Explicit Substitution Calculi

Agenda for Today

1. Lambda-Calculus - A Brief Reminder
2. From Lambda-Calculus to Multiplicative Exponential Linear Logic
3. Lambda Calculi with Explicit Substitutions

Agenda

The λ-Calculus

$$\begin{array}{ll} \textbf{Terms}: & t, u ::= x \mid \lambda x.t \mid tu \\ \textbf{Contexts}: & \mathsf{C} ::= \Box \mid \lambda x.\mathsf{C} \mid \mathsf{C}t \mid t\mathsf{C} \end{array}$$

- We use $\mathbf{fv}(t)$ (resp. $\mathbf{bv}(t)$) to denote the set of free (resp. bound) variables of t.
- We work modulo alpha-conversion (renaming of bound variables) generated by the equation: $\lambda x.t \equiv \lambda y.t\{x\backslash y\}$ where y is fresh.
- $\mathsf{C}\langle t\rangle$ denotes the context C where the hole $\Box$ has been replaced by t. Possible capture of free variables, e.g. $(\lambda x.\Box)\langle x\rangle = \lambda x.x$.
- $\mathsf{C}\langle\!\langle t\rangle\!\rangle$ denotes the context C where the hole $\Box$ has been replaced by t without capturing any free variables. e.g. $(\lambda x.\Box)\langle\!\langle y\rangle\!\rangle = \lambda x.y$.

The λ-Calculus - Operational Semantics

- Only one rewriting rule:

$$(\lambda x.t)\ u \mapsto_\beta\ t\{x\backslash u\}$$

  where $t\{x\backslash u\}$ is a meta-level operation simultaneously replacing all the free occurrences of x in t by u.
- The reduction relation $\to_\beta$ is generated by the closure of $\mapsto_\beta$ by all contexts $\mathtt{C}$.

Alternative Definition:

$$\frac{}{(\lambda x.t)\ u \to_\beta\ t\{x\backslash u\}} \qquad \frac{t \to_\beta\ u}{\lambda x.t \to_\beta\ \lambda x.u} \qquad \frac{t \to_\beta\ u}{tv \to_\beta\ uv} \qquad \frac{t \to_\beta\ u}{vt \to_\beta\ vu}$$

Both definitions are modulo alpha-conversion.

Lambda-Calculus is Turing complete.

Examples

Erasing case: $(\lambda y.x)z \to_\beta x$

Duplicating case: $(\lambda y.yy)z \to_\beta zz$

Non-terminating case: $(\lambda y.yy)(\lambda y.yy) \to_\beta (\lambda y.yy)(\lambda y.yy) \to_\beta \ldots$

Contextual case: $\lambda z.(\lambda x.y)(\mathtt{II}) \to_\beta \lambda z.(\lambda x.y)\mathtt{I} \to_\beta \lambda z.y$

where $\mathtt{I} = \lambda z.z$.

The λ-Calculus - Simple Types

Types: $A ::= \iota \mid A \to B$

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i}\ \text{(axiom)}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to \text{ intro}) \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B}\ (\to \text{ elim})$$

- The axiom uses weakening.
- The application rule is additive.
- We denote by $\Gamma \vdash_\lambda t : A$ the corresponding derivability relation.
- A term t is (simply) typable if there exists a derivation $\Gamma \vdash_\lambda t : A$.

Some Salient Remarks about Simply Typed Lambda-Calculus

- Typical Curry-Howard interpretation.
- Provides only monomorphic information.
- Lack expressivity power but typability is decidable.
- Typability IMPLIES Strong Normalization, but the converse does not hold.

E.g. the term $\lambda x.xx$ is not typable

The λ-Calculus - Some Typical Meta-Properties

Theorem (Confluence)

If $t \to_\beta^* u$ and $t \to_\beta^* v$, then there is t' such that $u \to_\beta^* t'$ and $v \to_\beta^* t'$.

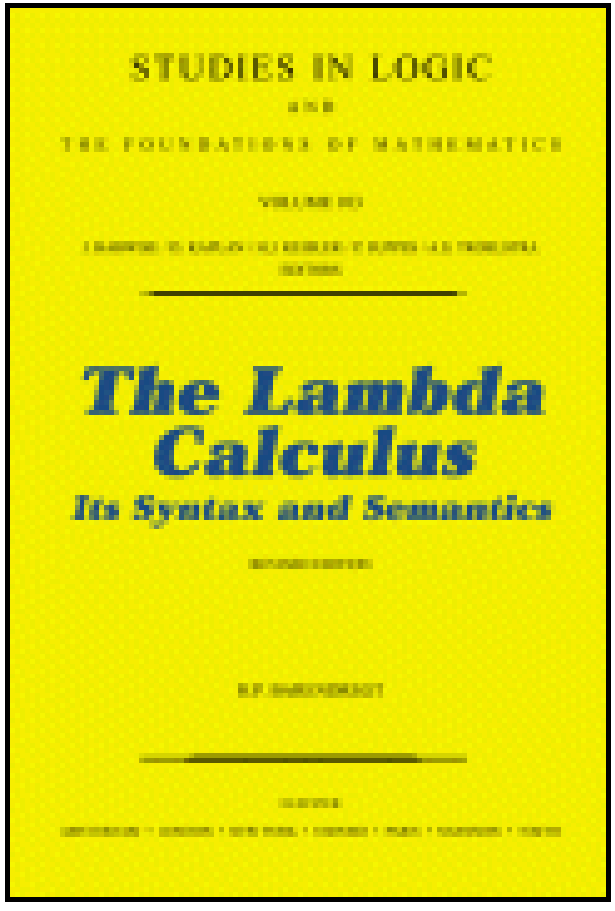
Theorem (Subject Reduction for Simple Types)

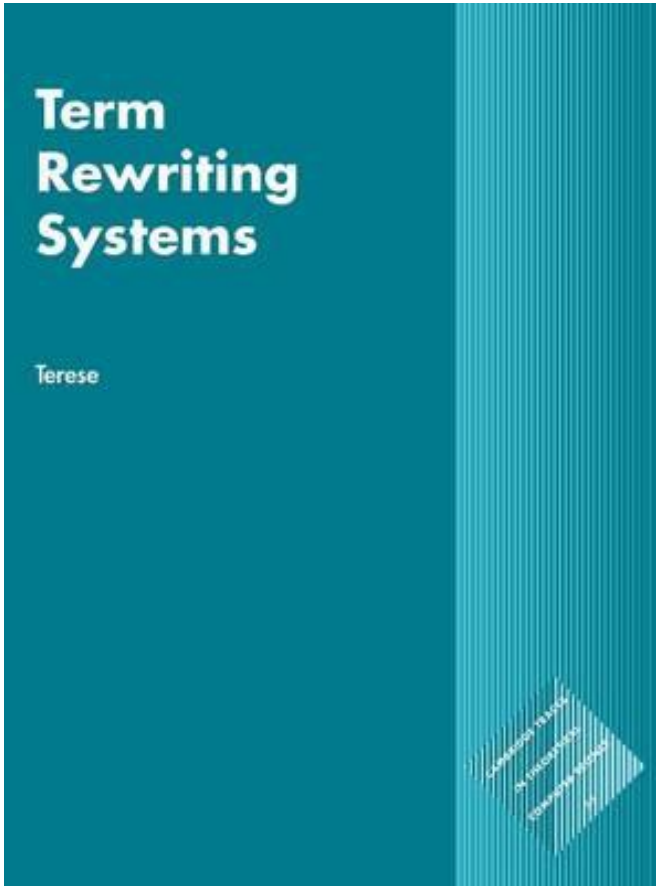
If $\Gamma \vdash_\lambda t : A$ and $t \to_\beta t'$, then $\Gamma \vdash_\lambda t' : A$.

Theorem (Strong Normalization)

If $\Gamma \vdash_\lambda t : A$, then $t \in SN(\beta)$, i.e. there is no infinite β-reduction sequence starting at t (every β-reduction sequence starting at t terminates).

To Go Further, Recommended Readings





Agenda

General Picture

λ-calculus $\Rightarrow$ intermediate language $\Rightarrow$ MELL Proof-Nets

- MELL is an extension of Linear Logic being able to capture the lambda-calculus.
- Weakening and contraction are handled in an explicit way.
- Interesting expressive power.

The intermediate language:

- Lambda-terms with explicit substitutions + axioms + reduction rules
- Explicit management of resources (erasure and duplication)
- Different alternatives: Λ-calculus, Linear Substitution Calculus, $\lambda\mathtt{ex}$-calculus, $\lambda\mathtt{pn}$-calculus, $\lambda\mathtt{lxr}$-calculus, ...

Agenda

A First Refinement: the Λ-Calculus

Syntax:

Terms	t, u, v	$::=$	$x \mid \lambda x.t \mid tu \mid t[x\backslash u]$
List Contexts	$\mathtt{L}$	$::=$	$\Box \mid \mathtt{L}[x\backslash t]$
Term Contexts	$\mathtt{C}$	$::=$	$\Box \mid \lambda x.\mathtt{C} \mid \mathtt{C}t \mid t\mathtt{C} \mid \mathtt{C}[x\backslash t] \mid t[x\backslash \mathtt{C}]$

- Free and bound variables (written $\mathbf{fv}(\_)$ and $\mathbf{bv}(\_)$ resp.)
- Alpha-conversion:

$$\begin{array}{lcll} \lambda x.t & \equiv & \lambda y.t\{x\backslash y\} & y \text{ fresh} \\ t[x\backslash u] & \equiv & t\{x\backslash y\}[y\backslash u] & y \text{ fresh} \end{array}$$

- Pure terms: terms without explicit substitutions.
- Notations: $\mathtt{C}\langle t\rangle$ (possible capture) and $\mathtt{C}\langle\!\langle t\rangle\!\rangle$ (capture-free).

The Λ-calculus - Semantics

Equations and Rules:

$$t[x\backslash v][y\backslash u] \quad \equiv \quad t[y\backslash u][x\backslash v] \quad \text{if } x \notin \mathbf{fv}(u) \ \& \ y \notin \mathbf{fv}(v)$$

$$\mathrm{L}\langle \lambda x.t\rangle u \quad \mapsto_{\mathtt{dB}} \quad \mathrm{L}\langle t[x\backslash u]\rangle \quad \text{if no capture of free variables}$$

$$t[x\backslash u] \quad \mapsto_{\mathtt{subs}} \quad t\{x\backslash u\}$$

- The context L allows reduction at a distance (as in PN).
- Capture of variables is avoided by using alpha-conversion.
- $\mathtt{dB}$ (resp. $\mathtt{subs}$) corresponds to multiplicative (resp. exponential) steps in MELL.
- $\rightarrow$ is the closure by all the contexts of the rewriting rules $\{\mathtt{dB}, \mathtt{subs}\}$.
- $\simeq$ is the reflexive, symmetric, transitive, closed by proof-net contexts relation on proof-nets generated by alpha-conversion and the equation $\equiv$.
- The reduction relation $\rightarrow_\Lambda$ is defined as follows:

$$t \rightarrow_\Lambda t' \text{ iff } \exists t_1, t_2 \text{ such that } t \simeq t_1 \rightarrow t_2 \simeq t'$$

- Example:
$(\lambda z.\lambda x.xx)wy \rightarrow_{\mathtt{dB}} (\lambda x.xx)[z\backslash w]y \rightarrow_{\mathtt{dB}} (xx)[x\backslash y][z\backslash w] \rightarrow_{\mathtt{subs}} (yy)[z\backslash w] \rightarrow_{\mathtt{subs}} yy.$

Implementing Substitution

At least two ways to implement substitution $t\{x\backslash u\}$:

- By induction on the structure of terms.
- By induction on the number of free occurrences of x in t.

Reasoning by Induction on the Structure of Terms

The $\lambda\mathtt{ex}$-calculus

$t[y\backslash u][x\backslash v]$	$\equiv$	$t[x\backslash v][y\backslash u]$	if $x \notin \mathbf{fv}(u)$ & $y \notin \mathbf{fv}(v)$
$\mathrm{L}\langle\lambda x.t\rangle v$	$\mapsto_{\mathtt{dB}}$	$\mathrm{L}\langle t[x\backslash v]\rangle$	if no capture of free variables
$y[x\backslash v]$	$\mapsto_{\mathtt{ex}}$	y	if $x \neq y$
$x[x\backslash v]$	$\mapsto_{\mathtt{ex}}$	v	
$(t\ u)[x\backslash v]$	$\mapsto_{\mathtt{ex}}$	$t[x\backslash v]u[x\backslash v]$	
$(\lambda y.t)[x\backslash v]$	$\mapsto_{\mathtt{ex}}$	$\lambda y.t[x\backslash v]$	if $x \neq y$ & no capture of free variables
$t[y\backslash u][x\backslash v]$	$\mapsto_{\mathtt{ex}}$	$t[x\backslash v][y\backslash u[x\backslash v]]$	if $x \in \mathbf{fv}(u)$ & and no capture of free variables

- A congruence is generated by the equation $\equiv$ and alpha-conversion. The reduction relation is gerated by the rewriting rules modulo this congruence (as before).
- Example: $(\lambda x.xx)y \to_{\mathtt{dB}} (xx)[x\backslash y] \to_{\mathtt{ex}} x[x\backslash y]x[x\backslash y] \to_{\mathtt{ex}} yx[x\backslash y] \to_{\mathtt{ex}} yy$.
- No possible translation into MELL PN.
- Condition $x \in \mathbf{fv}(u)$ in last rule is necessary for termination.

The Linear Substitution Calculus

$\mathrm{L}\langle\lambda x.t\rangle u$	$\mapsto_{\mathtt{dB}}$	$\mathrm{L}\langle t[x\backslash u]\rangle$	if no capture of free variables
$t[x\backslash v]$	$\mapsto_{\mathtt{gc}}$	t	if $x \notin \mathbf{fv}(t)$
$\mathrm{C}\langle\!\langle x\rangle\!\rangle[x\backslash u]$	$\mapsto_{\mathtt{ls}}$	$\mathrm{C}\langle\!\langle u\rangle\!\rangle[x\backslash u]$	if no capture of free variables

- Rule $\mapsto_{\mathtt{ls}}$ implements linear substitution.
- The reduction relation is gerated by the rewriting rules modulo alpha-conversion.
- **Example:** $(\lambda x.xx)y \to_{\mathtt{dB}} (xx)[x\backslash y] \to_{\mathtt{ls}} (yx)[x\backslash y] \to_{\mathtt{ls}} (yy)[x\backslash y] \to_{\mathtt{gc}} yy$
- No direct translation into (Girard) MELL PN.
- Very good rewriting properties (e.g. residual theory).

The λpn-Calculus

- A calculus with explicit substitutions enjoying good properties.
  - Full Composition
  - Confluence on terms and open terms
  - Preservation of β-Strong Normalization.
  - Strong Normalization.
- Admits a translation to (Girard) PN.
- Reduction faithfully follows cut-elimination of Girard PN.

Equivalence

The Notion of Head Context:

$$\mathtt{H} ::= \Box \mid \lambda x.\mathtt{H} \mid \mathtt{H}t \mid \mathtt{H}[x\backslash t]$$

The Equivalence:

$$\begin{array}{lcll}
\lambda x.t & \equiv & \lambda y.t\{x\backslash y\} & y \text{ fresh} \\
t[x\backslash u] & \equiv & t\{x\backslash y\}[y\backslash u] & y \text{ fresh} \\
\mathtt{H}\langle t\rangle[x\backslash u] & \equiv & \mathtt{H}\langle t[x\backslash u]\rangle & \text{if } x \notin \mathbf{fv}(\mathtt{H}) \text{ and no capture of free variables}
\end{array}$$

Particular Instances:

$$\begin{array}{lcll}
\lambda x.t & \equiv & \lambda y.t\{x\backslash y\} & y \text{ fresh} \\
t[x\backslash u] & \equiv & t\{x\backslash y\}[y\backslash u] & y \text{ fresh} \\
t[y\backslash v][x\backslash u] & \equiv & t[x\backslash u][y\backslash v] & \text{if } y \notin \mathbf{fv}(u) \ \&\ x \notin \mathbf{fv}(v) \\
(\lambda y.t)[x\backslash u] & \equiv & \lambda y.t[x\backslash u] & \text{if no capture of free variables} \\
(tv)[x\backslash u] & \equiv & t[x\backslash u]v & \text{if } x \notin \mathbf{fv}(v)
\end{array}$$

Reduction

The Notion of Box Context:

$$\mathtt{B} ::= t\Box \mid t[x\backslash\Box]$$

Notation:

$$\mathtt{B}[y\backslash u] ::= \begin{cases} t[y\backslash u]\Box & \text{if } \mathtt{B} = t\Box \\ t[y\backslash u][x\backslash\Box] & \text{if } \mathtt{B} = t[x\backslash\Box] \end{cases}$$

The Rewriting Rules:

$\mathtt{L}\langle\lambda x.t\rangle u$	$\mapsto_{\mathtt{dB}}$	$\mathtt{L}\langle t[x\backslash u]\rangle$	if no capture of free variables
$x[x\backslash u]$	$\mapsto_{\mathtt{var}}$	u	
$t[x\backslash u]$	$\mapsto_{\mathtt{gc}}$	t	$x \notin \mathbf{fv}(t)$
$\mathtt{B}\langle\!\langle v\rangle\!\rangle[x\backslash u]$	$\mapsto_{\mathtt{arg}}$	$\mathtt{B}\langle\!\langle v[x\backslash u]\rangle\!\rangle$	$x \in \mathbf{fv}(v), x \notin \mathbf{fv}(\mathtt{B})$, no capture of free variables
$\mathtt{B}\langle\!\langle v\rangle\!\rangle[x\backslash u]$	$\mapsto_{\mathtt{dup}}$	$\mathtt{B}[x\backslash u]\langle\!\langle v\rangle\!\rangle[x\backslash u]$	$x \in \mathbf{fv}(v), x \in \mathbf{fv}(\mathtt{B})$, no capture of free variables

The Reduction Relations pn and λpn

- One-step reduction relation $\to_{\mathtt{pn}}$ on λpn-terms:

$$\frac{t \mapsto_{\mathtt{var,gc,arg,dup}} u}{t \to_{\mathtt{pn}} u} \qquad \frac{t \to_{\mathtt{pn}} u}{\mathsf{C}\langle t\rangle \to_{\mathtt{pn}} \mathsf{C}\langle u\rangle}$$

- One-step reduction relation $\to$ on λpn-terms:

$$\frac{t \mapsto_{\mathtt{dB,var,gc,arg,dup}} u}{t \to u} \qquad \frac{t \to u}{\mathsf{C}\langle t\rangle \to \mathsf{C}\langle u\rangle}$$

- Reduction relation $\to_{\lambda\mathtt{pn}}$ on λpn-terms:

$$t \to_{\lambda\mathtt{pn}} t' \text{ iff } \exists t_1, t_2 \text{ such that } t \equiv t_1 \to t_2 \equiv t'$$

Example

$$
\begin{array}{ll}
(\lambda z.\lambda y.\lambda x.yxx)wuv & \rightarrow_{\mathtt{dB}} \\
(\lambda y.\lambda x.yxx)[z\backslash w]uv & \rightarrow_{\mathtt{gc}} \\
(\lambda y.\lambda x.yxx)uv & \rightarrow_{\mathtt{dB}} \\
(\lambda x.yxx)[y\backslash u]v & \rightarrow_{\mathtt{dB}} \\
(yxx)[x\backslash v][y\backslash u] & \equiv \\
(y[y\backslash u]xx)[x\backslash v] & \rightarrow_{\mathtt{var}} \\
(uxx)[x\backslash v] & \rightarrow_{\mathtt{dup}} \\
((ux)[x\backslash v]x)[x\backslash v] & \equiv \\
((ux)[x\backslash v]x')[x'\backslash v] & \rightarrow_{\mathtt{arg}} \\
((ux)[x\backslash v]x'[x'\backslash v]) & \rightarrow_{\mathtt{var}} \\
((ux)[x\backslash v]v) & \rightarrow_{\mathtt{arg}} \\
((ux[x\backslash v])v) & \rightarrow_{\mathtt{var}} \\
((uv)v) &
\end{array}
$$

Some Properties

Lemma (Stability of Free Variables)

- *If* $t \equiv u$, *then* $\mathbf{fv}(t) = \mathbf{fv}(u)$.
- *If* $t \to_{\mathtt{dB},\mathtt{var},\mathtt{arg},\mathtt{dup}} u$, *then* $\mathbf{fv}(t) = \mathbf{fv}(u)$.
- *If* $t \to_{\mathtt{gc}} u$, *then* $\mathbf{fv}(t) \supseteq \mathbf{fv}(u)$.

Full Composition

Definition

A calculus with explicit substitutions $\mathcal{R}$ has the **Full Composition (FC)** property iff $t[x\backslash u] \to^*_{\mathcal{R}} t\{x\backslash u\}$ for all terms t, u.

Exercice : Define the operation $\_\{\_\backslash\_\}$ on $\lambda\mathtt{pn}$-terms and show that $\lambda\mathtt{pn}$ enjoys full composition by showing that $t[x\backslash u] \to^*_{\mathtt{pn}} t\{x\backslash u\}$.

Exercice : Give a variant of $\lambda\mathtt{pn}$ not enjoying full composition.

Computing pn-normal forms

Lemma (Uniqueness of pn)

The system pn *is confluent and terminating and thus every term has a unique* pn*-normal form, denoted* $\mathtt{pn}(t)$.

Exercice : Let $\mathtt{I} = \lambda w.w$. Compute the pn-normal form of the term $\lambda w.(zz(x[x\backslash a]y[y\backslash b]))[z\backslash \mathtt{I}]$.

Purification

Exercice : Define an inductive function $\texttt{proj}$ such that $\texttt{proj}(t) = \texttt{pn}(t)$.

1. Let us define $\texttt{proj}$ by induction as follows:

$$\begin{array}{lcl} \texttt{proj}(x) & := & x \\ \texttt{proj}(tu) & := & \texttt{proj}(t)\texttt{proj}(u) \\ \texttt{proj}(\lambda x.t) & := & \lambda x.\texttt{proj}(t) \\ \texttt{proj}(t[x\backslash u]) & := & \texttt{proj}(t)\{x\backslash\texttt{proj}(u)\} \end{array}$$

2. Prove that $\texttt{proj}(t) = \texttt{pn}(t)$ by induction on t.

Exercice : Verify the previous property on $\lambda w.(zz(x[x\backslash a]y[y\backslash b]))[z\backslash \texttt{I}]$.

Confluence on terms

Lemma (Projection)

Let t be a $\lambda\mathtt{pn}$-term. If $t \to_{\lambda\mathtt{pn}} t'$, then $\mathtt{proj}(t) \to_{\beta}^{*} \mathtt{proj}(t')$.

Lemma (Simulation)

Let t be a pure term. If $t \to_{\beta} t'$, then $t \to^{+}_{\lambda\mathtt{pn}} t'$.

Theorem (Confluence)

Let t, t_1, t_2 be $\lambda\mathtt{pn}$-terms. If $t \to^{*}_{\lambda\mathtt{pn}} t_1$ and $t \to^{*}_{\lambda\mathtt{pn}} t_2$, then there is $\mathtt{t}_3$ s.t. $t_1 \to^{*}_{\lambda\mathtt{pn}} t_3$ and $t_2 \to^{*}_{\lambda\mathtt{pn}} t_3$.

Confluence on open terms

Open Terms: $t, u ::= X_\Delta \mid x \mid \lambda x.t \mid t\ u \mid t[x\backslash u]$ (Δ is a set of variables)
For example $\lambda x.X_{\{x,y\}}z$ is an open term.

Theorem (Confluence on open terms)

Let t, t_1, t_2 be open λpn-terms. If $t \to^*_{\lambda\text{pn}} t_1$ and $t \to^*_{\lambda\text{pn}} t_2$, then there is $\mathtt{t}_3$ s.t. $t_1 \to^*_{\lambda\text{pn}} t_3$ and $t_2 \to^*_{\lambda\text{pn}} t_3$.

Observe that the following diagram is joinable thanks to equivalence $\equiv$:

$$X_{\{x,y\}}[y\backslash W][x\backslash Z_{\{y\}}[y\backslash W]] \ {}^*\!\leftarrow ((\lambda x.X_{\{x,y\}})Z_{\{y\}})[y\backslash W] \to^* X_{\{x,y\}}[x\backslash Z_{\{y\}}[y\backslash W]][y\backslash W]$$

Preservation of Strong Normalization (PSN)

Definition (**PSN)**

Let $\mathcal{O}_1 \subseteq \mathcal{O}_2$. Let R_1 be a relation on $\mathcal{O}_1$ and R_2 be a relation on $\mathcal{O}_2$. Then R_2 is said to preserve strong normalization of R_1 iff $t \in SN(R_1)$ implies $t \in SN(R_2)$.

Theorem (PSN for $\lambda\mathtt{pn}$)

Let t be a λ-term. If $t \in SN(\beta)$, then $t \in SN(\lambda\mathtt{pn})$.

Discussion

- The design of a calculus with explicit substitution enjoying confluence on open terms and PSN at the same time was an open open for many years.
- In particular, $\lambda\sigma$ of Abadi-Cardelli-Curien-Lévy does not enjoy PSN, a result due to Melliès.

How confluence on open terms can be lost

Consider the $\lambda\mathtt{x}$-calculus:

$$
\begin{array}{lll}
(\lambda x.t)\ u & \to_B & t[x\backslash u] \\
(t\ u)[x\backslash v] & \to & (t[x\backslash v]\ u[x\backslash v]) \\
(\lambda y.t)[x\backslash v] & \to & \lambda y.t[x\backslash v] \qquad \text{if } x \neq y \,\&\, y \notin \mathbf{fv}(v) \\
t[x\backslash v] & \to & t \qquad \text{if } x \notin \mathbf{fv}(t) \\
x[x\backslash v] & \to & v
\end{array}
$$

Observe that the following diagram is not joinable:

$$X_{\{x,y\}}[x\backslash Z_{\{y\}}][y\backslash W] \leftarrow (\lambda x.X_{\{x,y\}})Z_{\{y\}}[y\backslash W] \to^* X_{\{x,y\}}[y\backslash W][x\backslash Z_{\{y\}}[y\backslash W]]$$

How PSN can be lost

Consider the $\lambda\mathtt{x}$-calculus:

$$\begin{array}{lcll}
(\lambda x.t)\ u & \to_B & t[x\backslash u] & \\
(t\ u)[x\backslash v] & \to & (t[x\backslash v]\ u[x\backslash v]) & \\
(\lambda y.t)[x\backslash v] & \to & \lambda y.t[x\backslash v] & \text{if } x \neq y \,\&\, y \notin \mathbf{fv}(v) \\
t[x\backslash v] & \to & t & \text{if } x \notin \mathbf{fv}(t) \\
x[x\backslash v] & \to & v &
\end{array}$$

Define $\lambda\mathtt{x}c = \lambda\mathtt{x} \cup \{\mathtt{Comp}\}$, where:

$$(\mathtt{Comp}) \qquad t[x\backslash u][y\backslash v] \to\ t[x\backslash u[y\backslash v]] \text{ if } y \notin \mathbf{fv}(t)$$

The $\lambda\mathtt{x}c$-calculus does not preserve β-strong normalization:

- Define $\alpha_0 = [x\backslash(\lambda y.u)u]$ and $\alpha_{m+1} = [x\backslash u\alpha_m]$
- Show that $u\alpha_0\alpha_{m+1} \to^* \ldots u\alpha_{m+1}\alpha_{m+2}\ldots$
- Show that $u\alpha_{m+1}\alpha_{n+1} \to^* \ldots u\alpha_m\alpha_{n+1}\ldots$
- Show that there is a λ-typable term t (thus $t \in SN(\beta)$) that admits an infinite reduction sequence in the new system $\lambda\mathtt{x}c$ (thus $t \notin SN(\lambda\mathtt{x}c)$):

$$
\begin{array}{ll}
\lambda u.(\lambda x.(\lambda y.u)u)((\lambda x'.u)u) & \to_B \\
\lambda u.((\lambda y.u)u)[x\backslash(\lambda x'.u)u] & \to_{\mathtt{s}}^* \\
\lambda u.((\lambda y.u[x\backslash(\lambda x'.u)u])u[x\backslash(\lambda x'.u)u]) & = \\
\lambda u.((\lambda y.u\alpha_0)u\alpha_0) & \to_B \\
\lambda u.(u\alpha_0)[y\backslash u\alpha_0] & = \\
\lambda u.(u\alpha_0\alpha_1) & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_1\alpha_2 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_0\alpha_2 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_2\alpha_3 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_1\alpha_3 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_0\alpha_3 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_3\alpha_4 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_2\alpha_4 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_1\alpha_4 \ldots & \to_{\lambda\mathtt{x}c}^* \\
\ldots u\alpha_0\alpha_4 \ldots & \ldots
\end{array}
$$

A Special (Maximal) Strategy for λpn-Reduction...

The **strategy** $\rightsquigarrow$ on terms is given by an inductive definition.

$$\frac{}{(\lambda x.t)u\overline{u_n} \rightsquigarrow t[x\backslash u]\overline{u_n}} \qquad \frac{t \rightsquigarrow t'}{\lambda x.t \rightsquigarrow \lambda x.t'} \qquad \frac{\overline{u_n} \in \mathtt{NF}_{\lambda\mathrm{pn}} \quad t \rightsquigarrow t'}{x\overline{u_n}t\overline{v_m} \rightsquigarrow x\overline{u_n}t'\overline{v_m}}$$

$$\frac{u \in SN(\lambda\mathrm{pn})}{t[x\backslash u]\overline{v_n} \rightsquigarrow t\{x\backslash u\}\overline{v_n}} \qquad \frac{u \notin SN(\lambda\mathrm{pn}) \qquad u \rightsquigarrow u'}{t[x\backslash u]\overline{v_n} \rightsquigarrow t[x\backslash u']\overline{v_n}}$$

... which is Perpetual

Theorem (Perpetuality)

If $t \rightsquigarrow t'$ *and* $t' \in SN(\lambda\mathbf{pn})$, *then* $t \in SN(\lambda\mathbf{pn})$

- Equivalently $t \notin SN(\lambda\mathbf{pn})$ implies $t' \notin SN(\lambda\mathbf{pn})$.
- Remark that call-by-name is not a perpetual strategy.

Inductive Characterisation of $SN(\lambda\mathbf{pn})$

$$\frac{t_1, \ldots, t_n \in \mathtt{ISN} \qquad n \geq 0}{x t_1 \ldots t_n \in \mathtt{ISN}} \qquad \frac{u[x\backslash v] t_1 \ldots t_n \in \mathtt{ISN} \qquad n \geq 0}{(\lambda x.u) v t_1 \ldots t_n \in \mathtt{ISN}}$$

$$\frac{u\{x\backslash v\} t_1 \ldots t_n \in \mathtt{ISN} \qquad v \in \mathtt{ISN} \qquad n \geq 0}{u[x\backslash v] t_1 \ldots t_n \in \mathtt{ISN}} \qquad \frac{u \in \mathtt{ISN}}{\lambda x.u \in \mathtt{ISN}}$$

Lemma (Inductive Characterization of Strongly Normalizable Terms)

$SN(\lambda\mathbf{pn}) = \mathtt{ISN}$.

Proof.

Uses the Perpetuality Theorem. □