Solvability for Generalized Applications

Delia Kesner and Loïc Peyrot

Université Paris Cité, IRIF, CNRS, France

FSCD, August 3rd, 2022

CBN and CBV generalized applications

- Generalized applications (GA) = a Curry-Howard interpretation of natural deduction with generalized elimination rules.
- First call-by-name (CBN) calculus ΛJ by Joachimski & Matthes (2000).
- First call-by-value (CBV) calculus ΛJ_v by Espírito Santo (2020).
- We use new variants with distance (λJ_n and λJ_v).

The Syntax of Terms

$$\frac{}{\Gamma, A \vdash A}\ \mathsf{AX} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to_i$$

$$\frac{\Gamma \vdash A \to B \qquad \Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma \vdash C}\ \to_e$$

The Syntax of Terms

$$\frac{}{\Gamma, \quad A \vdash \quad A}\ \mathsf{AX} \qquad \frac{\Gamma, \quad A \vdash \quad B}{\Gamma \vdash \quad A \to B}\ \to_i$$

$$\frac{\Gamma \vdash \quad A \to B \qquad \Gamma \vdash \quad A \qquad \Gamma, \quad B \vdash \quad C}{\Gamma \vdash \quad C}\ \to_e$$

The Syntax of Terms

$$\frac{}{\Gamma, x : A \vdash x : A}\ \mathsf{AX} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ \to_i$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A \qquad \Gamma, y : B \vdash r : C}{\Gamma \vdash t(u, y.r) : C}\ \to_e$$

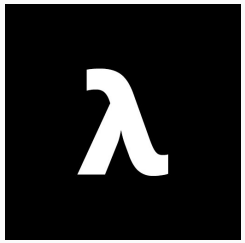
$$\begin{array}{lrl} \textsf{(Terms)} & t, u, r & ::= v \mid t(u, y.r) \\ \textsf{(Values)} & v & ::= x \mid \lambda x.t \end{array}$$

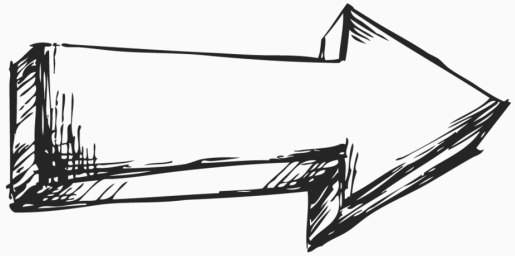
Intuition

$t(u, y.r) \rightsquigarrow \mathsf{let}\ y = tu\ \mathsf{in}\ r$

What is the place of generalized applications in the theory of programming languages?

Towards implementation





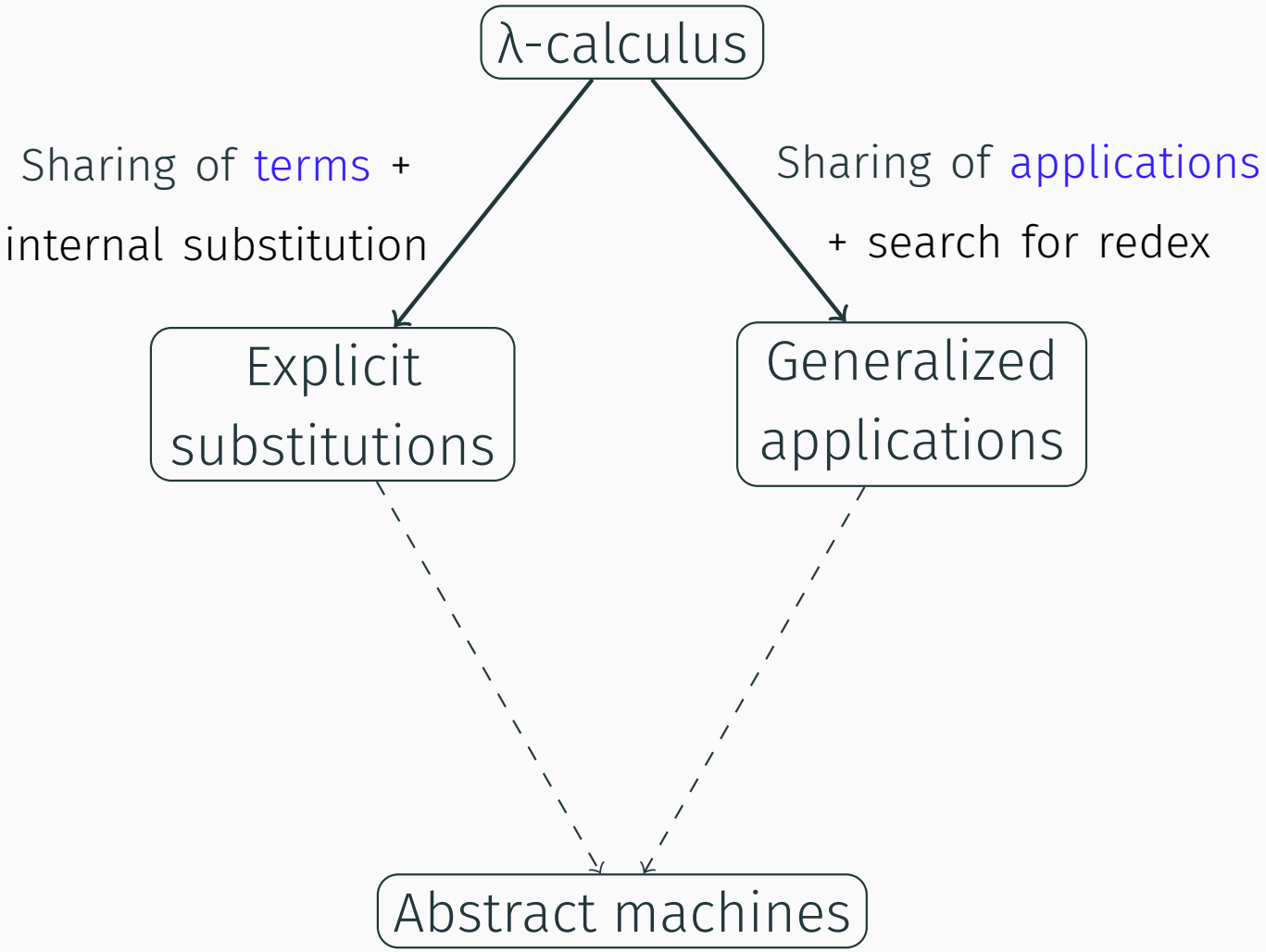
The λ-calculus

- A minimal syntax: $x \mid \lambda x.t \mid tu$.
- A single computational rule: $(\lambda x.t)u \to_\beta t\{x/u\}$.
- Meta-level substitution.
- Nondeterminism.

Abstract machines

- Internal management of substitution (environment).
- Explicit search for a redex (stack).
- Determinism.

Extending the λ-calculus



Calculi with explicit substitutions

Explicit substitutions offer:

- Sharing of terms: $t[x/u] \rightsquigarrow$ let x = u in t (environment).

Calculi with explicit substitutions

Explicit substitutions offer:

- Sharing of terms: $t[x/u] \rightsquigarrow$ let x = u in t (environment).
- An internal treatment of substitution.

$$\left.\begin{array}{lll} (\lambda x.t)u & \to_{\mathrm{B}} & t[x/u] \\ t[x/u] & \to_{\mathrm{sub}} & \cdots \end{array}\right\} \text{Two phases of computation}$$

Characteristics of generalized applications

Features of generalized applications:

- Sharing of applications: $t(u, y.r) \rightsquigarrow$ let y = tu in r.

Characteristics of generalized applications

Features of generalized applications:

- Sharing of applications: $t(u, y.r) \rightsquigarrow$ let y = tu in r.
- Unique computational rule (external substitution):

$$(\lambda x.t)(u, y.r) \to_\beta r\{y/t\{x/u\}\}$$

Characteristics of generalized applications

Features of generalized applications:

- Sharing of applications: $t(u, y.r) \rightsquigarrow$ let y = tu in r.
- Unique computational rule (external substitution):

$$(\lambda x.t)(u, y.r) \to_\beta r\{y/t\{x/u\}\}$$

- Explicit search for a redex (stack):

$$t(u, x.r)(u', y.r') \to_\pi t(u, x.r(u', y.r'))$$

Generalized applications encode the stack

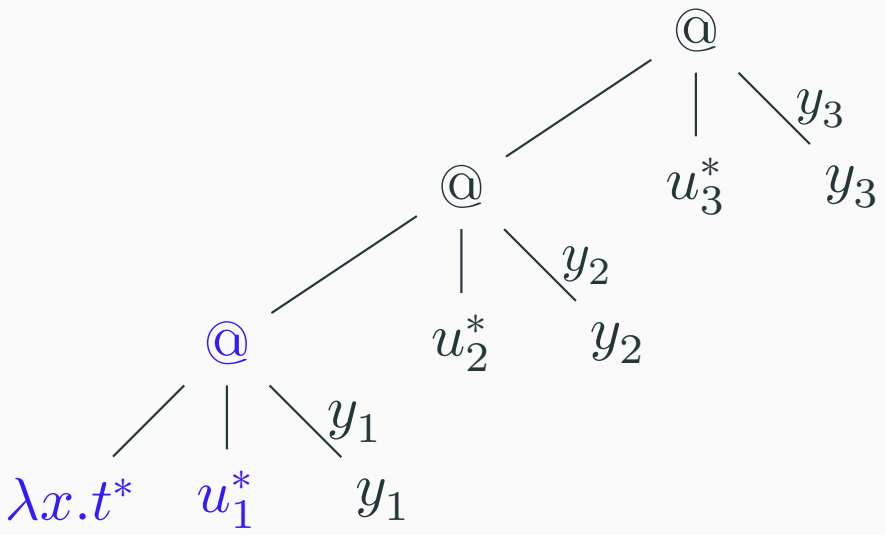
Example

$$((\lambda x.t)u_1u_2u_3)^* = (\lambda x.t^*)(u_1^*, y_1.y_1)(u_2^*, y_2.y_2)(u_3^*, y_3.y_3)$$

Generalized applications encode the stack

Example

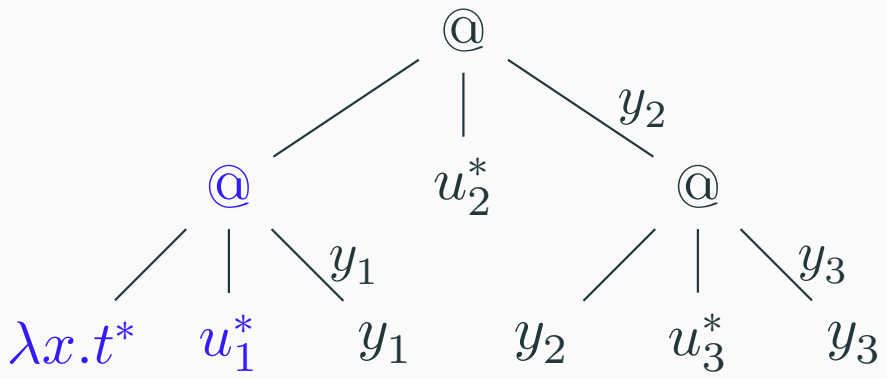
$$(\lambda x.t^*)(u_1^*, y_1.y_1)(u_2^*, y_2.y_2)(u_3^*, y_3.y_3)$$



Generalized applications encode the stack

Example

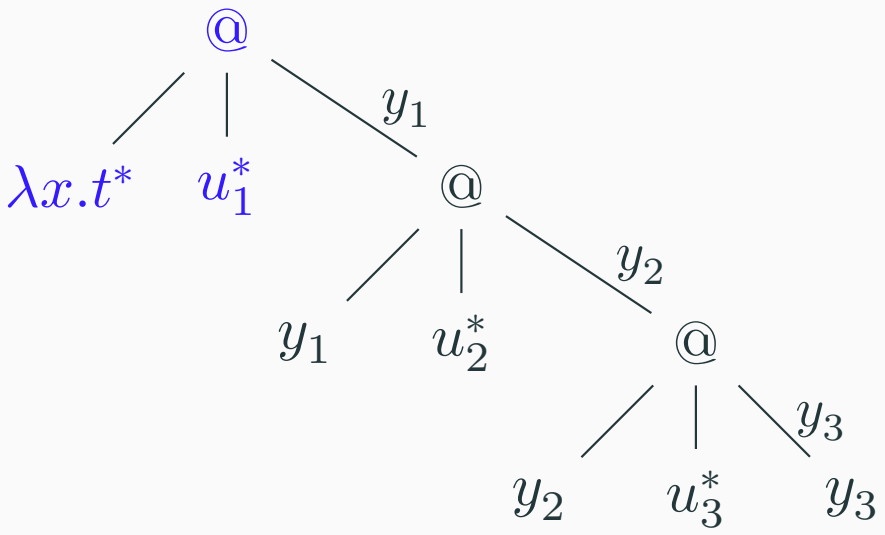
$$(\lambda x.t^*)(u_1^*, y_1.y_1)(u_2^*, y_2.y_2)(u_3^*, y_3.y_3)$$
$$\to_\pi (\lambda x.t^*)(u_1^*, y_1.y_1)(u_2, y_2.y_2(u_3, y_3.y_3))$$



Generalized applications encode the stack

Example

$$(\lambda x.t^*)(u_1^*, y_1.y_1)(u_2^*, y_2.y_2)(u_3^*, y_3.y_3)$$
$$\to_\pi (\lambda x.t^*)(u_1^*, y_1.y_1)(u_2, y_2.y_2(u_3, y_3.y_3))$$
$$\to_\pi (\lambda x.t^*)({u_1}^*, y_1.y_1(u_2^*, y_2.y_2(u_3^*, y_3.y_3)))$$



The original calculi

$$\left.\begin{array}{rcl} (\lambda x.t)(u, y.r) & \to_\beta & r\{y/t\{x/u\}\} \\ t(u, x.r)(u', y.r') & \to_\pi & t(u, x.r(u', y.r')) \end{array}\right\} \Lambda J$$

The original calculi

$$\left.\begin{array}{rcl} (\lambda x.t)(u, y.r) & \to_{\beta} & r\{y/t\{x/u\}\} \\ t(u, x.r)(u', y.r') & \to_{\pi} & t(u, x.r(u', y.r')) \end{array}\right\} \Lambda J$$

$$\left.\begin{array}{rcl} (\lambda x.t)(u, y.r) & \to_{\beta \mathrm{v}} & r\{y \backslash\backslash t\{x \backslash\backslash u\}\} \\ t(u, x.r)(u', y.r') & \to_{\pi} & t(u, x.r(u', y.r')) \end{array}\right\} \Lambda J_{v}$$

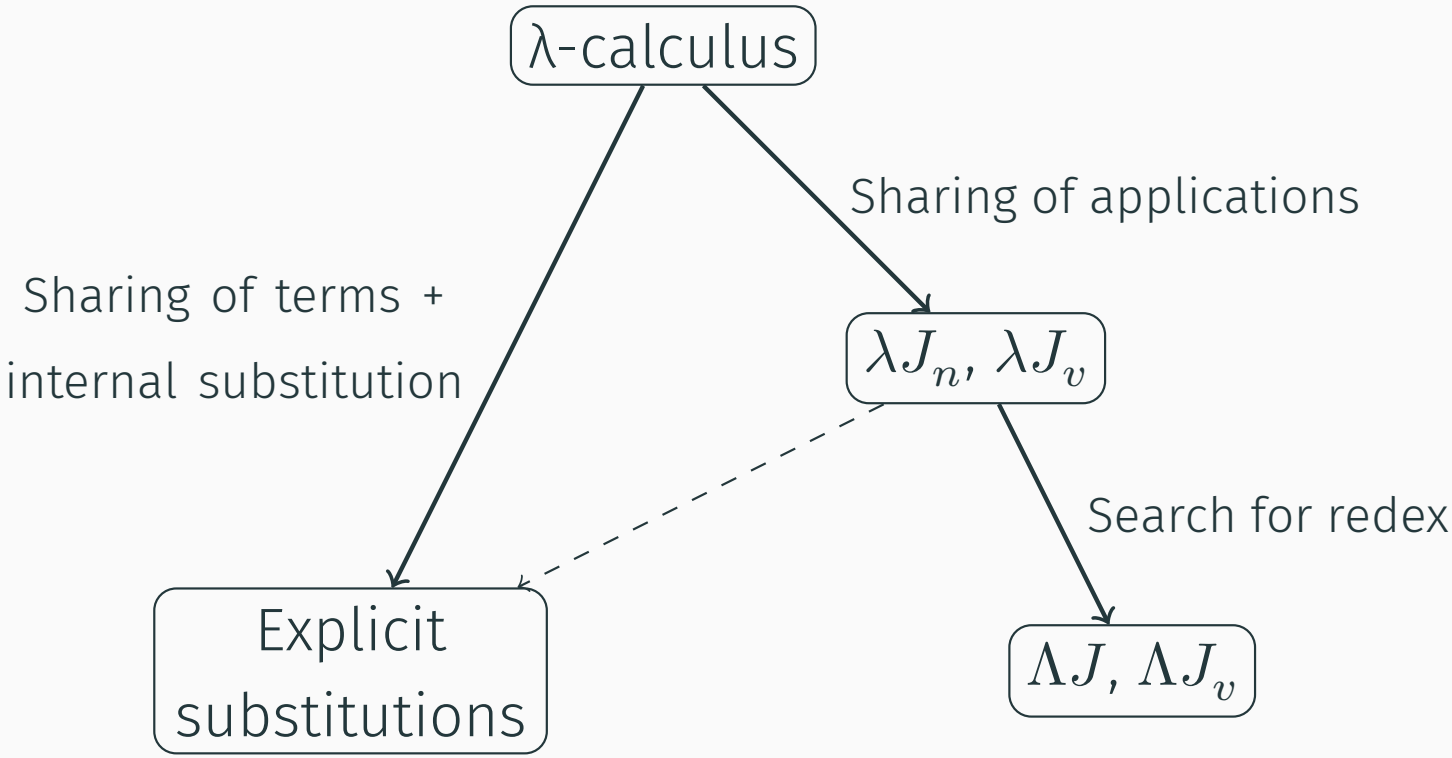
The original calculi

$$\left.\begin{array}{rcl} (\lambda x.t)(u, y.r) & \to_{\beta} & r\{y/t\{x/u\}\} \\ t(u, x.r)(u', y.r') & \to_{\pi} & t(u, x.r(u', y.r')) \end{array}\right\} \Lambda J$$

$$\left.\begin{array}{rcl} (\lambda x.t)(u, y.r) & \to_{\beta \mathrm{v}} & r\{y \backslash\backslash t\{x \backslash\backslash u\}\} \\ t(u, x.r)(u', y.r') & \to_{\pi} & t(u, x.r(u', y.r')) \end{array}\right\} \Lambda J_v$$

What about λJ_n and λJ_v?

Different levels of abstraction



Stuck reductions

The variants do not use a permutation rule.

But β-reduction alone is not sufficient to reach normal forms.

$$z(u_1, y_1.\lambda x.x)(u_2, y_2.y_2) \not\twoheadrightarrow_\beta$$

Stuck reductions

π-permutations can be used to unblock beta-reduction.

$$z(u_1, y_1.\lambda x.x)(u_2, y_2.y_2) \to_\pi z(u_1, y_1.(\lambda x.x)(u_2, y_2.y_2)) \to_\beta z(u_1, y_1.u_2)$$

Distance integrates necessary permutations into computation

Idea: distance

$$\beta + \text{needed permutations} = \mathrm{d}\beta$$

Distance integrates necessary permutations into computation

Idea: distance

$$\beta + \text{needed permutations} = \mathrm{d}\beta$$

Examples

- $x_1(x_2, y_1.y_1)(x_3, y_2.y_2) \nrightarrow$
- $z(u_1, y_1.\lambda x.x)(u_2, y_2.y_2) \to_{\mathrm{d}\beta} z(u_1, y_1.u_2)$

The distant calculi

$$\left.\begin{array}{lll} \mathtt{D}\langle\lambda x.t\rangle(u, y.r) & \to_{\mathrm{d}\beta} & r\{y/\mathtt{D}\langle t\{x/u\}\rangle\} \end{array}\right\}\lambda J_n$$

$$\left.\begin{array}{lll} \mathtt{D}\langle\lambda x.t\rangle(u, y.r) & \to_{\mathrm{d}\beta_{\mathrm{v}}} & r\{y\backslash\backslash\mathtt{D}\langle t\{x\backslash\backslash u\}\rangle\} \end{array}\right\}\lambda J_v$$

$\mathtt{D}$ = a series of generalized applications.

Goal of the paper

We show some benefits of GA with the case study of a crucial semantical property: solvability.

Towards solvability

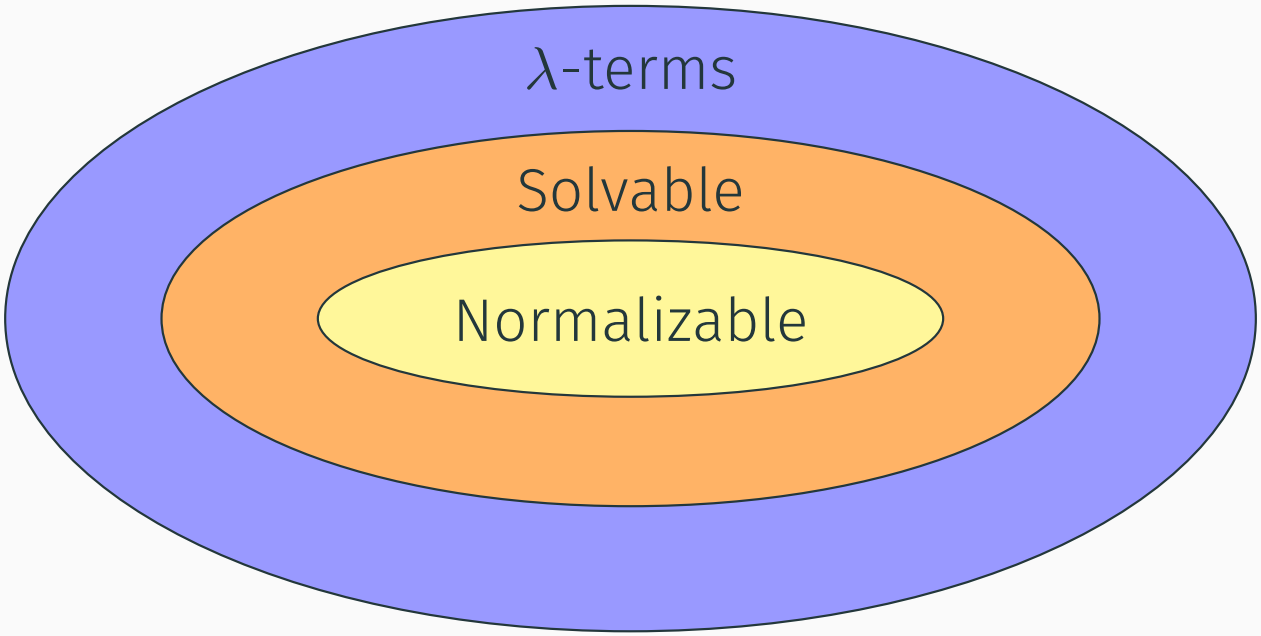
Goal: define meaningless terms.

- A first approach: meaningless = non normalizable.
  Collapse ✗
- The solution: meaningless = unsolvable. ✓

Solvability

Definition (Solvable term)

A term t is solvable *iff* there is a *head context* H such that $\mathsf{H}\langle t\rangle \to^* \lambda x.x.$



Operational characterization of CBN solvability

CBN solvability in the λ-calculus has an operational characterization:

Theorem

t is CBN solvable $\iff$ t has a head normal form
$\iff$ t is head-normalizable.

CBN solvability in generalized applications

Head normalization extends naturally to generalized applications.

Definition

A term t is CBN solvable *iff* there is a head context H such that $\mathsf{H}\langle t\rangle \to_{\mathrm{d}\beta}^{*} \mathsf{D}\langle \lambda x.x\rangle$.

D = list of "garbage" applications $t(u, x.r)$ where $x \notin \mathrm{fv}(r)$.

Example

$\Omega(x, y.\lambda x.x)$ is solvable.

A problem with CBV

Plotkin's original CBV is defective due to premature normal forms.

$$(\lambda y.\lambda x.xx)(zz)(\lambda x.xx) \not\mapsto_{\beta \mathrm{v}}$$

Characterizing CBV solvability

Characterizations were given in variants of the λ-calculus with:

Permutations (Carraro & Guerrieri, 2014)

$$(\lambda y.\lambda x.xx)(zz)(\lambda x.xx) \to_{\sigma_1} (\lambda y.(\lambda x.xx)(\lambda x.xx))(zz) \not\to_{\beta \mathrm{v}}$$

Explicit substitutions (Accattoli & Paolini, 2012)

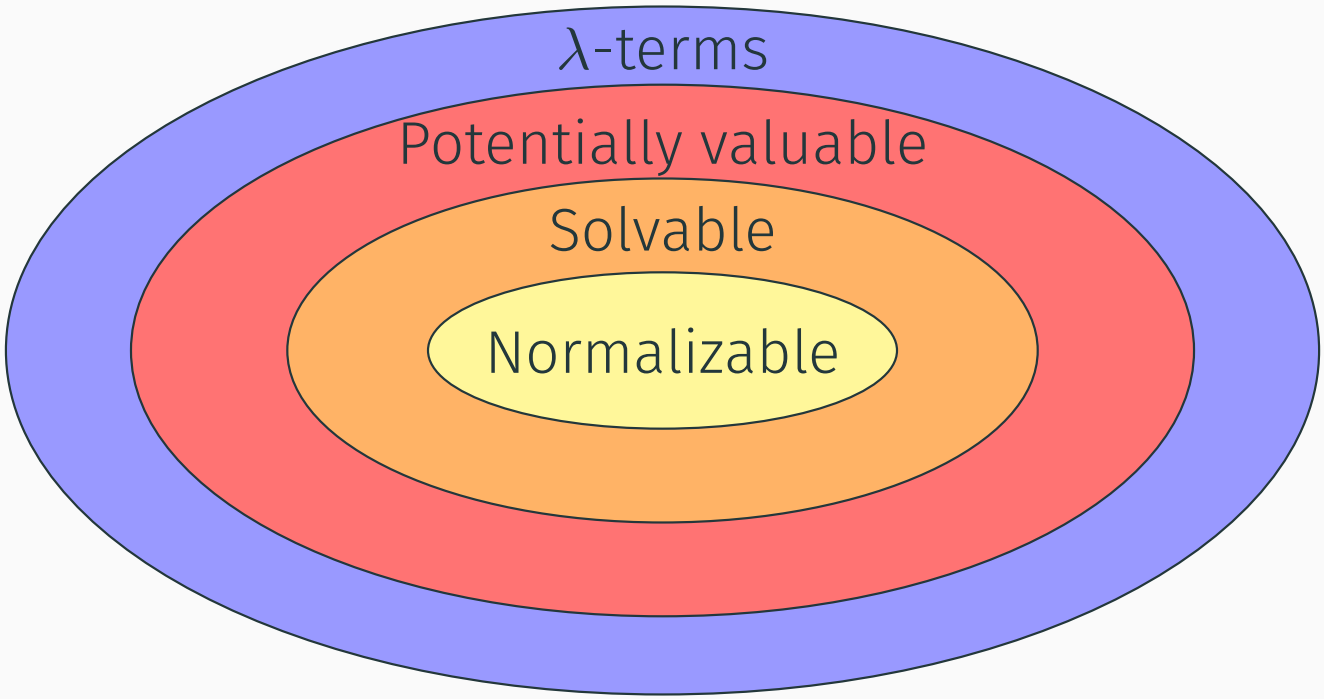
$$(\lambda y.\lambda x.xx)(zz)(\lambda x.xx) \to^* ((\lambda x.xx)(\lambda x.xx))[y/zz] \not\to^*$$

Potential valuability

The characterizations of CBV solvability rely on potential valuability.

Definition (Potential valuability for GA)

A term t is potentially valuable *iff* there is a *distant context* D and a value v such that $\mathsf{D}\langle t\rangle \to^*_{\mathrm{d}\beta_{\mathrm{v}}} v$.



CBV solvability for generalized applications

Definition

A term t is CBV solvable *iff* there is a head context H such that $\mathsf{H}\langle t\rangle \to^*_{\mathrm{d}\beta_{\mathrm{v}}} \lambda x.x$.

Example

In CBV, the term $x(\Omega, y.\lambda x.x)$ is not solvable, because the argument Ω cannot be erased.

CBV solvability with generalized applications

Operational characterizations of CBV solvability are natural with generalized applications.

- The reduction relation refines head reduction.
- It is based on the operational characterization of potential valuability.

Particularities of the operational characterizations

Operational characterizations:

- Are provided for distant and non-distant calculi.
- Have simple normal forms
  (of the shape $\lambda\vec{x}.y(u_1, z_1.r_1) \dots (u_n, z_n.r_n)$).

A characterization of CBV solvability is possible without explicit substitutions and permutations rules.

Different notions of solvability?

$\mathrm{solv}(\lambda J_n)$ $\mathrm{solv}(\Lambda J)$ $\mathrm{solv}(\lambda J_v)$ $\mathrm{solv}(\Lambda J_v)$

CBN solvability (λ-calculus, explicit substitutions)

CBV solvability (λ-calculus, λ with permutations, explicit substitutions)

Logical models

To relate the different notions of solvability semantically, we provide non-idempotent intersection type systems.

Theorem (Logical characterization)

t normalizable $\iff$ t typable $\iff$ t solvable.

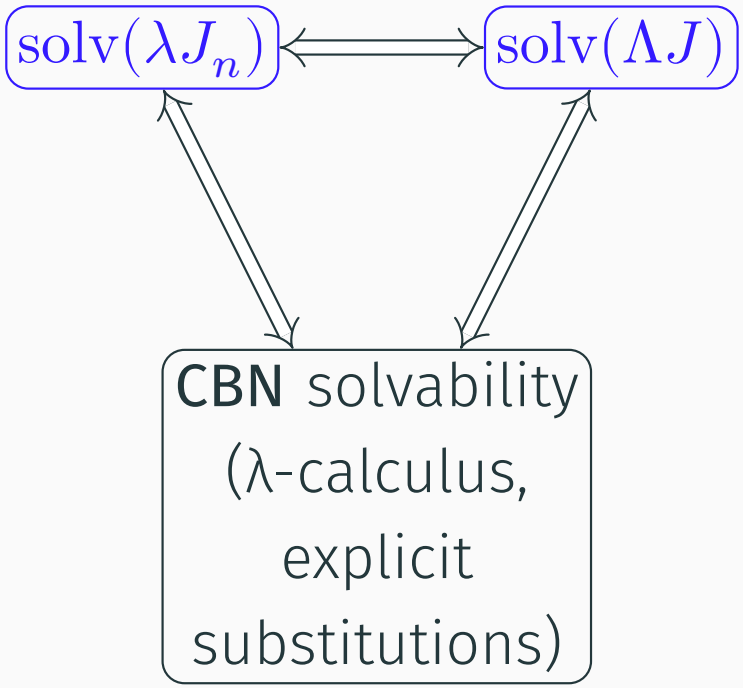
We show preservation of solvability simply by proving preservation of typability.

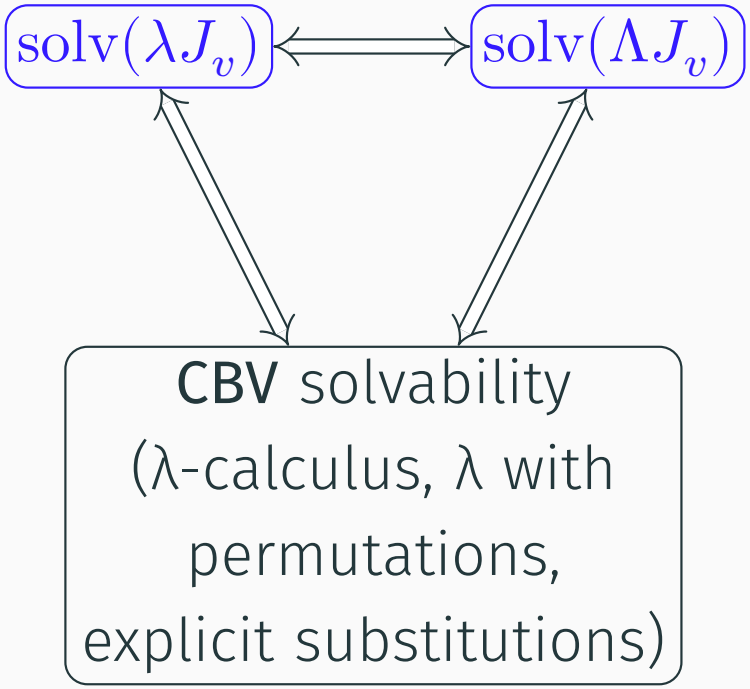
Why non-idempotence?

Non-idempotence brings a quantitative flavor to intersection types, with:

- Upper bounds on the number of reduction steps and size of normal form, and
- Combinatorial proofs of normalization.

Two unique notions of solvability





Future works

- A partial genericity lemma.
- Investigate call-by-value solvability and potential valuability semantically.
- Describe the transformation from generalized applications to abstract machines.
- Exact bounds with tight types.

Thank you!