A model of PCF in Guarded Type Theory

Marco Paviotti[1] Rasmus Møgelberg[1] Lars Birkedal[2]

[1]IT University of Copenhagen

[2]Aarhus University

June 23th, 2015
MFPS 2015
Nijmegen, Netherlands

Guarded Type Theory

Birkedal and Møgelberg '12

In Type Theory

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

In Guarded Type Theory

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

- $\mathrm{next} : A \to \triangleright A$
- $\circledast : \triangleright(A \to B) \to \triangleright A \to \triangleright B$
- $\mathrm{fix}\colon (\triangleright A \to A) \to A$ s.t. $f(\mathrm{next}(\mathrm{fix}(f))) = \mathrm{fix}(f)$
- $X \cong A \times \triangleright X$

Guarded Type Theory

In Type Theory

unrestricted fix-point $\mathrm{fix}\colon (A \to A) \to A$ is inconsistent

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

In Guarded Type Theory

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

- $\mathrm{next} : A \to \triangleright A$
- $\circledast\colon \triangleright(A \to B) \to \triangleright A \to \triangleright B$
- $\mathrm{fix}\colon (\triangleright A \to A) \to A$ s.t. $f(\mathrm{next}(\mathrm{fix}(f))) = \mathrm{fix}(f)$
- $X \cong A \times \triangleright X$

Streams

$$\mathsf{Str}_A \cong A \times \mathsf{Str}_A$$

Streams in Coq

- $\mathsf{ones} = 1 :: \mathsf{ones}$ ✔
- $\mathsf{bad} = \mathsf{tail}\,\mathsf{bad}$ ✘
- $\mathsf{nats} = 0 :: \mathsf{map}\ (1+)\ \ \mathsf{nats}$ ✘

Streams

$$\mathsf{Str}_A \cong A \times \mathsf{Str}_A$$

Streams in Coq

- ones $=$ 1 : : ones ✔
- bad $=$ tail bad ✘
- nats $=$ 0 : : map $(1+)$ nats ✘

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

Guarded Streams

$\_ :: \_ : A \to \triangleright \mathsf{Str}^g_A \to \mathsf{Str}^g_A$ $\quad$ $\mathsf{head} : \mathsf{Str}^g_A \to A$ $\quad$ $\mathsf{tail} : \mathsf{Str}^g_A \to \triangleright \mathsf{Str}^g_A$

- ones $=$ 1 : : *ones* : Str^g_A ✔
- bad $=$ tail bad $\not{:}$ Str^g_A ✘
- nats $=$ 0 : : next(map $(1+)$) $\circledast$ nats : Str^g_A ✔

Model of Guarded Type Theory Birkedal and Møgelberg '12

The category of presheaves over ω

$$X \quad X(1) \xleftarrow{r_1} X(2) \quad .. \xleftarrow{r_{n-1}} X(n) \xleftarrow{r_n} ..$$

$$\triangleright X \quad 1 \xleftarrow{!} X(1) \quad .. \xleftarrow{r_{n-2}} X(n-1) \xleftarrow{r_n} ..$$

$\mathrm{Str}^g_A \cong A \times \triangleright \mathrm{Str}^g_A$ **Guarded Streams**

$$\mathrm{Str}^g_A \quad A \times 1 \xleftarrow{r_1} A \times (A \times 1) \xleftarrow{r_2} A \times (A \times A \times 1)$$

$$\triangleright \mathrm{Str}^g_A \quad 1 \xleftarrow{!} A \times 1 \xleftarrow{r_2} A \times A \times 1$$

$$A \times \triangleright \mathrm{Str}^g_A \quad A \times 1 \xleftarrow{r_1} A \times A \times 1 \xleftarrow{r_2} A \times A \times A \times 1$$

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

$$\sigma, \tau := \mathbf{nat} \mid \sigma \to \tau$$

$$L, M, N := \underline{n} \mid x \mid \lambda x.M \mid \mathsf{pred}\ \ M \mid \mathsf{succ}\ \ M \mid \mathsf{Y}\ \ M \mid \mathsf{ifz}\ \ L\ \ M\ \ N$$

$$\frac{}{\Gamma, x : \sigma, \Delta \vdash x : \sigma} \qquad \frac{}{\Gamma \vdash \underline{n} : \mathbf{nat}}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.M) : \sigma \to \tau} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

$$\frac{\Gamma \vdash M : \mathbf{nat}}{\Gamma \vdash \mathsf{succ}\ \ M : \mathbf{nat}} \qquad \frac{\Gamma \vdash M : \mathbf{nat}}{\Gamma \vdash \mathsf{pred}\ \ M : \mathbf{nat}}$$

$$\frac{\Gamma \vdash M : \sigma \to \sigma}{\Gamma \vdash \mathsf{Y}_\sigma\ \ M : \sigma}$$

$$\frac{\Gamma \vdash L : \mathbf{nat} \quad \Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \mathsf{ifz}\ \ L\ \ M\ \ N : \sigma}$$

Big-step semantics

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

$$M \Downarrow^{k} Q$$

Big-step semantics

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

explicit step counting

$$M \Downarrow^{k} Q$$

Big-step semantics

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

explicit step counting

$$M \Downarrow^k Q$$

Predicates on values

can define $M \Downarrow^k v$ as $M \Downarrow^k \lambda v'.v = v'$

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

explicit step counting

$$M \Downarrow^k Q$$

Predicates on values

can define $M \Downarrow^k v$ as $M \Downarrow^k \lambda v'.v = v'$

$$v \Downarrow^0 Q \stackrel{\text{def}}{=\!=} Q(v)$$

Big-step semantics

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

explicit step counting

$$M \Downarrow^k Q$$

Predicates on values

can define $M \Downarrow^k v$ as $M \Downarrow^k \lambda v'.v = v'$

$$v \Downarrow^0 Q \stackrel{\text{def}}{=\!=} Q(v)$$

$$MN \Downarrow^{k+m} Q \stackrel{\text{def}}{=\!=} M \Downarrow^k Q'$$

$$\text{where } Q'(\lambda x.L) = L[N/x] \Downarrow^m Q$$

Big-step semantics

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

explicit step counting

$$M \Downarrow^{k} Q$$

Predicates on values

can define $M \Downarrow^{k} v$ as $M \Downarrow^{k} \lambda v'.v = v'$

$$v \Downarrow^{0} Q \stackrel{\text{def}}{=} Q(v)$$

$$MN \Downarrow^{k+m} Q \stackrel{\text{def}}{=} M \Downarrow^{k} Q'$$

$$\text{where } Q'(\lambda x.L) = L[N/x] \Downarrow^{m} Q$$

$$\mathsf{Y}_{\sigma}\ M \Downarrow^{k+1} Q \stackrel{\text{def}}{=} \triangleright(M(\mathsf{Y}_{\sigma}\ M) \Downarrow^{k} Q)$$

Big-step semantics

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

explicit step counting

$$M \Downarrow^k Q$$

Predicates on values

can define $M \Downarrow^k v$ as $M \Downarrow^k \lambda v'.v = v'$

$$v \Downarrow^0 Q \stackrel{\text{def}}{=\!=} Q(v)$$

$$MN \Downarrow^{k+m} Q \stackrel{\text{def}}{=\!=} M \Downarrow^k Q'$$

$$\text{where } Q'(\lambda x.L) = L[N/x] \Downarrow^m Q$$

$$\mathrm{Y}_\sigma \ \ M \Downarrow^{k+1} Q \stackrel{\text{def}}{=\!=} \triangleright(M(\mathrm{Y}_\sigma \ \ M) \Downarrow^k Q)$$

Synchronising with the type theory

Small-Step Operational Semantics

$$\frac{}{(\lambda x:\sigma.M)(N) \to^0 M[N/x]} \qquad \frac{}{\mathsf{Y}_\sigma\ \ M \to^1 M(\mathsf{Y}_\sigma\ \ M)}$$

$$\frac{M \to^k M'}{M(N) \to^k M'(N)}$$

Let $\to_*^0$ be the reflexive, transitive closure of $\to^0$.

$$M \Rightarrow^0 Q \stackrel{\text{def}}{=\!=} \Sigma N \colon \mathtt{Term}_{\mathtt{PCF}}. M \to_*^0 N \text{ and } Q(N)$$

$$M \Rightarrow^{k+1} Q \stackrel{\text{def}}{=\!=} \Sigma M', M'' \colon \mathtt{Term}_{\mathtt{PCF}}. M \to_*^0 M' \text{ and } M' \to^1 M'' \text{ and } \triangleright(M'' \Rightarrow^k Q)$$

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

Lemma

$M \Downarrow^k v \Leftrightarrow M \Rightarrow^k v$

Outline

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

Lifting Monad

$$LA \cong A + \triangleright LA$$

Lifting monad

- $\eta : A \to LA \qquad \theta : \triangleright LA \to LA$
- Time step operation : $\delta = \theta \circ \mathsf{next} : LA \to LA$
- Bottom element $\bot = \mathsf{fix}(\theta)$
- 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

$$LA \cong A + \triangleright LA$$

Lifting monad $\qquad L\mathbb{N} \cong \mathbb{N} + \triangleright L\mathbb{N}$

$$\begin{array}{ll}
L\mathbb{N} & \mathbb{N}+1 \xleftarrow{r_1} \mathbb{N}+\mathbb{N}+1 \xleftarrow{r_2} \mathbb{N}+\mathbb{N}+\mathbb{N}+1 \\
\triangleright L\mathbb{N} & 1 \xleftarrow{\quad !\quad} \mathbb{N}+1 \xleftarrow{\quad r_1 \quad} \mathbb{N}+\mathbb{N}+1 \\
\mathbb{N}+\triangleright L\mathbb{N} & \mathbb{N}+1 \xleftarrow{r_1} \mathbb{N}+\mathbb{N}+1 \xleftarrow{r_2} \mathbb{N}+\mathbb{N}+\mathbb{N}+1
\end{array}$$

Interpreting PCF

- Interpreting Types

$$[\![\mathbf{nat}]\!] \stackrel{\text{def}}{=\!=} L\mathbb{N}$$

$$[\![\tau \to \sigma]\!] \stackrel{\text{def}}{=\!=} [\![\tau]\!] \to [\![\sigma]\!]$$

- All types are $\triangleright$–algebras with $\theta_\sigma : \triangleright[\![\sigma]\!] \to [\![\sigma]\!]$
- Interpreting terms $[\![t]\!] : [\![\Gamma]\!] \to [\![\sigma]\!]$

$$[\![\Gamma \vdash \mathsf{Y}_\sigma \;\; M]\!](\gamma) = (\mathsf{fix}_{[\![\sigma]\!]})(\lambda x : \triangleright[\![\sigma]\!].\theta_\sigma(\mathsf{next}([\![M]\!](\gamma))) \circledast x))$$

Interpreting PCF

- Interpreting Types

$$[\![\mathbf{nat}]\!] \stackrel{\mathrm{def}}{=\!=} L\mathbb{N}$$

$$[\![\tau \to \sigma]\!] \stackrel{\mathrm{def}}{=\!=} [\![\tau]\!] \to [\![\sigma]\!]$$

- All types are $\triangleright$–algebras with $\theta_\sigma : \triangleright[\![\sigma]\!] \to [\![\sigma]\!]$
- Interpreting terms $[\![t]\!] : [\![\Gamma]\!] \to [\![\sigma]\!]$

$$[\![\Gamma \vdash \mathsf{Y}_\sigma \;\; M]\!](\gamma) = (\mathsf{fix}_{[\![\sigma]\!]})(\lambda x : \triangleright[\![\sigma]\!].\theta_\sigma(\mathsf{next}([\![M]\!](\gamma))) \circledast x))$$

can be thought of
$\theta \circ \triangleright[\![M]\!]$

Interpreting PCF

- Interpreting Types

$$[\![\mathbf{nat}]\!] \stackrel{\text{def}}{=\!=} L\mathbb{N}$$

$$[\![\tau \to \sigma]\!] \stackrel{\text{def}}{=\!=} [\![\tau]\!] \to [\![\sigma]\!]$$

- All types are $\triangleright$–algebras with $\theta_\sigma : \triangleright[\![\sigma]\!] \to [\![\sigma]\!]$
- Interpreting terms $[\![t]\!] : [\![\Gamma]\!] \to [\![\sigma]\!]$

$$[\![\Gamma \vdash \mathsf{Y}_\sigma \ \ M]\!](\gamma) = (\mathsf{fix}_{[\![\sigma]\!]})(\lambda x : \triangleright[\![\sigma]\!].\theta_\sigma(\mathsf{next}([\![M]\!](\gamma))) \circledast x))$$

Lemma

Let $\Gamma \vdash M : \sigma \to \sigma$ *then* $[\![\mathsf{Y}_\sigma \ \ M]\!] = \delta_\sigma \circ [\![M(\mathsf{Y}_\sigma \ \ M)]\!]$

Soundness

Theorem (Soundness)

Let M be a closed term of type τ, if $M \Downarrow^k v$ then $[\![M]\!](*) = \delta^k [\![v]\!](*)$

Outline

- Operational Semantics of PCF
- Denotational Semantics
- **Computational Adequacy** if $[\![M]\!](*) = \delta^k [\![v]\!](*)$ then $M \Downarrow^k v$
- Discussion

Logical Relation

Adequacy proved by (proof-relevant) logical relation

$$d \; \mathcal{R}_\tau \; M$$

Define $\mathcal{R}_\tau$ by induction on τ

$$\eta(v) \; \mathcal{R}_{\mathbf{nat}} \; M \stackrel{\text{def}}{=\!=} M \Downarrow^0 v$$

$$\theta_{\mathbf{nat}}(r) \; \mathcal{R}_{\mathbf{nat}} \; M \stackrel{\text{def}}{=\!=} \Sigma M', M'' \colon \mathtt{Term}_{\mathtt{PCF}}. M \to^0_* M' \text{ and } M' \to^1 M'' \text{ and } r \mathrel{\triangleright\mathcal{R}_{\mathbf{nat}}} \mathrm{next}(M'')$$

Logical Relation

Adequacy proved by (proof-relevant) logical relation

$$d \ \mathcal{R}_\tau \ M$$

Define $\mathcal{R}_\tau$ by induction on τ

$$\eta(v) \ \mathcal{R}_{\mathbf{nat}} \ M \stackrel{\text{def}}{=\!=} M \Downarrow^0 v$$

$$\theta_{\mathbf{nat}}(r) \ \mathcal{R}_{\mathbf{nat}} \ M \stackrel{\text{def}}{=\!=} \Sigma M', M'' : \mathtt{Term}_{\mathrm{PCF}}. M \to^0_* M' \text{ and } M' \to^1 M'' \text{ and } r \triangleright \mathcal{R}_{\mathbf{nat}} \ \mathrm{next}(M'')$$

$L\mathbb{N} \cong \mathbb{N} + \triangleright L\mathbb{N}$

an element in this type is either of the form $\eta(v)$ or $\theta_{\mathbf{nat}}(r)$

Logical Relation

Adequacy proved by (proof-relevant) logical relation

$$d \; \mathcal{R}_\tau \; M$$

Define $\mathcal{R}_\tau$ by induction on τ

$$\eta(v) \; \mathcal{R}_{\mathbf{nat}} \; M \stackrel{\text{def}}{=\!=} M \Downarrow^0 v$$

$$\theta_{\mathbf{nat}}(r) \; \mathcal{R}_{\mathbf{nat}} \; M \stackrel{\text{def}}{=\!=} \Sigma M', M'' : \mathtt{Term}_{\mathtt{PCF}}. M \to^0_* M' \text{ and } M' \to^1 M'' \text{ and } r \triangleright\mathcal{R}_{\mathbf{nat}} \; \mathsf{next}(M'')$$

Delayed Relation $\triangleright\mathcal{R}$

$t \; \triangleright\mathcal{R}_{\mathbf{nat}} \; u$

delayed version of $\mathcal{R}$

Logical Relation

Adequacy proved by (proof-relevant) logical relation

$$d \;\mathcal{R}_\tau\; M$$

Define $\mathcal{R}_\tau$ by induction on τ

$$\begin{aligned}
\eta(v) \;\mathcal{R}_{\mathbf{nat}}\; M &\stackrel{\text{def}}{=\!=} M \Downarrow^0 v \\
\theta_{\mathbf{nat}}(r) \;\mathcal{R}_{\mathbf{nat}}\; M &\stackrel{\text{def}}{=\!=} \Sigma M', M'' \colon \mathtt{Term}_{\mathrm{PCF}}.M \to^0_* M' \\
&\qquad \text{and } M' \to^1 M'' \text{ and } r \rhd \mathcal{R}_{\mathbf{nat}} \; \mathsf{next}(M'') \\
f \;\mathcal{R}_{\tau\to\sigma}\; M &\stackrel{\text{def}}{=\!=} \Pi\alpha \colon [\![\tau]\!], \\
&\qquad N \colon \mathtt{Term}_{\mathrm{PCF}}.\alpha \;\mathcal{R}_\tau\; N \implies f(\alpha) \;\mathcal{R}_\sigma\; (MN)
\end{aligned}$$

Adequacy

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* $[\![t]\!](\vec{\alpha}) \; \mathcal{R}_\tau \; t[\vec{t}/\vec{x}]$

Theorem (Computational Adequacy)

If M *is a closed term of type* **nat** *then* $M \Downarrow^k v$ *iff* $[\![M]\!](*) = \delta^k [\![v]\!]$*.*

Outline

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

Type theory vs. Topos logic

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

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

$$\exists k. \exists v. \, \mathsf{Y}_{\mathbf{nat}} \; (\lambda x.x) \Downarrow^{k} v$$

Proof (Sketch)

- The argument is by Guarded Recursion: assume $\triangleright(\exists k. \exists v. \, \mathsf{Y}_{\mathbf{nat}} \; (\lambda x.x) \Downarrow^{k} v)$
- by property of $\mathtt{Set}^{\omega^{\mathrm{op}}}$ $\exists k. \exists v. \triangleright(\mathsf{Y}_{\mathbf{nat}} \; (\lambda x.x) \Downarrow^{k} v)$
- which implies $\exists k. \exists v. \, \mathsf{Y}_{\mathbf{nat}} \; (\lambda x.x) \Downarrow^{k+1} v$

In Type Theory

$$\Sigma k. \Sigma v. \, \mathsf{Y}_{\mathbf{nat}} \; \lambda x.x \Downarrow^{k} v$$

is not globally inhabited

Type theory vs. Topos logic

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

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

$$\exists k.\exists v.\, \mathsf{Y}_{\mathbf{nat}}\ (\lambda x.x) \Downarrow^{k} v$$

Proof (Sketch)

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

In Type Theory

$$\Sigma k.\Sigma v.\, \mathsf{Y}_{\mathbf{nat}}\ \lambda x.x \Downarrow^{k} v$$

is not globally inhabited

Type theory vs. Topos logic

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

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

$$\exists k.\exists v.\, \mathrm{Y}_{\mathbf{nat}}\ (\lambda x.x) \Downarrow^{k} v$$

Proof (Sketch)

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

In Type Theory

$$\Sigma k.\Sigma v.\, \mathrm{Y}_{\mathbf{nat}}\ \ \lambda x.x \Downarrow^{k} v$$

is not globally inhabited

Type theory vs. Topos logic

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

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

$$\exists k.\exists v.\, \mathsf{Y}_{\mathbf{nat}}\ (\lambda x.x) \Downarrow^{k} v$$

Proof (Sketch)

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

In Type Theory

$$\Sigma k.\Sigma v.\, \mathsf{Y}_{\mathbf{nat}}\ \ \lambda x.x \Downarrow^{k} v$$

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!

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!