Two Guarded Recursive Powerdomains for Applicative Simulation

Rasmus Ejlers Møgelberg  Andrea Vezzosi

IT University of Copenhagen

Types 2022
How to model programming languages in type theory?

- \( \lambda \)-calculus with non-determinism

\[
M, N ::= M \ M \mid \lambda x. M \mid M \ or \ N
\]

- May-convergence predicate

\[
\begin{align*}
\lambda x. M \Downarrow \diamond & \implies \lambda x. M \\
\hline
M \Downarrow \diamond \lambda x. M' & \quad N \Downarrow \diamond V' & \quad M'[V'/x] \Downarrow \diamond V \\
\hline
MN \Downarrow \diamond V
\end{align*}
\]

\[
\begin{align*}
M \Downarrow \diamond V & \quad N \Downarrow \diamond V \\
\hline
M \ or \ N \Downarrow \diamond V
\end{align*}
\]
How to model programming languages in type theory?

▶ Idea: Model non-determinism using powersets

\[ \llbracket M \rrbracket : P_f(\text{Val}) \]

▶ Model divergence using coinductive partiality monad

\[ L A \simeq A + LA \]

▶ How to combine these?
Overview

- Guarded recursion
- Finite powerset as a HIT
- A powerdomain for may-convergence
  - Applicative may-similarity is a congruence
- A powerdomain for must-convergence
  - Applicative must-similarity is a congruence
Overview

- Guarded recursion
- Finite powerset as a HIT
- A powerdomain for may-convergence
  - Applicative may-similarity is a congruence
- A powerdomain for must-convergence
  - Applicative must-similarity is a congruence
- Everything done on paper in Clocked Cubical Type Theory
- Should be possible to formalise this in Guarded Cubical Agda
Guarded recursion

- Work in Clocked Cubical Type Theory\(^1\)
- Guarded fixed point operator
  \[
  \text{fix}^\kappa : (\triangleright^\kappa A \to A) \to A
  \]
- Guarded recursive types
  \[
  L^\kappa A \simeq A + \triangleright^\kappa L^\kappa A
  \]
- Define coinductive *partiality monad* \(L(A) = \forall \kappa. L^\kappa A\)
  \[
  L(A) \simeq A + L(A)
  \]
- Both \(L\) and \(L^\kappa\) are monads on Set

---

Finite (non-empty) powersets as HITs

\[
\{\emptyset\} : A \to \mathcal{P}_f(A)
\]

\[
\cup : \mathcal{P}_f(A) \to \mathcal{P}_f(A) \to \mathcal{P}_f(A)
\]

assoc : \(\Pi(X Y Z : \mathcal{P}_f(A)). X \cup (Y \cup Z) = (X \cup Y) \cup Z\)

comm : \(\Pi(X Y : \mathcal{P}_f(A)). X \cup Y = Y \cup X\)

idem : \(\Pi(X : \mathcal{P}_f(A)). X \cup X = X\)

trunc : isSet(\(\mathcal{P}_f(A)\))

- \(\mathcal{P}_f(A)\) is free semilattice structure on \(A\)
- \(\mathcal{P}_f\) is a monad on Set
A guarded powerdomain for may-equivalence

- Define may-powerdomain

\[ P^\land_\kappa(A) \simeq P_f(A + \triangleright^\kappa P^\land_\kappa(A)) \]

- Operations

\[ \bigcup : P^\land_\kappa(A) \to P^\land_\kappa(A) \to P^\land_\kappa(A) \]
\[ \text{now} : A \to P^\land_\kappa(A) \]
\[ \text{step} : \triangleright^\kappa P^\land_\kappa(A) \to P^\land_\kappa(A) \]

- Free delay-algebra and semilattice structure on \( A \)

- So monad on \( \text{Set} \)
Applicative similarity

- $R$ is an applicative may-simulation if $M \ R \ N$ and $M \Downarrow \lambda x.M'$ implies

  $$\exists N'. \ N \Downarrow \lambda y.N' \land (\forall (V:Val).M'[V/x] \ R \ N'[V/x])$$

- $\leq$ is greatest (coinductive) applicative may-simulation
Proving applicative similarity is a congruence

- Adaptation of Pitts’ method
- A semantic domain

\[
\text{SVal}^\kappa \overset{\text{def}}{=} \triangleright^\kappa (\text{SVal}^\kappa \rightarrow \text{P}^\kappa (\text{SVal}^\kappa))
\]

\[
D^\kappa \overset{\text{def}}{=} \text{P}^\kappa (\text{SVal}^\kappa)
\]

- Define

\[
\llbracket - \rrbracket^\kappa : \Lambda \rightarrow D^\kappa
\]

\[
\leq^\kappa : D^\kappa \times \Lambda \rightarrow \text{Prop}
\]

- **Lemma.** If \( M \) and \( N \) are closed terms then \( M \leq^\diamond N \) is equivalent to \( \forall \kappa. \llbracket M \rrbracket^\kappa \leq^\kappa N \).

- **Theorem.** If \( M \leq^\diamond N \) and \( C[\_] \) is a context then also \( C[M] \leq^\diamond C[N] \).
Must-convergence predicate

Define \(\Downarrow^\square: \Lambda \times P_f(\text{Val}) \rightarrow \text{Prop}\) as

\[
\begin{align*}
M \Downarrow^\square X & \quad N \Downarrow^\square Y \\
\hline
M \text{ or } N \Downarrow^\square X \cup Y \\
\hline
\text{\(\forall(x.\ M') \in X, \ V \in Y. \ M'[V/y] \Downarrow^\square Z_{x.\ M', \ V}\)} \\
\hline
M \ N \Downarrow^\square \cup_{V' \in X, \ V \in Y} Z_{V', \ V}
\end{align*}
\]
Powerdomain definition

Define

\[ P_\Box^\kappa(A) \overset{\text{def}}{=} L^\kappa(P_f(A)) \]
\[ \simeq P_f(A) + \triangleright^\kappa P_\Box^\kappa(A) \]

Semilattice structure

\[ \text{now}_L(x) \cup \text{now}_L(y) \overset{\text{def}}{=} \text{now}_L(x \cup y) \]
\[ \text{step}_\Box(x) \cup \text{step}_\Box(y) \overset{\text{def}}{=} \text{step}_\Box(\lambda(\alpha:\kappa).x[\alpha] \cup y[\alpha]) \]
\[ \text{step}_\Box(x) \cup \text{now}_L(y) \overset{\text{def}}{=} \text{step}_\Box(\lambda(\alpha:\kappa).(x[\alpha] \cup \text{now}_L(y))) \]
Powerdomain definition

Define

\[ P^\kappa(A) \overset{\text{def}}{=} L^\kappa(P_f(A)) \]
\[ \simeq P_f(A) + \triangleright^\kappa P^\kappa(A) \]

Semilattice structure

\[ \text{now}_L(x) \cup \text{now}_L(y) \overset{\text{def}}{=} \text{now}_L(x \cup y) \]
\[ \text{step}_\Box(x) \cup \text{step}_\Box(y) \overset{\text{def}}{=} \text{step}_\Box(\lambda(\alpha : \kappa).x[\alpha] \cup y[\alpha]) \]
\[ \text{step}_\Box(x) \cup \text{now}_L(y) \overset{\text{def}}{=} \text{step}_\Box(\lambda(\alpha : \kappa).(x[\alpha] \cup \text{now}_L(y))) \]

Not a monad!

Bind is not associative

Proved must-applicative similarity a congruence
Defined two guarded powerdomains and related these to may- and must-similarity

Unfortunately $P^K$ is not a monad

It is a monad up to weak bisimilarity

Future work:

A theory of algebraic effects and guarded recursion