice Bohm's thm for $I = \lambda x.x$ and $K = \lambda x.\lambda y.x$.

Böhm's theorem

Theorem [Bohm, 68]

Let M and N be 2 distinct normal forms. Then for any x and y, there exists a context $C[\,]$ such that:

$$C[M] \xrightarrow{*} x \quad \text{and} \quad C[N] \xrightarrow{*} y$$

Corollary Any (consistent) semantics of the λ-calculus cannot identify 2 distinct normal forms.

Notice Distinct normal forms means not η-interconvertible.

Exercice Bohm's thm for $I = \lambda x.x$ and $K = \lambda x.\lambda y.x$.

Böhm's theorem

Theorem [Bohm, 68]

Let M and N be 2 distinct normal forms. Then for any x and y, there exists a context $C[\,]$ such that:

$$C[M] \xrightarrow{*} x \quad \text{and} \quad C[N] \xrightarrow{*} y$$

Corollary Any (consistent) semantics of the λ-calculus cannot identify 2 distinct normal forms.

Notice Distinct normal forms means not η-interconvertible.

Exercice Bohm's thm for $I = \lambda x.x$ and $K = \lambda x.\lambda y.x$.

Böhm's theorem

Theorem [Bohm, 68]

Let M and N be 2 distinct normal forms. Then for any x and y, there exists a context $C[\,]$ such that:

$$C[M] \xrightarrow{*} x \quad \text{and} \quad C[N] \xrightarrow{*} y$$

Corollary Any (consistent) semantics of the λ-calculus cannot identify 2 distinct normal forms.

Notice Distinct normal forms means not η-interconvertible.

Exercice Bohm's thm for $I = \lambda x.x$ and $K = \lambda x.\lambda y.x$.

Terms without normal forms

Lemma It is inconsistent to identify all terms without normal forms

Proof: Take $M = xa\Omega,\ N = y\Omega b$ where $\Omega = (\lambda x.xx)(\lambda x.xx)$

Let $C[\,] = (\lambda x.\lambda y.[\,])K(KI)$

Then $C[M] \xrightarrow{*} a$ and $C[N] \xrightarrow{*} b$

Question Which terms can be consistently identified ?

Easy terms `[Bohm, Jacopini]` $I = \Omega$ is consistent !

Terms without normal forms

Lemma It is inconsistent to identify all terms without normal forms

Proof: Take $M = xa\Omega,\ N = y\Omega b$ where $\Omega = (\lambda x.xx)(\lambda x.xx)$

Let $C[\,] = (\lambda x.\lambda y.[\,])K(KI)$

Then $C[M] \xrightarrow{*} a$ and $C[N] \xrightarrow{*} b$

Question Which terms can be consistently identified ?

Easy terms `[Bohm, Jacopini]` $I = \Omega$ is consistent !

Terms without normal forms

Lemma It is inconsistent to identify all terms without normal forms

Proof: Take $M = xa\Omega,\ N = y\Omega b$ where $\Omega = (\lambda x.xx)(\lambda x.xx)$

Let $C[\,] = (\lambda x.\lambda y.[\,])K(KI)$

Then $C[M] \xrightarrow{*} a$ and $C[N] \xrightarrow{*} b$

Question Which terms can be consistently identified ?

Easy terms `[Bohm, Jacopini]` $I = \Omega$ is consistent !

Terms without normal forms

Lemma It is inconsistent to identify all terms without normal forms

Proof: Take $M = xa\Omega,\ N = y\Omega b$ where $\Omega = (\lambda x.xx)(\lambda x.xx)$

Let $C[\,] = (\lambda x.\lambda y.[\,])K(KI)$

Then $C[M] \xrightarrow{*} a$ and $C[N] \xrightarrow{*} b$

Question Which terms can be consistently identified ?

Easy terms `[Bohm, Jacopini]` $I = \Omega$ is consistent !

Terms without normal forms

Definition [Wadsworth, 72] M is totally undefined iff

for all $C[\,]$, if $C[M] \xrightarrow{*} nf$, then $C[N] \xrightarrow{*} nf$ for any N.

Fact: Ω is totally undefined.
$xa\Omega$ and $y\Omega b$ are not totally undefined.

Exercice:
Find other terms totally undefined. Try with $\Delta_3 = \lambda x.xxx$, $K = \lambda x.\lambda y.x$ and $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

Terms without normal forms

Definition [Wadsworth, 72] M is totally undefined iff

for all $C[\,]$, if $C[M] \xrightarrow{*} nf$, then $C[N] \xrightarrow{*} nf$ for any N.

Fact: Ω is totally undefined.
$xa\Omega$ and $y\Omega b$ are not totally undefined.

Exercice:
Find other terms totally undefined. Try with $\Delta_3 = \lambda x.xxx$, $K = \lambda x.\lambda y.x$ and $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

Terms without normal forms

Definition [Wadsworth, 72] M is totally undefined iff

for all $C[\,]$, if $C[M] \xrightarrow{*} nf$, then $C[N] \xrightarrow{*} nf$ for any N.

Fact: Ω is totally undefined.
$xa\Omega$ and $y\Omega b$ are not totally undefined.

Exercice:
Find other terms totally undefined. Try with $\Delta_3 = \lambda x.xxx$, $K = \lambda x.\lambda y.x$ and $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

Terms without normal forms

Definition [Wadsworth, 72] M is totally undefined iff

for all $C[\,]$, if $C[M] \xrightarrow{*} nf$, then $C[N] \xrightarrow{*} nf$ for any N.

Fact: Ω is totally undefined.
$xa\Omega$ and $y\Omega b$ are not totally undefined.

Exercice:
Find other terms totally undefined. Try with $\Delta_3 = \lambda x.xxx$, $K = \lambda x.\lambda y.x$ and $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

Terms without normal forms

Definition [Wadsworth, 72] M is in head normal form (hnf) iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.xM_1M_2\dots M_n \qquad (m,n \geq 0)$$

head variable

M not in hnf iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.(\underline{\lambda x.P)Q}M_1M_2\dots M_n \qquad (m,n \geq 0)$$

head redex

Proposition: M totally undefined iff M has no hnf.

Terms without normal forms

Definition [Wadsworth, 72] M is in head normal form (hnf) iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.xM_1M_2\dots M_n \qquad (m,n \geq 0)$$

head variable

M not in hnf iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.(\underline{\lambda x.P)Q}M_1M_2\dots M_n \qquad (m,n \geq 0)$$

head redex

Proposition: M totally undefined iff M has no hnf.

Terms without normal forms

Definition [Wadsworth, 72] M is in head normal form (hnf) iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.xM_1M_2\dots M_n \qquad (m,n \geq 0)$$

head variable

M not in hnf iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.(\underline{\lambda x.P)Q}M_1M_2\dots M_n \qquad (m,n \geq 0)$$

head redex

Proposition: M totally undefined iff M has no hnf.

Terms without normal forms

Definition [Wadsworth, 72] M is in head normal form (hnf) iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.xM_1M_2\dots M_n \qquad (m,n \geq 0)$$

head variable

M not in hnf iff

$$M = \lambda x_1.\lambda x_2.\dots\lambda x_m.(\underline{\lambda x.P)Q}M_1M_2\dots M_n \qquad (m,n \geq 0)$$

head redex

Proposition: M totally undefined iff M has no hnf.

Bohm trees (1/3)

Definition [72] The Bohm tree $\mathrm{BT}(M)$ of M is defined(?) as follows:

(1) If M has no hnf, $\mathrm{BT}(M) = \perp$

(2) If $M \xrightarrow{*} \lambda x_1.\lambda x_2 \ldots \lambda x_m.xM_1 M_2 \ldots M_n$, then

$$\mathrm{BT}(M) = \lambda x_1.\lambda x_2 \ldots \lambda x_m.x$$

$$\mathrm{BT}(M_1) \quad \mathrm{BT}(M_2) \quad \cdots \quad \mathrm{BT}(M_n)$$

Exercices Compute $\mathrm{BT}(I)$, $\mathrm{BT}(K)$, $\mathrm{BT}(\Omega)$, $\mathrm{BT}(Y)$, ...
BT ($L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L$) where
L = λabcdefghijklmnopqstuvwxyzr. (r (t h i s i s a f i x e d p o i n t c o m b i n a t o r))

Bohm trees (1/3)

Definition [72] The Bohm tree $\mathrm{BT}(M)$ of M is defined(?) as follows:

(1) If M has no hnf, $\mathrm{BT}(M) = \perp$

(2) If $M \xrightarrow{*} \lambda x_1.\lambda x_2 \ldots \lambda x_m.xM_1 M_2 \ldots M_n$, then

$$\mathrm{BT}(M) = \lambda x_1.\lambda x_2 \ldots \lambda x_m.x$$

$$\mathrm{BT}(M_1) \quad \mathrm{BT}(M_2) \quad \cdots \quad \mathrm{BT}(M_n)$$

Exercices Compute $\mathrm{BT}(I)$, $\mathrm{BT}(K)$, $\mathrm{BT}(\Omega)$, $\mathrm{BT}(Y)$, ...
BT ($L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L$) where
L = λabcdefghijklmnopqstuvwxyzr. (r (t h i s i s a f i x e d p o i n t c o m b i n a t o r))

Bohm trees (1/3)

Definition [72] The Bohm tree $\mathrm{BT}(M)$ of M is defined(?) as follows:

(1) If M has no hnf, $\mathrm{BT}(M) = \perp$

(2) If $M \xrightarrow{*} \lambda x_1.\lambda x_2 \ldots \lambda x_m.xM_1 M_2 \ldots M_n$, then

$$\mathrm{BT}(M) = \lambda x_1.\lambda x_2 \ldots \lambda x_m.x$$

$$\mathrm{BT}(M_1) \quad \mathrm{BT}(M_2) \quad \cdots \quad \mathrm{BT}(M_n)$$

Exercices Compute $\mathrm{BT}(I)$, $\mathrm{BT}(K)$, $\mathrm{BT}(\Omega)$, $\mathrm{BT}(Y)$, ...
BT ($L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L$) where
L = λabcdefghijklmnopqstuvwxyzr. (r (t h i s i s a f i x e d p o i n t c o m b i n a t o r))

Bohm trees (1/3)

Definition [72] The Bohm tree $\mathrm{BT}(M)$ of M is defined(?) as follows:

(1) If M has no hnf, $\mathrm{BT}(M) = \perp$

(2) If $M \xrightarrow{*} \lambda x_1.\lambda x_2 \ldots \lambda x_m.xM_1 M_2 \ldots M_n$, then

$$\mathrm{BT}(M) = \lambda x_1.\lambda x_2 \ldots \lambda x_m.x$$

$$\mathrm{BT}(M_1) \quad \mathrm{BT}(M_2) \quad \cdots \quad \mathrm{BT}(M_n)$$

Exercices Compute $\mathrm{BT}(I)$, $\mathrm{BT}(K)$, $\mathrm{BT}(\Omega)$, $\mathrm{BT}(Y)$, ...
BT ($L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L\,L$) where
L = λabcdefghijklmnopqstuvwxyzr. (r (t h i s i s a f i x e d p o i n t c o m b i n a t o r))

Bohm trees (2/3)

Theorem [74] Let $M \equiv_{BT} N$ iff $BT(M) = BT(N)$. Then $\equiv_{BT}$ is a (consistent) semantics of the λ-calculus.

Proof: (1) $M \xrightarrow{*} N$ implies $BT(M) = BT(N)$.
by Church-Rosser.

(2) $BT(M) = BT(N)$ implies $BT(C[M]) = BT(C[N])$.
by completeness of inside-out reductions.

Bohm trees (2/3)

Theorem [74] Let $M \equiv_{BT} N$ iff $BT(M) = BT(N)$. Then $\equiv_{BT}$ is a (consistent) semantics of the λ-calculus.

Proof: (1) $M \xrightarrow{*} N$ implies $BT(M) = BT(N)$.
by Church-Rosser.

(2) $BT(M) = BT(N)$ implies $BT(C[M]) = BT(C[N])$.
by completeness of inside-out reductions.

Bohm trees (2/3)

Theorem [74] Let $M \equiv_{BT} N$ iff $BT(M) = BT(N)$. Then $\equiv_{BT}$ is a (consistent) semantics of the λ-calculus.

Proof: (1) $M \xrightarrow{*} N$ implies $BT(M) = BT(N)$.
by Church-Rosser.

(2) $BT(M) = BT(N)$ implies $BT(C[M]) = BT(C[N])$.
by completeness of inside-out reductions.

Bohm trees (2/3)

Theorem [74] Let $M \equiv_{BT} N$ iff $BT(M) = BT(N)$. Then $\equiv_{BT}$ is a (consistent) semantics of the λ-calculus.

Proof: (1) $M \xrightarrow{*} N$ implies $BT(M) = BT(N)$.
by Church-Rosser.

(2) $BT(M) = BT(N)$ implies $BT(C[M]) = BT(C[N])$.
by completeness of inside-out reductions.

Bohm trees (3/3)

Facts [74] All Scott's semantics are quotients of equality of Bohm trees: D_∞, $P\omega$, T^ω, filter models, Jim Morris' extensional equiv.

Bohm trees (3/3)

Facts [74] All Scott's semantics are quotients of equality of Bohm trees: D_∞, $P\omega$, T^ω, filter models, Jim Morris' extensional equiv.

Bohm trees (3/3)

Facts [74] All Scott's semantics are quotients of equality of Bohm trees: D_∞, $P\omega$, T^ω, filter models, Jim Morris' extensional equiv.

Bohm trees (3/3)

Facts [74] All Scott's semantics are quotients of equality of Bohm trees: D_∞, $P\omega$, T^ω, filter models, Jim Morris' extensional equiv.