Operational Semantics Using the Partiality Monad

Nils Anders Danielsson (Göteborg)

Shonan Meeting 026:
Coinduction for computation structures and programming languages

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n° 247219.
This presentation does not necessarily reflect the views of the ERC or the EU. The EU is not liable for any use of the presented information.

Total Definitional Interpreters

Nils Anders Danielsson (Göteborg)

Shonan Meeting 026:
Coinduction for computation structures and programming languages

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n° 247219.
This presentation does not necessarily reflect the views of the ERC or the EU. The EU is not liable for any use of the presented information.

Outline

Using partiality monad to:

- Define semantics of partial language:
  total definitional interpreter.
- Prove type soundness.
- Prove compiler correctness.

A partial language

A language with two effects:
non-termination and crashes.

$$t ::= c \mid x \mid \lambda x.t \mid t\ t$$

Represented using well-scoped de Bruijn indices:

$$
\begin{aligned}
&\mathbf{data}\ \mathit{Tm}\ (n : \mathbb{N}) : \mathit{Set}\ \mathbf{where} \\
&\quad \mathsf{con} : \mathbb{N} \to \mathit{Tm}\ n \\
&\quad \mathsf{var} : \mathit{Fin}\ n \to \mathit{Tm}\ n \\
&\quad \mathsf{lam} : \mathit{Tm}\ (1 + n) \to \mathit{Tm}\ n \\
&\quad \mathsf{app} : \mathit{Tm}\ n \to \mathit{Tm}\ n \to \mathit{Tm}\ n
\end{aligned}
$$

A partial language

A language with two effects:
non-termination and crashes.

$$t ::= c \mid x \mid \lambda x.t \mid t\ t$$

Represented using well-scoped de Bruijn indices:

$$
\begin{array}{l}
\textbf{data}\ \mathit{Tm}\ (n : \mathbb{N}) : \mathit{Set}\ \textbf{where} \\
\quad \mathsf{con} : \mathbb{N} \to \mathit{Tm}\ n \\
\quad \mathsf{var} : \mathit{Fin}\ n \to \mathit{Tm}\ n \\
\quad \mathsf{lam} : \mathit{Tm}\ (1 + n) \to \mathit{Tm}\ n \\
\quad \mathsf{app} : \mathit{Tm}\ n \to \mathit{Tm}\ n \to \mathit{Tm}\ n
\end{array}
$$

Environments and values

Based on closures:

$$
\begin{aligned}
&\mathit{Env} : \mathbb{N} \to \mathit{Set} \\
&\mathit{Env}\, n = \mathit{Vec}\ \mathit{Value}\, n
\end{aligned}
$$

$$
\begin{aligned}
&\textbf{data}\ \mathit{Value} : \mathit{Set}\ \textbf{where} \\
&\quad \mathsf{con} : \mathbb{N} \to \mathit{Value} \\
&\quad \mathsf{clo} : \mathit{Tm}\,(1 + n) \to \mathit{Env}\, n \to \mathit{Value}
\end{aligned}
$$

Definitional interpreter

$$
\begin{aligned}
[\![\_]\!] &: \mathit{Tm}\,n \rightarrow \mathit{Env}\,n \rightarrow \mathit{Value} \\
[\![\ \mathsf{con}\ i\ ]\!]\ \rho &= \mathsf{con}\ i \\
[\![\ \mathsf{var}\ x\ ]\!]\ \rho &= \mathit{lookup}\ x\ \rho \\
[\![\ \mathsf{lam}\ t\ ]\!]\ \rho &= \mathsf{clo}\ t\ \rho \\
[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\ \rho &= [\![\ t_1\ ]\!]\ \rho \bullet [\![\ t_2\ ]\!]\ \rho
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: \mathit{Value} \rightarrow \mathit{Value} \rightarrow \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= [\![\ t_1\ ]\!]\ (v_2 :: \rho)
\end{aligned}
$$

Definitional interpreter

$$\begin{array}{lcl}
[\![_]\!] & : & \mathit{Tm}\, n \;\rightarrow\; \mathit{Env}\, n \;\rightarrow\; \mathit{Value} \\
[\![\ \mathsf{con}\ i\ ]\!]\, \rho & = & \mathsf{con}\ i \qquad\qquad\qquad \Leftarrow \\
[\![\ \mathsf{var}\ x\ ]\!]\, \rho & = & \mathit{lookup}\ x\ \rho \\
[\![\ \mathsf{lam}\ t\ ]\!]\, \rho & = & \mathsf{clo}\ t\ \rho \\
[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\, \rho & = & [\![\ t_1\ ]\!]\, \rho \bullet [\![\ t_2\ ]\!]\, \rho
\end{array}$$

$$\begin{array}{lcl}
\bullet & : & \mathit{Value} \;\rightarrow\; \mathit{Value} \;\rightarrow\; \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 & = & [\![\ t_1\ ]\!]\, (v_2 :: \rho)
\end{array}$$

Definitional interpreter

$$
\begin{aligned}
[\![\_]\!] &: \mathit{Tm}\, n \to \mathit{Env}\, n \to \mathit{Value} \\
[\![\, \mathsf{con}\ i \,]\!]\, \rho &= \mathsf{con}\ i \\
[\![\, \mathsf{var}\ x \,]\!]\, \rho &= \mathit{lookup}\ x\ \rho \qquad \Leftarrow \\
[\![\, \mathsf{lam}\ t \,]\!]\, \rho &= \mathsf{clo}\ t\ \rho \\
[\![\, \mathsf{app}\ t_1\ t_2 \,]\!]\, \rho &= [\![\, t_1 \,]\!]\, \rho \bullet [\![\, t_2 \,]\!]\, \rho
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: \mathit{Value} \to \mathit{Value} \to \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= [\![\, t_1 \,]\!]\, (v_2 :: \rho)
\end{aligned}
$$

Definitional interpreter

$$
\begin{aligned}
[\![\_]\!] &: \mathit{Tm}\,n \to \mathit{Env}\,n \to \mathit{Value} \\
[\![\ \mathsf{con}\ i\ ]\!]\ \rho &= \mathsf{con}\ i \\
[\![\ \mathsf{var}\ x\ ]\!]\ \rho &= \mathit{lookup}\ x\ \rho \\
[\![\ \mathsf{lam}\ t\ ]\!]\ \rho &= \mathsf{clo}\ t\ \rho \qquad \Leftarrow \\
[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\ \rho &= [\![\ t_1\ ]\!]\ \rho \bullet [\![\ t_2\ ]\!]\ \rho
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: \mathit{Value} \to \mathit{Value} \to \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= [\![\ t_1\ ]\!]\ (v_2 :: \rho)
\end{aligned}
$$

Definitional interpreter

$$\llbracket \_ \rrbracket \;:\; \mathit{Tm}\, n \;\to\; \mathit{Env}\, n \;\to\; \mathit{Value}$$
$$\llbracket\, \textsf{con}\; i \,\rrbracket\, \rho \;=\; \textsf{con}\; i$$
$$\llbracket\, \textsf{var}\; x \,\rrbracket\, \rho \;=\; \mathit{lookup}\; x\; \rho$$
$$\llbracket\, \textsf{lam}\; t \,\rrbracket\, \rho \;=\; \textsf{clo}\; t\; \rho$$
$$\llbracket\, \textsf{app}\; t_1\; t_2 \,\rrbracket\, \rho \;=\; \llbracket\, t_1 \,\rrbracket\, \rho \bullet \llbracket\, t_2 \,\rrbracket\, \rho \qquad \Leftarrow$$

$$\_\bullet\_ \;:\; \mathit{Value} \;\to\; \mathit{Value} \;\to\; \mathit{Value}$$
$$\textsf{clo}\; t_1\; \rho \bullet v_2 \;=\; \llbracket\, t_1 \,\rrbracket\, (v_2 :: \rho)$$

Definitional interpreter

$$
\begin{aligned}
[\![\_]\!] &: \mathit{Tm}\,n \to \mathit{Env}\,n \to \mathit{Value} \\
[\![\ \mathsf{con}\ i\ ]\!]\,\rho &= \mathsf{con}\ i \\
[\![\ \mathsf{var}\ x\ ]\!]\,\rho &= \mathit{lookup}\ x\ \rho \\
[\![\ \mathsf{lam}\ t\ ]\!]\,\rho &= \mathsf{clo}\ t\ \rho \\
[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\,\rho &= [\![\ t_1\ ]\!]\,\rho \bullet [\![\ t_2\ ]\!]\,\rho
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: \mathit{Value} \to \mathit{Value} \to \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= [\![\ t_1\ ]\!]\,(v_2 :: \rho) \qquad \Leftarrow
\end{aligned}
$$

Definitional interpreter

$$
\begin{aligned}
[\![\_]\!] &: \mathit{Tm}\, n \to \mathit{Env}\, n \to \mathit{Value} \\
[\![\ \mathsf{con}\ i\ ]\!]\, \rho &= \mathsf{con}\ i \\
[\![\ \mathsf{var}\ x\ ]\!]\, \rho &= \mathit{lookup}\ x\ \rho \\
[\![\ \mathsf{lam}\ t\ ]\!]\, \rho &= \mathsf{clo}\ t\ \rho \\
[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\, \rho &= [\![\ t_1\ ]\!]\, \rho \bullet [\![\ t_2\ ]\!]\, \rho
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: \mathit{Value} \to \mathit{Value} \to \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= [\![\ t_1\ ]\!]\, (v_2 :: \rho)
\end{aligned}
$$

- Does not work in Agda, Coq...
- Call-by-value? Call-by-name?

Relational, big-step semantics

- Can define inductive big-step semantics:

  $$\rho \vdash t \Downarrow v$$

- Very similar to definitional interpreter, but we cannot "run" the semantics.
- Does not distinguish between non-termination and crashes.

Relational, big-step semantics

- Can add coinductive big-step semantics:

$$\rho \vdash t \Uparrow$$

- Problem: Duplication of rules.
- Problem: Have we forgotten a rule?

Alternative

Definitional interpreter + partiality monad

$\Rightarrow$

functional, big-step semantics.

Partiality monad

Partiality monad

$$
\begin{array}{l}
\textbf{codata}\ \_{}_{\bot}\ (A\ :\ Set)\ :\ Set\ \textbf{where} \\
\quad \mathsf{now}\ \ :\ A\ \ \to\ A_{\bot} \\
\quad \mathsf{later}\ :\ A_{\bot}\ \to\ A_{\bot}
\end{array}
$$

- *Constructive* partiality monad,
  not maybe monad or exception monad.
- $A_{\bot}\ \approx\ \nu C.\ A + C$.

Partiality monad

$$
\begin{array}{l}
\textbf{codata } \_{}_\bot \ (A : Set) : Set \textbf{ where} \\
\quad \textsf{now} : A \to A_\bot \\
\quad \textsf{later} : A_\bot \to A_\bot
\end{array}
$$

- *Constructive* partiality monad,
  not maybe monad or exception monad.
- $A_\bot \approx \nu C.\, A + C$.

Partiality monad

$$\textbf{codata}\ \_{}_\bot\ (A : \mathit{Set}) : \mathit{Set}\ \textbf{where}$$
$$\begin{aligned} \mathsf{now} &: A \to A_\bot \\ \mathsf{later} &: A_\bot \to A_\bot \end{aligned}$$

- *Constructive* partiality monad,
  not maybe monad or exception monad.
- $A_\bot \approx \nu C.\, A + C$.

Partiality monad

$$\textbf{codata}\ \_{}_{\bot}\ (A\ :\ Set)\ :\ Set\ \textbf{where}$$
$$\quad \mathsf{now}\ :\ A \to A_{\bot}$$
$$\quad \mathsf{later}\ :\ A_{\bot} \to A_{\bot}$$

$$never\ :\ A_{\bot}$$
$$never\ =\ \mathsf{later}\ never$$

$$\_{\gg\!=}\_\ :\ A_{\bot} \to (A \to B_{\bot}) \to B_{\bot}$$
$$\mathsf{now}\ x \gg\!= f = f\ x$$
$$\mathsf{later}\ x \gg\!= f = \mathsf{later}\ (x \gg\!= f)$$

Partiality monad

$$\textbf{codata}\ \_{}_{\perp}\ (A\ :\ Set)\ :\ Set\ \textbf{where}$$
$$\begin{aligned} &\textsf{now} &&: A &&\to A_{\perp} \\ &\textsf{later} &&: A_{\perp} &&\to A_{\perp} \end{aligned}$$

What is the right notion of equality for $A_{\perp}$?

$$\begin{aligned} &\textsf{later}\ (\textsf{later}\ (\textsf{now}\ 5)) &&\approx\ \textsf{now}\ 5 \\ &\textsf{later}\ never &&\approx\ never \\ &\textsf{now}\ 5 &&\not\approx\ never \end{aligned}$$

Equality up to finite differences in the number of later constructors.

Total definitional interpreter

Definitional interpreter

$$
\begin{aligned}
[\![\_]\!] &: \mathit{Tm}\, n \to \mathit{Env}\, n \to \mathit{Value} \\
[\![\, \mathsf{con}\ i \,]\!]\, \rho &= \mathsf{con}\ i \\
[\![\, \mathsf{var}\ x \,]\!]\, \rho &= \mathit{lookup}\ x\ \rho \\
[\![\, \mathsf{lam}\ t \,]\!]\, \rho &= \mathsf{clo}\ t\ \rho \\
[\![\, \mathsf{app}\ t_1\ t_2 \,]\!]\, \rho &= [\![\, t_1 \,]\!]\, \rho \bullet [\![\, t_2 \,]\!]\, \rho
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: \mathit{Value} \to \mathit{Value} \to \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= [\![\, t_1 \,]\!]\, (v_2 :: \rho)
\end{aligned}
$$

Definitional interpreter

With Maybe monad:

$$
\begin{aligned}
\llbracket \_ \rrbracket &: \mathit{Tm}\,n \to \mathit{Env}\,n \to \mathit{Maybe}\ \mathit{Value} \\
\llbracket\ \mathsf{con}\ i\ \rrbracket\,\rho &= \mathit{return}\,(\mathsf{con}\ i) \\
\llbracket\ \mathsf{var}\ x\ \rrbracket\,\rho &= \mathit{return}\,(\mathit{lookup}\ x\ \rho) \\
\llbracket\ \mathsf{lam}\ t\ \rrbracket\,\rho &= \mathit{return}\,(\mathsf{clo}\ t\ \rho) \\
\llbracket\ \mathsf{app}\ t_1\ t_2\ \rrbracket\,\rho &= \llbracket\ t_1\ \rrbracket\,\rho \gg\!= \lambda\, v_1 \to \\
&\ \llbracket\ t_2\ \rrbracket\,\rho \gg\!= \lambda\, v_2 \to \\
&\ v_1 \bullet v_2
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: \mathit{Value} \to \mathit{Value} \to \mathit{Maybe}\ \mathit{Value} \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= \llbracket\ t_1\ \rrbracket\,(v_2 :: \rho) \\
\mathsf{con}\ i_1 \bullet v_2 &= \mathit{fail}
\end{aligned}
$$

Definitional interpreter

With $Maybe$ monad:

$$
\begin{aligned}
[\![\_]\!] &: Tm\ n \to Env\ n \to Maybe\ Value \\
[\![\ \mathsf{con}\ i\ ]\!]\ \rho &= return\ (\mathsf{con}\ i) \\
[\![\ \mathsf{var}\ x\ ]\!]\ \rho &= return\ (lookup\ x\ \rho) \\
[\![\ \mathsf{lam}\ t\ ]\!]\ \rho &= return\ (\mathsf{clo}\ t\ \rho) \\
[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\ \rho &= [\![\ t_1\ ]\!]\ \rho \gg\!= \lambda\, v_1 \to \\
&\quad\ [\![\ t_2\ ]\!]\ \rho \gg\!= \lambda\, v_2 \to \\
&\quad\ v_1 \bullet v_2
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: Value \to Value \to Maybe\ Value \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= [\![\ t_1\ ]\!]\ (v_2 :: \rho) \\
\mathsf{con}\ i_1 \bullet v_2 &= fail
\end{aligned}
$$

Total definitional interpreter

With $Maybe + \_{}_\bot$:

$$
\begin{aligned}
&[\![\_]\!] : Tm\,n \to Env\,n \to (Maybe\ Value)_\bot \\
&[\![\ \mathsf{con}\ i\ ]\!]\ \rho = return\,(\mathsf{con}\ i) \\
&[\![\ \mathsf{var}\ x\ ]\!]\ \rho = return\,(lookup\ x\ \rho) \\
&[\![\ \mathsf{lam}\ t\ ]\!]\ \rho = return\,(\mathsf{clo}\ t\ \rho) \\
&[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\ \rho = [\![\ t_1\ ]\!]\ \rho \gg\!= \lambda\, v_1 \to \\
&\qquad [\![\ t_2\ ]\!]\ \rho \gg\!= \lambda\, v_2 \to \\
&\qquad v_1 \bullet v_2
\end{aligned}
$$

$$
\begin{aligned}
&\_\bullet\_ : Value \to Value \to (Maybe\ Value)_\bot \\
&\mathsf{clo}\ t_1\ \rho \bullet v_2 = \mathsf{later}\ ([\![\ t_1\ ]\!]\ (v_2 :: \rho)) \\
&\mathsf{con}\ i_1 \bullet v_2 = fail
\end{aligned}
$$

Total definitional interpreter

With $Maybe + \_{}_\bot$:

$$
\begin{aligned}
[\![\_]\!] &: Tm\,n \to Env\,n \to (Maybe\ Value)_\bot \\
[\![\ \mathsf{con}\ i\ ]\!]\ \rho &= return\,(\mathsf{con}\ i) \\
[\![\ \mathsf{var}\ x\ ]\!]\ \rho &= return\,(lookup\ x\ \rho) \\
[\![\ \mathsf{lam}\ t\ ]\!]\ \rho &= return\,(\mathsf{clo}\ t\ \rho) \\
[\![\ \mathsf{app}\ t_1\ t_2\ ]\!]\ \rho &= [\![\ t_1\ ]\!]\ \rho \gg= \lambda\, v_1 \to \\
&\quad\ [\![\ t_2\ ]\!]\ \rho \gg= \lambda\, v_2 \to \\
&\quad\ v_1 \bullet v_2
\end{aligned}
$$

$$
\begin{aligned}
\_\bullet\_ &: Value \to Value \to (Maybe\ Value)_\bot \\
\mathsf{clo}\ t_1\ \rho \bullet v_2 &= \mathsf{later}\,([\![\ t_1\ ]\!]\ (v_2 :: \rho)) \\
\mathsf{con}\ i_1 \bullet v_2 &= fail
\end{aligned}
$$

Total definitional interpreter

$$[\![\_]\!] : \mathit{Tm}\, n \;\to\; \mathit{Env}\, n \;\to\; (\mathit{Maybe}\ \mathit{Value})_{\perp}$$

- Type signature $\Rightarrow$ deterministic.
- Type signature $\Rightarrow$ computable.
- No “duplication of rules”.
- Impossible to forget a case.
- Classically equivalent to
  $\rho \vdash t \Downarrow v$ plus $\rho \vdash t \Uparrow$.

Type soundness

Types

Coinductive simple types (to allow non-termination):

$$
\begin{array}{l}
\textbf{codata}\ Ty\ :\ Set\ \textbf{where} \\
\quad \mathsf{nat}\quad :\ Ty \\
\quad \_\!\rightarrow\!\_\ :\ Ty\ \rightarrow\ Ty\ \rightarrow\ Ty
\end{array}
$$

$$
\begin{array}{l}
\tau\ :\ Ty \\
\tau\ =\ \tau\ \rightarrow\ \mathsf{nat}
\end{array}
$$

Typing relation:

$$\Gamma \vdash t : \sigma$$

Type soundness

Statement of type soundness:

$$[\,] \vdash t : \sigma \;\; \to \;\; [\![\, t \,]\!]\, [\,] \not\approx \; \mathit{fail}$$

Proof: Generalise, then nested corecursion/ structural recursion.

Compiler correctness

Virtual machine

$$
\begin{array}{l}
\textbf{data}\ \mathit{State}\ :\ \mathit{Set}\ \textbf{where} \\
\quad \vdots
\end{array}
$$

$$
\begin{array}{llll}
\multicolumn{4}{l}{\textbf{data}\ \mathit{Result}\ :\ \mathit{Set}\ \textbf{where}} \\
\quad \mathsf{continue} & : & \mathit{State} & \to\ \mathit{Result} \\
\quad \mathsf{done} & : & \mathit{VM\text{-}Value} & \to\ \mathit{Result} \\
\quad \mathsf{crash} & : & & \ \mathit{Result}
\end{array}
$$

$$
\begin{array}{l}
\mathit{step}\ :\ \mathit{State}\ \to\ \mathit{Result} \\
\mathit{step}\ \ldots\ =\ \ldots \\
\vdots \\
\mathit{step}\ \_\ \ =\ \mathsf{crash}
\end{array}
$$

Virtual machine

Small-step, total, functional semantics:

$$
\begin{aligned}
&\mathit{exec} : \mathit{State} \to (\mathit{Maybe}\ \mathit{VM\text{-}Value})_{\bot} \\
&\mathit{exec}\ s = \textbf{case}\ \mathit{step}\ s\ \textbf{of} \\
&\quad \textsf{continue}\ s' \to \textsf{later}\ (\mathit{exec}\ s') \\
&\quad \textsf{done}\ v \to \mathit{return}\ v \\
&\quad \textsf{crash} \to \mathit{fail}
\end{aligned}
$$

- Type signature ⇒ deterministic.
- Type signature ⇒ computable.
- *Possible* to forget a case.

Compiler correctness

Small-step, total, functional semantics:

$$exec \;:\; State \;\to\; (Maybe\ VM\text{-}Value)_{\bot}$$

Compilers:

$$\begin{aligned} comp &\;:\; Tm\,0 \;\to\; State \\ comp_{\mathsf{v}} &\;:\; Value \;\to\; VM\text{-}Value \end{aligned}$$

Compiler correctness

Small-step, total, functional semantics:

$$exec \;:\; State \;\to\; (Maybe\ VM\text{-}Value)_\bot$$

Compilers:

$$\begin{aligned} comp &\;:\; Tm\,0 \;\to\; State \\ comp_{\mathsf{v}} &\;:\; Value \;\to\; VM\text{-}Value \end{aligned}$$

Compiler correctness:

$$\begin{aligned} &(t \;:\; Tm\,0) \;\to \\ &exec\,(comp\,t) \;\approx \\ &\quad (\llbracket\, t \,\rrbracket\,[\,] \;\gg\!= \;\lambda\, v \;\to\; return\,(comp_{\mathsf{v}}\; v)) \end{aligned}$$

Conclusions

- Definitional interpreter $+$ partiality monad
  $\Rightarrow$
  functional, total, big-step semantics.
- Can mechanise (some) meta-theory.
- See paper for non-determinism,
  contextual equivalence, applicative bisimilarity.

Bonus slides

Operational semantics?

- Very close to usual, relational big-step operational semantics.
- Not compositional.
- Values contain closures.

Typing rules

$$\Gamma \vdash \mathsf{con}\ i : \mathsf{nat} \qquad \Gamma \vdash \mathsf{var}\ x : \mathit{lookup}\ x\ \Gamma$$

$$\frac{\sigma :: \Gamma \vdash t : \tau}{\Gamma \vdash \mathsf{lam}\ t : \sigma \rightarrow \tau}$$

$$\frac{\Gamma \vdash t_1 : \sigma \rightarrow \tau \qquad \Gamma \vdash t_2 : \sigma}{\Gamma \vdash \mathsf{app}\ t_1\ t_2 : \tau}$$

Applicative bisimilarity, part 1

$$
\begin{array}{l}
\textbf{data}\ \_\!\approx_\bot\!\_ \ :\ (\mathit{Maybe}\ \mathit{Value})_\bot \to (\mathit{Maybe}\ \mathit{Value})_\bot \to \mathit{Set}\ \textbf{where} \\
\quad \mathsf{now} : u \approx_{\mathsf{MV}} v \to \mathsf{now}\ u \approx_\bot \mathsf{now}\ v \\
\quad \mathsf{later} : \infty\,(x \approx_\bot y) \to \mathsf{later}\ x \approx_\bot \mathsf{later}\ y \\
\quad \mathsf{later}^{\mathrm{l}} : x \approx_\bot y \to \mathsf{later}\ x \approx_\bot y \\
\quad \mathsf{later}^{\mathrm{r}} : x \approx_\bot y \to x \approx_\bot \mathsf{later}\ y
\end{array}
$$

$$
\begin{array}{l}
\textbf{data}\ \_\!\approx_{\mathsf{MV}}\!\_ \ :\ \mathit{Maybe}\ \mathit{Value} \to \mathit{Maybe}\ \mathit{Value} \to \mathit{Set}\ \textbf{where} \\
\quad \mathsf{just} : u \approx_{\mathsf{V}} v \to \mathsf{just}\ u \approx_{\mathsf{MV}} \mathsf{just}\ v \\
\quad \mathsf{nothing} : \mathsf{nothing} \approx_{\mathsf{MV}} \mathsf{nothing}
\end{array}
$$

Applicative bisimilarity, part 2

$$
\begin{aligned}
&\textbf{data}\ \_{\approx_{\mathsf{V}}}\_ : \mathit{Value} \to \mathit{Value} \to \mathit{Set}\ \textbf{where} \\
&\quad \mathsf{con} : \mathsf{con}\ i \approx_{\mathsf{V}} \mathsf{con}\ i \\
&\quad \mathsf{clo} : (\forall\ v \to \\
&\qquad\qquad \infty\,(\llbracket\, t_1 \,\rrbracket\,(v :: \rho_1) \approx_{\bot} \llbracket\, t_2 \,\rrbracket\,(v :: \rho_2))) \to \\
&\qquad\quad \mathsf{clo}\ t_1\ \rho_1 \approx_{\mathsf{V}} \mathsf{clo}\ t_2\ \rho_2
\end{aligned}
$$

$$
\begin{aligned}
&\_{\approx_{\mathsf{T}}}\_ : \mathit{Tm}\ n \to \mathit{Tm}\ n \to \mathit{Set} \\
&t_1 \approx_{\mathsf{T}} t_2 = \forall\ \rho \to \llbracket\, t_1 \,\rrbracket\,\rho \approx_{\bot} \llbracket\, t_2 \,\rrbracket\,\rho
\end{aligned}
$$