Reducibility candidates modulo isomorphisms

Cristian Sottile
UNQ & ICC (UBA–CONICET)

Alejandro Díaz-Caro
Inria, LORIA & UNQ

37th Symposium on Implementation and Application of Functional Languages

Facultad de Ingeniería, Montevideo, Uruguay

October 3, 2025

CONICET

UBA
Universidad de Buenos Aires
Argentina virtus robur et studium

Universidad Nacional de Quilmes

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Inria

Outline

Proposal

we have **System F/$_\sim$** we want $\mathcal{SN}$ we need $RED/_\sim$

Outline

Proposal

we have **System F/$_\sim$** we want $\mathcal{SN}$ we need $RED/_\sim$

Outline

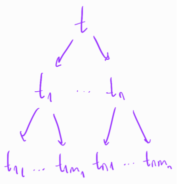
- Termination
- Reducibility
  - STLC
  - System F
  - System F modulo isomorphisms

What and why

Intuition

$$(\lambda x.\, t)\; s$$

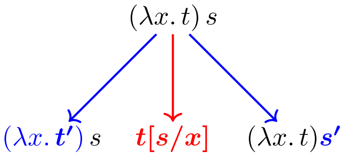
$$(\lambda x.\, \boldsymbol{t'})\; s \qquad \boldsymbol{t[s/x]} \qquad (\lambda x.\, t)\,\boldsymbol{s'}$$

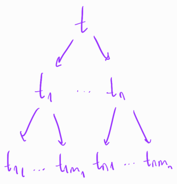


$\mathcal{SN}$ = all branches finite

What and why

Intuition





$\mathcal{SN}$ = all branches finite

Why?

- safety core language (without fix)
- freedom at implementation
- (my take) should–have unless intended

How (not by induction on terms)

Induction does not work

$$\overbrace{(\lambda x.\, t)\; s}^{\text{new untested redex}}$$

$$\downarrow$$

$$\underbrace{\boldsymbol{t[s/x]}}_{>(\lambda x.t)s}$$

How (not by induction on terms)

Induction does not work

$$\overbrace{(\lambda x.\, t)\ s}^{\text{new untested redex}}$$
$$\downarrow$$
$$\underbrace{\boldsymbol{t[s/x]}}_{>(\lambda x.t)s}$$

$\mathcal{SN}$ of subterms is **not enough**

How (not by induction on terms)

Induction does not work

new untested redex

$$\overbrace{(\lambda x.\, t)\ s}^{\text{new untested redex}}$$

$$\downarrow$$

$$\underbrace{\boldsymbol{t[s/x]}}_{>(\lambda x.t)s}$$

$\mathcal{SN}$ of subterms is **not enough**

We need more: also remain $\mathcal{SN}$ when applied

How (not by induction on terms)

Induction does not work

$$(\lambda x.\, t)\; s\; u$$

$$\downarrow$$

$$\underbrace{\boldsymbol{t}[\boldsymbol{s}/\boldsymbol{x}]}_{\mathcal{SN}\text{ by IH}}\; u$$

How (not by induction on terms)

Induction does not work

$$(\lambda x.\, t)\; s\; u$$

$$\downarrow$$

$$\underbrace{\boldsymbol{t[s/x]}}_{\mathcal{SN}\text{ by IH}}\; u$$

Remaining $\mathcal{SN}$ is **not enough**

How (not by induction on terms)

Induction does not work

$$(\lambda x.\, t)\; s\; u$$

$$\downarrow$$

$$\underbrace{\boldsymbol{t[s/x]}}_{\mathcal{SN} \text{ by IH}}\; u$$

Remaining $\mathcal{SN}$ is **not enough**

We need more: recursively remain $\mathcal{SN}$ when applied

Reducibility for STLC — Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$t: \quad A_1 \quad \to \quad \cdots \quad \to \quad A_n \quad \to \quad \tau$$
$$\downarrow$$

Reducibility for STLC Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$\begin{array}{cccccccc} t: & A_1 & \to & \cdots & \to & A_n & \to & \tau \\ & & & & & & & \downarrow \\ t & & & & & & & \end{array}$$

Reducibility for STLC

Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$\begin{array}{ccccccccc} t: & A_1 & \to & \cdots & \to & A_n & \to & \tau \\ & \Cup & & & & & & \downarrow \\ t & s_1 & & & & & & \end{array}$$

Reducibility for STLC — Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$\begin{array}{ccccccccc} t: & A_1 & \to & \cdots & \to & A_n & \to & \tau \\ & \rotatebox{90}{$\in$} & & \cdots & & & & \downarrow \\ t & s_1 & & \cdots & & & & \end{array}$$

Reducibility for STLC — Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$\begin{array}{cccccccc} t: & A_1 & \to & \cdots & \to & A_n & \to & \tau \\ & \Cup & & \cdots & & \Cup & & \downarrow \\ t & s_1 & & \cdots & & s_n & & \end{array}$$

Reducibility for STLC

Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$
\begin{array}{ccccccccc}
t: & A_1 & \to & \cdots & \to & A_n & \to & \tau \\
 & \Cup & & \cdots & & \Cup & & \downarrow \\
t & s_1 & & \cdots & & s_n & \in &
\end{array}
$$

Reducibility for STLC

Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$\begin{array}{cccccccc} t: & A_1 & \rightarrow & \cdots & \rightarrow & A_n & \rightarrow & \tau \\ & \Cup & & \cdots & & \Cup & & \downarrow \\ t & s_1 & & \cdots & & s_n & \in & \mathcal{SN}_\tau \end{array}$$

Reducibility for STLC — Fetching stage

Intuition

- We need terms to behave well under all possible uses
- We need to know all the possible uses

$$\begin{array}{cccccccc} t: & A_1 & \to & \cdots & \to & A_n & \to & \tau \\ & \Cup & & \cdots & & \Cup & & \downarrow \\ t & s_1 & & \cdots & & s_n & \in & \mathcal{SN}_\tau \end{array}$$

The RED–set

$$\begin{array}{rcl} [\![\tau]\!] & = & \mathcal{SN}_\tau \\ [\![A \to B]\!] & = & \{\, t \in A \to B \mid \forall s \in [\![A]\!].\ ts \in [\![B]\!] \,\} \end{array}$$

Reducibility for STLC — Testing stage

Intuition

test $\lambda x.t$ for $[\![A \to B]\!]$

$$(\lambda x.t)s$$

$$(\lambda x.\boldsymbol{t'})s \qquad \boldsymbol{t}[\boldsymbol{s}/\boldsymbol{x}] \qquad (\lambda x.\,t)\boldsymbol{s'}$$

CR3: neutrality + induction on B: all one–step reducts RED implies RED

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad\qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] \;=\; \{\, t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \,\}$$

Example Let $I_\forall = \forall X.X \to X$

$$t : \quad \forall X \quad X \quad \to \quad X$$

$$t$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad\qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] \;=\; \{\, t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \,\}$$

Example Let $I_\forall = \forall X.X \to X$

$$t : \quad \forall X \quad \underset{\Cup}{X} \quad \to \quad X$$

$$t \qquad X$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad\qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] \;=\; \{\, t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \,\}$$

Example Let $I_\forall = \forall X.X \to X$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & \Cup & & \\ t & X & x^X & & \end{array}$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] \;=\; \{\, t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \,\}$$

Example Let $I_\forall = \forall X.X \to X$

$$t : \quad \forall X \quad X \quad \to \quad X$$

$$\qquad\qquad\quad \Cup$$

$$t \quad X \quad x^X \quad \in \quad \mathcal{SN}_X$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] = \{ t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \}$$

Example Let $I_\forall = \forall X.X \to X$

$$t : \quad \forall X \quad X \quad \to \quad X$$
$$\Cup$$
$$t \quad X \quad x^X \quad \in \quad \mathcal{SN}_X$$

$$t : \quad \forall X \quad X \quad \to \quad X$$

$$t$$

Reducibility for System F — Problem

Unlike $\lambda^{\rightarrow}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\rightarrow}$ approach fails

$$[\![\forall X.A]\!] = \{ t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \}$$

Example Let $I_\forall = \forall X.X \rightarrow X$

$$t : \forall X \quad X \rightarrow X$$

$$\Uparrow$$

$$t \quad X \quad x^X \in \mathcal{SN}_X$$

$$t : \forall X \quad X \rightarrow X$$

$$t \quad I_\forall$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] = \{ t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \}$$

Example Let $I_\forall = \forall X.X \to X$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & \Cup & & \\ t & X & x^X & \in & \mathcal{SN}_X \end{array}$$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & I_\forall & & \\ & & \Cup & & \\ t & I_\forall & t & & \end{array}$$

Reducibility for System F — Problem

Unlike $\lambda^{\rightarrow}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\rightarrow}$ approach fails

$$[\![\forall X.A]\!] = \{ t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \}$$

Example Let $I_\forall = \forall X.X \to X$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & \Cup & & \\ t & X & x^X & \in & \mathcal{SN}_X \end{array}$$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & I_\forall & \to & \\ & & \Cup & & \\ t & I_\forall & t & I_\forall & \end{array}$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] = \{ t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \}$$

Example Let $I_\forall = \forall X.X \to X$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & \Cup & & \\ t & X & x^X & \in & \mathcal{SN}_X \end{array}$$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & I_\forall & \to & I_\forall \\ & & \Cup & & \Cup \\ t & I_\forall & t & I_\forall & t \end{array}$$

Reducibility for System F — Problem

Unlike $\lambda^{\rightarrow}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\rightarrow}$ approach fails

$$[\![\forall X.A]\!] = \{ t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \}$$

Example Let $I_\forall = \forall X.X \rightarrow X$

$$\begin{array}{ccccc} t: & \forall X & X & \rightarrow & X \\ & & \Cup & & \\ t & X & x^X & \in & \mathcal{SN}_X \end{array}$$

$$\begin{array}{ccccccc} t: & \forall X & X & \rightarrow & X & & \\ & & I_\forall & \rightarrow & I_\forall & \rightarrow & \\ & & \Cup & & \Cup & & \\ t & I_\forall & t & I_\forall & t & I_\forall & \end{array}$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] = \{ t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \}$$

Example Let $I_\forall = \forall X.X \to X$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & \Cup & & \\ t & X & x^X & \in & \mathcal{SN}_X \end{array}$$

$$\begin{array}{cccccccc} t: & \forall X & X & \to & X & & \\ & & I_\forall & \to & I_\forall & \to & I_\forall \\ & & \Cup & & \Cup & & \Cup \\ t & I_\forall & t & I_\forall & t & I_\forall & t \end{array}$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad\qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] \;=\; \{\, t \in \forall X.A \mid \forall B \in \mathcal{K}.\; tB \in [\![A[B/X]]\!] \,\}$$

Example Let $I_\forall = \forall X.X \to X$

$$\begin{array}{ccccc} t: & \forall X & X & \to & X \\ & & \Cup & & \\ t & X & x^X & \in & \mathcal{SN}_X \end{array}$$

$$\begin{array}{cccccccc} t: & \forall X & X & \to & X & & & \\ & & I_\forall & \to & I_\forall & \to & I_\forall & \dots \\ & & \Cup & & \Cup & & \Cup & \\ t & I_\forall & t & I_\forall & t & I_\forall & t & \dots \end{array}$$

Reducibility for System F — Problem

Unlike $\lambda^{\to}$, not so easy to find the RED–set

Type application substitutes both the term *and the type*

$$\Lambda X.t : \forall X.A \qquad\qquad (\Lambda X.t)B : A[B/X]$$

Following $\lambda^{\to}$ approach fails

$$[\![\forall X.A]\!] \;=\; \{\, t \in \forall X.A \mid \forall B \in \mathcal{K}.\ tB \in [\![A[B/X]]\!] \,\}$$

Example Let $I_\forall = \forall X.X \to X$

$$\begin{array}{cccccc} t: & \forall X & X & \to & X \\ & & \rotatebox{90}{$\in$} & & \\ t & X & x^X & \in & \mathcal{SN}_X \end{array}$$

$$\begin{array}{cccccccc} t: & \forall X & X & \to & X & & & \\ & & I_\forall & \to & I_\forall & \to & I_\forall & \dots \\ & & \rotatebox{90}{$\in$} & & \rotatebox{90}{$\in$} & & \rotatebox{90}{$\in$} & \\ t & I_\forall & t & I_\forall & t & I_\forall & t & \dots \end{array}$$

Parametric RED–set

- Avoid substitution: stop at X
- Parameterize: save $I_\forall$ into a mapping $\rho : \mathsf{TVar} \to$ RED-set

Reducibility for System F

Fetching stage

But...

what to put into ρ for B?

B is a type   we can't put $[\![B]\!]$ what RED–set?

Reducibility for System F — Fetching stage

But...

what to put into ρ for B?

B is a type   we can't put $[\![B]\!]$ what RED–set?

Candidates

1. Abstractly describe RED–set by properties

Reducibility for System F — Fetching stage

But...

what to put into ρ for B?

B is a type — we can't put $[\![B]\!]$ — what RED–set?

Candidates

1. Abstractly describe RED–set by properties
2. Define the notion of **reducibility candidate of a type**: $\mathcal{R}_A$

any set satisfying CR1, CR2 and CR3

Reducibility for System F — Fetching stage

But...

what to put into ρ for B?

B is a type — we can't put $[\![B]\!]$ — what RED–set?

Candidates

1. Abstractly describe RED–set by properties
2. Define the notion of **reducibility candidate of a type**: $\mathcal{R}_A$

   any set satisfying CR1, CR2 and CR3
3. Parameterize $[\![\cdot]\!]$ by a map $\rho : \mathsf{TVar} \to \mathcal{R}$

$$[\![X]\!]_\rho \quad = \quad \rho(X)$$

Reducibility for System F

Fetching stage

But...

what to put into ρ for B?

B is a type   we can't put $[\![B]\!]$ what RED–set?

Candidates

1. Abstractly describe RED–set by properties
2. Define the notion of **reducibility candidate of a type**: $\mathcal{R}_A$

   any set satisfying CR1, CR2 and CR3

3. Parameterize $[\![\cdot]\!]$ by a map $\rho : \mathsf{TVar} \to \mathcal{R}$

$$[\![X]\!]_\rho \quad = \quad \rho(X)$$

4. Make $\forall$–step range over all $\mathcal{R}$ for any type B

$$[\![\forall X.A]\!]_\rho \quad = \quad \{\, t \in \forall X.A \mid \forall B \in \mathcal{K}, \mathcal{C}_B \in \mathcal{R}_B.\ tB \in [\![A]\!]_{[\rho \cdot X \mapsto \mathcal{C}_B]} \,\}$$

Reducibility for System F — Testing stage

when testing $\Lambda X.t : \forall X.A$

$$(\Lambda X.t)B$$

$$\swarrow \qquad \searrow$$

$$(\Lambda X.t)B \qquad \boldsymbol{t[B/X]}$$

tests from $[\![A]\!]_{\rho\cdot[\mathcal{C}_B/X]}$ are left to the CR3 induction

Simply typed λ–calculus modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \quad \sim \quad (A \to B) \times (A \to C)$$

Simply typed λ–calculus modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \quad \sim \quad (A \to B) \times (A \to C)$$

Typing rules

$$\frac{\Gamma \vdash t : A \qquad A \sim B}{\Gamma \vdash t : B}$$

Simply typed λ–calculus modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \quad \sim \quad (A \to B) \times (A \to C)$$

Typing rules

$$\frac{\Gamma \vdash t : A \qquad A \sim B}{\Gamma \vdash t : B}$$

Equivalence on terms

$$\begin{array}{ccc} \lambda x.\langle t, s\rangle & \leftrightarrows & \langle \lambda x.t, \lambda x.s\rangle \\ \langle t, s\rangle u & \leftrightarrows & \langle tu, su\rangle \end{array}$$

Simply typed λ–calculus modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \quad \sim \quad (A \to B) \times (A \to C)$$

Typing rules

$$\frac{\Gamma \vdash t : A \qquad A \sim B}{\Gamma \vdash t : B}$$

Equivalence on terms

$$\begin{array}{ccc} \lambda x.\langle t, s\rangle & \leftrightarrows & \langle \lambda x.t, \lambda x.s\rangle \\ \langle t, s\rangle u & \leftrightarrows & \langle tu, su\rangle \end{array}$$

Reduction

$$\to \;::=\; \rightleftarrows^* \circ \hookrightarrow$$

Simply typed λ–calculus modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \quad \sim \quad (A \to B) \times (A \to C)$$

Typing rules

$$\frac{\Gamma \vdash t : A \qquad A \sim B}{\Gamma \vdash t : B}$$

Equivalence on terms

$$\begin{array}{ccc} \lambda x.\langle t, s\rangle & \leftrightarrows & \langle \lambda x.t, \lambda x.s\rangle \\ \langle t, s\rangle u & \leftrightarrows & \langle tu, su\rangle \end{array}$$

Reduction

$$\to \; ::= \; \rightleftarrows^* \circ \hookrightarrow$$

$$\langle \lambda x.x, \lambda x.\lambda y.x\rangle s \;\rightleftarrows\; (\lambda x.\langle x, \lambda y.x\rangle)s \;\hookrightarrow\; \langle s, \lambda y.s\rangle$$

(Simple) Reducibility modulo isomorphisms — Problems

Problems

Problem 1: fetching (later) Problem 2: testing

(Simple) Reducibility modulo isomorphisms — Problems

Problems

Problem 1: fetching (later)

Problem 2: testing

Problem 2.a: Lack of neutrality

Problem 2.b: Knowing all the u_i

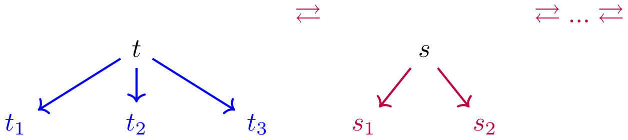
(Simple) Reducibility modulo isomorphisms — Problems

Problems

Problem 1: fetching (later) Problem 2: testing

Problem 2.a: Lack of neutrality Problem 2.b: Knowing all the u_i

Testing



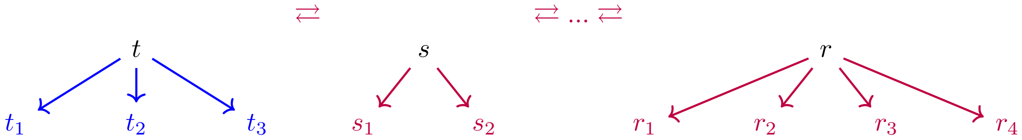
(Simple) Reducibility modulo isomorphisms — Problems

Problems

Problem 1: fetching (later) Problem 2: testing

Problem 2.a: Lack of neutrality Problem 2.b: Knowing all the u_i

Testing



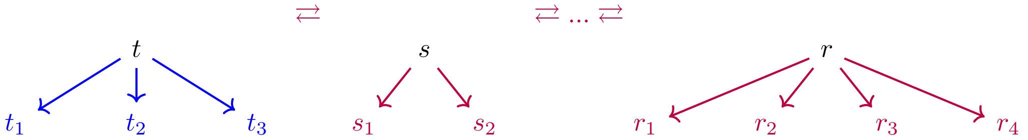
(Simple) Reducibility modulo isomorphisms — Problems

Problems

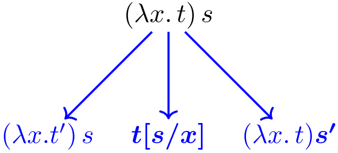
Problem 1: fetching (later) Problem 2: testing

Problem 2.a: Lack of neutrality Problem 2.b: Knowing all the u_i

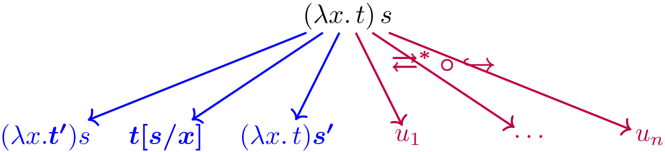
Testing



STLC



STLC/$\sim$



(Simple) Reducibility modulo isomorphisms — Solution 2.a

Solving lack of neutrality

- Neutrality is used to test one eliminator at a time
- Constructors commutation breaks neutrality
- "Local" testing does not work

(Simple) Reducibility modulo isomorphisms Solution 2.a

Solving lack of neutrality

- Neutrality is used to test one eliminator at a time
- Constructors commutation breaks neutrality
- "Local" testing does not work

Fetching all at once

$$[\![A_1 \to \ldots \to A_n \to \tau]\!]$$

(Simple) Reducibility modulo isomorphisms — Solution 2.a

Solving lack of neutrality

- Neutrality is used to test one eliminator at a time
- Constructors commutation breaks neutrality
- "Local" testing does not work

Fetching all at once

$$[\![A_1 \to \ldots \to A_n \to \tau]\!]$$

Testing all possible eliminators $\vec{u} = (u_1, \ldots, u_n)$ from $[\![A_1]\!] \times \ldots \times [\![A_n]\!]$

(Simple) Reducibility modulo isomorphisms — Solution 2.a

Solving lack of neutrality

- Neutrality is used to test one eliminator at a time
- Constructors commutation breaks neutrality
- "Local" testing does not work

Fetching all at once

$$[\![A_1 \to \ldots \to A_n \to \tau]\!]$$

Testing all possible eliminators $\vec{u} = (u_1, \ldots, u_n)$ from $[\![A_1]\!] \times \ldots \times [\![A_n]\!]$

$$(\lambda x.t)s\vec{u}$$

$(\lambda x.\boldsymbol{t'})s\vec{u}$ $\quad$ $\boldsymbol{t[s/x]}\vec{u}$ $\quad$ $(\lambda x.t)\boldsymbol{s'}\vec{u}$ $\quad$ $\boldsymbol{(\lambda x.t)}s\vec{u}'$

(Simple) Reducibility modulo isomorphisms — Solution 2.a

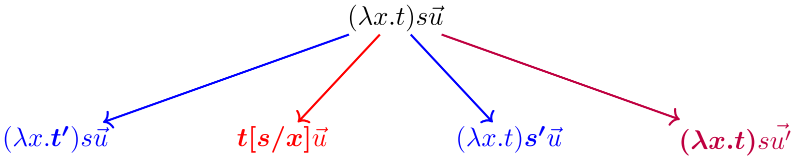
Solving lack of neutrality

- Neutrality is used to test one eliminator at a time
- Constructors commutation breaks neutrality
- "Local" testing does not work

Fetching all at once

$$[\![A_1 \to \ldots \to A_n \to \tau]\!]$$

Testing all possible eliminators $\vec{u} = (u_1, \ldots, u_n)$ from $[\![A_1]\!] \times \ldots \times [\![A_n]\!]$



Remark the proof obligation is now $\mathcal{SN}$ instead of RED

System F (polymorphic λ–calculus) modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \sim (A \to B) \times (A \to C)$$
$$\forall X.(A \times B) \sim (\forall X.A) \times (\forall X.B)$$

System F (polymorphic λ–calculus) modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \sim (A \to B) \times (A \to C)$$
$$\forall X.(A \times B) \sim (\forall X.A) \times (\forall X.B)$$

Typing rules

$$\frac{\Gamma \vdash t : A \qquad A \sim B}{\Gamma \vdash t : B}$$

System F (polymorphic λ–calculus) modulo isomorphisms

Equivalence on types

$$A \to (B \times C) \quad \sim \quad (A \to B) \times (A \to C)$$
$$\forall X.(A \times B) \quad \sim \quad (\forall X.A) \times (\forall X.B)$$

Typing rules

$$\frac{\Gamma \vdash t : A \qquad A \sim B}{\Gamma \vdash t : B}$$

Equivalence on terms

$$\lambda x.\langle t, s\rangle \quad \leftrightarrows \quad \langle \lambda x.t, \lambda x.s\rangle$$
$$\langle t, s\rangle u \quad \leftrightarrows \quad \langle tu, su\rangle$$
$$\Lambda X.\langle t, s\rangle \quad \leftrightarrows \quad \langle \Lambda X.t, \Lambda X.s\rangle$$
$$\langle t, s\rangle A \quad \leftrightarrows \quad \langle tA, sA\rangle$$

System F (polymorphic λ–calculus) modulo isomorphisms

Equivalence on types

$$
\begin{array}{rcl}
A \to (B \times C) & \sim & (A \to B) \times (A \to C) \\
\forall X.(A \times B) & \sim & (\forall X.A) \times (\forall X.B)
\end{array}
$$

Typing rules

$$
\frac{\Gamma \vdash t : A \qquad A \sim B}{\Gamma \vdash t : B}
$$

Equivalence on terms

$$
\begin{array}{rcl}
\lambda x.\langle t, s\rangle & \leftrightarrows & \langle \lambda x.t, \lambda x.s\rangle \\
\langle t, s\rangle u & \leftrightarrows & \langle tu, su\rangle \\
\Lambda X.\langle t, s\rangle & \leftrightarrows & \langle \Lambda X.t, \Lambda X.s\rangle \\
\langle t, s\rangle A & \leftrightarrows & \langle tA, sA\rangle
\end{array}
$$

Reduction

$$
\to \; ::= \; \rightleftarrows^* \circ \hookrightarrow
$$

Reducibility for System F/∼ — Problem 2.a again

Testing/∼

we agreed on testing "globally"

$$(\Lambda X.t)B\vec{u}$$

$$(\Lambda X.\boldsymbol{t'})B\vec{u} \qquad \boldsymbol{t[B/X]}\vec{u} \qquad (\Lambda X.t)B\vec{\boldsymbol{u'}}$$

Reducibility for System F/$\sim$ — Problem 2.a again

Testing/$\sim$ we agreed on testing "globally"

$$(\Lambda X.t)B\vec{u}$$

$$\swarrow \qquad \downarrow \qquad \searrow$$

$$(\Lambda X.\boldsymbol{t'})B\vec{u} \qquad \boldsymbol{t[B/X]}\vec{u} \qquad (\Lambda X.t)B\vec{\boldsymbol{u'}}$$

But...

which are the possible $\vec{u}$ of $[\![X]\!]_\rho$?

- $[\![X]\!]_\rho = \rho(X)$ is any candidate
- we only know CR1, CR2 and CR3
- we don't have enough information to range over $\vec{u}$

Alternative RED for System F: Parigot candidates

Giving structure to candidates

Alternative RED for System F: Parigot candidates

Giving structure to candidates

Family of sets of candidates inductively!

$$\frac{}{\mathcal{SN}_A \in \mathcal{R}_A} \qquad \frac{U \in \mathcal{R}_A \quad V \in \mathcal{R}_B}{U \tilde{\rightarrow} V \in \mathcal{R}_{A \to B}}$$

$$\frac{X \subseteq \mathcal{R}_A}{\bigcap X \in \mathcal{R}_A} \qquad \frac{(U_B \in \mathcal{R}_{A[B/X]})_{B \in \mathcal{K}}}{\tilde{\forall} B.U_B \in \mathcal{R}_{\forall X.A}}$$

Alternative RED for System F: Parigot candidates

Giving structure to candidates

Family of sets of candidates inductively!

$$\frac{}{\mathcal{SN}_A \in \mathcal{R}_A} \qquad \frac{U \in \mathcal{R}_A \quad V \in \mathcal{R}_B}{U \tilde{\rightarrow} V \in \mathcal{R}_{A \rightarrow B}}$$

$$\frac{X \subseteq \mathcal{R}_A}{\bigcap X \in \mathcal{R}_A} \qquad \frac{(U_B \in \mathcal{R}_{A[B/X]})_{B \in \mathcal{K}}}{\tilde{\forall} B.U_B \in \mathcal{R}_{\forall X.A}}$$

Example

$$\frac{\overline{\mathcal{SN}_{A_1} \in \mathcal{R}_{A_1}} \qquad \begin{array}{c}\dfrac{\overline{\mathcal{SN}_{A_n} \in \mathcal{R}_{A_n}} \qquad \overline{\mathcal{SN}_X \in \mathcal{R}_X}}{\mathcal{SN}_{A_n} \tilde{\rightarrow} \mathcal{SN}_X \in \mathcal{R}_{A_n \rightarrow X}} \\ \vdots \end{array}}{\mathcal{SN}_{A_1} \tilde{\rightarrow} \dots \tilde{\rightarrow} \mathcal{SN}_{A_n} \tilde{\rightarrow} \mathcal{SN}_X \in \mathcal{R}_{A_1 \rightarrow \dots \rightarrow A_n \rightarrow X}}$$

Alternative RED for System F: Parigot candidates

Giving structure to candidates

Family of sets of candidates inductively!

$$\frac{}{\mathcal{SN}_A \in \mathcal{R}_A} \qquad \frac{U \in \mathcal{R}_A \quad V \in \mathcal{R}_B}{U \tilde{\to} V \in \mathcal{R}_{A\to B}}$$

$$\frac{X \subseteq \mathcal{R}_A}{\bigcap X \in \mathcal{R}_A} \qquad \frac{(U_B \in \mathcal{R}_{A[B/X]})_{B\in\mathcal{K}}}{\tilde{\forall} B.U_B \in \mathcal{R}_{\forall X.A}}$$

Example

$$\frac{\frac{}{\mathcal{SN}_{A_1} \in \mathcal{R}_{A_1}} \qquad \frac{\vdots \qquad \frac{\mathcal{SN}_{A_n} \in \mathcal{R}_{A_n} \qquad \mathcal{SN}_X \in \mathcal{R}_X}{\mathcal{SN}_{A_n} \tilde{\to} \mathcal{SN}_X \in \mathcal{R}_{A_n \to X}}}{}}{\mathcal{SN}_{A_1} \tilde{\to} \dots \tilde{\to} \mathcal{SN}_{A_n} \tilde{\to} \mathcal{SN}_X \in \mathcal{R}_{A_1\to\dots\to A_n\to X}}$$

Fetching

- the RED–set

$$[\![A \to B]\!]_\rho = \frac{[\![A]\!]_\rho \in \mathcal{R}_A \qquad [\![B]\!]_\rho \in \mathcal{R}_B}{[\![A]\!]_\rho \tilde{\to} [\![B]\!]_\rho \in \mathcal{R}_{A\to B}}$$

- the eliminators $\vec{u}$

$$[\![\overline{\mathcal{SN}_X \in \mathcal{R}_X}]\!]_\rho^\perp = \varepsilon$$

Alternative RED for System F: Parigot candidates

Giving structure to candidates

Family of sets of candidates inductively!

$$\frac{}{\mathcal{SN}_A \in \mathcal{R}_A} \qquad \frac{U \in \mathcal{R}_A \quad V \in \mathcal{R}_B}{U \tilde{\to} V \in \mathcal{R}_{A \to B}}$$

$$\frac{X \subseteq \mathcal{R}_A}{\bigcap X \in \mathcal{R}_A} \qquad \frac{(U_B \in \mathcal{R}_{A[B/X]})_{B \in \mathcal{K}}}{\tilde{\forall} B.U_B \in \mathcal{R}_{\forall X.A}}$$

Example

$$\dfrac{\dfrac{}{\mathcal{SN}_{A_1} \in \mathcal{R}_{A_1}} \qquad \dfrac{\dfrac{\mathcal{SN}_{A_n} \in \mathcal{R}_{A_n} \qquad \mathcal{SN}_X \in \mathcal{R}_X}{\mathcal{SN}_{A_n} \tilde{\to} \mathcal{SN}_X \in \mathcal{R}_{A_n \to X}}}{\vdots}}{\mathcal{SN}_{A_1} \tilde{\to} \dots \tilde{\to} \mathcal{SN}_{A_n} \tilde{\to} \mathcal{SN}_X \in \mathcal{R}_{A_1 \to \dots \to A_n \to X}}$$

Fetching

- the RED–set

$$[\![A \to B]\!]_\rho = \frac{[\![A]\!]_\rho \in \mathcal{R}_A \qquad [\![B]\!]_\rho \in \mathcal{R}_B}{[\![A]\!]_\rho \tilde{\to} [\![B]\!]_\rho \in \mathcal{R}_{A \to B}}$$

- the eliminators $\vec{u}$

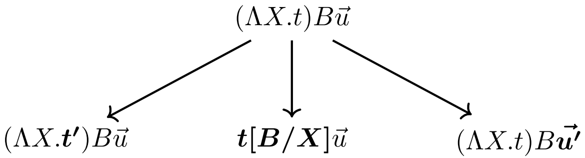
$$[\![\overline{\mathcal{SN}_X \in \mathcal{R}_X}]\!]^\perp_\rho = \varepsilon$$

$$\vdots$$

$$[\![\overline{\mathcal{SN}_A \tilde{\to} \mathcal{SN}_B \in \mathcal{R}_{A \to B}}]\!]^\perp_\rho = (u)_{u \in \mathcal{SN}_A}$$

Reducibility for System F/$\sim$ — Problem 2.a solved

Testing/$\sim$



Now

which are the possible $\vec{u}$ of $[\![X]\!]_\rho$?

- $[\![X]\!]_\rho = \rho(X)$ is any candidate
- we only know ~~CR1, CR2 and CR3~~ **have structure for $\rho(X)$**
- we ~~don't~~ **do!** have enough information
- **range over the eliminators $\vec{u}$ in $\rho(X)$**

RED for System F/$\sim$ Fetching/$\sim$

Problem 1

Against which terms should $\langle \lambda x^A.t, \lambda x^A.s \rangle$ be tested?

RED for System F/$\sim$ — Fetching/$\sim$

Problem 1

Against which terms should $\langle \lambda x^A.t, \lambda x^A.s \rangle$ be tested?

But modulo isomorphisms

- types are part of an equivalence class
- restrictions comes from *all* the class

$$\langle \lambda x^A.t, \lambda x^A.s \rangle$$

$$(A \to B) \times (A \to C) \qquad A \to (B \times C)$$

RED for System F/$\sim$ — Fetching/$\sim$

Problem 1

Against which terms should $\langle \lambda x^A.t, \lambda x^A.s \rangle$ be tested?

But modulo isomorphisms

- types are part of an equivalence class
- restrictions comes from *all* the class

$$\langle \lambda x^A.t, \lambda x^A.s \rangle$$

$$(A \to B) \times (A \to C) \qquad A \to (B \times C)$$

Recall that

- $[\![\cdot]\!]_.$ follows types fetching restrictions
- by induction $m(A)$
- $m(A)$ is a stable measure on types

RED for System F/$\sim$ — Fetching/$\sim$

Problem 1

Against which terms should $\langle \lambda x^A.t, \lambda x^A.s\rangle$ be tested?

But modulo isomorphisms

- types are part of an equivalence class
- restrictions comes from *all* the class

$$\langle \lambda x^A.t, \lambda x^A.s\rangle$$

$$(A \to B)\times(A\to C) \qquad A \to (B\times C)$$

Recall that

- $[\![\cdot]\!].$ follows types fetching restrictions
- by induction $m(A)$
- $m(A)$ is a stable measure on types

$$\begin{aligned}
[\![A]\!]_\rho \quad = \quad & \textstyle\bigcap_{A_1\times A_2\sim A}\{\,[\![A_1]\!]_\rho \,\tilde{\times}\, [\![A_2]\!]_\rho\,\}\\
\cap \quad & \textstyle\bigcap_{A_1\to A_2\sim A}\{\,[\![A_1]\!]_\rho \,\tilde{\to}\, [\![A_2]\!]_\rho\,\}\\
\cap \quad & \textstyle\bigcap_{\forall X.A'\sim A}\{\,\tilde{\forall} B. \cap\{\,[\![A']\!]_{\rho\cdot[\mathcal{C}_B/X]}\,\}\,\}
\end{aligned}$$

RED for System F/$\sim$ — Parigot candidates/$\sim$

Changes in the family $\mathcal{R}_*$ due to isomorphisms

RED for System F/$\sim$ Parigot candidates/$\sim$

Changes in the family $\mathcal{R}_*$ due to isomorphisms

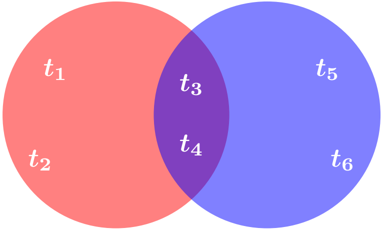
Family of sets of candidates inductively!

$$\frac{}{\mathcal{SN}_A \in \mathcal{R}_A} \qquad \frac{U \in \mathcal{R}_A \quad V \in \mathcal{R}_B}{U \tilde{\to} V \in \mathcal{R}_{A\to B}}$$

$$\frac{X \subseteq \mathcal{R}_A}{\bigcap X \in \mathcal{R}_A} \qquad \frac{(U_B \in \mathcal{R}_{A[B/X]})_{B\in\mathcal{K}}}{\tilde{\forall} B.U_B \in \mathcal{R}_{\forall X.A}}$$

$$\frac{F \in \mathcal{R}_A \quad G \in \mathcal{R}_B \quad A \equiv B}{F \cap G \in \mathcal{R}_A}$$

$A \to (B \times C)$ $\qquad$ $(A \to B) \times (A \to C)$



RED for System F/$\sim$ Testing/$\sim$

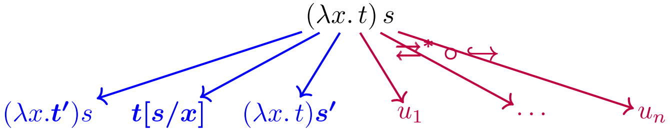
Problem 2.b

What are the one–step reducts of a term?

RED for System F/∼ — Testing/∼

Problem 2.b

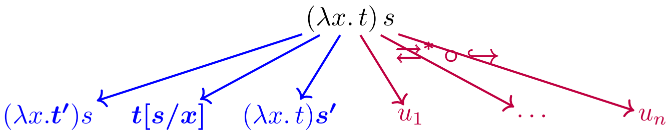
What are the one–step reducts of a term?



RED for System F/$\sim$ — Testing/$\sim$

Problem 2.b

What are the one–step reducts of a term?



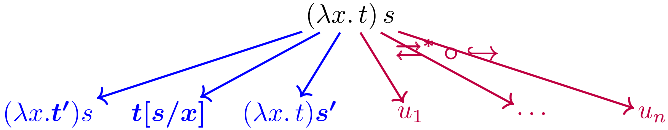
Solution

1. characterize classes of terms

RED for System F/$\sim$ — Testing/$\sim$

Problem 2.b

What are the one–step reducts of a term?



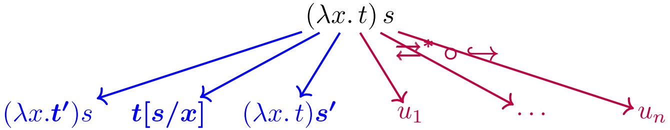
Solution

1. characterize classes of terms
2. look at the one–step redex of each shape

RED for System F/∼ — Testing/∼

Problem 2.b

What are the one–step reducts of a term?

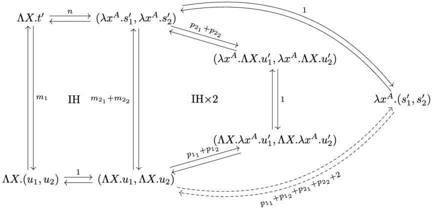


Solution

1. characterize classes of terms
2. look at the one–step redex of each shape

Lemma 3.2 (The class of type abstractions). *If $\Lambda X.t' \rightleftarrows^n s$, then s is equal to:*

1. *$\Lambda X.s'$ with $t' \rightleftarrows^m s'$ and $m \leq n$*
2. *$\lambda x^A.s'$ with $t' \rightleftarrows^{m_1} \lambda x^A.r$, $s' \rightleftarrows^{m_2} \Lambda X.r$, $m_1+1+m_2 \leq n$, and $X \notin FV(A)$*
3. *$\langle s'_1, s'_2\rangle$ with $t' \rightleftarrows^{m_1} \langle r_1, r_2\rangle$, $s'_i \rightleftarrows^{m_{2_i}} \Lambda X.r_i$, and $m_1 + 1 + m_{2_1} + m_{2_2} \leq n$*
4. *$\pi_{\forall X.A}s'$ with $t' \rightleftarrows^{m_1} \pi_A r$, $s' \rightleftarrows^{m_2} \Lambda X.r$, and $m_1 + 1 + m_2 \leq n$*



Conclusions

- We need to prove $\mathcal{SN}$ in a System F modulo isomorphisms
- The calculus has no neutrality
- Parigot's approach to reducibility does not use neutrality
- We are adapting Parigot's technique to modulo isomorphisms by
  1. relating candidates of isomorphic types
  2. fetching restrictions from all the equivalents
  3. characterizing the one–step reducts of all term–classes