Strategies as Higher-Order Recursion Schemes

Pierre Clairambault
ENS Lyon

Andrzej Murawski
University of Warwick

26 Juin 2014
Abstraction and Verification in Semantics
IHP, Paris

I. Introduction

Higher-order program verification

Example Stolen from Olivier Serre's talk.

$$
\begin{aligned}
\mathtt{Main} &= \mathtt{MakeReport}\,\mathtt{Nil} \\
\mathtt{MakeReport}\,x &= \mathtt{if} * \mathtt{then}\,(\mathtt{commit}\,x) \\
&\quad\ \mathtt{else}\,(\mathtt{AddData}\,x\,\mathtt{MakeReport}) \\
\mathtt{AddData}\,y\,\phi &= \mathtt{if} * \mathtt{then}\,(\phi\,(\mathtt{Error}\,\mathtt{End})) \\
&\quad\ \mathtt{else}\,(\phi\,(\mathtt{Cons}\,(-,y)))
\end{aligned}
$$

Question: Can we commit an error?
$\hookrightarrow$ **By MSO model-checking (Ong):** Yes, there is a branch:

$$\mathtt{if} \xrightarrow{2} \mathtt{if} \xrightarrow{1} \mathtt{if} \xrightarrow{1} \mathtt{commit} \xrightarrow{1} \mathtt{Error}$$

Question: If I am given another candidate for this program, can I check their equivalence?
$\hookrightarrow$ **If nonterminals do not have higher-order parameters:** Yes, by Sénizergues' result.
$\hookrightarrow$ **Otherwise:** Open problem: **HORSE**

Higher-order open program verification

Problem: What if we do not have the full program, but just:

$$\begin{aligned}\mathtt{AddData}\,y\,\phi \;=\; & \mathtt{if}\ *\ \mathtt{then}\,(\phi\,(\mathtt{Error}\,\mathtt{End}))\\ & \mathtt{else}\,(\phi\,(\mathtt{Cons}\,(-, y)))\end{aligned}$$

Questions:

1. Can we check MSO properties compositionally?
2. If AddData is replaced by AddData′ by an optimizer (or an intern), can we check if they are equivalent? Superficially, it seems harder than HORSE:

   $$\mathtt{AddData} : o \to (o \to o) \to o$$

   but is it strictly harder?

In this talk, we give some answers, mainly to 2. For 1, see also Ong and Tsukada's recent CSL-LICS paper.

II. Higher-order program equivalence in semantics

The language under study: PCF_2

Types.

$$A ::= \mathrm{Bool} \mid A \to A$$

Terms.

$$\begin{array}{rcl} M, N & ::= & \lambda x.\, M \mid x \mid M\, N \mid Y \mid \\ & & tt \mid ff \mid \mathrm{if}\, M\, \mathrm{then}\, N_1\, \mathrm{else}\, N_2 \end{array}$$

Typing rules. Standard, generating a typing relation $\Gamma \vdash M : A$

Reduction.

$$\begin{array}{rcl} (\lambda x.\, M)\, N & \rightsquigarrow & M[N/x] \\ \mathrm{if}\, tt\, \mathrm{then}\, N_1\, \mathrm{else}\, N_2 & \rightsquigarrow & N_1 \\ \mathrm{if}\, ff\, \mathrm{then}\, N_1\, \mathrm{else}\, N_2 & \rightsquigarrow & N_2 \\ Y\, M & \rightsquigarrow & M\,(Y\, M) \end{array}$$

When are two programs the same?

Observational equivalence. A closed program $\vdash M : \mathrm{Bool}$ **converges**, written $M\downarrow$, iff

$$M \rightsquigarrow^* \mathit{tt}, \mathit{ff}$$

Two programs $\vdash M : A$, $\vdash N : A$ are **observationally equivalent** iff they are indistinguishable:

$$M \simeq N \Leftrightarrow (\forall \mathcal{C}[-],\ \mathcal{C}[M]\downarrow \Leftrightarrow\ \mathcal{C}[N]\downarrow)$$

Loader's undecidability.

Theorem (Loader)

The equivalence $\simeq$ is undecidable, even on **finitary** *(Y-free)* PCF_2.

However in the above, $\mathcal{C}$ is chosen in PCF_2 itself...

Computational effects and notions of observation

Idea. In practice, functional programs are often executed in an **unsafe** environment, with **computational effects**.

Computational effects.

- **Control operators.** Basic exception mechanism.

$$\begin{array}{rcl} \mathrm{call/cc} & : & ((A \to B) \to A) \to A \\ \mathrm{call/cc}\,(\lambda f.\, E[f\ M]) & \rightsquigarrow & M \end{array}$$

- **Ground state.** Combinators for **storing** and **reading** boolean values.

- **Higher-order state.** Combinators for **storing** and **reading** values of arbitrary types.

[Warning: all of this talk is in call-by-name, but relies on tools that can be applied as well to call-by-value]

Semantic analysis of observation

Semantic representation of PCF_2 observed by:

Effect	**Model**	**Decidable?**
None	non-effective (quotient of terms/strategies)	NO (Loader)
Control	Sequential Algorithms (Berry & Curien, Cartwright & Curien & Felleisen)	YES
Control (ground, arbitrary) State	Hyland-Ong games Innocent strategies	
(ground, arbitrary) State	Hyland-Ong games Observable parts of Innocent strategies	

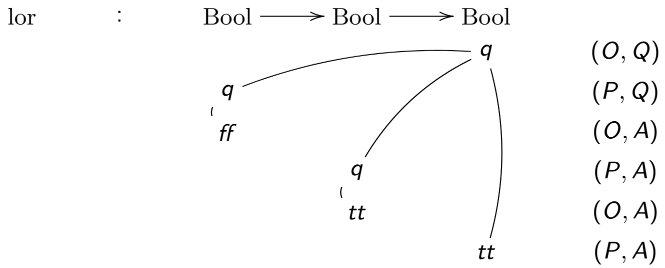
Semantic analysis of observation

Semantic representation of PCF_2 observed by:

Effect	Model	Decidable?
None	non-effective (quotient of terms/strategies)	NO (Loader)
Control	Sequential Algorithms (Berry & Curien, Cartwright & Curien & Felleisen)	YES
Control (ground, arbitrary) State	Hyland-Ong games Innocent strategies	$\Leftrightarrow$ HORSE
(ground, arbitrary) State	Hyland-Ong games Observable parts of Innocent strategies	$\Leftrightarrow$ HORSE

Hyland-Ong games and innocent strategies

Hyland-Ong games represents terms by enumerating their **dialogues** with the environment:



- The line is the **justification** relation, indicating the function/argument relationship.
- **Questions** are function/parameter calls, **Answers** are returns.
- Deterministic, purely functional terms are entirely specified by their plays which are **P-views** (Opponent always points to the previous move) and **well-bracketed** (Player always answers the latest unanswered question).

Well-bracketed P-views and branches of terms

Well-bracketed P-views correspond to syntactic branches of terms. For instance, this P-view:

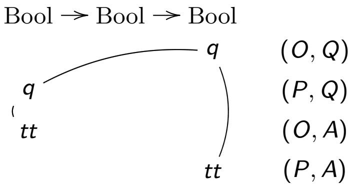
$$
\begin{array}{lcccccc}
\mathrm{lor} & : & \mathrm{Bool} & \longrightarrow & \mathrm{Bool} & \longrightarrow & \mathrm{Bool} & \\
 & & & & & & q & (O,Q) \\
 & & q & & & & & (P,Q) \\
 & & \mathit{ff} & & & & & (O,A) \\
 & & & & q & & & (P,Q) \\
 & & & & \mathit{tt} & & & (O,A) \\
 & & & & & & \mathit{tt} & (P,A)
\end{array}
$$

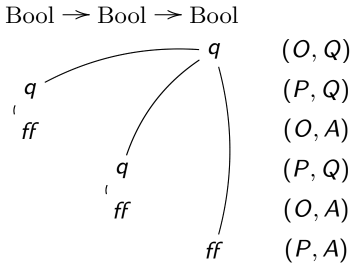
corresponds to the partial term:

$$\lambda b_1^{\mathrm{Bool}}.\,\lambda b_2^{\mathrm{Bool}}.\,\mathrm{if}\ b_1\ \mathrm{then}\ []\ \mathrm{else}\ (\mathrm{if}\ b_2\ \mathrm{then}\ \mathit{tt}\ \mathrm{else}\ [])$$

P-views and branches of terms

To fill in the holes, adjoin the two additional P-views:





in total giving the term:

$$\lambda b_1^{\mathrm{Bool}}.\,\lambda b_2^{\mathrm{Bool}}.\,\mathrm{if}\ b_1\ \mathrm{then}\,[tt]\,\mathrm{else}\,(\mathrm{if}\ b_2\ \mathrm{then}\ tt\ \mathrm{else}\,[ff])$$

which we call the **PCF Böhm tree** (Curien) of lor.

Non purely functional behaviour

Control. The behaviour of $\mathrm{call/cc}$ is expressed by **non well-bracketed** plays:

$$
\begin{array}{cccccccccc}
\mathrm{call/cc} & : & ((\mathrm{Bool} & \longrightarrow & \mathrm{Bool}) & \longrightarrow & \mathrm{Bool}) & \longrightarrow & \mathrm{Bool} & \\
 & & & & & & & & q & (O, Q) \\
 & & & & & & q & & & (P, Q) \\
 & & & & q & & & & & (O, Q) \\
 & & q & & & & & & & (P, Q) \\
 & & tt & & & & & & & (O, A) \\
 & & & & & & & & tt & (P, A)
\end{array}
$$

State. The presence of state is witnessed by **expanded plays**, *i.e.* not necessarily P-views:

$$
\begin{array}{cc}
\mathrm{Bool} & \\
q & (O, Q) \\
tt & (P, A) \\
q & (O, Q) \\
ff & (P, A)
\end{array}
$$

Arbitrary plays can be realized using terms with state and control.

PCF Böhm trees as normal forms w.r.t. control and state

Innocent strategies as PCF Böhm trees. In general, innocent (well-bracketed) strategies can be represented syntactically:

$$\frac{}{\Gamma \vdash \bot, tt, ff : \mathrm{Bool}} \qquad \frac{\Gamma, \overrightarrow{\ldots, x_i : A_i, \ldots} \vdash M : \mathrm{Bool}}{\Gamma \vdash \lambda \overrightarrow{x}.M : \overrightarrow{A} \to \mathrm{Bool}}$$

$$\frac{\overrightarrow{\Gamma \vdash M_i : A_i} \quad \Gamma \vdash N_1, N_2 : \mathrm{Bool} \quad (x : \overrightarrow{A} \to \mathrm{Bool}) \in \Gamma}{\Gamma \vdash \mathrm{if}\ x\ \overrightarrow{M}\ \mathrm{then}\ N_1\ \mathrm{else}\ N_2 : \mathrm{Bool}}$$

Construction. The PCF Böhm tree of a term M can be generated compositionally through its game semantics, or by infinitary rewriting.

Strategies as sets of plays. On the other hand, **all finite P-views are explorable by an environment with ground state and control**. So:

Proposition

Two terms of PCF_2 have the same PCF Böhm tree iff they are indistinsguishable by a context with state and control.

Observable Böhm trees as normal forms w.r.t. state

What if there is no control? The following PCF Böhm trees are distinct, but **indistinguishable** without $\mathrm{call/cc}$:

$$x : \mathrm{Bool} \to \mathrm{Bool} \vdash \mathrm{if}\ x\ tt\ \mathrm{then}\ \bot\ \mathrm{else}\ \bot$$

$$x : \mathrm{Bool} \to \mathrm{Bool} \vdash \mathrm{if}\ x\ ff\ \mathrm{then}\ \bot\ \mathrm{else}\ \bot$$

The call to x is not **observable** in a terminating well-bracketed computation.

Definition

An if statement in a PCF Böhm tree is **observable** iff, just by following $\mathrm{then}/\mathrm{else}$ branches, we can eventually reach a tt/ff.

Proposition

Two terms M_1, M_2 of PCF_2 are distinguishable by a context with state (without control) iff their PCF Böhm trees have the same observable prefixes.

III. PCF Böhm trees as recursion schemes

III.1 The λY-calculus and Böhm trees with binders

A first step: the λY-calculus

We first treat the boolean-free case – the λY-calculus.

Types. Simple types on one atom o.

$$A, B ::= o \mid A \to B$$

Terms.

$$M, N ::= x \mid \lambda x^A.M \mid M\ N \mid Y_A$$

Typing rules.

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma \vdash Y_A : (A \to A) \to A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : A \to B}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

Reduction.

$$\begin{array}{rcll} (\lambda x^A.M)\ N & \rightsquigarrow_\beta & M[N/x] & \\ Y_A\ M & \rightsquigarrow_\delta & M\ (Y_A\ M) & \\ M^{A \to B} & \rightsquigarrow_\eta & \lambda x^A.\, M\, x & (x \text{ fresh}) \end{array}$$

Böhm trees of λY-terms

λY-calculus and HORS. **Tree signatures** are represented by first-order contexts:

$$\Sigma = \{c_1 : o^{p_1} \to o, \ldots, c_n : o^{p_n} \to o\}$$

Proposition (Salvati & Walukiewicz)

HORS on the tree signature Σ correspond to λY-terms:

$$\Sigma \vdash M : o$$

Böhm trees of λY-terms. If $\Gamma \vdash M : A$ is a λY-term then its Böhm tree:

$$\Gamma \vdash BT(M) : A$$

is an infinitary term, defined as the upper bound of η-long β-normal forms of Y-free approximations of M.

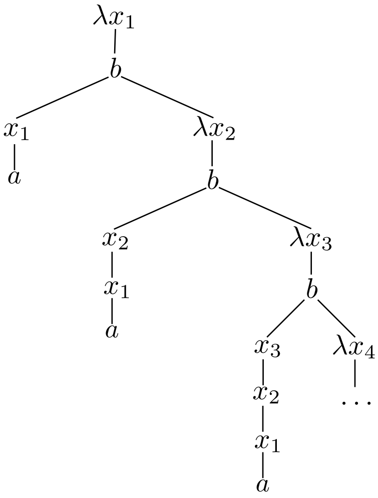
- For $\Sigma \vdash M : o$ a HORS, $BT(M)$ is the infinite tree generated by it.
- For arbitrary λY-terms, as for PCF_2 they correspond to innocent strategies.

Binders in open Böhm trees

Consider the term G:

$$Y_{o\to(o\to o)\to o}\ (\lambda f^{o\to(o\to o)\to o}.\lambda y^{o}.\lambda x^{o\to o}.\ b\ (x\ y)\ (f\ (x\ y)))\ a$$

Its Böhm tree starts with:

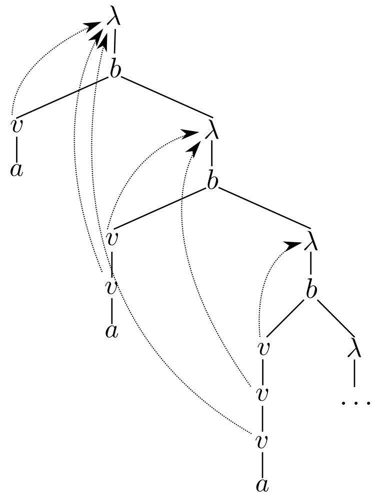


Binders in open Böhm trees

Consider the term G:

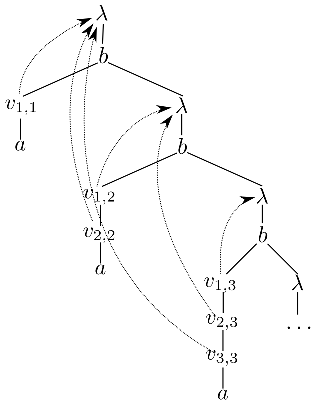
$$Y_{o\to(o\to o)\to o}\ (\lambda f^{o\to(o\to o)\to o}.\lambda y^{o}.\lambda x^{o\to o}.\ b\ (x\ y)\ (f\ (x\ y)))\ a$$

Its Böhm tree starts with:



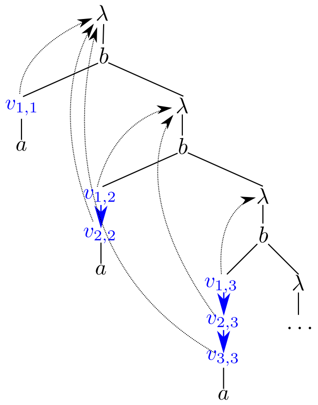
Undecidability with binding

We MSO-define a half-grid in the Böhm tree of G.



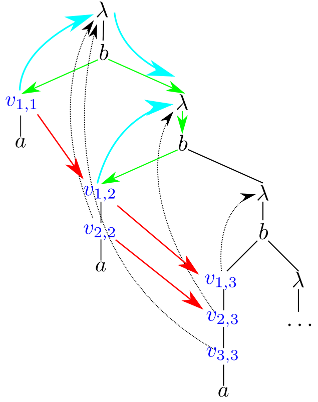
Undecidability with binding

We MSO-define a half-grid in the Böhm tree of G.



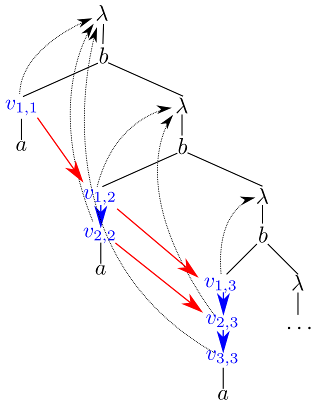
Undecidability with binding

We MSO-define a half-grid in the Böhm tree of G.



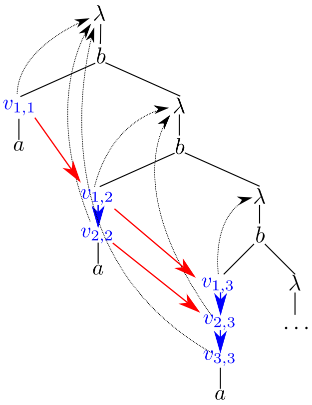
Undecidability with binding

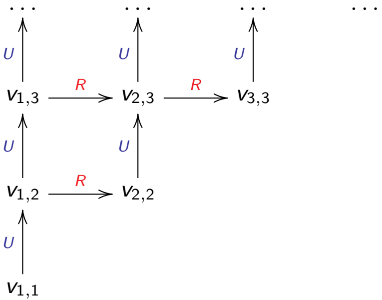
We MSO-define a half-grid in the Böhm tree of G.



Undecidability with binding

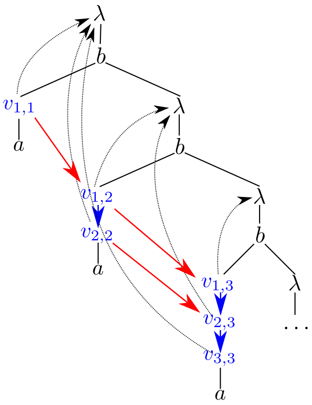
We MSO-define a half-grid in the Böhm tree of G.

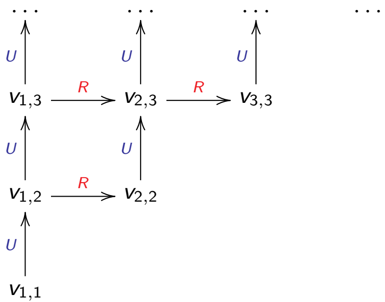




Undecidability with binding

We MSO-define a half-grid in the Böhm tree of G.





(Half-)grids have an undecidable MSO theory, therefore so do Böhm trees of open λY-terms.

III.2 Binder-free representation by De Bruijn levels

De Bruijn levels

Definition

De Bruijn levels are a variable naming convention where:

- Variable names are natural numbers,
- Each variable is given the smallest index not yet present in the context.

Example

The term

$$g : o \to o \to o \vdash \lambda f.f\ (\lambda x.g\ x\ (f\ (\lambda x.g\ x\ (f\ x))$$

is represented by:

$$0 : o \to o\ \ \to o \vdash \lambda 1.1\ (\lambda 2.0\ 2\ (1\ (\lambda 3.0\ 3\ (1\ 3))))$$

Proposition

Two terms M and M' have the same De Bruijn levels representation iff they are α-equivalent.

(not to be confused with **De Bruijn indices**)

Representation as binder-free normal forms

De Bruijn representations of terms are trees on the signature:

$z : o \qquad succ : o \to o \qquad var : o \to o \qquad app : o \to o \to o \qquad lam : o \to o \to o$

Or in other terms:

Definition

If $\vdash M : A$ is a λ-term, it has as De Bruijn representation, a Böhm tree:

$$\Gamma_{rep} \vdash \mathrm{rep}(M) : o$$

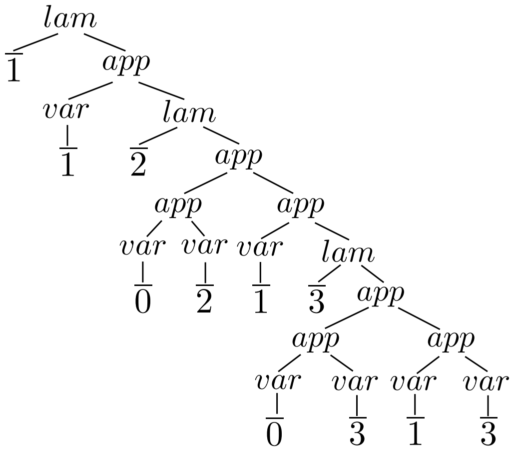
where Γ_{rep} is the context:

$$\{\, z : o,\ succ : o \to o,\ var : o \to o,\ app : o \to o \to o,\ lam : o \to o \to o \,\}$$

Note that even if M is not normal, $\mathrm{rep}(M)$ is!

Representation of λ-terms in Γ_{rep}

From $M = g : o \to o \to o \vdash \lambda f.f\ (\lambda x.g\ x\ (f\ (\lambda x.g\ x\ (f\ x))$, we build $\Gamma_{rep} \vdash \mathrm{rep}(M) : o$:



(Keeping in mind that $\overline{n} = succ\ (succ\ \ldots\ (succ\ z)\ldots))$

III.3 Internal Normalization by Evaluation

Normalization by evaluation

Semantic technique for computing normal forms of λ-terms.

Theorem (Martin-Löf, Berger & Schwichtenberg)

There is a set-theoretic interpretation of the simply-typed λ-calculus:

$$[\![-]\!] : \Lambda \to \mathrm{Set}$$

and for each type A, a function

$$\mathrm{reify} : [\![A]\!] \to \Lambda$$

such that for each term $\vdash M : A$,

$$\mathrm{reify}([\![M]\!]) \cong_{\beta\eta} M$$

is the β-normal η-long form of M.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$\begin{array}{rcl} [\![o]\!] & = & E \\ [\![x]\!]_\rho & = & \rho(x) \\ [\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) \end{array} \qquad \begin{array}{rcl} [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\ [\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} \end{array}$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$\begin{array}{rcl} \Downarrow_A & : & [\![A]\!] \to E \\ \Downarrow_o x & = & x \\ \Downarrow_{A \to B} x & = & \end{array}$$

$$\begin{array}{rcl} \Uparrow_A & : & E \to [\![A]\!] \\ \Uparrow_o e & = & e \\ \Uparrow_{A \to B} e & = & \end{array}$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$\begin{array}{rcl} [\![o]\!] & = & E \\ [\![x]\!]_\rho & = & \rho(x) \\ [\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) \end{array} \qquad \begin{array}{rcl} [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\ [\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} \end{array}$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$\begin{array}{rcl} \Downarrow_A & : & [\![A]\!] \to E \\ \Downarrow_o x & = & x \\ \Downarrow_{A\to B} x & = & lam\ n\ (\qquad\qquad\qquad\qquad) \quad (n \text{ fresh}) \end{array}$$

$$\begin{array}{rcl} \Uparrow_A & : & E \to [\![A]\!] \\ \Uparrow_o e & = & e \\ \Uparrow_{A\to B} e & = & \end{array}$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$\begin{array}{rcl} [\![o]\!] & = & E \\ [\![x]\!]_\rho & = & \rho(x) \\ [\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) \end{array} \qquad \begin{array}{rcl} [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\ [\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} \end{array}$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$\begin{array}{rcl} \Downarrow_A & : & [\![A]\!] \to E \\ \Downarrow_o x & = & x \\ \Downarrow_{A \to B} x & = & lam\ n\ (\qquad x \qquad\qquad) \quad (n \text{ fresh}) \end{array}$$

$$\begin{array}{rcl} \Uparrow_A & : & E \to [\![A]\!] \\ \Uparrow_o e & = & e \\ \Uparrow_{A \to B} e & = & \end{array}$$

by setting $\mathrm{nbe}(\mathrm{M}) = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$
\begin{array}{rcl}
[\![o]\!] & = & E \\
[\![x]\!]_\rho & = & \rho(x) \\
[\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho)
\end{array}
\qquad
\begin{array}{rcl}
[\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\
[\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}}
\end{array}
$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$
\begin{array}{rcl}
\Downarrow_A & : & [\![A]\!] \to E \\
\Downarrow_o x & = & x \\
\Downarrow_{A \to B} x & = & lam\ n\ (\quad x \quad (var\ n)\ )\ \ (n \text{ fresh})
\end{array}
$$

$$
\begin{array}{rcl}
\Uparrow_A & : & E \to [\![A]\!] \\
\Uparrow_o e & = & e \\
\Uparrow_{A \to B} e & = &
\end{array}
$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$
\begin{array}{rcl@{\qquad}rcl}
[\![o]\!] & = & E & [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\
[\![x]\!]_\rho & = & \rho(x) & [\![\lambda x^A . M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]} . [\![M]\!]_{\rho \oplus \{x \mapsto a\}} \\
[\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) & & &
\end{array}
$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$
\begin{array}{rcl}
\Downarrow_A & : & [\![A]\!] \to E \\
\Downarrow_o x & = & x \\
\Downarrow_{A \to B} x & = & lam\ n\ (\qquad x\ (\Uparrow_A (var\ \ n))\ )\quad (n \text{ fresh})
\end{array}
$$

$$
\begin{array}{rcl}
\Uparrow_A & : & E \to [\![A]\!] \\
\Uparrow_o e & = & e \\
\Uparrow_{A \to B} e & = &
\end{array}
$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$
\begin{array}{rclrcl}
[\![o]\!] & = & E & [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\
[\![x]\!]_\rho & = & \rho(x) & [\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} \\
[\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) & & &
\end{array}
$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$
\begin{array}{rcl}
\Downarrow_A & : & [\![A]\!] \to E \\
\Downarrow_o x & = & x \\
\Downarrow_{A \to B} x & = & lam\ n\ (\Downarrow_B (x\ (\Uparrow_A (var\ n)))) \quad (n \text{ fresh})
\end{array}
$$

$$
\begin{array}{rcl}
\Uparrow_A & : & E \to [\![A]\!] \\
\Uparrow_o e & = & e \\
\Uparrow_{A \to B} e & = &
\end{array}
$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$
\begin{array}{rclcrcl}
[\![o]\!] & = & E & \qquad & [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\
[\![x]\!]_\rho & = & \rho(x) & & [\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} \\
[\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) & & & &
\end{array}
$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$
\begin{array}{rcl}
\Downarrow_A & : & [\![A]\!] \to E \\
\Downarrow_o x & = & x \\
\Downarrow_{A \to B} x & = & lam\ n\ (\Downarrow_B (x\ (\Uparrow_A (var\ n)))) \quad (n \text{ fresh})
\end{array}
$$

$$
\begin{array}{rcl}
\Uparrow_A & : & E \to [\![A]\!] \\
\Uparrow_o e & = & e \\
\Uparrow_{A \to B} e & = & \boldsymbol{\lambda} x^{[\![A]\!]}.
\end{array}
$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$
\begin{array}{rcl}
[\![o]\!] & = & E \\
[\![x]\!]_\rho & = & \rho(x) \\
[\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho)
\end{array}
\qquad
\begin{array}{rcl}
[\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\
[\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}}
\end{array}
$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$
\begin{array}{rcl}
\Downarrow_A & : & [\![A]\!] \to E \\
\Downarrow_o x & = & x \\
\Downarrow_{A \to B} x & = & \mathit{lam}\ n\ (\Downarrow_B (x\ (\Uparrow_A (\mathit{var}\ n)))) \quad (n \text{ fresh})
\end{array}
$$

$$
\begin{array}{rcl}
\Uparrow_A & : & E \to [\![A]\!] \\
\Uparrow_o e & = & e \\
\Uparrow_{A \to B} e & = & \boldsymbol{\lambda} x^{[\![A]\!]}. \quad \mathit{app}\ e \quad x
\end{array}
$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$\begin{array}{rcl} [\![o]\!] & = & E \\ [\![x]\!]_\rho & = & \rho(x) \\ [\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho) \end{array} \qquad \begin{array}{rcl} [\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\ [\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}} \end{array}$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$\begin{array}{rcl} \Downarrow_A & : & [\![A]\!] \to E \\ \Downarrow_o x & = & x \\ \Downarrow_{A\to B} x & = & lam\ n\ (\Downarrow_B (x\ (\Uparrow_A (var\ n)))) \quad (n \text{ fresh}) \end{array}$$

$$\begin{array}{rcl} \Uparrow_A & : & E \to [\![A]\!] \\ \Uparrow_o e & = & e \\ \Uparrow_{A\to B} e & = & \boldsymbol{\lambda} x^{[\![A]\!]}. \quad app\ e\ (\Downarrow_A x) \end{array}$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Normalization by evaluation for the simply-typed λ-calculus

Step 1: Interpretation. Let E be a set containing representations of terms.

$$
\begin{array}{rcl}
[\![o]\!] & = & E \\
[\![x]\!]_\rho & = & \rho(x) \\
[\![M\ N]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho)
\end{array}
\qquad
\begin{array}{rcl}
[\![A \to B]\!] & = & [\![A]\!] \to [\![B]\!] \\
[\![\lambda x^A.M]\!]_\rho & = & \boldsymbol{\lambda} a^{[\![A]\!]}.[\![M]\!]_{\rho \oplus \{x \mapsto a\}}
\end{array}
$$

Where all the right hand side operations are operations on sets and functions.

Step 2: Reification. The normal form of $\vdash M : A$ can be **extracted** from $[\![M]\!]$ by the following:

$$
\begin{array}{rcl}
\Downarrow_A & : & [\![A]\!] \to E \\
\Downarrow_o x & = & x \\
\Downarrow_{A \to B} x & = & lam\ n\ (\Downarrow_B (x\ (\Uparrow_A (var\ n)))) \quad (n \text{ fresh})
\end{array}
$$

$$
\begin{array}{rcl}
\Uparrow_A & : & E \to [\![A]\!] \\
\Uparrow_o e & = & e \\
\Uparrow_{A \to B} e & = & \boldsymbol{\lambda} x^{[\![A]\!]}.\Uparrow_B app\ e\ (\Downarrow_A x)
\end{array}
$$

by setting $\mathrm{nbe(M)} = \Downarrow_A [\![M]\!]$.

Generating De Bruijn levels (Berger & Schwichtenberg)

Expressions. $e \in E$ are replaced with **indexed expressions**

$$f \in \mathbb{N} \to E = \widehat{E}$$

Constructors. var, lam, app are replaced with variants:

$$\begin{array}{rcl} \widehat{var} & = & \boldsymbol{\lambda} v^N . \boldsymbol{\lambda} n^N . var\ v : N \to \widehat{E} \\ \widehat{app} & = & \boldsymbol{\lambda} e_1^{\widehat{E}} . \boldsymbol{\lambda} e_2^{\widehat{E}} . \boldsymbol{\lambda} n^N . app\ (e_1\ n)\ (e_2\ n) : \widehat{E} \to \widehat{E} \to \widehat{E} \\ \widehat{lam} & = & \boldsymbol{\lambda} f^{N \to \widehat{E}} . \boldsymbol{\lambda} n^N . lam\ n\ (f\ n\ (succ\ n)) : (N \to \widehat{E}) \to \widehat{E} \end{array}$$

Reify and reflect. They are generalized:

$$\begin{array}{rclcrcl} \Downarrow_o x & = & x & \quad & \Downarrow_{A \to B} x & = & \widehat{lam}\ (\boldsymbol{\lambda} n^N . \Downarrow_B (x\ (\Uparrow_A \widehat{var}\ n))) \\ \Uparrow_o e & = & e & \quad & \Uparrow_{A \to B} e & = & \boldsymbol{\lambda} x^{[\![A]\!]} . \Uparrow_B \widehat{app}\ e\ (\Downarrow_B x) \end{array}$$

Normalization by evaluation. The interpretation $[\![-]\!]$ is now based on $\widehat{E}$ instead of E. NBE is obtained for $\vdash M : A$ by:

$$\mathrm{nbe}(M) = \Downarrow_A [\![M]\!]\ 0$$

Continuity and NBE for λY

Step 1. We adapt the construction to produce lazily infinite normal forms.

Model. Standard pointed ω-cpo model of the λY-calculus:

$$\begin{array}{rcl} [\![o]\!] & = & \widehat{E} \\ [\![Y_A]\!] & = & \boldsymbol{\lambda} f^A . \bigsqcup_n f^n(\bot) \end{array}$$

where E is the pointed ω-cpo of possibly infinite expressions.

Normalization by evaluation. From a λY-term $\vdash M : A$,

$$\mathrm{nbe}(M) = \Downarrow_A [\![M]\!]\ 0 \in E$$

$\hookrightarrow$ Obtain an infinite normal form.

Proof. By induction for finite normal forms, by soundness and continuity arguments for arbitrary terms.

Internalization within λY

Step 2. We internalize NBE for λY within the λY-calculus.

Expressions are terms of λY:

$$\Gamma_{rep} \vdash M : o$$

Term families is the type $\widehat{E} = o \to o$.

Interpretation is the substitution $A^* = A[o \to o/o]$ and $M^* = M[o \to o/o]$.

Term formers are the following:

$$\begin{array}{rcl} \widehat{var} & = & \lambda v^o.\lambda n^o.var\ v \\ \widehat{lam} & = & \lambda f^{o \to o \to o}.\lambda n^o.lam\ n\ (f\ \ n\ (succ\ n)) \\ \widehat{app} & = & \lambda e_1^{o \to o}.\lambda e_2^{o \to o}.\lambda n^o.app\ (e_1\ \ n)\ (e_2\ \ n) \end{array}$$

Reify/reflect are now terms of the λY-calculus:

$$\begin{array}{rclrcl} \downarrow_o & = & \lambda x^o.x & \downarrow_{A \to B} & = & \lambda x^{A^* \to B_2^*}.\widehat{lam}\ (\lambda n^o.\downarrow_B (x\ (\uparrow_A \widehat{var}\ \ n))) \\ \uparrow_o & = & \lambda e^o.e & \uparrow_{A \to B} & = & \lambda e^o.\lambda x^{A^*}.\uparrow_B \widehat{app}\ e\ (\downarrow_B x) \end{array}$$

Internalization within λY

Theorem

If $\vdash M : A$ is a λY-term, then the term M_{rep} defined as:

$$\Gamma_{rep} \vdash\!\downarrow_A \; M^* \; \overline{0} : o$$

satisfies:

$$BT(M_{rep}) = \mathrm{rep}(BT(M))$$

Moreover, the construction increases the **order** *by one.*

The NBE translation for PCF_2

Representation. In the ω-cpo E of infinitary terms $\Gamma_{pcf} \vdash M : o$, with:

$$\Gamma_{pcf} = \Gamma_{rep} \cup \{tt : o, ff : o, if : o \to o \to o \to o\}$$

Semantics. Domain semantics of PCF, based on:

$$[\![\mathrm{Bool}]\!] = \widehat{E} \to \widehat{E} \to \widehat{E}$$

Reflect and reify. Extended to booleans with:

$$\begin{array}{rcl} \Downarrow_A & : & [\![A]\!] \to \widehat{E} \\ \Downarrow_{\mathrm{Bool}} x & = & x\,\widehat{tt}\,\widehat{ff} \end{array} \qquad \begin{array}{rcl} \Uparrow_A & : & \widehat{E} \to [\![A]\!] \\ \Uparrow_{\mathrm{Bool}} e & = & \lambda x^{\widehat{E}}.\, \lambda y^{\widehat{E}}.\, \widehat{\mathrm{if}}\; e\, x\, y \end{array}$$

with $\widehat{tt} = \lambda\_.tt$, $\widehat{ff} = \lambda\_.ff$, $\widehat{\mathrm{if}} = \lambda e_1^{\widehat{E}}.\, \lambda e_2^{\widehat{E}}.\, \lambda n^{E}.\, \mathrm{if}\,(e_1\, n)\,(e_2\, n)$.

Internalization. Follows the same lines as for λY.

IV. Conclusions

Consequences

Theorem

For any term $\Gamma \vdash M : A$ of PCF_2, there is a recursion scheme that generates:

- *(the De Bruijn levels representation of) the PCF Böhm tree of M,*
- *Equivalently, the innocent strategy for M.*

Theorem

For any term $\Gamma \vdash M : A$ of PCF_2, there is a recursion scheme that generates the **observable prefix** *of the PCF Böhm tree of M.*

Proof.

Being an *observable node* is MSO-definable, so we deduce by logical reflection for schemes (Broadbent, Carayol, Ong, Serre). $\square$

Consequences

For equivalence. The following problems are equivalent to HORSE.

- Distinguishability of PCF_2 terms via contexts with state and control.
- Distinguishability of PCF_2 terms via contexts with state (but no control).

From MSO model-checking: The following problems are decidable:

- **Normalization:** Is a $\lambda Y/\mathrm{PCF}_2$ term equivalent to a Y-free term?
- **Finiteness:** Has a $\lambda Y/\mathrm{PCF}_2$ term a finite Böhm tree/strategy?
- **Solvability:** Has a $\lambda Y/\mathrm{PCF}_2$ term a head normal form?
- **Finite prefix:** Has a $\lambda Y/\mathrm{PCF}_2$ term a finite/regular prefix?

Or the same, with the **observable** prefix.