Operational Semantics Using the Partiality Monad

Nils Anders Danielsson (Göteborg)

ICFP 2012, København, 2012-09-10

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. I do not hold the copyright to the EU emblem or the ERC logo.

Outline

Using partiality monad to:

- Define semantics of partial language: *formal 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:

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

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:

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

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

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

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

Definitional interpreter

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

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

Definitional interpreter

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

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

Definitional interpreter

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

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

Definitional interpreter

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

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

Definitional interpreter

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

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

Definitional interpreter

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

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

- 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

$$
\begin{array}{llll}
\multicolumn{4}{l}{\textbf{data}\ \_{}_\perp\ (A\ :\ Set)\ :\ Set\ \textbf{where}} \\
\quad \mathsf{now} & : & A & \to\ A_\perp \\
\quad \mathsf{later} & : & \infty\,(A_\perp) & \to\ A_\perp
\end{array}
$$

- Non-strict data type (∞ suspension).
- $A_\perp \ \approx\ \nu C.\, A + C$.

Partiality monad

$$
\begin{array}{l}
\textbf{data}\ \_{}_\perp\ (A\ :\ \mathit{Set})\ :\ \mathit{Set}\ \textbf{where} \\
\quad \mathsf{now}\ \ :\ A \qquad\qquad\quad \to\ A_\perp \\
\quad \mathsf{later}\ :\ \infty\,(A_\perp)\ \to\ A_\perp
\end{array}
$$

- Non-strict data type (∞ suspension).
- $A_\perp \approx \nu C.\, A + C$.

Partiality monad

$$
\begin{array}{l}
\textbf{data}\ \_{}_{\perp}\ (A\ :\ Set)\ :\ Set\ \textbf{where} \\
\quad \mathsf{now}\ \ :\ A \qquad\qquad \to\ A_{\perp} \\
\quad \mathsf{later}\ :\ \infty\,(A_{\perp})\ \to\ A_{\perp}
\end{array}
$$

- Non-strict data type (∞ suspension).
- $A_{\perp}\ \approx\ \nu C.\, A + C$.

Partiality monad

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

$$\begin{array}{l} \mathit{never} : A_\bot \\ \mathit{never} = \mathsf{later}\ \mathit{never} \end{array}$$

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

Partiality monad

$$
\begin{array}{l}
\textbf{data}\ \__\perp\ (A\ :\ Set)\ :\ Set\ \textbf{where} \\
\quad \textsf{now}\ \ :\ A \qquad\qquad\ \to\ A_\perp \\
\quad \textsf{later}\ :\ \infty\ (A_\perp)\ \to\ A_\perp
\end{array}
$$

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

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

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

Total definitional interpreter

Definitional interpreter

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

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

Definitional interpreter

With Maybe monad:

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

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

Definitional interpreter

With Maybe monad:

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

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

Total definitional interpreter

With $\mathit{Maybe} + \_{}_{\perp}$:

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

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

Total definitional interpreter

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

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

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

Total definitional interpreter

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

- Obviously deterministic.
- Obviously computable: can test things.
- 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):

$$
\begin{aligned}
&\mathbf{data}\ Ty\ :\ Set\ \mathbf{where}\\
&\quad \mathsf{nat}\quad\ :\ Ty\\
&\quad \_\!\!\rightarrow\!\!\_\ :\ \infty\, Ty\ \rightarrow\ \infty\, Ty\ \rightarrow\ Ty
\end{aligned}
$$

$$
\begin{aligned}
&\tau\ :\ Ty\\
&\tau\ =\ \tau\ \rightarrow\ \mathsf{nat}
\end{aligned}
$$

Typing relation:

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

Type soundness

Statement of type soundness:

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

Proof: Generalise, then nested corecursion/ structural recursion.

Compiler correctness

Compiler correctness

Small-step, total, functional semantics:

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

Iterates step function.

Compiler correctness

Small-step, total, functional semantics:

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

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.

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.
- No built-in, congruent extensional equality.

Typing rules

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

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

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

Virtual machine

```
data State : Set where
  ⋮

data Result : Set where
  continue : State    → Result
  done     : VM-Value → Result
  crash    :            Result

step : State → Result
step ... = ...
⋮
step _ = crash
```

Virtual machine

$$
\begin{array}{l}
\mathit{exec} \;:\; \mathit{State} \;\to\; (\mathit{Maybe}\ \mathit{VM\text{-}Value})_{\perp} \\
\mathit{exec}\ s \;=\; \mathbf{case}\ \mathit{step}\ s\ \mathbf{of} \\
\quad \begin{array}{lcl}
\mathsf{continue}\ s' & \to & \mathsf{later}\ (\mathit{exec}\ s') \\
\mathsf{done}\ v & \to & \mathit{return}\ v \\
\mathsf{crash} & \to & \mathit{fail}
\end{array}
\end{array}
$$

- Obviously deterministic.
- Obviously computable.
- *Possible* to forget a case.

Applicative bisimilarity, part 1

$$
\begin{array}{lllllll}
\multicolumn{7}{l}{\textbf{data}\ \_\approx_\bot\_ : (\mathit{Maybe}\ \mathit{Value})_\bot \to} \\
\multicolumn{7}{l}{\qquad\qquad\qquad (\mathit{Maybe}\ \mathit{Value})_\bot \to \mathit{Set}\ \textbf{where}} \\
\quad \mathsf{now} & : & u \approx_{\mathsf{MV}} v & \to & \mathsf{now}\ u & \approx_\bot & \mathsf{now}\ v \\
\quad \mathsf{later} & : & \infty\,(x \approx_\bot y) & \to & \mathsf{later}\ x & \approx_\bot & \mathsf{later}\ y \\
\quad \mathsf{later}^{\mathrm{l}} & : & x \approx_\bot y & \to & \mathsf{later}\ x & \approx_\bot & y \\
\quad \mathsf{later}^{\mathrm{r}} & : & x \approx_\bot y & \to & x & \approx_\bot & \mathsf{later}\ y
\end{array}
$$

$$
\begin{array}{lllllll}
\multicolumn{7}{l}{\textbf{data}\ \_\approx_{\mathsf{MV}}\_ : \mathit{Maybe}\ \mathit{Value} \to} \\
\multicolumn{7}{l}{\qquad\qquad\qquad \mathit{Maybe}\ \mathit{Value} \to \mathit{Set}\ \textbf{where}} \\
\quad \mathsf{just} & : & u \approx_{\mathsf{V}} v & \to & \mathsf{just}\ u & \approx_{\mathsf{MV}} & \mathsf{just}\ v \\
\quad \mathsf{nothing} & : & & & \mathsf{nothing} & \approx_{\mathsf{MV}} & \mathsf{nothing}
\end{array}
$$

Applicative bisimilarity, part 2

$$
\begin{array}{l}
\textbf{data}\ \_{\approx_{\mathsf{V}}}\_\ :\ \mathit{Value}\ \to\ \mathit{Value}\ \to\ \mathit{Set}\ \textbf{where} \\
\quad \mathsf{con}\ :\ \mathsf{con}\ i\ \approx_{\mathsf{V}}\ \mathsf{con}\ i \\
\quad \mathsf{lam}\ :\ (\forall\ v\ \to \\
\qquad\qquad \infty\ (\llbracket\ t_1\ \rrbracket\ (v :: \rho_1)\ \approx_{\perp}\ \llbracket\ t_2\ \rrbracket\ (v :: \rho_2)))\ \to \\
\qquad\quad \mathsf{lam}\ t_1\ \rho_1\ \approx_{\mathsf{V}}\ \mathsf{lam}\ t_2\ \rho_2
\end{array}
$$

$$
\begin{array}{l}
\_{\approx_{\mathsf{T}}}\_\ :\ \mathit{Tm}\ n\ \to\ \mathit{Tm}\ n\ \to\ \mathit{Set} \\
t_1\ \approx_{\mathsf{T}}\ t_2\ =\ \forall\ \rho\ \to\ \llbracket\ t_1\ \rrbracket\ \rho\ \approx_{\perp}\ \llbracket\ t_2\ \rrbracket\ \rho
\end{array}
$$