A model of PCF in Guarded Type Theory

Marco Paviotti\textsuperscript{1}  Rasmus Møgelberg\textsuperscript{1}  Lars Birkedal\textsuperscript{2}

\textsuperscript{1}IT University of Copenhagen
\textsuperscript{2}Aarhus University

June 23th, 2015
MFPS 2015
Nijmegen, Netherlands
In Type Theory

unrestricted fix-point \( \text{fix}: (A \rightarrow A) \rightarrow A \) is inconsistent

\( \text{e.g.} \ \text{fix(id)} : A \) leads to every type to be inhabited

In Guarded Type Theory

restricted fix-points are allowed by using the \( \triangleright \) operator

- \( \text{next}: A \rightarrow \triangleright A \)
- \( \triangleright\!\times: \triangleright(A \rightarrow B) \rightarrow \triangleright A \rightarrow \triangleright B \)
- \( \text{fix}: (\triangleright A \rightarrow A) \rightarrow A \) \quad \text{s.t.} \quad f(\text{next}(\text{fix}(f))) = \text{fix}(f) \)
- \( X \cong A \times \triangleright X \)
In Type Theory

unrestricted fix-point \( \text{fix}: (A \to A) \to A \) is inconsistent
  e.g. \( \text{fix}(\text{id}): A \) leads to every type to be inhabited

In Guarded Type Theory

restricted fix-points are allowed by using the \( \triangleright \) operator

- next : \( A \to \triangleright A \)
- \( \otimes: \triangleright (A \to B) \to \triangleright A \to \triangleright B \)
- fix: \( \triangleright A \to A \) \( \) s.t. \( f(\text{next}(\text{fix}(f))) = \text{fix}(f) \)
- \( X \cong A \times \triangleright X \)
Stream $A \cong A \times \text{Str}_A$

Streams in Coq

- ones = 1 :: ones
- bad = tail bad
- nats = 0 :: map (1 +) nats
Streams

\[ \text{Str}_A \cong A \times \text{Str}_A \]

**Streams in Coq**

- ones = 1 :: ones ✓
- bad = tail bad ✗
- nats = 0 :: map (1 +) nats ✗

**Guarded Streams**

\[ \text{Str}^g_A \cong A \times \triangleright \text{Str}^g_A \]

- ones = 1 :: ones : \text{Str}^g_A ✓
- bad = tail bad :\text{Str}^g_A ✗
- nats = 0 :: next(map (1 +)) ⊙ nats : \text{Str}^g_A ✓
The category of presheaves over $\omega$

\[
\begin{align*}
X & \xleftarrow{r_1} X(1) & r_2 & \xleftarrow{r_3} X(2) & \ldots & \xleftarrow{r_{n-1}} X(n-1) & \xleftarrow{r_n} X(n-1) & \ldots \\
\triangleright X & 1 & \xleftarrow{!} X(1) & \ldots & \xleftarrow{r_{n-2}} X(n-1) & \xleftarrow{r_n} X(n-1) & \ldots
\end{align*}
\]

$\text{Str}^g_A \cong A \times \triangleright \text{Str}^g_A$

**Guarded Streams**

\[
\begin{align*}
\text{Str}^g_A & \quad A \times 1 \xleftarrow{r_1} A \times (A \times 1) \xleftarrow{r_2} A \times (A \times A \times 1) \\
\triangleright \text{Str}^g_A & \quad 1 \xleftarrow{!} A \times 1 \xleftarrow{r_2} A \times A \times 1 \\
A \times \triangleright \text{Str}^g_A & \quad A \times 1 \xleftarrow{r_1} A \times A \times 1 \xleftarrow{r_2} A \times A \times A \times 1
\end{align*}
\]
Can we do denotational semantics in Guarded Type Theory?

in particular, is it possible to model recursion with guarded recursion?
Can we do denotational semantics in Guarded Type Theory?

in particular, *is it possible to model recursion with guarded recursion?*

- **Motivations** Mechanising denotational semantics in a proof-assistant
- **Contributions**
  + Model of PCF in GTT
  + Adequacy Theorem proved in GTT

Similar to Escardo’s metric model \(^1\), but here the whole development is *entirely carried out within guarded type theory*

---

\(^1\)M.H. Escardo, “A metric model of PCF”. Presented at the *Workshop on Realizability Semantics and Applications*, 1999
Outline

- Operational Semantics of PCF
- Denotational Semantics
- Computational Adequacy
- Discussion
PCF

\[ \sigma, \tau := \text{nat} \mid \sigma \rightarrow \tau \]

\[ L, M, N := n \mid x \mid \lambda x. M \mid \text{pred} \ M \mid \text{succ} \ M \mid Y \ M \mid \text{ifz} \ L \ M \ N \]

\[
\begin{align*}
\Gamma, x : \sigma, \Delta \vdash x : \sigma & \quad & \Gamma \vdash n : \text{nat} \\
\Gamma, x : \sigma \vdash M : \tau & \quad & \Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma \\
\Gamma \vdash (\lambda x : \sigma. M) : \sigma \rightarrow \tau & \quad & \Gamma \vdash MN : \tau \\
\Gamma \vdash M : \text{nat} & \quad & \Gamma \vdash M : \text{nat} \\
\Gamma \vdash \text{succ} \ M : \text{nat} & \quad & \Gamma \vdash \text{pred} \ M : \text{nat} \\
\Gamma \vdash M : \sigma \rightarrow \sigma & \quad & \Gamma \vdash Y_\sigma \ M : \sigma \\
\Gamma \vdash L : \text{nat} & \quad & \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma \\
\Gamma \vdash \text{ifz} \ L \ M \ N : \sigma
\end{align*}
\]
The big-step relation is defined by induction on terms and indexes:

\[ M \downarrow^k Q \]
The big-step relation is defined by induction on terms and indexes:

\[ M \downarrow^k Q \]

explicit step counting
Big-step semantics

The big-step relation is defined by induction on terms and indexes:

\[ M \Downarrow^k Q \]

explicit step counting

**Predicates on values**

can define \( M \Downarrow^k v \) as \( M \Downarrow^k \lambda v'. v \equiv v' \)
The big-step relation is defined by induction on terms and indexes:

\[ M \Downarrow^k Q \]

**explicit step counting**

- \( \nu \Downarrow^0 Q \overset{\text{def}}{=} Q(\nu) \)

**Predicates on values**

- can define \( M \Downarrow^k \nu \) as \( M \Downarrow^k \lambda \nu'. \nu \equiv \nu' \)
Big-step semantics

The big-step relation is defined by induction on terms and indexes:

\[ M \downarrow^k Q \]

**explicit step counting**

\[ \nu \downarrow^0 Q \overset{\text{def}}{=} Q(\nu) \]

\[ MN \downarrow^{k+m} Q \overset{\text{def}}{=} M \downarrow^k Q' \]

where \( Q'(\lambda x.L) = L[N/x] \downarrow^m Q \)

**Predicates on values**

can define \( M \downarrow^k \nu \) as \( M \downarrow^k \lambda \nu'.\nu = \nu' \)
Big-step semantics

The big-step relation is defined by induction on terms and indexes:

\[ M \downarrow^k Q \]

**explicit step counting**

\[ \nu \downarrow^0 Q \overset{\text{def}}{=} Q(\nu) \]
\[ MN \downarrow^{k+m} Q \overset{\text{def}}{=} M \downarrow^k Q' \]
where \( Q'(\lambda x. L) = L[N/x] \downarrow^m Q \)
\[ Y_\sigma M \downarrow^{k+1} Q \overset{\text{def}}{=} \triangleright(M(Y_\sigma M) \downarrow^k Q) \]
The big-step relation is defined by induction on terms and indexes:

\[ M \Downarrow^k Q \]

### Predicates on values

- \( v \Downarrow^0 Q \) is defined as \( Q(v) \)
- \( MN \Downarrow^{k+m} Q \) is defined as \( M \Downarrow^k Q' \)
  - where \( Q'(\lambda x.L) = L[N/x] \Downarrow^m Q \)
- \( Y_\sigma M \Downarrow^{k+1} Q \) is defined as \( \triangleright (M(Y_\sigma M) \Downarrow^k Q) \)

### Synchronising with the type theory

- Explicit step counting

### Can define

- \( M \Downarrow^k \lambda v'.v = v' \)
Small-Step Operational Semantics

\[(\lambda x : \sigma. M)(N) \to^0 M[N/x] \quad Y_\sigma \ M \to^1 M(Y_\sigma \ M)\]

\[M \to^k M' \quad \frac{}{M(N) \to^k M'(N)}\]

Let \(\to^0_*\) be the reflexive, transitive closure of \(\to^0\).

\[M \Rightarrow^0 Q \overset{\text{def}}{=} \Sigma N : \text{Term}_{\text{PCF}}. M \to^0_* N \text{ and } Q(N)\]

\[M \Rightarrow^{k+1} Q \overset{\text{def}}{=} \Sigma M', M'' : \text{Term}_{\text{PCF}}. M \to^0_* M'\]

\[\text{and } M' \to^1 M'' \text{ and } \triangleright(M'' \Rightarrow^k Q)\]

Define \(M \Rightarrow^k v\) as \(M \Rightarrow^k \lambda v'. v = v'\)

**Lemma**

\[M \Downarrow^k v \iff M \Rightarrow^k v\]
Outline

- Operational Semantics of PCF
- Denotational Semantics
- Computational Adequacy
- Discussion
Lifting Monad

\[ LA \cong A + \triangleright LA \]

Lifting monad

- \( \eta : A \rightarrow LA \) \hspace{1cm} \( \theta : \triangleright LA \rightarrow LA \)
- Time step operation: \( \delta = \theta \circ \text{next} : LA \rightarrow LA \)
- Bottom element \( \bot = \text{fix}(\theta) \)
- \( LA \) is a free \( \triangleright \)-algebra on \( A \)
- \( L \) is the guarded recursive version of Capretta’s partiality monad\(^1\)

\(^1\)Venanzio Capretta, “General Recursion via Co-Inductive Types”, In *Logical Methods in Computer Science*, 2005
Lifting monad

\[ LA \cong A + \triangleright LA \]

\[ LN \cong N + \triangleright LN \]

\[ \begin{array}{ccc}
LN & N + 1 & \xleftarrow{r_1} \ N + N + 1 & \xleftarrow{r_2} \ N + N + N + 1 \\
\triangleright LN & 1 & \xleftarrow{!} \ N + 1 & \xleftarrow{r_1} \ N + N + 1 \\
N + \triangleright LN & N + 1 & \xleftarrow{r_1} \ N + N + 1 & \xleftarrow{r_2} \ N + N + N + 1 \\
\end{array} \]
• Interpreting Types

\[
[nat] \overset{\text{def}}{=} LN
\]

\[
[\tau \rightarrow \sigma] \overset{\text{def}}{=} [\tau] \rightarrow [\sigma]
\]

• All types are \(\triangleright\)-algebras with \(\theta_\sigma : \triangleright[\sigma] \rightarrow [\sigma]\)

• Interpreting terms \([t] : [\Gamma] \rightarrow [\sigma]\)

\[
[\Gamma \vdash Y_\sigma \ M](\gamma) = (\text{fix}_{[\sigma]})(\lambda x : \triangleright[\sigma].\theta_\sigma(\text{next}([M](\gamma)))) \ast x)
\]
Interpreting PCF

- Interpreting Types

\[
\begin{align*}
\llbracket \text{nat} \rrbracket & \overset{\text{def}}{=} LN \\
\llbracket \tau \rightarrow \sigma \rrbracket & \overset{\text{def}}{=} \llbracket \tau \rrbracket \rightarrow \llbracket \sigma \rrbracket
\end{align*}
\]

- All types are \(\triangleright\)-algebras with \(\theta_{\sigma} : \triangleright[\sigma] \rightarrow [\sigma]\)

- Interpreting terms \([t] : [\Gamma] \rightarrow [\sigma]\)

\[
\llbracket \Gamma \vdash Y_{\sigma} \ M \rrbracket(\gamma) = (\text{fix}_{[\sigma]})(\lambda x : \triangleright[\sigma]. \theta_{\sigma}(\text{next}(\llbracket M \rrbracket(\gamma)))) \odot x)
\]

\text{can be thought of}
\[\theta \circ \triangleright[\text{M}]\]
Interpreting PCF

• Interpreting Types

\[
\begin{align*}
\llbracket \text{nat} \rrbracket & \xlongequal{\text{def}} LN \\
\llbracket \tau \to \sigma \rrbracket & \xlongequal{\text{def}} \llbracket \tau \rrbracket \to \llbracket \sigma \rrbracket
\end{align*}
\]

• All types are \(\triangleright\)-algebras with \(\theta_\sigma : \triangleright[\sigma] \to [\sigma]\)

• Interpreting terms \(\llbracket t \rrbracket : \llbracket \Gamma \rrbracket \to \llbracket \sigma \rrbracket\)

\[
\llbracket \Gamma \vdash Y_\sigma. M \rrbracket(\gamma) = (\text{fix}[\sigma])(\lambda x : \triangleright[\sigma]. \theta_\sigma(\text{next}(\llbracket M \rrbracket(\gamma)))) \star x)
\]

Lemma

Let \(\Gamma \vdash M : \sigma \to \sigma\) then \(\llbracket Y_\sigma. M \rrbracket = \delta_\sigma \circ [M(Y_\sigma. M)]\)
Soundness

Theorem (Soundness)

Let $M$ be a closed term of type $\tau$, if $M \downarrow^k \nu$ then

\[
\llbracket M \rrbracket(\star) = \delta^k \llbracket \nu \rrbracket(\star)
\]
Outline

- Operational Semantics of PCF
- Denotational Semantics
- **Computational Adequacy**
  
  \[
  \text{if } \llbracket M \rrbracket(\ast) = \delta^k \llbracket \nu \rrbracket(\ast) \text{ then } M \downarrow^k \nu
  \]

- Discussion
Adequacy proved by (proof-relevant) logical relation

\[ d \, \mathcal{R}_\tau \, M \]

Define \( \mathcal{R}_\tau \) by induction on \( \tau \)

\( \eta(v) \, \mathcal{R}_{\text{nat}} \, M \overset{\text{def}}{=} M \Downarrow^0 v \)

\( \theta_{\text{nat}}(r) \, \mathcal{R}_{\text{nat}} \, M \overset{\text{def}}{=} \Sigma M', M'' : \text{Term}_{\text{PCF}}.M \rightarrow^0 M' \)

and \( M' \rightarrow^1 M'' \) and \( r \, \mathcal{R}_{\text{nat}} \, \text{next}(M'') \)
Adequacy proved by (proof-relevant) logical relation

\[ d \mathcal{R}_\tau M \]

Define \( \mathcal{R}_\tau \) by induction on \( \tau \)

\[ \eta(v) \mathcal{R}_{\text{nat}} M \overset{\text{def}}{=} M \Downarrow^0 v \]

\[ \theta_{\text{nat}}(r) \mathcal{R}_{\text{nat}} M \overset{\text{def}}{=} \sum M', M'' : \text{Term}_{\text{PCF}}. M \rightarrow^0 M' \]

and \( M' \rightarrow^1 M'' \) and \( r >\mathcal{R}_{\text{nat}} \) next(\( M'' \))

\[ LN \cong N + >LN \]

an element in this type is either of the form \( \eta(v) \) or \( \theta_{\text{nat}}(r) \)
Adequacy proved by (proof-relevant) logical relation

\[ d \; \mathcal{R}_\tau \; M \]

Define \( \mathcal{R}_\tau \) by induction on \( \tau \)

\[
\eta(v) \; \mathcal{R}_{\text{nat}} \; M \overset{\text{def}}{=} M \Downarrow^0 v
\]

\[
\theta_{\text{nat}}(r) \; \mathcal{R}_{\text{nat}} \; M \overset{\text{def}}{=} \sum M', M'' : \text{Term}_{\text{PCF}}.M \rightarrow^0 M'
\]

and \( M' \rightarrow^1 M'' \) and \( r \; \mathcal{R}_{\text{nat}} \; \text{next}(M'') \)

Delayed Relation \( \triangleright \mathcal{R} \)

\[
t \; \triangleright \mathcal{R}_{\text{nat}} \; u
\]

delayed version of \( \mathcal{R} \)
Adequacy proved by (proof-relevant) logical relation

\[ d \; \mathcal{R}_\tau \; M \]

Define \( \mathcal{R}_\tau \) by induction on \( \tau \)

\[ \eta(v) \; \mathcal{R}_{\text{nat}} \; M \overset{\text{def}}{=} M \Downarrow^0 v \]

\[ \theta_{\text{nat}}(r) \; \mathcal{R}_{\text{nat}} \; M \overset{\text{def}}{=} \sum M', M'' : \text{Term}_{\text{PCF}}. M \rightarrow^0 M' \]

and \( M' \rightarrow^1 M'' \) and \( r \triangleright \mathcal{R}_{\text{nat}} \; \text{next}(M'') \)

\[ f \; \mathcal{R}_{\tau \rightarrow \sigma} \; M \overset{\text{def}}{=} \prod \alpha: [\tau], N : \text{Term}_{\text{PCF}}. \alpha \; \mathcal{R}_\tau \; N \implies f(\alpha) \; \mathcal{R}_\sigma \; (MN) \]
Lemma (Fundamental Lemma)

Let $\Gamma \vdash t : \tau$, suppose $\Gamma \equiv x_1 : \tau_1, \cdots, x_n : \tau_n$ and $t_i : \tau_i$, $\alpha_i : [\tau_i]$ and $\alpha_i \mathcal{R}_{[\tau_i]} t_i$ for $i \in \{1, \ldots, n\}$, then $\llbracket t \rrbracket(\overrightarrow{\alpha}) \mathcal{R}_\tau t[\overrightarrow{t}/\overrightarrow{x}]$.

Theorem (Computational Adequacy)

If $M$ is a closed term of type $\textbf{nat}$ then $M \downarrow^k v$ iff $\llbracket M \rrbracket(*) = \delta^k [v]$. 
Outline

• Operational Semantics of PCF

• Denotational Semantics

• Computational Adequacy

• Discussion
Type theory vs. Topos logic

Set_{\omega^{op}} also models the topos logic.

The following is derivable in the non-proof-relevant topos logic:

$$\exists k. \exists v. Y_{\text{nat}} (\lambda x. x) \Downarrow^k v$$

**Proof (Sketch)**

- The argument is by Guarded Recursion: assume
  $$\triangleright (\exists k. \exists v. Y_{\text{nat}} (\lambda x. x) \Downarrow^k v)$$
- by property of $\text{Set}_{\omega^{op}}$ $\exists k. \exists v. \triangleright(Y_{\text{nat}} (\lambda x. x) \Downarrow^k v)$
- which implies $\exists k. \exists v. Y_{\text{nat}} (\lambda x. x) \Downarrow^{k+1} v$

In Type Theory

$$\Sigma k. \Sigma v. Y_{\text{nat}} (\lambda x. x) \Downarrow^k v$$
is not globally inhabited
Type theory vs. Topos logic

Set\textsuperscript{\omega^\text{op}} also models the topos logic.

The following is derivable in the non-proof-relevant topos logic:

\[ \exists k. \exists v. Y_{\text{nat}}(\lambda x.x) \Downarrow^k v \]

Proof (Sketch)

- The argument is by Guarded Recursion: assume \[ \triangleright (\exists k. \exists v. Y_{\text{nat}}(\lambda x.x) \Downarrow^k v) \]
- by property of Set\textsuperscript{\omega^\text{op}} \[ \exists k. \exists v. \triangleright (Y_{\text{nat}}(\lambda x.x) \Downarrow^k v) \]
- which implies \[ \exists k. \exists v. Y_{\text{nat}}(\lambda x.x) \Downarrow^{k+1} v \]

In Type Theory

\[ \Sigma k. \Sigma v. Y_{\text{nat}}(\lambda x.x) \Downarrow^k v \]

is not globally inhabited
Type theory vs. Topos logic

Set^{\omega^{op}} also models the topos logic.

The following is derivable in the non-proof-relevant topos logic:

$$\exists k. \exists v. \ Y_{nat} \ (\lambda x. x) \downarrow^k v$$

Proof (Sketch)

- The argument is by Guarded Recursion: assume
  $$\triangleright (\exists k. \exists v. \ Y_{nat} \ (\lambda x. x) \downarrow^k v)$$
- by property of $\text{Set}^{\omega^{op}}$ \(\exists k. \exists v. \triangleright (Y_{nat} \ (\lambda x. x) \downarrow^k v)\)
- which implies \(\exists k. \exists v. \ Y_{nat} \ (\lambda x. x) \downarrow^{k+1} v\)

In Type Theory

$$\Sigma k. \Sigma v. \ Y_{nat} \lambda x. x \downarrow^k v$$

is not globally inhabited
Set^{\omega^{op}} also models the topos logic.

The following is derivable in the non-proof-relevant topos logic:

\[ \exists k. \exists v. Y_{\text{nat}} (\lambda x. x) \Downarrow^k v \]

**Proof (Sketch)**

- The argument is by Guarded Recursion: assume
  \[ \triangleright (\exists k. \exists v. Y_{\text{nat}} (\lambda x. x) \Downarrow^k v) \]
- by property of Set^{\omega^{op}} \[ \exists k. \exists v. \triangleright (Y_{\text{nat}} (\lambda x. x) \Downarrow^k v) \]
- which implies \[ \exists k. \exists v. Y_{\text{nat}} (\lambda x. x) \Downarrow^{k+1} v \]

**In Type Theory**

\[ \Sigma k. \Sigma v. Y_{\text{nat}} \quad \lambda x. x \Downarrow^k v \]

is not globally inhabited
Conclusions

We presented a model for PCF that is adequate w.r.t. the operational semantics.

The work has been carried out entirely in guarded type theory:

- Operational Semantics with explicit step-indexing is synchronised with the time steps in the type theory
- Denotational semantics with proof of adequacy

Main message

Guarded type theory as a meta theory for denotational semantics of programming languages.

Future work

- Using the model to reason about contextual equivalence
- FPC in guarded type theory

Thanks!
We presented a model for PCF that is adequate w.r.t. the operational semantics.
The work has been carried out entirely in guarded type theory:

- Operational Semantics with explicit step-indexing is synchronised with the time steps in the type theory
- Denotational semantics with proof of adequacy

**Main message**

Guarded type theory as a meta theory for denotational semantics of programming languages.

**Future work**

- Using the model to reason about contextual equivalence
- FPC in guarded type theory

Thanks!