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
Example Stolen from Olivier Serre’s talk.

\[
\text{Main} = \text{MakeReport} \text{Nil}
\]

\[
\text{MakeReport} x = \text{if } \ast \text{ then } (\text{commit } x)
\]

\[
\text{else } (\text{AddData} x \text{ MakeReport})
\]

\[
\text{AddData} y \phi = \text{if } \ast \text{ then } (\phi (\text{Error End}))
\]

\[
\text{else } (\phi (\text{Cons } (-, y)))
\]

Question: Can we commit an error?

\[\iff \text{ By MSO model-checking (Ong): Yes, there is a branch:}\]

\[
\begin{array}{c}
\text{if } \xrightarrow{2} \text{ if } \xrightarrow{1} \text{ if } \xrightarrow{1} \text{ commit } \xrightarrow{1} \text{ Error}
\end{array}
\]

Question: If I am given another candidate for this program, can I check their equivalence?

\[\iff \text{ If nonterminals do not have higher-order parameters: Yes, by Sénizergues’ result.}\]

\[\iff \text{ Otherwise: Open problem: HORSE}\]
Problem: What if we do not have the full program, but just:

$$\text{AddData } y \phi = \text{if } \ast \text{ then}(\phi(\text{Error End})) \quad \text{else}(\phi(\text{Cons}(-, y)))$$

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:

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

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.

\[ A ::= \text{Bool} \mid A \to A \]

Terms.

\[ M, N ::= \lambda x. M \mid x \mid M \, N \mid Y \mid tt \mid ff \mid \text{if } M \text{ then } N_1 \text{ else } N_2 \]

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

Reduction.

\[ (\lambda x. \, M) \, N \rightsquigarrow M[N/x] \]
\[ \text{if } tt \text{ then } N_1 \text{ else } N_2 \rightsquigarrow N_1 \]
\[ \text{if } ff \text{ then } N_1 \text{ else } N_2 \rightsquigarrow N_2 \]
\[ Y \, M \rightsquigarrow M (Y \, M) \]
When are two programs the same?

**Observational equivalence.** A closed program ⊢ M : Bool **converges**, written \( M \downarrow \), iff

\[
M \leadsto^* \text{tt, ff}
\]

Two programs ⊢ M : A, ⊢ N : A are **observationally equivalent** iff they are indistinguishable:

\[
M \simeq N \iff (\forall C[-], \ C[M] \downarrow \iff C[N] \downarrow)
\]

**Loader’s undecidability.**

**Theorem (Loader)**

*The equivalence \( \simeq \) is undecidable, even on **finitary** (\( Y \)-free) PCF\(_2\).*

However in the above, \( C \) is chosen in PCF\(_2\) itself...
Idea. In practice, functional programs are often executed in an unsafe environment, with computational effects.

Computational effects.

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

  \[
  \text{call/cc} : ((A \rightarrow B) \rightarrow A) \rightarrow A \\
  \text{call/cc}\ (\lambda f. E[f\ M]) \rightsquigarrow M
  \]

- **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 representation of PCF₂ observed by:

<table>
<thead>
<tr>
<th>Effect</th>
<th>Model</th>
<th>Decidable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>non-effective (quotient of terms/strategies)</td>
<td>NO (Loader)</td>
</tr>
<tr>
<td>Control</td>
<td>Sequential Algorithms (Berry &amp; Curien, Cartwright &amp; Curien &amp; Felleisen)</td>
<td>YES</td>
</tr>
<tr>
<td>Control (ground, arbitrary)</td>
<td>Hyland-Ong games</td>
<td></td>
</tr>
<tr>
<td>State (ground, arbitrary)</td>
<td>Innocent strategies</td>
<td></td>
</tr>
<tr>
<td>(ground, arbitrary) State</td>
<td>Hyland-Ong games</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Observable parts of Innocent strategies</td>
<td></td>
</tr>
</tbody>
</table>
Semantic representation of PCF$_2$ observed by:

<table>
<thead>
<tr>
<th>Effect</th>
<th>Model</th>
<th>Decidable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>non-effective (quotient of terms/strategies)</td>
<td>NO (Loader)</td>
</tr>
<tr>
<td>Control</td>
<td>Sequential Algorithms</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>(Berry &amp; Curien, Cartwright &amp; Curien &amp; Felleisen)</td>
<td></td>
</tr>
<tr>
<td>Control (ground, arbitrary)</td>
<td>Hyland-Ong games</td>
<td>⇔ HORSE</td>
</tr>
<tr>
<td>State</td>
<td>Innocent strategies</td>
<td></td>
</tr>
<tr>
<td>(ground, arbitrary) State</td>
<td>Hyland-Ong games</td>
<td>⇔ HORSE</td>
</tr>
<tr>
<td>(ground, arbitrary) State</td>
<td>Observable parts of Innocent strategies</td>
<td></td>
</tr>
</tbody>
</table>
Hyland-Ong games and innocent strategies

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

\[
\text{lor} : \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}
\]

\[
q \overset{(O, Q)}{\rightarrow} \overset{(P, Q)}{\rightarrow} \overset{(O, A)}{\rightarrow} \overset{(P, A)}{\rightarrow} \overset{(O, A)}{\rightarrow} \overset{(P, A)}{\rightarrow}
\]

- 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:

\[
\text{lor} : \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}
\]

corresponds to the partial term:

\[
\lambda b_1^{\text{Bool}}. \lambda b_2^{\text{Bool}}. \text{if } b_1 \text{ then } [] \text{ else (if } b_2 \text{ then } \text{tt} \text{ else } [])
\]
P-views and branches of terms

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

\[
\begin{align*}
\text{Bool} & \rightarrow \text{Bool} \\
q & \rightarrow q \rightarrow \text{Bool} \\
\text{tt} & \rightarrow q \rightarrow \text{Bool} \\
(O, Q) & \rightarrow (P, Q) \\
(P, Q) & \rightarrow (O, A) \\
(q & \rightarrow (P, A) \\
\text{Bool} & \rightarrow \text{Bool} \\
q & \rightarrow q \rightarrow \text{Bool} \\
\text{ff} & \rightarrow q \rightarrow \text{Bool} \\
(O, A) & \rightarrow (P, Q) \\
(P, Q) & \rightarrow (O, A) \\
(ff & \rightarrow (P, A) \\
\text{in total giving the term:} \\
\lambda b_1^{\text{Bool}}. \lambda b_2^{\text{Bool}}. \text{if } b_1 \text{ then } [\text{tt}] \text{ else (if } b_2 \text{ then } \text{tt} \text{ else } [\text{ff}]) \\
\text{which we call the } \text{PCF Böhm tree (Curien)} \text{ of } \text{lor}.
\end{align*}
\]
Non purely functional behaviour

Control. The behaviour of call/cc is expressed by non well-bracketed plays:

\[ \text{call/cc} : ((\text{Bool} \rightarrow \text{Bool}) \rightarrow \text{Bool}) \rightarrow \text{Bool} \]

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

\[
\begin{array}{c}
\text{Bool} \\
q \\
tt \\
q \\
q \\
\end{array}
\begin{array}{c}
(\text{O, Q}) \\
(\text{P, Q}) \\
(\text{O, Q}) \\
(\text{P, Q}) \\
(\text{O, A}) \\
(\text{P, A}) \\
\end{array}
\]

Arbitrary plays can be realized using terms with state and control.
Innocent strategies as PCF Böhm trees. In general, innocent (well-bracketed) strategies can be represented syntactically:

\[
\Gamma \vdash \bot, tt, ff : \text{Bool} \\
\Gamma, \ldots, x_i : A_i, \ldots \vdash M : \text{Bool} \\
\Gamma \vdash \lambda \overrightarrow{x}. M : \overrightarrow{A} \rightarrow \text{Bool} \\
\Gamma \vdash M_i : A_i \quad \Gamma \vdash N_1, N_2 : \text{Bool} \quad (x : \overrightarrow{A} \rightarrow \text{Bool}) \in \Gamma \\
\Gamma \vdash \text{if } x \overrightarrow{M} \text{ then } N_1 \text{ else } N_2 : \text{Bool}
\]

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 indistingsuishable by a context with state and control.*
What if there is no control? The following PCF Böhm trees are distinct, but **indistinguishable** without call/cc:

\[
x : \text{Bool} \to \text{Bool} \vdash \text{if } x \text{ tt then } \bot \text{ else } \bot
\]

\[
x : \text{Bool} \to \text{Bool} \vdash \text{if } x \text{ ff then } \bot \text{ else } \bot
\]

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 then/else branches, we can eventually reach a \( tt/ff \).

**Proposition**

Two terms \( M_1, M_2 \) of \( \text{PCF}_2 \) 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 $\lambda Y$-calculus and Böhm trees with binders
A first step: the $\lambda Y$-calculus

We first treat the boolean-free case – the $\lambda Y$-calculus.

Types. Simple types on one atom $o$.

\[ A, B ::= o \mid A \rightarrow B \]

Terms.

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

Typing rules.

\[
\begin{align*}
\Gamma, x : A & \vdash x : A \\
\Gamma & \vdash Y_A : (A \rightarrow A) \rightarrow A \\
\Gamma & \vdash \lambda x^A.M : A \rightarrow B
\end{align*}
\]

\[
\begin{align*}
\Gamma & \vdash M : A \rightarrow B \\
\Gamma & \vdash N : A \\
\Gamma & \vdash M \ N : B
\end{align*}
\]

Reduction.

\[
\begin{align*}
(\lambda x^A.M) \ N & \rightsquigarrow_\beta M[N/x] \\
Y_A \ M & \rightsquigarrow_\delta M (Y_A \ M) \\
M^{A \rightarrow B} & \rightsquigarrow_\eta \lambda x^A.M \ x \quad (x \ \text{fresh})
\end{align*}
\]
Böhm trees of \(\lambda Y\)-terms

\(\lambda Y\)-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 \(\Sigma\) correspond to \(\lambda Y\)-terms:

\[
\Sigma \vdash M : o
\]

Böhm trees of \(\lambda Y\)-terms. If \(\Gamma \vdash M : A\) is a \(\lambda Y\)-term then its Böhm tree:

\[
\Gamma \vdash BT(M) : A
\]

is an infinitary term, defined as the upper bound of \(\eta\)-long \(\beta\)-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 \(\lambda Y\)-terms, as for \(\text{PCF}_2\) they correspond to innocent strategies.
Binders in open Böhm trees

Consider the term $G$:

$$Y_{o \rightarrow (o \rightarrow o) \rightarrow o} (\lambda f^{o \rightarrow (o \rightarrow o) \rightarrow o} \cdot \lambda y^o \cdot \lambda x^{o \rightarrow o} \cdot b (x \ y) (f \ (x \ y))) \ a$$

Its Böhm tree starts with:
Binders in open Böhm trees

Consider the term $G$:

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

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 $G$. 

![Diagram of Böhm tree with nodes and arrows]
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 $\lambda$-$Y$-terms.
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 $\lambda Y$-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

\[ g : o \rightarrow o \rightarrow o \vdash \lambda f. f \ (\lambda x. g \ x \ (f \ (\lambda x. g \ x \ (f \ x))) \]

is represented by:

\[ 0 : o \rightarrow o \rightarrow o \vdash \lambda 1.1 \ (\lambda 2.0 \ 2 \ (1 \ (\lambda 3.0 \ 3 \ (1 \ 3)))) \]

Proposition

Two terms \( M \) and \( M' \) have the same De Bruijn levels representation iff they are \( \alpha \)-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:

\[
\begin{align*}
z & : o \\
succ & : o \to o \\
var & : o \to o \\
app & : o \to o \to o \\
lam & : o \to o \to o
\end{align*}
\]

Or in other terms:

**Definition**

If \( \vdash M : A \) is a \( \lambda \)-term, it has as De Bruijn representation, a Böhm tree:

\[
\Gamma_{rep} \vdash \text{rep}(M) : o
\]

where \( \Gamma_{rep} \) 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, \( \text{rep}(M) \) is!
Representation of $\lambda$-terms in $\Gamma_{\text{rep}}$

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

 así:

```
(Keeping in mind that $n = \text{succ} \ (\text{succ} \ldots (\text{succ} \ z)\ldots))
```
III.3 Internal Normalization by Evaluation
Normalization by evaluation

Semantic technique for computing normal forms of \( \lambda \)-terms.

**Theorem (Martin-Löf, Berger & Schwichtenberg)**

*There is a set-theoretic interpretation of the simply-typed \( \lambda \)-calculus:*

\[
[-] : \Lambda \to \text{Set}
\]

*and for each type \( A \), a function*

\[
\text{reify} : [A] \to \Lambda
\]

*such that for each term \( \vdash M : A \),*

\[
\text{reify}([M]) \cong_{\beta\eta} M
\]

*is the \( \beta \)-normal \( \eta \)-long form of \( M \).*
Normalization by evaluation for the simply-typed $\lambda$-calculus

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

$$\begin{align*}
[\text{o}] &= E \\
[x]_\rho &= \rho(x) \\
[M \; N]_\rho &= [M]_\rho([N]_\rho)
\end{align*}$$

$[\lambda x^A. M]_\rho = \lambda a^{[A]}.[M]_{\rho \oplus \{x \mapsto a\}}$

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:

$$\begin{align*}
\Downarrow_A &: \quad [[A]] \to E \\
\Downarrow_\circ x &= x \\
\Downarrow_{A \to B} x &= \\
\Upparrow_A &: \quad E \to [[A]] \\
\Upparrow_\circ e &= e \\
\Upparrow_{A \to B} e &= \\
\text{by setting } \text{nbe}(M) = \Downarrow_A [[M]].
\end{align*}$$
Normalization by evaluation for the simply-typed $\lambda$-calculus

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

$$
\begin{align*}
[[\text{o}]] &= E \\
[[x]]_{\rho} &= \rho(x) \\
[[M \ N]]_{\rho} &= [[M]]_{\rho}([[N]]_{\rho})
\end{align*}
$$

$[[A \rightarrow B]] = [[A]] \rightarrow [[B]]$

$$
\begin{align*}
[[\lambda x^A. M]]_{\rho} &= \lambda a^{[A]} [[M]]_{\rho \oplus \{ x \mapsto a \}}
\end{align*}
$$

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:

$$
\begin{align*}
\downarrow_A & : [[A]] \rightarrow E \\
\downarrow_\text{o} x &= x \\
\downarrow_{A \rightarrow B} x &= \text{lam} \ n \ ( \ ) \ (n \ \text{fresh}) \\
\uparrow_A & : E \rightarrow [[A]] \\
\uparrow_{\text{o}} e &= e \\
\uparrow_{A \rightarrow B} e &= \\
\text{by setting } \text{nbe}(M) = \downarrow_A [[M]].
\end{align*}
$$
Normalization by evaluation for the simply-typed \( \lambda \)-calculus

**Step 1: Interpretation.** Let \( E \) be a set containing representations of terms.

\[
\begin{align*}
[\circ] & = E \\
[x]_{\rho} & = \rho(x) \\
[M \ N]_{\rho} & = [M]_{\rho}(\[N\]_{\rho})
\end{align*}
\]

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

\[
\begin{align*}
\downarrow_A : & \quad [A] \to E \\
\downarrow_{\circ} x & = x \\
\downarrow_{A \to B} x & = \text{lam } n \ (x \quad ) (n \ \text{fresh})
\end{align*}
\]

\[
\begin{align*}
\uparrow_A : & \quad E \to [A] \\
\uparrow_{\circ} e & = e \\
\uparrow_{A \to B} e & =
\end{align*}
\]

by setting \( \text{nbe}(M) = \downarrow_A [M] \).
Normalization by evaluation for the simply-typed \( \lambda \)-calculus

**Step 1: Interpretation.** Let \( E \) be a set containing representations of terms.

\[
\begin{align*}
\llbracket o \rrbracket &= E \\
\llbracket x \rrbracket_\rho &= \rho(x) \\
\llbracket M \ N \rrbracket_\rho &= \llbracket M \rrbracket_\rho(\llbracket N \rrbracket_\rho)
\end{align*}
\]

\( [A \rightarrow B] = [A] \rightarrow [B] \)

\( [\lambda x^A . M]_\rho = \lambda a^{[A]} . [M]_\rho \oplus \{ x \mapsto a \} \)

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 \( \llbracket M \rrbracket \) by the following:

\[
\begin{align*}
\downarrow_A &= [A] \rightarrow E \\
\downarrow_o x &= x \\
\downarrow_{A \rightarrow B} x &= \text{lam } n \left( x \quad (\text{var } n) \right) \quad (n \ \text{fresh}) \\
\end{align*}
\]

\[
\begin{align*}
\uparrow_A &= E \rightarrow [A] \\
\uparrow_o e &= e \\
\uparrow_{A \rightarrow B} e &=
\end{align*}
\]

by setting \( \text{nbe}(M) = \downarrow_A [M] \).
Normalization by evaluation for the simply-typed $\lambda$-calculus

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

$$
\begin{align*}
[o] & = E \\
[x]_\rho & = \rho(x) \\
[M \cdot N]_\rho & = [M]_\rho([N]_\rho)
\end{align*}
$$

$[A \rightarrow B] = [A] \rightarrow [B]$

$[\lambda x^A.M]_\rho = \lambda a^{[A]}.[M]_\rho \oplus \{x \mapsto a\}$

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:

$$
\begin{align*}
\Downarrow_A : [A] \rightarrow E \\
\Downarrow_0 x & = x \\
\Downarrow_{A \rightarrow B} x & = \text{lam } n \left( x \left( \uparrow_A \left( \text{var } n \right) \right) \right) \ (n \text{ fresh})
\end{align*}
$$

$$
\begin{align*}
\uparrow_A : E \rightarrow [A] \\
\uparrow_0 e & = e \\
\uparrow_{A \rightarrow B} e & = \\
\end{align*}
$$

by setting $\text{nbe}(M) = \Downarrow_A [M]$. 
Normalization by evaluation for the simply-typed λ-calculus

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

\[
\begin{align*}
\llbracket o \rrbracket &= E \\
\llbracket x \rrbracket^\rho &= \rho(x) \\
\llbracket M \; N \rrbracket^\rho &= \llbracket M \rrbracket^\rho(\llbracket N \rrbracket^\rho)
\end{align*}
\]

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 $\llbracket M \rrbracket$ by the following:

\[
\begin{align*}
\downarrow_A &= \llbracket A \rrbracket \to E \\
\downarrow_o \; x &= x \\
\downarrow_{A \to B} \; x &= \text{lam} \; n \left( \downarrow_B \left( x \llbracket A \to B \rrbracket \text{var} \; n \right) \right) \quad (n \text{ fresh})
\end{align*}
\]

\[
\begin{align*}
\uparrow_A &= E \to \llbracket A \rrbracket \\
\uparrow_o \; e &= e \\
\uparrow_{A \to B} \; e &= \text{by setting } \text{nbe}(M) = \downarrow_A \; \llbracket M \rrbracket.
\end{align*}
\]
Normalization by evaluation for the simply-typed $\lambda$-calculus

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

\[
\begin{align*}
\llbracket o \rrbracket &= E \\
\llbracket x \rrbracket_\rho &= \rho(x) \\
\llbracket M \quad N \rrbracket_\rho &= \llbracket M \rrbracket_\rho(\llbracket N \rrbracket_\rho)
\end{align*}
\]

$[A \to B] = [A] \to [B]$

$[\lambda x^A.M]_\rho = \lambda a^{[A]}.[M]_{\rho \oplus \{x \mapsto a\}}$

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 $\llbracket M \rrbracket$ by the following:

\[
\begin{align*}
\downarrow_A : [A] &\to E \\
\downarrow o \quad x &= x \\
\downarrow_{A \to B} \quad x &= \text{lam} \quad n \ (\downarrow_B (x \ (\uparrow_A (\text{var} \quad n)))) \ (n \text{ fresh})
\end{align*}
\]

\[
\begin{align*}
\uparrow_A : E &\to [A] \\
\uparrow o \quad e &= e \\
\uparrow_{A \to B} \quad e &= \lambda x^{[A]}
\end{align*}
\]

by setting $\text{nbe}(M) = \downarrow_A \llbracket M \rrbracket$. 

Normalization by evaluation for the simply-typed $\lambda$-calculus

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

$[o] = E$

$[\lambda x^A.M]_\rho = \lambda a \uparrow [A].[M]_{\rho \oplus \{x\mapsto a\}}$

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

$[A \rightarrow B] = [A] \rightarrow [B]$

$[\lambda x^A.M]_\rho = \lambda a \uparrow [A].[M]_{\rho \oplus \{x\mapsto a\}}$

$\uparrow [A] : E \rightarrow [A]$

$\uparrow [o] e = \lambda x^A.M_ho = \lambda a \uparrow [A].[M]_{\rho \oplus \{x\mapsto a\}}$

$\uparrow [A \rightarrow B] e = \lambda x^A.M_ho = \lambda a \uparrow [A].[M]_{\rho \oplus \{x\mapsto a\}}$

$\downarrow [o] x = x$

$\downarrow [A \rightarrow B] x = \mathit{lam} n \left( \downarrow [B] \left( x \left( \uparrow [A] (\mathit{var} n) \right) \right) \right)$ (n fresh)

$\downarrow [A] \rightarrow E$

$\downarrow [A \rightarrow B] x = \mathit{lam} n \left( \downarrow [B] \left( x \left( \uparrow [A] (\mathit{var} n) \right) \right) \right)$ (n fresh)

by setting $\mathit{nbe}(M) = \downarrow [A] [M]$. 

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

$\uparrow [A] : E \rightarrow [A]$

$\uparrow [o] e = e$

$\uparrow [A \rightarrow B] e = \lambda x^A.M_ho = \lambda a \uparrow [A].[M]_{\rho \oplus \{x\mapsto a\}}$
Normalization by evaluation for the simply-typed \(\lambda\)-calculus

**Step 1: Interpretation.** Let \(E\) be a set containing representations of terms.

\[
\begin{align*}
\llbracket o \rrbracket &= E \\
\llbracket x \rrbracket_\rho &= \rho(x) \\
\llbracket M \ N \rrbracket_\rho &= \llbracket M \rrbracket_\rho(\llbracket N \rrbracket_\rho)
\end{align*}
\]

\[
\begin{align*}
\llbracket A \to B \rrbracket &= \llbracket A \rrbracket \to \llbracket B \rrbracket \\
\llbracket \lambda x^A.M \rrbracket_\rho &= \lambda a^{[A]}.[M]_{\rho \oplus \{x \mapsto a\}}
\end{align*}
\]

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 \(\llbracket M \rrbracket\) by the following:

\[
\begin{align*}
\downarrow_A : & \quad \llbracket A \rrbracket \to E \\
\downarrow_o x &= x \\
\downarrow_{A\to B} x &= \text{lam } n \left( \downarrow_B \left( x \left( \uparrow_A (\text{var } n) \right) \right) \right) \quad (n \text{ fresh})
\end{align*}
\]

\[
\begin{align*}
\uparrow_A : & \quad E \to \llbracket A \rrbracket \\
\uparrow_o e &= e \\
\uparrow_{A\to B} e &= \lambda x^{[A]}. \quad \text{app } e \left( \downarrow_A x \right)
\end{align*}
\]

by setting \(\text{nbe}(M) = \downarrow_A \llbracket M \rrbracket\).
Normalization by evaluation for the simply-typed $\lambda$-calculus

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

$$
\begin{align*}
\llbracket o \rrbracket &= E \\
\llbracket x \rrbracket_\rho &= \rho(x) \\
\llbracket M \mathbin{;} N \rrbracket_\rho &= \llbracket M \rrbracket_\rho(\llbracket N \rrbracket_\rho)
\end{align*}
$$

$\llbracket A \to B \rrbracket = [A] \to [B]$  
$\llbracket \lambda x^A . M \rrbracket_\rho = \lambda a^{[A]} . [M]_\rho \oplus \{x \mapsto a\}$

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 $\llbracket M \rrbracket$ by the following:

$$
\begin{align*}
\mathbin{\downarrow}_A & : [A] \to E \\
\mathbin{\downarrow}_o x &= x \\
\mathbin{\downarrow}_{A \to B} x &= \text{lam } n \left( \mathbin{\downarrow}_B \left( x \left( \mathbin{\uparrow}_A \left( \text{var } n \right) \right) \right) \right) \quad (n \text{ fresh})
\end{align*}
$$

$$
\begin{align*}
\mathbin{\uparrow}_A & : E \to [A] \\
\mathbin{\uparrow}_o e &= e \\
\mathbin{\uparrow}_{A \to B} e &= \lambda x^{[A]} . \mathbin{\uparrow}_B \text{app } e \left( \mathbin{\downarrow}_A x \right)
\end{align*}
$$

by setting $\text{nbe}(M) = \mathbin{\downarrow}_A \llbracket M \rrbracket$. 
Expressions. $e \in E$ are replaced with **indexed expressions**

$$f \in \mathbb{N} \rightarrow E = \hat{E}$$

Constructors. $\text{var}, \text{lam}, \text{app}$ are replaced with variants:

$$\hat{\text{var}} = \lambda v^N. \lambda n^N. \text{var} \; v : N \rightarrow \hat{E}$$

$$\hat{\text{app}} = \lambda e_1^\hat{E}. \lambda e_2^\hat{E}. \lambda n^N. \text{app} (e_1 \; n) (e_2 \; n) : \hat{E} \rightarrow \hat{E} \rightarrow \hat{E}$$

$$\hat{\text{lam}} = \lambda f^{N \rightarrow \hat{E}}. \lambda n^N. \text{lam} \; n \; (f \; n \; (\text{succ} \; n)) : (N \rightarrow \hat{E}) \rightarrow \hat{E}$$

Reify and reflect. They are generalized:

$$\downarrow_o x = x$$
$$\uparrow_o e = e$$

$$\downarrow_{A \rightarrow B} x = \hat{\text{lam}} (\lambda n^N. \downarrow_B (x (\uparrow_A \hat{\text{var}} \; n)))$$

$$\uparrow_{A \rightarrow B} e = \lambda x^{[A]}. \uparrow_B \hat{\text{app}} \; e (\downarrow_B x)$$

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

$$\text{nbe}(M) = \downarrow_A [M]_0$$
Continuity and NBE for $\lambda Y$

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

**Model.** Standard pointed $\omega$-cpo model of the $\lambda Y$-calculus:

$$
\begin{align*}
\llbracket o \rrbracket & = \hat{E} \\
\llbracket Y_A \rrbracket & = \lambda f^A. \bigsqcup_n f^n(\bot)
\end{align*}
$$

where $E$ is the pointed $\omega$-cpo of possibly infinite expressions.

**Normalization by evaluation.** From a $\lambda Y$-term $\vdash M : A$,

$$nbe(M) = \downarrow_A \llbracket M \rrbracket 0 \in E$$

$\rightarrow$ Obtain an infinite normal form.

**Proof.** By induction for finite normal forms, by soundness and continuity arguments for arbitrary terms.
Step 2. We internalize NBE for $\lambda Y$ within the $\lambda Y$-calculus.

**Expressions** are terms of $\lambda Y$:

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

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

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

**Term formers** are the following:

- $\widehat{\text{var}} = \lambda v^o. \lambda n^o. \text{var } v$
- $\widehat{\text{lam}} = \lambda f^{o\rightarrow o\rightarrow o}. \lambda n^o. \text{lam } n (f n (\text{succ } n))$
- $\widehat{\text{app}} = \lambda e_1^{o\rightarrow o}. \lambda e_2^{o\rightarrow o}. \lambda n^o. \text{app } (e_1 n) (e_2 n)$

**Reify/reflect** are now terms of the $\lambda Y$-calculus:

- $\downarrow o = \lambda x^o.x$
- $\downarrow_{A\rightarrow B} = \lambda x^{A^*\rightarrow B^*}. \text{lam } (\lambda n^o. \downarrow_B (x (\uparrow_A \text{var } n)))$
- $\uparrow o = \lambda e^o.e$
- $\uparrow_{A\rightarrow B} = \lambda e^o. \lambda x^{A^*}. \uparrow_B \text{app } e (\downarrow_B x)$
Internalization within $\lambda Y$

**Theorem**

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

$$\Gamma_{rep} \vdash_{A} M^* \bar{0} : o$$

satisfies:

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

Moreover, the construction increases the **order** by one.
The NBE translation for PCF₂

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

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

Semantics. Domain semantics of PCF, based on:

$$[[\text{Bool}]] = \hat{E} \rightarrow \hat{E} \rightarrow \hat{E}$$

Reflect and reify. Extended to booleans with:

$$\downarrow_A : [[A]] \rightarrow \hat{E} \quad \uparrow_A : \hat{E} \rightarrow [[A]]$$

$$\downarrow_{\text{Bool}} x = x \hat{tt} \hat{ff} \quad \uparrow_{\text{Bool}} e = \lambda x^{\hat{E}}. \lambda y^{\hat{E}}. \hat{\text{if}} \ e \ x \ y$$

with $\hat{tt} = \lambda_. tt$, $\hat{ff} = \lambda_. ff$, $\hat{\text{if}} = \lambda e_1^{\hat{E}}. \lambda e_2^{\hat{E}}. \lambda n^{\hat{E}}. \text{if} \ (e_1 \ n) \ (e_2 \ n)$.

Internalization. Follows the same lines as for $\lambda Y$. 
IV. Conclusions
Consequences

Theorem

For any term $\Gamma \vdash M : A$ of $\text{PCF}_2$, 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 $\text{PCF}_2$, 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).
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 \(\lambda Y/\text{PCF}_2\) term equivalent to a \(Y\)-free term?
- **Finiteness**: Has a \(\lambda Y/\text{PCF}_2\) term a finite Böhm tree/strategy?
- **Solvability**: Has a \(\lambda Y/\text{PCF}_2\) term a head normal form?
- **Finite prefix**: Has a \(\lambda Y/\text{PCF}_2\) term a finite/regular prefix?

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