Denotational Semantics in Synthetic Guarded Domain Theory

Marco Paviotti

Ph.D. defence

October 13th, 2016

Overview

Overview

Civil engineering vs. Software engineering

- In Civil Engineering, the mathematical foundation is physics: houses and skyscrapers are mathematically designed to be stable
- In Computer Science, the mathematical foundation is logic

Unfortunately, Computer science does not yet allow for mathematically correct implementations of software.

Foundations of Computer Science

Theoretical Computer science is devoted to laying the mathematical foundations of computer science.

Designing:

- “Good” Programming languages
- Specification languages
- Tools that implement them (e.g. Proof assistants)

λ Functional Programming

The λ-calculus underpins functional programming

- Recursion

$$\text{fact } x = \text{if } x == 0 \text{ then } 1 \text{ else } x * \text{ fact } (x - 1)$$

- Recursive Types

$$\text{data List } = \text{Empty } \mid \text{Cons Int List}$$

Semantics of Programming languages

- Aim: mathematically specifying the behaviour of programs

Semantics of Programming languages

- Aim: mathematically specifying the behaviour of programs
- This is allows to prove properties using a mathematical description

Semantics of Programming languages

- Aim: mathematically specifying the behaviour of programs
- This is allows to prove properties using a mathematical description
- Operational semantics describe how a program computes

Semantics of Programming languages

- Aim: mathematically specifying the behaviour of programs
- This is allows to prove properties using a mathematical description
- Operational semantics describe how a program computes
- Denotational semantics describe what a program is.
  - inspires new languages, prove soundness of formal systems and logics

Denotational semantics in set theory

Domain Theory

Let M be a well-typed program from A to B.

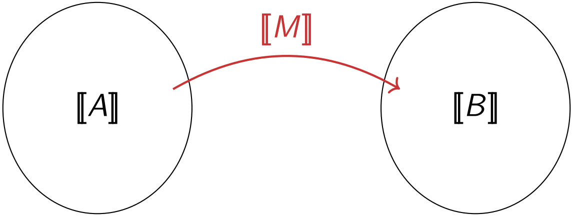
Its mathematical meaning is that of a function between two domains

Denotational semantics in set theory

Domain Theory

Let M be a well-typed program from A to B.

Its mathematical meaning is that of a function between two domains

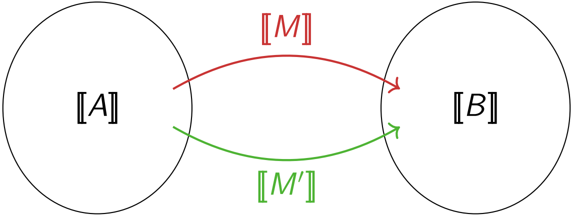


Denotational semantics in set theory

Domain Theory

Let M be a well-typed program from A to B.

Its mathematical meaning is that of a function between two domains



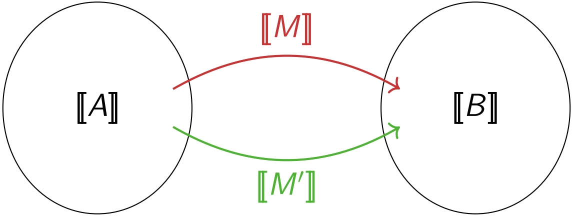
Theorem. If M reduces to M' then $[\![M]\!] = [\![M']\!]$

Denotational semantics in set theory

Domain Theory

Let M be a well-typed program from A to B.

Its mathematical meaning is that of a function between two domains



Theorem (Computational Adequacy).
If $[\![M]\!] = [\![M']\!]$ then $M \approx_{CTX} M'$

λ Semantics of Functional Programming

- Recursion

$$\text{fact } x = \text{if } x == 0 \text{ then } 1 \text{ else } x * \text{ fact } (x - 1)$$

λ Semantics of Functional Programming

- Recursion

$$\text{fact } x = \text{if } x == 0 \text{ then } 1 \text{ else } x * \text{ fact } (x-1)$$

$\text{Y } (\lambda f.\text{if } x == 0 \text{ then } 1 \text{ else } x * f(x-1))$

λ Semantics of Functional Programming

- Recursion

$$\text{fact } x = \text{if } x == 0 \text{ then } 1 \text{ else } x * \text{ fact } (x - 1)$$

- Recursive Types

$$\text{data List} = \text{Empty } \mid \text{Cons Int List}$$

λ Semantics of Functional Programming

- Recursion

$$\text{fact } x = \text{if } x == 0 \text{ then } 1 \text{ else } x * \text{ fact } (x - 1)$$

- Recursive Types

$$\text{data List} = \text{Empty} \mid \text{Cons Int List}$$

$X \cong 1 + \mathbb{Z} \times X$

λ Semantics of Functional Programming

- Recursion

$$\text{fact } x = \text{if } x == 0 \text{ then } 1 \text{ else } x * \text{ fact } (x - 1)$$

- Recursive Types

$$\text{data List} = \text{Empty} \mid \text{Cons Int List}$$

- Domain theory was invented for giving semantics of Recursion and Recursive Types (Domains as Ordered Sets)

The downsides of Denotational semantics

Denotational semantics do not scale:

- It relies on Set Theory, hence it needs a lot of structure
- The model does not scale for more featureful languages

Solution :

- Synthetic Domain Theory[1] replaces Set Theory with a theory with in-built domain theoretic structure (Domains as sets)

[1] Rosolini 1986, Taylor 1991, Hylland 1991, Phoa 1991, Simpson 2002

Formalising Denotational Semantics

Formalising Domain Theory:

- A huge line of research is devoted to reformulating mathematical theories into proof-assistants
- However, most proof-assistants (e.g. Coq, Agda) rely on another meta-theory...

Type Theory

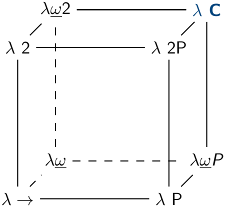
Relates *logic* and *computer science* under the principle
"*Proofs as programs, Formulas as types*"

Curry-Howard isomorphism

A is **Provable** $\Longleftrightarrow$ A is **Inhabited**

Type Theory

Relates *logic* and *computer science* under the principle
"*Proofs as programs, Formulas as types*"



Curry-Howard isomorphism

A is **Provable** $\Longleftrightarrow$ A is **Inhabited**

Denotational semantics in type theory

Formalising programming languages inside proof-assistants

- Domains as types
- All the definitions are explicit with the prover's logic
- Need to encode domain theory in type theory first [1]
  - Domain theory relies on set theory
  - The underlying theory does not understand the new definitions
  - Hard to use

[1] Capretta 2005, Benton and Kennedy and Varming 2009, Danielsson 2012

Guarded Type Theory [Birkedal and Møgelberg, 2012]

- It is a type theory with a notion of time
- If A is inhabited then also later A is inhabited
- Let f be a function from later A to A. Then there exists a unique "guarded" fixed-point for f.

Contributions

Denotational semantics of programming languages with recursion in guarded type theory under the slogan

"*Recursion in Guarded Recursion*"

Contributions

Denotational semantics of programming languages with recursion in guarded type theory under the slogan

"*Recursion in Guarded Recursion*"

- Simply-typed λ-calculus with Recursion (PCF)
- λ-calculus with Recursive Types (FPC)

Remarks

The development is entirely inside the type theory

Denotational semantics in Guarded Type theory

Types as domains

- combining *synthetic* and *type-theoretic* approaches
- Homotopy Theory to HoTT as Domain Theory to gDTT [1]
- more abstract representations and easier proofs

The price to pay is intentionality

- the model counts steps
- we define a logical relation to prove extensional computational adequacy

[1] Guarded dependent Type Theory with Coinductive Types, FoSSaCS, 2016

Guarded recursion and Separation Logic

In this very thesis I proved correct (w.r.t. some specifications) these 5 lines of assembly code

$$\begin{aligned}
&\mathsf{mov}\ \ \mathsf{ESI}, \mathit{info};\\
&\mathsf{mov}\ \ \mathsf{EDI}, [\mathsf{ESI}];\\
&\mathsf{mov}\ [\mathsf{EDI}],\ 0;\\
&\mathsf{add}\ \ \mathsf{EDI}, 4;\\
&\mathsf{mov}\ [\mathsf{ESI}],\ \mathsf{EDI}\,.
\end{aligned}$$

The specification language is a step-indexed variant of separation logic [1]

$$\mathbb{N} \times (\Sigma \to \mathsf{Prop}) \to \mathsf{Prop}$$

which validates the Guarded recursion Principle

[1] J.Jensen, N.Benton and A. Kennedy. High-level separation logic for low-level code. POPL 2016

Pubblications and Manuscripts

- A model of PCF in Guarded Type Theory. M. Paviotti, Rasmus E. Møgelberg and Lars Birkedal. In *Proceedings of Mathematical Foundations of Programming Semantics*, 2015.
- Denotational semantics of recursive types in Synthetic Guarded Domain Theory. In *Proceedings of Logic in Computer Science*, 2016.
- Formally Verifying Exceptions for Low-level code with Separation Logic, Marco Paviotti and Jesper Bengtson.

Synthetic Guarded Domain Theory

Guarded recursion

Guarded Type theory

- is a type theory with a time modality $\blacktriangleright$ pronounced "later"

$$
\begin{aligned}
\mathrm{fix} &: (\blacktriangleright X \to X) \to X \\
\mathrm{fix}(f) &= f(\mathrm{next}(\mathrm{fix}(f))) \\
\mathrm{next} &: X \to \blacktriangleright X \\
\circledast &: \blacktriangleright(X \to Y) \to \blacktriangleright X \to \blacktriangleright Y
\end{aligned}
$$

Guarded Type theory

- is a type theory with a time modality $\blacktriangleright$ pronounced "later"

$$\begin{aligned}
\mathrm{fix} &: (\blacktriangleright X \to X) \to X \\
\mathrm{fix}(f) &= f(\mathrm{next}(\mathrm{fix}(f))) \\
\mathrm{next} &: X \to \blacktriangleright X \\
\circledast &: \blacktriangleright(X \to Y) \to \blacktriangleright X \to \blacktriangleright Y
\end{aligned}$$

- Can solve equations as long as the $\blacktriangleright$ modality guards the recursive variable

$$X \cong \blacktriangleright((N \to X) \to 2)$$

- is an abstract form of step-indexing

Guarded Type theory

- is a type theory with a time modality $\blacktriangleright$ pronounced "later"

$$
\begin{aligned}
\mathrm{fix} &: (\blacktriangleright X \to X) \to X \\
\mathrm{fix}(f) &= f(\mathrm{next}(\mathrm{fix}(f))) \\
\mathrm{next} &: X \to \blacktriangleright X \\
\circledast &: \blacktriangleright(X \to Y) \to \blacktriangleright X \to \blacktriangleright Y
\end{aligned}
$$

- Can solve equations as long as the $\blacktriangleright$ modality guards the recursive variable

$$X \cong \blacktriangleright((N \to X) \to 2)$$

- is an abstract form of step-indexing
- used for relational reasoning and program logics for : $\mu\alpha.\tau$, ref τ, non-determinism and concurrency

Guarded Type Theory

Guarded recursive types are useful for checking productivity

$$\mathsf{Str}_A^g \cong A \times \blacktriangleright \mathsf{Str}_A^g$$

Guarded Streams

- $\mathsf{ones} = 1 :: \mathit{ones} : \mathsf{Str}_A^g$ ✔
- $\mathsf{bad} = \mathsf{tail}\,\mathsf{bad}$ ✘
- $\mathsf{nats} = 0 :: \mathrm{next}(\mathsf{map}\ (1+)) \circledast\ \mathsf{nats} : \mathsf{Str}_A^g$ ✔

PCF in guarded type theory

Big step operational semantics

- $M \Downarrow^{k} v$ is defined as an *inductive type*
- Fixed Point operator counts steps

$$\mathsf{Y}_\sigma \;\; M \Downarrow^{k+1} v =_{\text{def}} \blacktriangleright (M(\mathsf{Y}_\sigma \;\; M) \Downarrow^{k} v)$$

Big step operational semantics

Synchronising with the type theory

- $M \Downarrow^k v$ is defined as an *inductive type*
- Fixed Point operator counts steps

$$\mathsf{Y}_\sigma \;\; M \Downarrow^{k+1} v =_{\mathrm{def}} \blacktriangleright (M(\mathsf{Y}_\sigma \;\; M) \Downarrow^k v)$$

Denotational semantics

Lifting monad

$$LA =_{\mathrm{def}} A + \blacktriangleright LA$$

$$\eta : A \to LA \qquad \Theta : \blacktriangleright LA \to LA \qquad \delta = \Theta \circ \mathrm{next} : LA \to LA$$

- Satisfies $\bot = \delta(\bot)$.

Related

Similar to Escardó's Metric Lifting

It is the guarded recursive version of Capretta's coinductive lifting monad

Lifting monad

$$LA =_{\mathrm{def}} A + \blacktriangleright LA$$

$$\eta : A \to LA \qquad \Theta : \blacktriangleright LA \to LA \qquad \delta = \Theta \circ \mathrm{next} : LA \to LA$$

- Satisfies $\bot = \delta(\bot)$.
- LA is a *free* $\blacktriangleright$–algebra on A.

  Let $f : A \to B$, $\Theta_B : \blacktriangleright B \to B$, define

$$\hat{f} : LA \to B$$
$$\hat{f}(\eta(a)) = f(a)$$
$$\hat{f}(\Theta_{LA}(x)) = \Theta_B(\blacktriangleright(\hat{f})(x))$$

Related

Similar to Escardó's Metric Lifting

It is the guarded recursive version of Capretta's coinductive lifting monad

Denotational semantics

$$\llbracket \textbf{nat} \rrbracket =_{\mathrm{def}} L\mathbb{N}$$
$$\llbracket \sigma \to \tau \rrbracket =_{\mathrm{def}} \llbracket \sigma \rrbracket \to \llbracket \tau \rrbracket$$
$$\Theta_\sigma : \blacktriangleright \llbracket \sigma \rrbracket \to \llbracket \sigma \rrbracket$$

Denotational semantics

$$\llbracket \textbf{nat} \rrbracket =_{\mathrm{def}} L\mathbb{N}$$
$$\llbracket \sigma \to \tau \rrbracket =_{\mathrm{def}} \llbracket \sigma \rrbracket \to \llbracket \tau \rrbracket$$
$$\Theta_\sigma : \blacktriangleright \llbracket \sigma \rrbracket \to \llbracket \sigma \rrbracket$$

- Interpretation of terms

$$\llbracket x_1 : A_1, \ldots, x_1 : A_1 \vdash t : B \rrbracket : \llbracket A_1 \rrbracket \times \cdots \times \llbracket A_n \rrbracket \to \llbracket B \rrbracket$$

- Case of fixed point

$$\llbracket \mathsf{Y}_\sigma \;\; M \rrbracket =_{\mathrm{def}} \mathrm{fix}(\Theta_\sigma \circ \blacktriangleright(\llbracket M \rrbracket))$$
$$\llbracket \mathsf{Y}_\sigma \;\; M \rrbracket = \delta_\sigma \circ \llbracket M(\mathsf{Y}_\sigma \;\; M) \rrbracket$$

Denotational semantics

$$[\![\mathbf{nat}]\!] =_{\mathrm{def}} L\mathbb{N}$$
$$[\![\sigma \to \tau]\!] =_{\mathrm{def}} [\![\sigma]\!] \to [\![\tau]\!]$$
$$\Theta_\sigma : \blacktriangleright[\![\sigma]\!] \to [\![\sigma]\!]$$

- Interpretation of terms

$$[\![x_1 : A_1, \ldots, x_1 : A_1 \vdash t : B]\!] : [\![A_1]\!] \times \cdots \times [\![A_n]\!] \to [\![B]\!]$$

- Case of fixed point

$$[\![\mathsf{Y}_\sigma \;\; M]\!] =_{\mathrm{def}} \mathrm{fix}(\Theta_\sigma \circ \blacktriangleright([\![M]\!]))$$
$$[\![\mathsf{Y}_\sigma \;\; M]\!] = \delta_\sigma \circ [\![M(\mathsf{Y}_\sigma \;\; M)]\!]$$

- Case of if statement

$$[\![\mathsf{ifz}\, L \;\; M \;\; N]\!] =_{\mathrm{def}} (\widehat{\mathsf{ifz}}_{[\![M]\!],[\![N]\!]})[\![L]\!]$$

Adequacy

Theorem. If $\vdash M : \mathbf{nat}$ then $M \Downarrow^k v \iff [\![M]\!] = \delta^k([\![v]\!])$

Adequacy

Theorem. If $\vdash M : \mathbf{nat}$ then $M \Downarrow^k v \iff [\![M]\!] = \delta^k([\![v]\!])$

- Note

$$42 \models [\![(\mathsf{Y}_{\mathbf{nat}}\ (\lambda x : \mathbf{nat}.x))]\!] = \delta^{42}[\![0]\!]$$

- So also need

$$42 \models (\mathsf{Y}_{\mathbf{nat}}\ (\lambda x : \mathbf{nat}.x)) \Downarrow^{42} \underline{0}$$

Adequacy

Theorem. If $\vdash M : \mathbf{nat}$ then $M \Downarrow^k v \iff [\![M]\!] = \delta^k([\![v]\!])$

- Note

$$42 \models [\![(\mathsf{Y}_{\mathbf{nat}}\ (\lambda x : \mathbf{nat}.x))]\!] = \delta^{42}[\![0]\!]$$

- So also need

$$42 \models (\mathsf{Y}_{\mathbf{nat}}\ (\lambda x : \mathbf{nat}.x)) \Downarrow^{42} \underline{0}$$

Recall

$$\mathsf{Y}_\sigma\ \ M \Downarrow^{k+1} v =_{\mathrm{def}} \blacktriangleright (M(\mathsf{Y}_\sigma\ \ M) \Downarrow^k v)$$

Adequacy

Theorem. If $\vdash M : \mathbf{nat}$ then $M \Downarrow^k v \iff [\![M]\!] = \delta^k([\![v]\!])$

- Note
$$42 \models [\![(\mathsf{Y}_{\mathbf{nat}}\ (\lambda x : \mathbf{nat}.x))]\!] = \delta^{42}[\![0]\!]$$
- So also need
$$42 \models (\mathsf{Y}_{\mathbf{nat}}\ (\lambda x : \mathbf{nat}.x)) \Downarrow^{42} \underline{0}$$

Proof by Logical Relation argument

Lemma. If $M : \sigma$ closed then $[\![M]\!]\ \mathcal{R}_\sigma\ M$

FPC in guarded type theory

Big-step operational semantics

- $M \Downarrow^k v$ is an inductive type
- Such that

$$\texttt{unfold}\,(\texttt{fold}\,(M)) \Downarrow^k v = \blacktriangleright(M \Downarrow^{k-1} v)$$

Big-step operational semantics

- $M \Downarrow^k v$ is an inductive type
- Such that

Synchronising with the type theory

$$\texttt{unfold}\,(\texttt{fold}\,(M)) \Downarrow^k v = \blacktriangleright(M \Downarrow^{k-1} v)$$

Denotational Semantics – Types

- Define
$$[\![\Theta \vdash \tau]\!] : U^{|\Theta|} \to U$$
- By
$$\begin{aligned}
[\![\Theta \vdash \alpha]\!](\rho) &=_{\mathrm{def}} \rho(\alpha) \\
[\![\Theta \vdash 1]\!](\rho) &=_{\mathrm{def}} L1 \\
&\cdots \\
[\![\Theta \vdash \tau_1 + \tau_2]\!](\rho) &=_{\mathrm{def}} L([\![\Theta \vdash \tau_1]\!](\rho) + [\![\Theta \vdash \tau_2]\!](\rho)) \\
[\![\Theta \vdash \mu\alpha.\tau]\!](\rho) &=_{\mathrm{def}} \blacktriangleright([\![\Theta \vdash \tau]\!](\rho, [\![\mu\alpha.\tau]\!](\rho)))
\end{aligned}$$
- **Theorem.** $[\![\Theta \vdash \mu\alpha.\tau]\!](\rho) = \blacktriangleright([\![\Theta \vdash \tau[\mu\alpha.\tau/\alpha]]\!](\rho))$
- $\Theta_\tau : \blacktriangleright [\![\tau]\!] \to [\![\tau]\!]$

Denotational Semantics

- Interpretation of terms

$$[\![x_1 : A_1, \ldots, x_1 : A_1 \vdash t : B]\!] : [\![A_1]\!] \times \cdots \times [\![A_n]\!] \to [\![B]\!]$$

- Case of the folding and unfolding operations

$$[\![\Gamma \vdash \texttt{fold}\ M]\!](\gamma) = \mathrm{next}([\![M]\!](\gamma))$$

$$[\![\Gamma \vdash \texttt{unfold}\ M]\!](\gamma) = \theta_{\tau[\mu\alpha.\tau/\alpha]}([\![M]\!](\gamma))$$

Recall

$$[\![\Theta \vdash \mu\alpha.\tau]\!](\rho) = \blacktriangleright([\![\Theta \vdash \tau[\mu\alpha.\tau/\alpha]]\!](\rho))$$

Denotational Semantics

- Interpretation of terms

$$[\![x_1 : A_1, \ldots, x_1 : A_1 \vdash t : B]\!] : [\![A_1]\!] \times \cdots \times [\![A_n]\!] \to [\![B]\!]$$

- Case of the folding and unfolding operations

$$[\![\Gamma \vdash \texttt{fold}\ M]\!](\gamma) = \mathrm{next}([\![M]\!](\gamma))$$

$$[\![\Gamma \vdash \texttt{unfold}\ M]\!](\gamma) = \theta_{\tau[\mu\alpha.\tau/\alpha]}([\![M]\!](\gamma))$$

Recall

$$[\![\Theta \vdash \mu\alpha.\tau]\!](\rho) = \blacktriangleright([\![\Theta \vdash \tau[\mu\alpha.\tau/\alpha]]\!](\rho))$$

$[\![M]\!](\gamma)) : [\![\tau[\mu\alpha.\tau/\alpha]]\!]$

Denotational Semantics

- Interpretation of terms

$$[\![x_1 : A_1, \ldots, x_1 : A_1 \vdash t : B]\!] : [\![A_1]\!] \times \cdots \times [\![A_n]\!] \to [\![B]\!]$$

- Case of the folding and unfolding operations

$$[\![\Gamma \vdash \texttt{fold } M]\!](\gamma) = \mathrm{next}([\![M]\!](\gamma))$$

$$[\![\Gamma \vdash \texttt{unfold } M]\!](\gamma) = \theta_{\tau[\mu\alpha.\tau/\alpha]}([\![M]\!](\gamma))$$

Recall

$$[\![\Theta \vdash \mu\alpha.\tau]\!](\rho) = \blacktriangleright([\![\Theta \vdash \tau[\mu\alpha.\tau/\alpha]]\!](\rho))$$

$\boxed{[\![M]\!](\gamma)) : [\![\mu\alpha.\tau]\!]}$

Computational adequacy

Logical relation

$$\mathcal{R}_\tau : [\![\tau]\!] \times \mathtt{Term}_{\mathrm{FPC}} \to U$$

- Defined by guarded recursion and induction on the types

$$\eta(*) \ \mathcal{R}_1 \ M =_{\mathrm{def}} M \Downarrow^0 \langle\rangle$$
$$\Theta_1(\alpha) \ \mathcal{R}_1 \ M =_{\mathrm{def}} M \to_*^1 M' \text{ and } \alpha \blacktriangleright \mathcal{R}_1 \ \mathrm{next}(M')$$

- For recursive types

$$\alpha \ \mathcal{R}_{\mu\alpha.\tau} \ M =_{\mathrm{def}} \mathtt{unfold}\ M \to_*^1 M' \text{ and } \alpha \blacktriangleright \mathcal{R}_{\tau[\mu\alpha.\tau/\alpha]} \ \mathrm{next}(M')$$

- **Lemma.** If $M : \sigma$ closed then $[\![M]\!] \ \mathcal{R}_\sigma \ M$
- **Theorem.** If $\vdash M : 1$ then $M \Downarrow^k v \iff [\![M]\!] = \delta^k([\![v]\!])$

Extensional adequacy

Weak Bisimulation Logical relation

- Relation on denotable terms

$$\approx_\tau : [\![\tau]\!] \to [\![\tau]\!] \to U$$

defined by guarded recursion and induction on the types

- Relates computations produce the same value or diverge

$$\begin{aligned}
\eta(v) \approx_1 \eta(v') &=_{\mathrm{def}} v = v' \\
\eta(v) \approx_1 \beta &=_{\mathrm{def}} \Sigma n. \beta = (\delta_1)^n([\![v]\!]) \\
\alpha \approx_1 \eta(v) &=_{\mathrm{def}} \Sigma n. \alpha = (\delta_1)^n([\![v]\!]) \\
\Theta_1(\alpha') \approx_1 \Theta_1(\beta') &=_{\mathrm{def}} \alpha' \blacktriangleright \approx_1 \beta' \\
&\cdots \\
x \approx_{\mu\alpha.\tau} y &=_{\mathrm{def}} x \blacktriangleright \approx_{\tau[\mu\alpha.\tau/\alpha]} y
\end{aligned}$$

Recovering the global behaviour

- Idea originally due to Atkey and McBride
- We reformulate the interpretation s.t.

$$[\![1]\!]^{\mathrm{gl}} \cong 1 + [\![1]\!]^{\mathrm{gl}} A$$

- Lift $\approx$ to

$$\approx^{\mathrm{gl}}_{\sigma} \colon [\![\sigma]\!]^{\mathrm{gl}} \to [\![\sigma]\!]^{\mathrm{gl}} \to U$$

- Interpretation of terms $\Gamma \vdash M : \sigma$

$$[\![M]\!]^{\mathrm{gl}} : ([\![\Gamma]\!]^{\mathrm{gl}} \to [\![\sigma]\!]^{\mathrm{gl}})$$

- such that

$$\text{if } \delta^{\mathrm{gl}}(x) \approx^{\mathrm{gl}}_{1} \delta^{\mathrm{gl}}(y) \text{ then } x \approx^{\mathrm{gl}}_{1} y$$

Extensional Computational Adequacy

Theorem.

- If $\Gamma \vdash M, N : \tau$, and
- $[\![M]\!]^{\mathrm{global}} \approx^{\mathrm{global}}_{\Gamma,\tau} [\![N]\!]^{\mathrm{global}}$, and
- $C[-] : (\Gamma, \tau) \to \mathbf{nat}$, then

$$\prod n.(\Sigma k. C[M] \Downarrow^{k} n) \iff (\Sigma l. C[N] \Downarrow^{l} n)$$

Conclusions

Topos logic vs Type theory

- Could have probably done all this in topos logic
- But

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

Topos logic vs Type theory

- Could have probably done all this in topos logic
- But

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

- On the other hand

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

is not globally inhabited

Conclusions

- A step towards the formalisation of functional programming in modern proof-assistants
- *Recursion as Guarded recursion*: easier models and proofs

Main message

Guarded type theory is a natural setting for denotational semantics.

Future work

- General References, Co-inductive Types and Effects

Conclusions

- A step towards the formalisation of functional programming in modern proof-assistants
- *Recursion as Guarded recursion*: easier models and proofs

Main message

Guarded type theory is a natural setting for denotational semantics.

Future work

- General References, Co-inductive Types and Effects

Thanks!

Big step operational semantics

$$
\begin{aligned}
v \Downarrow^0 Q &=_{\mathrm{def}} Q(v) \\
\mathsf{pred}\ \ M \Downarrow^k Q &=_{\mathrm{def}} M \Downarrow^k (\lambda x.\Sigma n : \mathbb{N}.x = \underline{n} \text{ and } Q(\underline{n-1})) \\
\mathsf{succ}\ \ M \Downarrow^k Q &=_{\mathrm{def}} M \Downarrow^k (\lambda x.\Sigma n : \mathbb{N}.x = \underline{n} \text{ and } Q(\underline{n+1})) \\
\mathsf{Y}_\sigma\ \ M \Downarrow^{k+1} Q &=_{\mathrm{def}} \blacktriangleright (M(\mathsf{Y}_\sigma\ \ M) \Downarrow^k Q) \\
MN \Downarrow^{k+m} Q &=_{\mathrm{def}} M \Downarrow^k Q' \\
&\qquad \text{where } Q'(\lambda x.L) = L[N/x] \Downarrow^m Q \\
\mathsf{ifz}\ \ L\ \ M\ \ N \Downarrow^{k+m} Q &=_{\mathrm{def}} L \Downarrow^k Q' \\
&\qquad \text{where } Q'(\underline{0}) = M \Downarrow^m Q \text{ and } Q'(\underline{n+1}) = N \Downarrow^m Q
\end{aligned}
$$

Big step operational semantics

$$
\begin{aligned}
v \Downarrow^0 Q &=_{\mathrm{def}} Q(v) \\
\mathsf{pred}\ \ M \Downarrow^k Q &=_{\mathrm{def}} M \Downarrow^k (\lambda x.\Sigma n : \mathbb{N}.x = \underline{n} \text{ and } Q(\underline{n-1})) \\
\mathsf{succ}\ \ M \Downarrow^k Q &=_{\mathrm{def}} M \Downarrow^k (\lambda x.\Sigma n : \mathbb{N}.x = \underline{n} \text{ and } Q(\underline{n+1})) \\
\mathsf{Y}_\sigma\ \ M \Downarrow^{k+1} Q &=_{\mathrm{def}} \blacktriangleright (M(\mathsf{Y}_\sigma\ \ M) \Downarrow^k Q) \\
MN \Downarrow^{k+m} Q &=_{\mathrm{def}} M \Downarrow^k Q' \\
&\quad \text{where } Q'(\lambda x.L) = L[N/x] \Downarrow^m Q \\
\mathsf{ifz}\ \ L\ \ M\ \ N \Downarrow^{k+m} Q &=_{\mathrm{def}} L \Downarrow^k Q' \\
&\quad \text{where } Q'(\underline{0}) = M \Downarrow^m Q \text{ and } Q'(\underline{n+1}) = N \Downarrow^m Q
\end{aligned}
$$

Synchronising with the type theory

Big-step operational semantics

$$
\begin{aligned}
v \Downarrow^k Q &=_{\text{def}} Q(v, k) \\
\texttt{case}\ L\ \texttt{of}\ \texttt{inl}\ x_1.M; \texttt{inr}\ x_2.N \Downarrow^k Q &=_{\text{def}} L \Downarrow^k Q' \\
\text{where}\ & \begin{aligned} Q'(\texttt{inl}\ L, l) &=_{\text{def}} M[L/x_1] \Downarrow^l Q \\ Q'(\texttt{inr}\ L, l) &=_{\text{def}} N[L/x_2] \Downarrow^l Q \end{aligned} \\
\texttt{fst}\ L \Downarrow^k Q &=_{\text{def}} L \Downarrow^k Q' \\
&\quad \text{where}\ Q'(\langle M, N\rangle, m) =_{\text{def}} M \Downarrow^m Q \\
\texttt{snd}\ L \Downarrow^k Q &=_{\text{def}} L \Downarrow^k Q' \\
&\quad \text{where}\ Q'(\langle M, N\rangle, m) =_{\text{def}} N \Downarrow^m Q \\
MN \Downarrow^k Q &=_{\text{def}} M \Downarrow^k Q' \\
\text{where}\ Q'(\lambda x.L, m) &=_{\text{def}} L[N/x] \Downarrow^m Q \\
\texttt{unfold}\ M \Downarrow^k Q &=_{\text{def}} M \Downarrow^k Q' \\
\text{where}\ Q'(\texttt{fold}\ N, m+1) &=_{\text{def}} \blacktriangleright (N \Downarrow^m Q)
\end{aligned}
$$

Weak Bisimulation Logical relation

$$\eta(v) \approx_1 \eta(v') =_{\mathrm{def}} v = v'$$

$$\eta(v) \approx_1 \beta =_{\mathrm{def}} \Sigma n.\beta = (\delta_1)^n(\llbracket v \rrbracket)$$

$$\alpha \approx_1 \eta(v) =_{\mathrm{def}} \Sigma n.\alpha = (\delta_1)^n(\llbracket v \rrbracket)$$

$$\Theta_1(\alpha') \approx_1 \Theta_1(\beta') =_{\mathrm{def}} \alpha' \blacktriangleright \approx_1 \beta'$$

$$\cdots$$

$$x \approx_{\mu\alpha.\tau} y =_{\mathrm{def}} x \blacktriangleright \approx_{\tau[\mu\alpha.\tau/\alpha]} y$$

The topos of trees ($\mathbf{Set}^{\omega^{op}}$)

The category of presheaves over ω

$$
\begin{array}{lccccccccc}
X & X(1) & \xleftarrow{r_1} & X(2) & \quad & .. & \xleftarrow{r_{n-1}} & X(n) & \xleftarrow{r_n} & .. \\[1em]
\blacktriangleright X & 1 & \xleftarrow{!} & X(1) & \quad & .. & \xleftarrow{r_{n-2}} & X(n-1) & \xleftarrow{r_n} & ..
\end{array}
$$

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

$$
\begin{array}{lccccc}
\mathrm{Str}^g_A & A \times 1 & \xleftarrow{r_1} & A \times (A \times 1) & \xleftarrow{r_2} & A \times (A \times A \times 1) \\[1em]
\blacktriangleright \mathrm{Str}^g_A & 1 & \xleftarrow{!} & A \times 1 & \xleftarrow{r_2} & A \times A \times 1 \\[1em]
A \times \blacktriangleright \mathrm{Str}^g_A & A \times 1 & \xleftarrow{r_1} & A \times A \times 1 & \xleftarrow{r_2} & A \times A \times A \times 1
\end{array}
$$

Lifting monad

$$LA =_{\mathrm{def}} A + \blacktriangleright LA$$

Lifting monad

$$LA =_{\mathrm{def}} A + \blacktriangleright LA$$

- $L\mathbb{N}$ in model

$$\begin{tikzcd}
 & & & & \iddots & \vdots & \\
 & & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
1 & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
\mathbb{N} & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
L\mathbb{N}(1) & L\mathbb{N}(2) \arrow[l] & L\mathbb{N}(3) \arrow[l] & L\mathbb{N}(4) \arrow[l] & L\mathbb{N}(5) \arrow[l] & L\mathbb{N}(6) \arrow[l] \dots &
\end{tikzcd}$$

In model

$$\eta(42) : L\mathbb{N}$$

$$\begin{tikzcd}
 & & & & \iddots & \vdots & \\
 & & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
1 & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
\mathbb{N} & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots
\end{tikzcd}$$

$$L\mathbb{N}(1) \longleftarrow L\mathbb{N}(2) \longleftarrow L\mathbb{N}(3) \longleftarrow L\mathbb{N}(4) \longleftarrow L\mathbb{N}(5) \longleftarrow L\mathbb{N}(6) \dots$$

In model

$$\delta^2_{LA}(\eta(42)) : L\mathbb{N}$$

$$\begin{tikzcd}
 & & & & \iddots & \vdots & \\
 & & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \cdots \\
 & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \cdots \\
 & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \cdots \\
1 & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \cdots \\
\mathbb{N} & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \cdots
\end{tikzcd}$$

$$L\mathbb{N}(1) \longleftarrow L\mathbb{N}(2) \longleftarrow L\mathbb{N}(3) \longleftarrow L\mathbb{N}(4) \longleftarrow L\mathbb{N}(5) \longleftarrow L\mathbb{N}(6) \ldots$$

In model

$$\perp_{LA} : L\mathbb{N}$$

$$\begin{tikzcd}
 & & & \iddots & & \vdots & \\
 & & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
1 & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
\mathbb{N} & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots
\end{tikzcd}$$

$$L\mathbb{N}(1) \longleftarrow L\mathbb{N}(2) \longleftarrow L\mathbb{N}(3) \longleftarrow L\mathbb{N}(4) \longleftarrow L\mathbb{N}(5) \longleftarrow L\mathbb{N}(6) \dots$$

Forcing semantics

$$2 \models \delta^2_{LA}(\eta(42)) = \bot_{LA}$$
$$3 \not\models \delta^2_{LA}(\eta(42)) = \bot_{LA}$$

$$\begin{tikzcd}
 & & & & \iddots & \vdots & \\
 & & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
 & 1 \arrow[dl] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
1 & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots \\
\mathbb{N} & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \mathbb{N} \arrow[l] & \dots
\end{tikzcd}$$

$$L\mathbb{N}(1) \longleftarrow L\mathbb{N}(2) \longleftarrow L\mathbb{N}(3) \longleftarrow L\mathbb{N}(4) \longleftarrow L\mathbb{N}(5) \longleftarrow L\mathbb{N}(6) \dots$$

Construction of fixed points

- Given $f : \blacktriangleright X \to X$:

$$\begin{tikzcd}
\{*\} & X(1) \arrow[l] & X(2) \arrow[l, "r_1"'] & \dots \arrow[l] \\
X(1) \arrow[from=1-1, "f_1"'] & X(2) \arrow[l, "r_1"'] \arrow[from=1-2, "f_2"'] & X(3) \arrow[l, "r_2"'] \arrow[from=1-3, "f_3"'] & \dots \arrow[l]
\end{tikzcd}$$

- Construct $\mathrm{fix}_X(f) : 1 \to X$:

$$\begin{tikzcd}
1 \arrow[d, "f_1"] & 1 \arrow[l] \arrow[d, "f_2 \circ f_1"] & 1 \arrow[l] \arrow[d, "f_3 \circ f_2 \circ f_1"] & \dots \arrow[l] \\
X(1) & X(2) \arrow[l, "r_1"'] & X(3) \arrow[l, "r_2"'] & \dots \arrow[l]
\end{tikzcd}$$

- Fixed points are unique

Construction of fixed points

- Given $f : \blacktriangleright X \to X$:

$$\begin{tikzcd}
\{*\} & X(1) \arrow[l] \arrow[d, "f_2"'] & X(2) \arrow[l, "r_1"'] \arrow[d, "f_3"'] & \dots \arrow[l] \\
X(1) \arrow[u, leftarrow, "f_1"] & X(2) \arrow[l, "r_1"'] & X(3) \arrow[l, "r_2"'] & \dots \arrow[l]
\end{tikzcd}$$

- Construct $\operatorname{fix}_X(f) : 1 \to X$:

$$\begin{tikzcd}
1 \arrow[d, "f_1"] & 1 \arrow[l] \arrow[d, "f_2 \circ f_1"] & 1 \arrow[l] \arrow[d, "f_3 \circ f_2 \circ f_1"] & \dots \arrow[l] \\
X(1) & X(2) \arrow[l, "r_1"'] & X(3) \arrow[l, "r_2"'] & \dots \arrow[l]
\end{tikzcd}$$

- Fixed points are unique

Guarded recursive types as fixed points

Universe closed under $\blacktriangleright$

$$\frac{\Gamma \vdash A : \blacktriangleright U}{\Gamma \vdash \triangleright A : U}$$

$$\mathrm{El}(\triangleright(\mathrm{next}(A))) = \blacktriangleright \mathrm{El}(A)$$

Guarded recursive types as fixed points

Universe closed under $\blacktriangleright$

$$\frac{\Gamma \vdash A : \blacktriangleright U}{\Gamma \vdash \triangleright A : U}$$

$$\mathrm{El}(\triangleright(\mathrm{next}(A))) = \blacktriangleright \mathrm{El}(A)$$

Guarded Streams

- Type of streams as fixed point for universe map

$$\mathrm{S}(\mathrm{int}) = \mathrm{fix}(\lambda X : \blacktriangleright U. \mathbb{Z} \times \triangleright X)$$

- Then

$$\begin{aligned} \mathrm{El}(\mathrm{S}(\mathrm{int})) &= \mathrm{El}(\mathbb{Z} \times \triangleright(\mathrm{next}(\mathrm{S}(\mathrm{int})))) \\ &= \mathbb{Z} \times \blacktriangleright \mathrm{El}(\mathrm{S}(\mathrm{int})) \end{aligned}$$

Guarded dependent type theory (gDTT)

$$
\begin{aligned}
\mathcal{R}_\tau &: [\![\tau]\!] \to \mathtt{Term}_{\mathtt{PCF}} \to U \\
\mathrm{next}(\ \mathcal{R}_\tau\ ) &: \blacktriangleright([\![\tau]\!] \to \mathtt{Term}_{\mathtt{PCF}} \to U) \\
\mathrm{next}(\ \mathcal{R}_\tau\ ) \circledast (-) \circledast (-) &: \blacktriangleright[\![\tau]\!] \to \blacktriangleright \mathtt{Term}_{\mathtt{PCF}} \to \blacktriangleright U
\end{aligned}
$$

- Define (using $\triangleright : \blacktriangleright U \to U$)

$$
\begin{aligned}
\blacktriangleright \mathcal{R}_\tau \ &=_{\mathrm{def}} \triangleright \circ (\mathrm{next}(\ \mathcal{R}_\tau\ ) \circledast (-) \circledast (-)) \\
&: \blacktriangleright[\![\tau]\!] \to \blacktriangleright \mathtt{Term}_{\mathtt{PCF}} \to U
\end{aligned}
$$

Guarded dependent type theory (gDTT)

$$\begin{aligned}
\mathcal{R}_\tau \quad &: [\![\tau]\!] \to \mathtt{Term}_{\mathtt{PCF}} \to U \\
\mathrm{next}(\ \mathcal{R}_\tau\ ) &: \blacktriangleright([\![\tau]\!] \to \mathtt{Term}_{\mathtt{PCF}} \to U) \\
\mathrm{next}(\ \mathcal{R}_\tau\ ) \circledast (-) \circledast (-) &: \blacktriangleright[\![\tau]\!] \to \blacktriangleright\mathtt{Term}_{\mathtt{PCF}} \to \blacktriangleright U
\end{aligned}$$

- Define (using $\triangleright : \blacktriangleright U \to U$)

$$\begin{aligned}
\blacktriangleright\mathcal{R}_\tau \ &=_{\mathrm{def}} \triangleright \circ (\mathrm{next}(\ \mathcal{R}_\tau\ ) \circledast (-) \circledast (-)) \\
&: \blacktriangleright[\![\tau]\!] \to \blacktriangleright\mathtt{Term}_{\mathtt{PCF}} \to U
\end{aligned}$$

- Special syntax

$$x \ \blacktriangleright\mathcal{R}_\tau \ M = \blacktriangleright[y \leftarrow x, N \leftarrow M].(y \ \mathcal{R}_\tau \ N)$$

Syntax: multiple clocks

- Idea originally due to Atkey and McBride
- Clock variable context $\Delta = \kappa_1, \ldots, \kappa_n$

$$\frac{\Gamma \vdash_\Delta A : \mathsf{Type} \qquad \vdash_\Delta \kappa}{\Gamma \vdash_\Delta \overset{\kappa}{\blacktriangleright} A : \mathsf{Type}}$$

$$\mathtt{fix}^\kappa : (\overset{\kappa}{\blacktriangleright} X \to X) \to X$$

- etc

Universal quantification over clocks

$$\frac{\Gamma \vdash_{\Delta,\kappa} A : \mathsf{Type} \qquad \kappa \notin \mathsf{fc}(\Gamma)}{\Gamma \vdash_{\Delta} \forall\kappa.A : \mathsf{Type}}$$

$$\frac{\Gamma \vdash_{\Delta,\kappa} t : A \qquad \kappa \notin \mathsf{fc}(\Gamma)}{\Gamma \vdash_{\Delta} \Lambda\kappa.t : \forall\kappa.A}$$

$$\frac{\Gamma \vdash_{\Delta} t : \forall\kappa.A \qquad \vdash_{\Delta} \kappa'}{\Gamma \vdash_{\Delta} t[\kappa'] : A[\kappa'/\kappa]}$$

- Allows controlled elimination of $\blacktriangleright$

$$\mathsf{force} : \forall\kappa. \blacktriangleright^{\kappa} A \to \forall\kappa.A$$

- Clock quantification is right adjoint to clock weakening

Recovering the global behaviour

- Idea originally due to Atkey and McBride
- We reformulate the interpretation
- $L^{\mathrm{gl}}1 = \forall\kappa.L1$ so

$$[\![1]\!]^{\mathrm{gl}} \cong 1 + [\![1]\!]^{\mathrm{gl}}$$

- Lift $\approx$

$$\approx^{\mathrm{gl}}_{\sigma} \colon \forall\kappa.[\![\sigma]\!] \to \forall\kappa.[\![\sigma]\!] \to U$$

$$x \approx^{\mathrm{gl}}_{\sigma} y = \forall\kappa.x[\kappa] \approx_{\sigma} y[\kappa]$$

- Interpretation of terms $\Gamma \vdash M : \sigma$

$$[\![M]\!]^{\mathrm{gl}} : ([\![\Gamma]\!]^{\mathrm{gl}} \to [\![\sigma]\!]^{\mathrm{gl}})$$

$$[\![M]\!]^{\mathrm{gl}} = \Lambda\kappa.[\![M]\!]$$

- Can prove

$$\text{if } \delta^{\mathrm{gl}}(x) \approx^{\mathrm{gl}}_{1} \delta^{\mathrm{gl}}(y) \text{ then } \forall\kappa. \blacktriangleright^{\kappa} (x[\kappa] \approx_{\sigma} y[\kappa]) \text{ then } x \approx^{\mathrm{gl}}_{1} y$$

An extensional model (Bizjak and M, MFPS 2015)

- Type in context $\Delta = \emptyset$ is a set
- Type in context $\Delta = \kappa$ is an object in the topos of trees

$$
\begin{tikzcd}
X(1) & X(2) \arrow[l, "r_1"'] & X(3) \arrow[l, "r_2"'] & \dots \arrow[l]
\end{tikzcd}
$$

- Type in context $\Delta = \kappa, \kappa'$ is a diagram of form

$$
\begin{tikzcd}
\vdots \arrow[d] & \vdots \arrow[d] & \vdots \arrow[d] & & \vdots \arrow[d] \\
X(1,3) \arrow[d] & X(2,3) \arrow[l] \arrow[d] & X(3,3) \arrow[l] \arrow[d] \arrow[rr, bend right] & \cdots \arrow[l] & X(3) \arrow[d] \\
X(1,2) \arrow[d] & X(2,2) \arrow[l] \arrow[d] \arrow[rrr, bend right] & X(3,2) \arrow[l] \arrow[d] & \cdots \arrow[l] & X(2) \arrow[d] \\
X(1,1) \arrow[rrrr, bend right] & X(2,1) \arrow[l] & X(3,1) \arrow[l] & \cdots \arrow[l] & X(1)
\end{tikzcd}
$$