Verified computation over real numbers and other continuous objects II

Sewon Park[†]

March 3, 2022 (Akita, Japan)
The second Japan-Russia workshop on effective descriptive set theory, computable analysis and automata

[†] Kyoto University, Japan
[†] supported by JSPS KAKENHI Grant number JP18H03203.

Simple imperative language

(initial) state command (final) state

Simple imperative language

(initial) state			command	(final) state
x_1	$=$	0		
x_2	$=$	0		
x_3	$=$	0		
	$\vdots$			
x_d	$=$	0		

Simple imperative language

(initial) state			command	(final) state
x_1	$=$	0		
x_2	$=$	0		
x_3	$=$	0	$x_1 := 42$	
	$\vdots$			
x_d	$=$	0		

Simple imperative language

(initial) state		
x_1	$=$	0
x_2	$=$	0
x_3	$=$	0
	$\vdots$	
x_d	$=$	0

command

$x_1 := 42$

(final) state		
x_1	$=$	42
x_2	$=$	0
x_3	$=$	0
	$\vdots$	
x_d	$=$	0

Simple imperative language

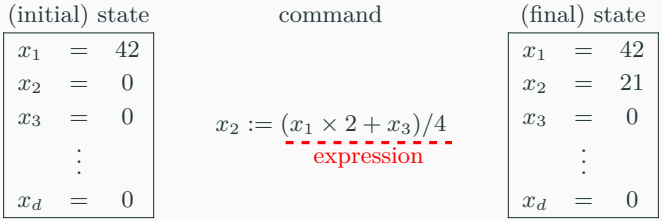
(initial) state		
x_1	$=$	42
x_2	$=$	0
x_3	$=$	0
	$\vdots$	
x_d	$=$	0

command

$$x_2 := (x_1 \times 2 + x_3)/4$$

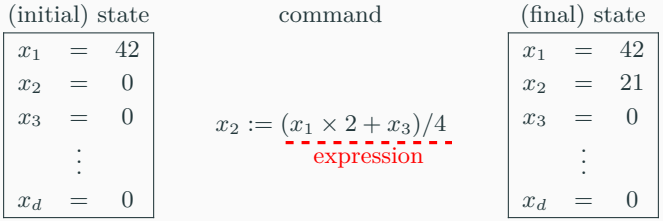
(final) state		
x_1	$=$	42
x_2	$=$	21
x_3	$=$	0
	$\vdots$	
x_d	$=$	0

Simple imperative language



commands c ::= $c_1; c_2 \mid x := e \mid$ `if` e `then` c_1 `else` c_2 `end` $\mid$ `while` e `do` c `end`

Simple imperative language



commands c ::= $c_1; c_2 \mid x := e \mid$ `if` e `then` c_1 `else` c_2 `end` $\mid$ `while` e `do` c `end`

1. A simple model of computation
2. Models practical programming languages
3. Pre-post condition specification:
   For all state satisfying ϕ, the command results states in ψ.

$$\{\phi\}\ c\ \{\psi\}$$

$$\{\text{any state}\}\ c\ \{x \text{ stores } \pi\}$$

4. Hoare-style verification calculus for proving specifications

Imperative language for real computation

Imperative language where

- Variables store real numbers exactly (x_2 stores π not 3.141592)
- Operations computed exactly ($\pi \times \sqrt{2} \rightsquigarrow \pi\sqrt{2}$)
- Implementable
- Reasoning based on the programmers' mathematical knowledge. No artificial rounding and truncation error analysis.

Infinite representation of reals

(initial) state

abstract			implementation
x_1	=	42	$42.00000000\cdots$
x_2	=	π	$3.141592653\cdots$
x_3	=	$\sqrt{2}$	$1.414213562\cdots$
	$\vdots$		
x_d	=	3.14	$3.140000000\cdots$

$\{x_2 = \pi \wedge x_3 = \sqrt{2}\}$

command

$x_1 := x_2 \times x_3$

(final) state

abstract			implementation
x_1	=	$\pi\sqrt{2}$	$4.442882936\cdots$
x_2	=	π	$3.141592653\cdots$
x_3	=	$\sqrt{2}$	$1.414213562\cdots$
	$\vdots$		
x_d	=	3.14	$3.140000000\cdots$

$\{x_1 = \pi\sqrt{2}\}$

Imperative language where

- Variables store real numbers exactly (x_2 stores π not 3.141592)
- Operations computed exactly ($\pi \times \sqrt{2} \rightsquigarrow \pi\sqrt{2}$)
- Implementable
- Reasoning based on the programmers' mathematical knowledge. No artificial rounding and truncation error analysis.

Previously...

Yesterday, we

- identified a core fragment of imperative exact real computation software as a simple imperative language with partial comparison of reals and choose
- reviewed computable analysis in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ and specified some monads for various partiality and nondeterminism
- reviewed Plotkin powerdomain and specified some representations on them

Previously...

Yesterday, we

- identified a core fragment of imperative exact real computation software as a simple imperative language with partial comparison of reals and choose
- reviewed computable analysis in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ and specified some monads for various partiality and nondeterminism
- reviewed Plotkin powerdomain and specified some representations on them

Today, we

- define formal semantics both denotational and operational using the elements specified today
- devise proof rules for formal verification that is sound w.r.t. the denotational semantics
- suggest an extension structure of the language with further continuous objects
- introduce some further designs toward having limits and multi-limits

Recap I: $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

- Category of assemblies (generalized representations) $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ with computable mappings

Recap I: $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

- Category of assemblies (generalized representations) $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ with computable mappings
- $\mathsf{M} : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for multi-functions $f : X \to \mathsf{M}(Y)$
- $\flat : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions diverge out-of-domain $f : X \to \flat Y$
- $\sharp : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions signal corrections out-of-domain $f : X \to \sharp Y$
- $\natural : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for general partial functions without out-of-domain spec. $f : X \to \natural Y$

Recap I: $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

- Category of assemblies (generalized representations) $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ with computable mappings
- $\mathsf{M} : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for multi-functions $f : X \to \mathsf{M}(Y)$
- $\flat : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions diverge out-of-domain $f : X \to \flat Y$
- $\sharp : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions signal corrections out-of-domain $f : X \to \sharp Y$
- $\natural : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for general partial functions without out-of-domain spec. $f : X \to \natural Y$

universal collection of morhpisms for any effective assembly of real numbers $\mathbf{R}$:

1. constants $0, 1 : \mathbf{1} \to \mathbf{R}$
2. field operators $+, \times : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$, $- : \mathbf{R} \to \mathbf{R}$, and $^{-1} : \mathbf{R} \to \flat\mathbf{R}$,
3. order relation $\lesssim : \mathbf{R} \times \mathbf{R} \to \flat\mathbf{2}$
4. limit operation $\mathsf{lim} : \mathbf{R}^{\mathbf{N}} \to \sharp\mathbf{R}$

Recap I: $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

- Category of assemblies (generalized representations) $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ with computable mappings
- $\mathsf{M} : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for multi-functions $f : X \to \mathsf{M}(Y)$
- $\flat : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions diverge out-of-domain $f : X \to \flat Y$
- $\sharp : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions signal corrections out-of-domain $f : X \to \sharp Y$
- $\natural : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for general partial functions without out-of-domain spec. $f : X \to \natural Y$

universal collection of morhpisms for any effective assembly of real numbers $\mathbf{R}$:

1. constants $0, 1 : \mathbf{1} \to \mathbf{R}$
2. field operators $+, \times : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$, $- : \mathbf{R} \to \mathbf{R}$, and $^{-1} : \mathbf{R} \to \flat\mathbf{R}$,
3. order relation $\lesssim : \mathbf{R} \times \mathbf{R} \to \flat\mathbf{2}$
4. limit operation $\mathsf{lim} : \mathbf{R}^{\mathbf{N}} \to \sharp\mathbf{R}$
5. nondeterminism construct $\mathsf{choose}_n : \flat\mathbf{2}^n \to \flat\mathsf{M}(\mathbf{n})$

$$\mathsf{choose}_n(b_1, \cdots, b_n) := \begin{cases} \{i \mid b_i = \mathtt{true}\} & \text{if } \{i \mid b_i = \mathtt{true}\} \neq \emptyset \\ \flat & \text{otherwise} \end{cases}$$

Recap I: $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

- Category of assemblies (generalized representations) $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ with computable mappings
- $\mathsf{M} : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for multi-functions $f : X \to \mathsf{M}(Y)$
- $\flat : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions diverge out-of-domain $f : X \to \flat Y$
- $\sharp : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for partial functions signal corrections out-of-domain $f : X \to \sharp Y$
- $\natural : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ for general partial functions without out-of-domain spec. $f : X \to \natural Y$

universal collection of morhpisms for any effective assembly of real numbers $\mathbf{R}$:

1. constants $0, 1 : \mathbf{1} \to \mathbf{R}$
2. field operators $+, \times : \mathbf{R} \times \mathbf{R} \to \mathbf{R}$, $- : \mathbf{R} \to \mathbf{R}$, and $^{-1} : \mathbf{R} \to \flat\mathbf{R}$,
3. order relation $\lesssim : \mathbf{R} \times \mathbf{R} \to \mathbf{K}$
4. limit operation $\mathsf{lim} : \mathbf{R}^{\mathbf{N}} \to \sharp\mathbf{R}$
5. nondeterminism construct $\mathsf{choose}_n : \mathbf{K}^n \to \flat\mathsf{M}(\mathbf{n})$

$$\mathsf{choose}_n(b_1, \cdots, b_n) := \begin{cases} \{i \mid b_i = \mathtt{true}\} & \text{if } \{i \mid b_i = \mathtt{true}\} \neq \emptyset \\ \{\flat\} & \text{otherwise} \end{cases}$$

When $\mathbf{K} := \flat\mathbf{2}$

Recap II: Powerdomain

- For a set X, the Plotkin powerdomain $\mathcal{P}(X_\perp)$ over its flat domain $X_\perp$ is a poset on

$$\{S \subseteq A \mid S \neq \emptyset, S \text{ is finite or } \perp \in S\}$$

whose partial order is

$$P \sqsubseteq Q \quad \iff \quad (\perp \in P \wedge P \subseteq Q \cup \{\perp\}) \vee (\perp \notin P \wedge P = Q) \ .$$

is a cpo with $\{\perp\}$ the least element

Recap II: Powerdomain

- For a set X, the Plotkin powerdomain $\mathcal{P}(X_\perp)$ over its flat domain $X_\perp$ is a poset on

$$\{S \subseteq A \mid S \neq \emptyset, S \text{ is finite or } \perp \in S\}$$

  whose partial order is

$$P \sqsubseteq Q \quad \iff \quad (\perp \in P \wedge P \subseteq Q \cup \{\perp\}) \vee (\perp \notin P \wedge P = Q) \ .$$

  is a cpo with $\{\perp\}$ the least element
- set-valued condition $b : X \to \mathcal{P}(\mathbb{2}_\perp)$ and set-value command $c : X \to \mathcal{P}(X_\perp)$

$$[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]\gamma = \bigcup_{v \in b(\gamma)} \begin{cases} [\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]^\dagger \circ c(\gamma) & \text{if } v = \texttt{true} \\ \{\gamma\} & \text{if } v = \texttt{false} \\ \{\perp\} & \text{if } v = \perp \end{cases}$$

Recap II: Powerdomain

- For a set X, the Plotkin powerdomain $\mathcal{P}(X_\perp)$ over its flat domain $X_\perp$ is a poset on

$$\{S \subseteq A \mid S \neq \emptyset, S \text{ is finite or } \perp \in S\}$$

  whose partial order is

$$P \sqsubseteq Q \quad \iff \quad (\perp \in P \wedge P \subseteq Q \cup \{\perp\}) \vee (\perp \notin P \wedge P = Q) \ .$$

  is a cpo with $\{\perp\}$ the least element
- set-valued condition $b : X \to \mathcal{P}(\mathbb{2}_\perp)$ and set-value command $c : X \to \mathcal{P}(X_\perp)$

$$[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]\gamma = \bigcup_{v \in b(\gamma)} \begin{cases} [\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]^\dagger \circ c(\gamma) & \text{if } v = \texttt{true} \\ \{\gamma\} & \text{if } v = \texttt{false} \\ \{\perp\} & \text{if } v = \perp \end{cases}$$

- $\mathcal{W}_{b,c}(f : X \to \mathcal{P}(X_\perp)) = \gamma \mapsto \bigcup_{v \in b(\gamma)} \begin{cases} f^\dagger \circ c(\gamma) & \text{if } v = \texttt{true} \\ \{\gamma\} & \text{if } v = \texttt{false} \\ \{\perp\} & \text{if } v = \perp \end{cases}$

  is continuous

Recap II: Powerdomain

- For a set X, the Plotkin powerdomain $\mathcal{P}(X_\perp)$ over its flat domain $X_\perp$ is a poset on

$$\{S \subseteq A \mid S \neq \emptyset, S \text{ is finite or } \perp \in S\}$$

  whose partial order is

$$P \sqsubseteq Q \quad \iff \quad (\perp \in P \wedge P \subseteq Q \cup \{\perp\}) \vee (\perp \notin P \wedge P = Q) \ .$$

  is a cpo with $\{\perp\}$ the least element
- set-valued condition $b : X \to \mathcal{P}(\mathbb{2}_\perp)$ and set-value command $c : X \to \mathcal{P}(X_\perp)$

$$[\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]\gamma = \bigcup_{v \in b(\gamma)} \begin{cases} [\![\texttt{while}\ b\ \texttt{do}\ c\ \texttt{end}]\!]^\dagger \circ c(\gamma) & \text{if } v = \texttt{true} \\ \{\gamma\} & \text{if } v = \texttt{false} \\ \{\perp\} & \text{if } v = \perp \end{cases}$$

- $\mathcal{W}_{b,c}(f : X \to \mathcal{P}(X_\perp)) = \gamma \mapsto \bigcup_{v \in b(\gamma)} \begin{cases} f^\dagger \circ c(\gamma) & \text{if } v = \texttt{true} \\ \{\gamma\} & \text{if } v = \texttt{false} \\ \{\perp\} & \text{if } v = \perp \end{cases}$

  is continuous
- Defined $\mathbf{While}(b, c) := \mathbf{LFP}(\mathcal{W}_{b,c})$

Plotkin Powerdomain in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

Depending on the meaning of $\bot$, there can be different Plotkin powerdomains in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:

Definition (For $\star \in \{\flat, \sharp, \natural\}$)
Define the closure: for any non-empty subset $S \subseteq X \cup \{\star\}$,

$$S_\star \coloneqq \begin{cases} S \cup \{\star\} & \text{if } S \text{ is infinite} \\ S & \text{otherwise} \end{cases}$$

Then monad $\mathbf{P}_\star : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ which on an assembly $\mathbf{X}$ is the natural sub-assembly of $\mathsf{M}(\star\mathbf{X})$ induced by $_\star$:

$$\varphi \Vdash_{\mathbf{P}(\mathbf{X}_\star)} S \Leftrightarrow \varphi \Vdash_{\mathsf{M}(\star\mathbf{X})} S$$

Plotkin Powerdomain in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

Depending on the meaning of $\bot$, there can be different Plotkin powerdomains in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:

Definition (For $\star \in \{\flat, \sharp, \natural\}$)
Define the closure: for any non-empty subset $S \subseteq X \cup \{\star\}$,

$$S_\star \coloneqq \begin{cases} S \cup \{\star\} & \text{if } S \text{ is infinite} \\ S & \text{otherwise} \end{cases}$$

Then monad $\mathbf{P}_\star : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ which on an assembly $\mathbf{X}$ is the natural sub-assembly of $\mathsf{M}(\star\mathbf{X})$ induced by $_\star$:

$$\varphi \Vdash_{\mathbf{P}(\mathbf{X}_\star)} S \Leftrightarrow \varphi \Vdash_{\mathsf{M}(\star\mathbf{X})} S$$

Remark (For $\star \in \{\flat, \sharp, \natural\}$)
$\mathbf{P}_\star$ is a monad with the same set-theoretic definition of $\mathcal{P}(_\bot)$. I.e., Kleisli compositions of computable multi-functions are computable.

Plotkin Powerdomain in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

Depending on the meaning of $\bot$, there can be different Plotkin powerdomains in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:

Definition (For $\star \in \{\flat, \sharp, \natural\}$)
Define the closure: for any non-empty subset $S \subseteq X \cup \{\star\}$,

$$S_\star \coloneqq \begin{cases} S \cup \{\star\} & \text{if } S \text{ is infinite} \\ S & \text{otherwise} \end{cases}$$

Then monad $\mathbf{P}_\star : \mathsf{Asm}(\mathbb{N}^{\mathbb{N}}) \to \mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ which on an assembly $\mathbf{X}$ is the natural sub-assembly of $\mathsf{M}(\star\mathbf{X})$ induced by $_\star$:

$$\varphi \Vdash_{\mathbf{P}(\mathbf{X}_\star)} S \Leftrightarrow \varphi \Vdash_{\mathsf{M}(\star\mathbf{X})} S$$

Remark (For $\star \in \{\flat, \sharp, \natural\}$)
$\mathbf{P}_\star$ is a monad with the same set-theoretic definition of $\mathcal{P}(_\bot)$. I.e., Kleisli compositions of computable multi-functions are computable.

Remark
nondeterminism construct $\mathsf{choose}_n : \mathbf{K}^n \to \flat\mathsf{M}(\mathbf{n})$ appears only as

$\mathsf{choose}_n : \mathbf{K}^n \to \mathbf{P}(\mathbf{n}_\flat)$ or $\mathsf{choose}_n : \mathbf{K}^n \to \mathbf{P}(\mathbf{n}_\natural)$

but not $\mathsf{choose}_n : \mathbf{K}^n \to \mathbf{P}(\mathbf{n}_\sharp)$

Lemma

1. $\mathbf{P}_\flat$, *when* $\perp$ *is identified as nontermination:*
   the least fixed-point is uniformly computable: there is

$$\mathbf{LFP} : (\mathbf{P}(\mathbf{X}_\flat))^{\mathbf{N}} \to \flat\mathbf{P}(\mathbf{X}_\flat)$$

   defined on chains that computes the least fixed points.

Lemma

1. $\mathbf{P}_\flat$, *when* $\bot$ *is identified as nontermination:*
   the least fixed-point is uniformly computable: there is

$$\mathbf{LFP} : (\mathbf{P}(\mathbf{X}_\flat))^{\mathbf{N}} \to \flat\mathbf{P}(\mathbf{X}_\flat)$$

   defined on chains that computes the least fixed points.

2. $\mathbf{P}_\natural$, *when* $\bot$ *is identified as general partiality:*
   the least fixed-point is not uniformly computable.
   However, the while operator is uniformly computable: there is

$$\mathbf{While} : \mathbf{P}(\mathbf{2}_\natural)^{\mathbf{X}} \times \mathbf{P}(\mathbf{X}_\natural)^{\mathbf{X}} \to (\mathbf{X}_\natural)^{\mathbf{X}}$$

   computing the least fixed point of $\mathcal{W}_{b,c}$

Lemma

1. $\mathbf{P}_\flat$, *when* $\bot$ *is identified as nontermination:*
   the least fixed-point is uniformly computable: there is

$$\mathbf{LFP} : (\mathbf{P}(\mathbf{X}_\flat))^{\mathbf{N}} \to \flat\mathbf{P}(\mathbf{X}_\flat)$$

   defined on chains that computes the least fixed points.

2. $\mathbf{P}_\natural$, *when* $\bot$ *is identified as general partiality:*
   the least fixed-point is not uniformly computable.
   However, the while operator is uniformly computable: there is

$$\mathbf{While} : \mathbf{P}(\mathbf{2}_\natural)^{\mathbf{X}} \times \mathbf{P}(\mathbf{X}_\natural)^{\mathbf{X}} \to (\mathbf{X}_\natural)^{\mathbf{X}}$$

   computing the least fixed point of $\mathcal{W}_{b,c}$

3. $\mathbf{P}_\sharp$, *when* $\bot$ *identified as fixing: the least fixed-point nor the while operator is computable.*

Denotational Semantics

Syntax

data types	τ	$::=$	K	Kleenan (lazy Boolean)
		$\mid$	Z	integer
		$\mid$	R	real number
expressions	e	$::=$	$\texttt{true} \mid \texttt{false} \mid 0 \mid 1 \mid \cdots$	constant
		$\mid$	$e_1 + e_2 \mid e_1 - e_2 \mid e_1 \times e_2$	integer arithmetic
		$\mid$	$e_1 = e_2 \mid e_1 \leq e_2$	integer comparison
		$\mid$	$e_1 \dot{+} e_2 \mid e_1 \dot{-} e_2 \mid e_1 \dot{\times} e_2 \mid e^{-1}$	real arithmetic
		$\mid$	$e_1 \lesssim e_2$	partial real comparison
		$\mid$	$\mathsf{choose}(e_1, \cdots, e_n)$	nondeterminism
		$\mid$	$2^e \mid \iota(e)$	coercion from Z to R
commands	c	$::=$	$\texttt{skip}$	do nothing
		$\mid$	$c_1; c_2$	composition
		$\mid$	$x \coloneqq e$	assignment
		$\mid$	$\texttt{if}\ e\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \texttt{end}$	conditional
		$\mid$	$\texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}$	loop

Denotational Semantics

1. Given a state γ, an *expression* evaluates to a value:

$$x + y \rightsquigarrow_\gamma 42 \quad \text{when } x \rightsquigarrow_\gamma 21 \text{ and } y \rightsquigarrow_\gamma 21$$

2. Due to *nondeterminism*, there are several possible evaluations:

$$\texttt{choose}(\texttt{true}, \texttt{true}) \rightsquigarrow_\gamma 1 \text{ or } 2$$

3. There are nonterminating evaluations

$$0^{-1} \rightsquigarrow_\gamma \ \bot \qquad \pi \lesssim \pi \rightsquigarrow_\gamma \ \bot$$

Denotational Semantics

1. Given a state γ, an *expression* evaluates to a value:

$$x + y \rightsquigarrow_\gamma 42 \quad \text{when } x \rightsquigarrow_\gamma 21 \text{ and } y \rightsquigarrow_\gamma 21$$

2. Due to *nondeterminism*, there are several possible evaluations:

$$\mathsf{choose}(\mathtt{true}, \mathtt{true}) \rightsquigarrow_\gamma 1 \text{ or } 2$$

3. There are nonterminating evaluations

$$0^{-1} \rightsquigarrow_\gamma \ \bot \qquad \pi \lesssim \pi \rightsquigarrow_\gamma \ \bot$$

4. Denotation is the set of all nondeterministic evaluations:

$$[\![\mathsf{choose}(\mathtt{true}, \mathtt{true})]\!]\gamma = \{1, 2\}$$

Denotational Semantics

1. Given a state γ, an *expression* evaluates to a value:

$$x + y \rightsquigarrow_\gamma 42 \quad \text{when } x \rightsquigarrow_\gamma 21 \text{ and } y \rightsquigarrow_\gamma 21$$

2. Due to *nondeterminism*, there are several possible evaluations:

$$\mathsf{choose}(\mathtt{true}, \mathtt{true}) \rightsquigarrow_\gamma 1 \text{ or } 2$$

3. There are nonterminating evaluations

$$0^{-1} \rightsquigarrow_\gamma \ \bot \qquad \pi \lesssim \pi \rightsquigarrow_\gamma \ \bot$$

4. Denotation is the set of all nondeterministic evaluations:

$$\begin{aligned} [\![\mathsf{choose}(\mathtt{true}, \mathtt{true})]\!]\gamma &= \{1, 2\} \\ [\![\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1]\!]\gamma &= \{0, 1\} \end{aligned}$$

Denotational Semantics

1. Given a state γ, an *expression* evaluates to a value:

$$x + y \rightsquigarrow_\gamma 42 \quad \text{when } x \rightsquigarrow_\gamma 21 \text{ and } y \rightsquigarrow_\gamma 21$$

2. Due to *nondeterminism*, there are several possible evaluations:

$$\mathsf{choose}(\mathtt{true}, \mathtt{true}) \rightsquigarrow_\gamma 1 \text{ or } 2$$

3. There are nonterminating evaluations

$$0^{-1} \rightsquigarrow_\gamma \ \bot \qquad \pi \lesssim \pi \rightsquigarrow_\gamma \ \bot$$

4. Denotation is the set of all nondeterministic evaluations:

$$\begin{aligned}
[\![\mathsf{choose}(\mathtt{true}, \mathtt{true})]\!]\gamma &= \{1, 2\} \\
[\![\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1]\!]\gamma &= \{0, 1\} \\
[\![(\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1)^{-1}]\!]\gamma &= \{\bot, 1\}
\end{aligned}$$

Denotational Semantics

1. Given a state γ, an *expression* evaluates to a value:

$$x + y \rightsquigarrow_\gamma 42 \quad \text{when } x \rightsquigarrow_\gamma 21 \text{ and } y \rightsquigarrow_\gamma 21$$

2. Due to *nondeterminism*, there are several possible evaluations:

$$\mathsf{choose}(\mathtt{true}, \mathtt{true}) \rightsquigarrow_\gamma 1 \text{ or } 2$$

3. There are nonterminating evaluations

$$0^{-1} \rightsquigarrow_\gamma \ \bot \qquad \pi \lesssim \pi \rightsquigarrow_\gamma \ \bot$$

4. Denotation is the set of all nondeterministic evaluations:

$$\begin{aligned} [\![\mathsf{choose}(\mathtt{true}, \mathtt{true})]\!]\gamma &= \{1, 2\} \\ [\![\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1]\!]\gamma &= \{0, 1\} \\ [\![(\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1)^{-1}]\!]\gamma &= \{\bot, 1\} \end{aligned}$$

5. Denotation of a command is the set of all nondeterministic resulting states

$$[\![x := \mathsf{choose}(\mathtt{true}, \mathtt{true})]\!]\gamma = \{\gamma[x \mapsto 1], \gamma[x \mapsto 2]\}$$

Denotational Semantics

1. Given a state γ, an *expression* evaluates to a value:

$$x + y \rightsquigarrow_\gamma 42 \quad \text{when } x \rightsquigarrow_\gamma 21 \text{ and } y \rightsquigarrow_\gamma 21$$

2. Due to *nondeterminism*, there are several possible evaluations:

$$\mathsf{choose}(\mathtt{true}, \mathtt{true}) \rightsquigarrow_\gamma 1 \text{ or } 2$$

3. There are nonterminating evaluations

$$0^{-1} \rightsquigarrow_\gamma \ \bot \qquad \pi \lesssim \pi \rightsquigarrow_\gamma \ \bot$$

4. Denotation is the set of all nondeterministic evaluations:

$$\begin{aligned}
[\![\mathsf{choose}(\mathtt{true}, \mathtt{true})]\!]\gamma &= \{1, 2\} \\
[\![\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1]\!]\gamma &= \{0, 1\} \\
[\![(\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1)^{-1}]\!]\gamma &= \{\bot, 1\}
\end{aligned}$$

5. Denotation of a command is the set of all nondeterministic resulting states

$$\begin{aligned}
[\![x := \mathsf{choose}(\mathtt{true}, \mathtt{true})]\!]\gamma &= \{\gamma[x \mapsto 1], \gamma[x \mapsto 2]\} \\
[\![x := (\mathsf{choose}(\mathtt{true}, \mathtt{true}) - 1)^{-1}]\!]\gamma &= \{\bot, \gamma[x \mapsto 1]\}
\end{aligned}$$

Denotations of data types

$$[\![\mathsf{R}]\!] = |\mathbf{R}| \qquad [\![\mathsf{K}]\!] = |\mathbf{K}| \qquad [\![\mathsf{Z}]\!] = |\mathbf{Z}|$$

Denotations of data types and expressions

Denotations of data types

$$[\![\mathsf{R}]\!] = |\mathbf{R}| \qquad [\![\mathsf{K}]\!] = |\mathbf{K}| \qquad [\![\mathsf{Z}]\!] = |\mathbf{Z}|$$

Denotations of expressions

$$[\![e : \tau]\!]\gamma = \text{the set of all possible evaluations of } e \text{ under } \gamma \in \mathcal{P}([\![\tau]\!]_\perp)$$

Denotations of data types and expressions

Denotations of data types

$$[\![\mathsf{R}]\!] = |\mathbf{R}| \qquad [\![\mathsf{K}]\!] = |\mathbf{K}| \qquad [\![\mathsf{Z}]\!] = |\mathbf{Z}|$$

Denotations of expressions

$$\begin{aligned}
[\![e : \tau]\!]\gamma &= \text{the set of all possible evaluations of } e \text{ under } \gamma \in \mathcal{P}([\![\tau]\!]_\perp) \\
[\![\mathsf{c} : \tau]\!] &\coloneqq \gamma \mapsto \{\mathsf{c}\} \\
[\![v : \tau]\!] &\coloneqq \gamma \mapsto \{\gamma(v)\} \\
[\![t_1 \odot t_2 : \tau]\!] &\coloneqq \odot^\dagger \circ \langle [\![t_1 : \tau']\!], [\![t_2 : \tau']\!] \rangle \qquad \text{for } \odot \in \{+, \times, =, \leq, \lesssim\} \\
[\![\star t : \tau]\!] &\coloneqq \star^\dagger \circ [\![t : \tau']\!] \qquad \text{for } \star \in \{-, {}^{-1}, 2^{\square}, \iota\} \\
[\![\mathsf{choose}_n(e_1, \cdots, e_n) : \mathsf{Z}]\!] &\coloneqq \mathsf{choose}_n^\dagger \circ \langle [\![e_1 : \mathsf{K}]\!], \cdots, [\![e_n : \mathsf{K}]\!] \rangle
\end{aligned}$$

recall $f^\dagger(S, T) = \bigcup_{s \in S, t \in T} \begin{cases} f(s,t) & \text{if } s, t \neq \perp \\ \{\perp\} & \text{otherwise} \end{cases}$ for $f : X_1 \times X_2 \to \mathcal{P}(Y_\perp)$ and $x \mapsto \{x\}$ implicit

Denotations of data types and expressions

Denotations of data types

$$[\![\mathsf{R}]\!] = |\mathbf{R}| \qquad [\![\mathsf{K}]\!] = |\mathbf{K}| \qquad [\![\mathsf{Z}]\!] = |\mathbf{Z}|$$

Denotations of expressions

$$\begin{aligned}
[\![e : \tau]\!]\gamma &= \text{the set of all possible evaluations of } e \text{ under } \gamma \in \mathcal{P}([\![\tau]\!]_\perp) \\
[\![\mathsf{c} : \tau]\!] &\coloneqq \gamma \mapsto \{\mathsf{c}\} \\
[\![v : \tau]\!] &\coloneqq \gamma \mapsto \{\gamma(v)\} \\
[\![t_1 \odot t_2 : \tau]\!] &\coloneqq \odot^\dagger \circ \langle [\![t_1 : \tau']\!], [\![t_2 : \tau']\!] \rangle \qquad \text{for } \odot \in \{+, \times, =, \leq, \lesssim\} \\
[\![\star t : \tau]\!] &\coloneqq \star^\dagger \circ [\![t : \tau']\!] \qquad \text{for } \star \in \{-, {}^{-1}, 2^{\square}, \iota\} \\
[\![\mathsf{choose}_n(e_1, \cdots, e_n) : \mathsf{Z}]\!] &\coloneqq \mathsf{choose}_n^\dagger \circ \langle [\![e_1 : \mathsf{K}]\!], \cdots, [\![e_n : \mathsf{K}]\!] \rangle
\end{aligned}$$

recall $f^\dagger(S,T) = \bigcup_{s \in S, t \in T} \begin{cases} f(s,t) & \text{if } s,t \neq \perp \\ \{\perp\} & \text{otherwise} \end{cases}$ for $f : X_1 \times X_2 \to \mathcal{P}(Y_\perp)$ and $x \mapsto \{x\}$ implicit

Theorem
Identifying $\perp$ with $\natural$, hence $\mathcal{P}(\__\perp)$ with $\mathbf{P}_\natural$, under the standard representations, for any expression e, its denotation is computable $[\![e]\!] : \Sigma \to \mathcal{P}([\![\tau]\!]_\perp)$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

Denotations of commands

$[\![c]\!]\gamma$ = the set of possible resulting states of executing c under γ

Denotations of commands

$$
\begin{aligned}
[\![c]\!]\gamma &= \text{the set of possible resulting states of executing } c \text{ under } \gamma \\
[\![\texttt{skip}]\!] &\coloneqq \gamma \mapsto \{\gamma\} \\
[\![v \coloneqq e]\!] &\coloneqq (x \mapsto \{\gamma[v \mapsto x]\})^\dagger \circ [\![e]\!] \\
[\![c_1; c_2]\!] &\coloneqq [\![c_2]\!]^\dagger \circ [\![c_1]\!] \\
[\![\texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}]\!] &\coloneqq \mathrm{Cond} \circ \langle [\![e]\!], [\![c_1]\!], [\![c_2]\!] \rangle \\
[\![\texttt{while } e \texttt{ do } c \texttt{ end}]\!] &\coloneqq \mathbf{While}([\![b]\!], [\![S]\!])
\end{aligned}
$$

Denotations of commands

$$
\begin{aligned}
[\![c]\!]\gamma &= \text{the set of possible resulting states of executing } c \text{ under } \gamma \\
[\![\texttt{skip}]\!] &\coloneqq \gamma \mapsto \{\gamma\} \\
[\![v \coloneqq e]\!] &\coloneqq (x \mapsto \{\gamma[v \mapsto x]\})^\dagger \circ [\![e]\!] \\
[\![c_1; c_2]\!] &\coloneqq [\![c_2]\!]^\dagger \circ [\![c_1]\!] \\
[\![\texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}]\!] &\coloneqq \mathrm{Cond} \circ \langle [\![e]\!], [\![c_1]\!], [\![c_2]\!] \rangle \\
[\![\texttt{while } e \texttt{ do } c \texttt{ end}]\!] &\coloneqq \mathbf{While}([\![b]\!], [\![S]\!])
\end{aligned}
$$

Theorem
Identifying $\bot$ with $\natural$, hence $\mathcal{P}(\_\bot)$ with $\mathbf{P}_\natural$, under the standard representations, for each command c, its denotation is computable $[\![c]\!] : \Sigma \to \mathcal{P}(\Sigma_\bot)$

Denotations of programs

Ordinary program:

$$[\![\mathbf{input}(x_1 : \tau_1, \cdots, x_d : \tau_d)\ c\ \mathbf{return}\ e]\!] \coloneqq [\![e]\!]^\dagger \circ [\![c]\!](x_1, \cdots, x_d)$$

Denotations of programs

Ordinary program:

$$[\![\mathbf{input}(x_1 : \tau_1, \cdots, x_d : \tau_d)\ c\ \mathbf{return}\ e]\!] \coloneqq [\![e]\!]^\dagger \circ [\![c]\!](x_1, \cdots, x_d)$$

Limit program:

$$[\![\mathbf{lim}(p : \mathsf{Z}; x_1 : \tau_1, \cdots, x_d : \tau_d)\ c\ \mathbf{return}\ e]\!] \coloneqq \lim_{p \to -\infty}([\![e]\!]^\dagger \circ [\![c]\!](p, x_1, \cdots, x_d))$$

where

$$\lim(f : \mathsf{N} \to \mathcal{P}(\mathbb{R}_\perp)) := \begin{cases} x & \text{if } \forall n.\ \forall y \in f(n).\ y \neq \perp \wedge |x - y| \leq 2^{-n} \\ \perp & \text{otherwise} \end{cases}$$

Denotations of programs

Ordinary program:

$$[\![\mathbf{input}(x_1 : \tau_1, \cdots, x_d : \tau_d)\ c\ \mathbf{return}\ e]\!] \coloneqq [\![e]\!]^\dagger \circ [\![c]\!](x_1, \cdots, x_d)$$

Limit program:

$$[\![\mathbf{lim}(p : \mathsf{Z}; x_1 : \tau_1, \cdots, x_d : \tau_d)\ c\ \mathbf{return}\ e]\!] \coloneqq \lim_{p \to -\infty}([\![e]\!]^\dagger \circ [\![c]\!](p, x_1, \cdots, x_d))$$

where

$$\lim(f : \mathsf{N} \to \mathcal{P}(\mathbb{R}_\perp)) := \begin{cases} x & \text{if } \forall n.\ \forall y \in f(n).\ y \neq \perp \wedge |x - y| \leq 2^{-n} \\ \perp & \text{otherwise} \end{cases}$$

Theorem
Identifying $\perp$ with $\natural$, hence $\mathcal{P}(_\perp)$ with $\mathbf{P}_\natural$, under the standard representations,

for each ordinary program P, its denotation is computable

$$[\![P]\!] : [\![\tau_1]\!] \times \cdots \times [\![\tau_d]\!] \to \mathcal{P}([\![\tau]\!]_\perp)$$

and for each limit program P, its denotation is computable

$$[\![P]\!] : [\![\tau_1]\!] \times \cdots \times [\![\tau_d]\!] \to \mathbb{R}_\perp$$

Completeness

Theorem
Any computable partial real functions and multi-valued integer functions can be realized.

Proof.

1. our command language restricted on integers without choose is Turing complete

□

Completeness

Theorem
Any computable partial real functions and multi-valued integer functions can be realized.

Proof.

1. our command language restricted on integers without choose is Turing complete
2. we can simulate finite map structure $(a_1, b_1), (a_2, b_2), \cdots$ encoded in integer variables

□

Completeness

Theorem
Any computable partial real functions and multi-valued integer functions can be realized.

Proof.

1. our command language restricted on integers without choose is Turing complete
2. we can simulate finite map structure $(a_1, b_1), (a_2, b_2), \cdots$ encoded in integer variables
3. for any real variable x, we can nondeterministically compute its approximation $q_n : |x - q_n| \leq 2^{-n}$ for each n by rounding.

□

Completeness

Theorem
Any computable partial real functions and multi-valued integer functions can be realized.

Proof.

1. our command language restricted on integers without choose is Turing complete
2. we can simulate finite map structure $(a_1, b_1), (a_2, b_2), \cdots$ encoded in integer variables
3. for any real variable x, we can nondeterministically compute its approximation $q_n : |x - q_n| \leq 2^{-n}$ for each n by rounding.
4. Then, (see the whiteboard)

□

Formal Verification

Specification

$$\{\phi\}\ c\ \{\psi\}$$

All states satisfying ϕ make c terminate and results states satisfying ψ.

$$\{\phi\}\ c\ \{\psi\}$$

All states satisfying ϕ make c terminate and results states satisfying ψ.

Theorem
The first order logic over the structures of integers and reals connected via $\mathbb{Z} \ni z \mapsto z \in \mathbb{R}$ *and* $\mathbb{Z} \ni z \mapsto 2^z \in \mathbb{R}$ *is expressive for the expression language;*

for any expression e*, there is a predicate* $(\!|e|\!)(y)$ *that defines the denotation of* e

$(\!|e|\!)(v)$ *iff* v *is in (but* $\bot$ *is not in) the denotation of* e

$$\{\phi\}\ c\ \{\psi\}$$

All states satisfying ϕ make c terminate and results states satisfying ψ.

Theorem
The first order logic over the structures of integers and reals connected via $\mathbb{Z} \ni z \mapsto z \in \mathbb{R}$ *and* $\mathbb{Z} \ni z \mapsto 2^z \in \mathbb{R}$ *is expressive for the expression language;*

for any expression e*, there is a predicate* $(\!|e|\!)(y)$ *that defines the denotation of* e

$$(\!|e|\!)(v) \text{ iff } v \text{ is in (but } \bot \text{ is not in) the denotation of } e$$

For example, (in the simplified form)

$$(\!|x \lesssim y|\!)(v) \equiv (v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x)$$
$$(\!|\mathsf{choose}_n(e_1, \cdots, e_n)|\!)(v) \equiv (v = 1 \wedge (\!|e_1|\!)(\texttt{true})) \vee \cdots \vee (v = n \wedge (\!|e_n|\!)(\texttt{true}))$$

$$\{\phi\}\ c\ \{\psi\}$$

All states satisfying ϕ make c terminate and results states satisfying ψ.

Theorem
The first order logic over the structures of integers and reals connected via $\mathbb{Z} \ni z \mapsto z \in \mathbb{R}$ *and* $\mathbb{Z} \ni z \mapsto 2^z \in \mathbb{R}$ *is expressive for the expression language;*

for any expression e*, there is a predicate* $(\!|e|\!)(y)$ *that defines the denotation of* e

$$(\!|e|\!)(v) \text{ iff } v \text{ is in (but } \bot \text{ is not in) the denotation of } e$$

For example, (in the simplified form)

$$(\!|x \lesssim y|\!)(v) \equiv (v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x)$$
$$(\!|\mathsf{choose}_n(e_1, \cdots, e_n)|\!)(v) \equiv (v = 1 \wedge (\!|e_1|\!)(\texttt{true})) \vee \cdots \vee (v = n \wedge (\!|e_n|\!)(\texttt{true}))$$

$$\exists v.\ (\!|e|\!)(v) \quad \text{iff} \quad \bot \notin [\![e]\!]\gamma$$
$$\forall v.\ (\!|e|\!)(v) \Rightarrow \psi(v) \quad \text{iff} \quad \forall v \in [\![e]\!]\gamma \text{ satisfy } \psi$$

Proof Rule: assignment

$$\overline{\left\{\exists v.\ (\!|e|\!)(v) \wedge \forall v.\ (\!|e|\!)(v) \Rightarrow \psi[v/x]\right\}\ x \coloneqq e\ \left\{\psi\right\}}$$

For an initial state γ,

- γ satisfying $\exists v.\ (\!|e|\!)(v)$ ensures $\bot \notin [\![e]\!]\gamma$
- Resulting state is $\gamma[x \mapsto v]$ for each $v \in [\![e]\!]\gamma$
- If γ satisfies $\forall v.\ (\!|s|\!)(v) \Rightarrow \psi[v/x]$, any $v \in [\![e]\!]\gamma$, γ satisfies $\psi[v/x]$
- Hence, resulting states satisfy ψ

Proof Rule: assignment

$$\overline{\left\{\exists v.\ (\!|e|\!)(v) \wedge \forall v.\ (\!|e|\!)(v) \Rightarrow \psi[v/x]\right\}\ x \coloneqq e\ \left\{\psi\right\}}$$

For an initial state γ,

- γ satisfying $\exists v.\ (\!|e|\!)(v)$ ensures $\bot \notin [\![e]\!]\gamma$
- Resulting state is $\gamma[x \mapsto v]$ for each $v \in [\![e]\!]\gamma$
- If γ satisfies $\forall v.\ (\!|s|\!)(v) \Rightarrow \psi[v/x]$, any $v \in [\![e]\!]\gamma$, γ satisfies $\psi[v/x]$
- Hence, resulting states satisfy ψ

Example:

$$b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

Proof Rule: assignment

$$\overline{\left\{\exists v.\ (\!|e|\!)(v) \wedge \forall v.\ (\!|e|\!)(v) \Rightarrow \psi[v/x]\right\}\ x \coloneqq e\ \left\{\psi\right\}}$$

For an initial state γ,

- γ satisfying $\exists v.\ (\!|e|\!)(v)$ ensures $\bot \notin [\![e]\!]\gamma$
- Resulting state is $\gamma[x \mapsto v]$ for each $v \in [\![e]\!]\gamma$
- If γ satisfies $\forall v.\ (\!|s|\!)(v) \Rightarrow \psi[v/x]$, any $v \in [\![e]\!]\gamma$, γ satisfies $\psi[v/x]$
- Hence, resulting states satisfy ψ

Example:

$$\left\{(\exists v.\ (\!|x \lesssim y|\!)(v)) \wedge \forall v.\ (\!|x \lesssim y|\!)(v) \Rightarrow (b = \texttt{true})[v/b]\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

Proof Rule: assignment

$$\overline{\left\{\exists v.\ (\!|e|\!)(v) \wedge \forall v.\ (\!|e|\!)(v) \Rightarrow \psi[v/x]\right\}\ x \coloneqq e\ \left\{\psi\right\}}$$

For an initial state γ,

- γ satisfying $\exists v.\ (\!|e|\!)(v)$ ensures $\bot \notin [\![e]\!]\gamma$
- Resulting state is $\gamma[x \mapsto v]$ for each $v \in [\![e]\!]\gamma$
- If γ satisfies $\forall v.\ (\!|s|\!)(v) \Rightarrow \psi[v/x]$, any $v \in [\![e]\!]\gamma$, γ satisfies $\psi[v/x]$
- Hence, resulting states satisfy ψ

Example:

$$\left\{(\exists v.\ (\!|x \lesssim y|\!)(v)) \wedge \forall v.\ (\!|x \lesssim y|\!)(v) \Rightarrow (b = \texttt{true})[v/b]\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

$$\left\{(\exists v.\ (v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x)) \wedge \forall v.\ ((v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x)) \Rightarrow v = \texttt{true}\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

Proof Rule: assignment

$$\overline{\left\{\exists v.\ (\!|e|\!)(v) \wedge \forall v.\ (\!|e|\!)(v) \Rightarrow \psi[v/x]\right\}\ x \coloneqq e\ \left\{\psi\right\}}$$

For an initial state γ,

- γ satisfying $\exists v.\ (\!|e|\!)(v)$ ensures $\bot \notin [\![e]\!]\gamma$
- Resulting state is $\gamma[x \mapsto v]$ for each $v \in [\![e]\!]\gamma$
- If γ satisfies $\forall v.\ (\!|s|\!)(v) \Rightarrow \psi[v/x]$, any $v \in [\![e]\!]\gamma$, γ satisfies $\psi[v/x]$
- Hence, resulting states satisfy ψ

Example:

$$\left\{(\exists v.\ (\!|x \lesssim y|\!)(v)) \wedge \forall v.\ (\!|x \lesssim y|\!)(v) \Rightarrow (b = \texttt{true})[v/b]\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

$$\left\{(\exists v.\ (v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x)) \wedge \forall v.\ ((v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x)) \Rightarrow v = \texttt{true}\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

$$\left\{(x < y \vee y < x) \wedge x \leq y\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

Proof Rule: assignment

$$\overline{\left\{\exists v.\ (\!|e|\!)(v) \wedge \forall v.\ (\!|e|\!)(v) \Rightarrow \psi[v/x]\right\}\ x \coloneqq e\ \left\{\psi\right\}}$$

For an initial state γ,

- γ satisfying $\exists v.\ (\!|e|\!)(v)$ ensures $\bot \notin [\![e]\!]\gamma$
- Resulting state is $\gamma[x \mapsto v]$ for each $v \in [\![e]\!]\gamma$
- If γ satisfies $\forall v.\ (\!|s|\!)(v) \Rightarrow \psi[v/x]$, any $v \in [\![e]\!]\gamma$, γ satisfies $\psi[v/x]$
- Hence, resulting states satisfy ψ

Example:

$$\left\{(\exists v.\ (\!|x \lesssim y|\!)(v)) \wedge \forall v.\ (\!|x \lesssim y|\!)(v) \Rightarrow (b = \texttt{true})[v/b]\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

$$\left\{(\exists v.\ (v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x))\right.$$

$$\left.\wedge \forall v.\ ((v = \texttt{true} \wedge x < y) \vee (v = \texttt{false} \wedge y < x)) \Rightarrow v = \texttt{true}\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

$$\left\{(x < y \vee y < x) \wedge x \leq y\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

$$\left\{x < y\right\}\ b \coloneqq x \lesssim y\ \left\{b = \texttt{true}\right\}$$

Proof Rule: while loop

$$\frac{\left\{(\!|e|\!)(\texttt{true}) \wedge I \wedge V = \xi \wedge L = \xi'\right\}\ c\ \left\{I \wedge V \leq \xi - \xi' \wedge L = \xi'\right\}}{\left\{I\right\}\ \texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}\ \left\{I \wedge (\!|e|\!)(\texttt{false})\right\}}$$

- $I \wedge (\!|e|\!)(\texttt{true}) \Rightarrow L > 0$
- $I \Rightarrow ((\!|e|\!)(\texttt{true}) \vee (\!|e|\!)(\texttt{false}))$
- $I \wedge V \leq 0 \Rightarrow \forall k.\ (\!|e|\!)(k) \Rightarrow k = \texttt{false}$

Proof Rule: while loop

$$\frac{\left\{(\!|e|\!)(\texttt{true}) \wedge I \wedge V = \xi \wedge L = \xi'\right\}\ c\ \left\{I \wedge V \leq \xi - \xi' \wedge L = \xi'\right\}}{\left\{I\right\}\ \texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}\ \left\{I \wedge (\!|e|\!)(\texttt{false})\right\}}$$

- $I \wedge (\!|e|\!)(\texttt{true}) \Rightarrow L > 0$
- $I \Rightarrow ((\!|e|\!)(\texttt{true}) \vee (\!|e|\!)(\texttt{false}))$
- $I \wedge V \leq 0 \Rightarrow \forall k.\ (\!|e|\!)(k) \Rightarrow k = \texttt{false}$

- $(\!|e|\!)(\texttt{true})$ ensures $e \rightsquigarrow \texttt{true}$ and $(\!|e|\!)(\texttt{false})$ ensures $e \rightsquigarrow \texttt{false}$

Proof Rule: while loop

$$\frac{\left\{(\!|e|\!)(\texttt{true}) \wedge I \wedge V = \xi \wedge L = \xi'\right\}\ c\ \left\{I \wedge V \leq \xi - \xi' \wedge L = \xi'\right\}}{\left\{I\right\}\ \texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}\ \left\{I \wedge (\!|e|\!)(\texttt{false})\right\}}$$

- $I \wedge (\!|e|\!)(\texttt{true}) \Rightarrow L > 0$
- $I \Rightarrow ((\!|e|\!)(\texttt{true}) \vee (\!|e|\!)(\texttt{false}))$
- $I \wedge V \leq 0 \Rightarrow \forall k.\ (\!|e|\!)(k) \Rightarrow k = \texttt{false}$

- $(\!|e|\!)(\texttt{true})$ ensures $e \rightsquigarrow \texttt{true}$ and $(\!|e|\!)(\texttt{false})$ ensures $e \rightsquigarrow \texttt{false}$
- I is a loop-invariant formula, L is a loop-invariant quantity, and V is a loop-variant quantity

Proof Rule: while loop

$$\frac{\left\{(\!|e|\!)(\texttt{true}) \wedge I \wedge V = \xi \wedge L = \xi'\right\}\ c\ \left\{I \wedge V \leq \xi - \xi' \wedge L = \xi'\right\}}{\left\{I\right\}\ \texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}\ \left\{I \wedge (\!|e|\!)(\texttt{false})\right\}}$$

- $I \wedge (\!|e|\!)(\texttt{true}) \Rightarrow L > 0$
- $I \Rightarrow ((\!|e|\!)(\texttt{true}) \vee (\!|e|\!)(\texttt{false}))$
- $I \wedge V \leq 0 \Rightarrow \forall k.\ (\!|e|\!)(k) \Rightarrow k = \texttt{false}$

- $(\!|e|\!)(\texttt{true})$ ensures $e \rightsquigarrow \texttt{true}$ and $(\!|e|\!)(\texttt{false})$ ensures $e \rightsquigarrow \texttt{false}$
- I is a loop-invariant formula, L is a loop-invariant quantity, and V is a loop-variant quantity
- L bounds the decrement of V in each iteration

Proof Rule: while loop

$$\frac{\left\{(\!|e|\!)(\texttt{true}) \wedge I \wedge V = \xi \wedge L = \xi'\right\}\ c\ \left\{I \wedge V \leq \xi - \xi' \wedge L = \xi'\right\}}{\left\{I\right\}\ \texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}\ \left\{I \wedge (\!|e|\!)(\texttt{false})\right\}}$$

- $I \wedge (\!|e|\!)(\texttt{true}) \Rightarrow L > 0$
- $I \Rightarrow ((\!|e|\!)(\texttt{true}) \vee (\!|e|\!)(\texttt{false}))$
- $I \wedge V \leq 0 \Rightarrow \forall k.\ (\!|e|\!)(k) \Rightarrow k = \texttt{false}$

- $(\!|e|\!)(\texttt{true})$ ensures $e \rightsquigarrow \texttt{true}$ and $(\!|e|\!)(\texttt{false})$ ensures $e \rightsquigarrow \texttt{false}$
- I is a loop-invariant formula, L is a loop-invariant quantity, and V is a loop-variant quantity
- L bounds the decrement of V in each iteration
- Side-conditions ensure
  1. When I holds and $e \rightsquigarrow \texttt{true}$, L is positive
  2. When I holds, $e \rightsquigarrow \bot$ does not happen
  3. When I holds and V is negative, e evaluates only to $\texttt{false}$

Proof Rule: while loop

$$\frac{\left\{(\!|e|\!)(\texttt{true}) \wedge I \wedge V = \xi \wedge L = \xi'\right\}\ c\ \left\{I \wedge V \leq \xi - \xi' \wedge L = \xi'\right\}}{\left\{I\right\}\ \texttt{while}\ e\ \texttt{do}\ c\ \texttt{end}\ \left\{I \wedge (\!|e|\!)(\texttt{false})\right\}}$$

- $I \wedge (\!|e|\!)(\texttt{true}) \Rightarrow L > 0$
- $I \Rightarrow ((\!|e|\!)(\texttt{true}) \vee (\!|e|\!)(\texttt{false}))$
- $I \wedge V \leq 0 \Rightarrow \forall k.\ (\!|e|\!)(k) \Rightarrow k = \texttt{false}$

- $(\!|e|\!)(\texttt{true})$ ensures $e \rightsquigarrow \texttt{true}$ and $(\!|e|\!)(\texttt{false})$ ensures $e \rightsquigarrow \texttt{false}$
- I is a loop-invariant formula, L is a loop-invariant quantity, and V is a loop-variant quantity
- L bounds the decrement of V in each iteration
- Side-conditions ensure
  1. When I holds and $e \rightsquigarrow \texttt{true}$, L is positive
  2. When I holds, $e \rightsquigarrow \bot$ does not happen
  3. When I holds and V is negative, e evaluates only to $\texttt{false}$

Theorem
The proof rules are sound w.r.t. the denotational semantics.

Verification Condition

annotated commands	c	::=	`skip`	do nothing
		$\mid$	$c_1; c_2$	composition
		$\mid$	$x := e$	assignment
		$\mid$	`if` e `then` c_1 `else` c_2 `end`	conditional
		$\mid$	$\{I, V, L\}$`while` e `do` c `end`	loop

Verification Condition

annotated commands	c	$::=$	$\texttt{skip}$	do nothing
		$\mid$	$c_1; c_2$	composition
		$\mid$	$x := e$	assignment
		$\mid$	$\texttt{if } e \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}$	conditional
		$\mid$	$\{I, V, L\}\texttt{while } e \texttt{ do } c \texttt{ end}$	loop

Theorem
For c and a postcondition ψ, there is a precondition $\texttt{vc}(c, \psi)$

$$\left\{\texttt{vc}(c, \psi)\right\}\ c\ \left\{\psi\right\}$$

In order to show

$$\left\{\phi\right\}\ c\ \left\{\psi\right\}$$

it suffices to show

$$\phi \Rightarrow \texttt{vc}(c, \psi)$$

Verification Condition Example

For any $x \in \mathbb{R}$, compute $n \in \mathbb{Z}$ such that $x < 2^n$

```
int magnitude(REAL x){
    {x ∈ ℝ}
    let y := 1;
    let n := 0;

    while choose(y < x + 1, x < y) = 1
    do
        y = y × 2;
        n = n + 1
    end
    {x < 2^n}
    return n
}
```

Verification Condition Example

For any $x \in \mathbb{R}$, compute $n \in \mathbb{Z}$ such that $x < 2^n$

```
int magnitude(REAL x){
    {x ∈ ℝ}
    let y := 1;
    let n := 0;
    {I ≡ y = 2^n ∧ n ≥ 0      V ≡ x − y + 1      L ≡ 1}
    while choose(y < x + 1, x < y) = 1
    do
        y = y × 2;
        n = n + 1
    end
    {x < 2^n}
    return n
}
```

Extension

Denotational Semantics

In Set:

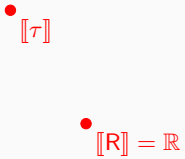
Denotational Semantics

In Set:

- Denotation of data type $[\![\tau]\!]$

Denotational Semantics

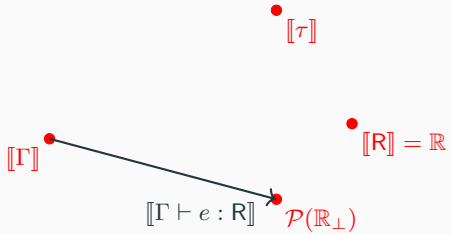
In Set:



- Denotation of data type $[\![\tau]\!]$

Denotational Semantics

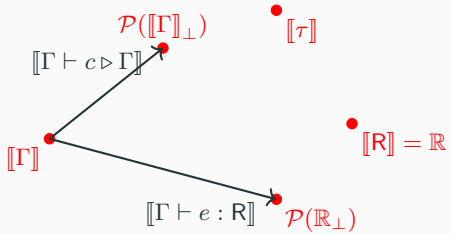
In Set:



- Denotation of data type $[\![\tau]\!]$
- Denotation of a context $[\![\Gamma]\!]$ of all states
- Denotation of a expression
$$[\![\Gamma \vdash e : \tau]\!] : [\![\Gamma]\!] \to \mathcal{P}([\![\tau]\!]_\perp)$$

Denotational Semantics

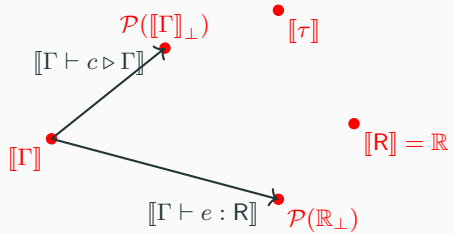
In Set:



- Denotation of data type $\llbracket\tau\rrbracket$
- Denotation of a context $\llbracket\Gamma\rrbracket$ of all states
- Denotation of a expression
  $$\llbracket\Gamma \vdash e : \tau\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\tau\rrbracket_\perp)$$
- Denotation of a command
  $$\llbracket\Gamma \vdash c \triangleright \Gamma\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\Gamma\rrbracket_\perp)$$

Denotational Semantics

In Set:



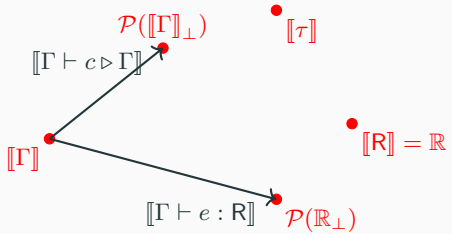
- Denotation of data type $\llbracket\tau\rrbracket$
- Denotation of a context $\llbracket\Gamma\rrbracket$ of all states
- Denotation of a expression
  $$\llbracket\Gamma \vdash e : \tau\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\tau\rrbracket_\perp)$$
- Denotation of a command
  $$\llbracket\Gamma \vdash c \triangleright \Gamma\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\Gamma\rrbracket_\perp)$$

Interpretation

In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:

Denotational Semantics

In Set:



- Denotation of data type $\llbracket\tau\rrbracket$
- Denotation of a context $\llbracket\Gamma\rrbracket$ of all states
- Denotation of a expression
  $$\llbracket\Gamma \vdash e : \tau\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\tau\rrbracket_\perp)$$
- Denotation of a command
  $$\llbracket\Gamma \vdash c \triangleright \Gamma\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\Gamma\rrbracket_\perp)$$

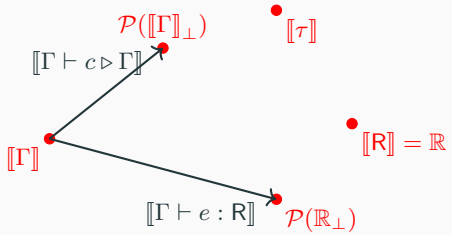
Interpretation

In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:

- Interpretation of data type $\langle\!\langle\tau\rangle\!\rangle$

Denotational Semantics

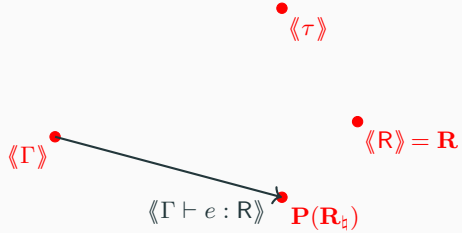
In Set:



- Denotation of data type $\llbracket\tau\rrbracket$
- Denotation of a context $\llbracket\Gamma\rrbracket$ of all states
- Denotation of a expression
$$\llbracket\Gamma \vdash e : \tau\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\tau\rrbracket_\perp)$$
- Denotation of a command
$$\llbracket\Gamma \vdash c \triangleright \Gamma\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\Gamma\rrbracket_\perp)$$

Interpretation

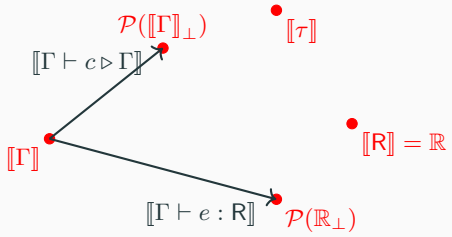
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Interpretation of data type $\langle\!\langle\tau\rangle\!\rangle$
- Interpretation of a context $\langle\!\langle\Gamma\rangle\!\rangle$ of all states
- Interpretation of a expression
$$\langle\!\langle\Gamma \vdash e : \tau\rangle\!\rangle : \langle\!\langle\Gamma\rangle\!\rangle \to \mathbf{P}(\langle\!\langle\tau\rangle\!\rangle_\natural)$$

Denotational Semantics

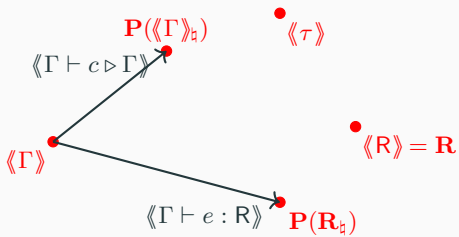
In Set:



- Denotation of data type $\llbracket\tau\rrbracket$
- Denotation of a context $\llbracket\Gamma\rrbracket$ of all states
- Denotation of a expression
$$\llbracket\Gamma \vdash e : \tau\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\tau\rrbracket_\perp)$$
- Denotation of a command
$$\llbracket\Gamma \vdash c \triangleright \Gamma\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\Gamma\rrbracket_\perp)$$

Interpretation

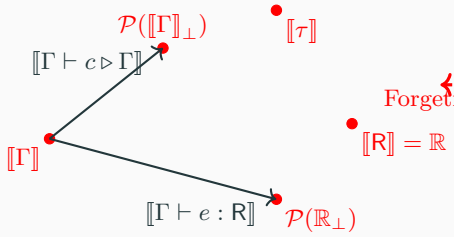
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Interpretation of data type $\langle\!\langle\tau\rangle\!\rangle$
- Interpretation of a context $\langle\!\langle\Gamma\rangle\!\rangle$ of all states
- Interpretation of a expression
$$\langle\!\langle\Gamma \vdash e : \tau\rangle\!\rangle : \langle\!\langle\Gamma\rangle\!\rangle \to \mathbf{P}(\langle\!\langle\tau\rangle\!\rangle_\natural)$$
- Interpretation of a command
$$\langle\!\langle\Gamma \vdash e \triangleright \Gamma\rangle\!\rangle : \langle\!\langle\Gamma\rangle\!\rangle \to \mathbf{P}(\langle\!\langle\Gamma\rangle\!\rangle_\natural)$$

Denotational Semantics

In Set:

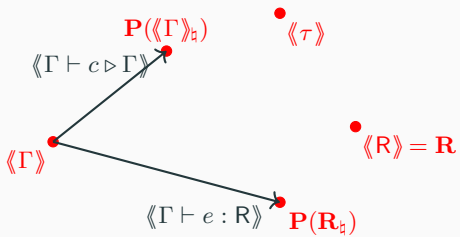


- Denotation of data type $\llbracket\tau\rrbracket$
- Denotation of a context $\llbracket\Gamma\rrbracket$ of all states
- Denotation of a expression
$$\llbracket\Gamma \vdash e : \tau\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\tau\rrbracket_\perp)$$
- Denotation of a command
$$\llbracket\Gamma \vdash c \triangleright \Gamma\rrbracket : \llbracket\Gamma\rrbracket \to \mathcal{P}(\llbracket\Gamma\rrbracket_\perp)$$

Forgetful Functor

Interpretation

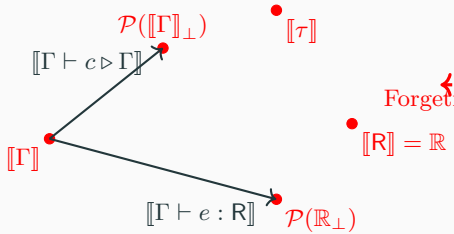
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Interpretation of data type $\langle\!\langle\tau\rangle\!\rangle$
- Interpretation of a context $\langle\!\langle\Gamma\rangle\!\rangle$ of all states
- Interpretation of a expression
$$\langle\!\langle\Gamma \vdash e : \tau\rangle\!\rangle : \langle\!\langle\Gamma\rangle\!\rangle \to \mathbf{P}(\langle\!\langle\tau\rangle\!\rangle_\natural)$$
- Interpretation of a command
$$\langle\!\langle\Gamma \vdash e \triangleright \Gamma\rangle\!\rangle : \langle\!\langle\Gamma\rangle\!\rangle \to \mathbf{P}(\langle\!\langle\Gamma\rangle\!\rangle_\natural)$$

Denotational Semantics

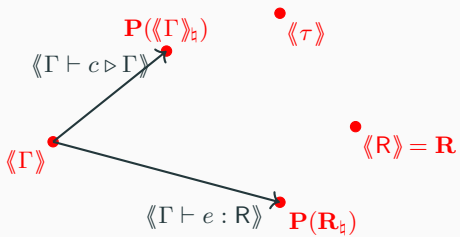
In Set:



Forgetful Functor

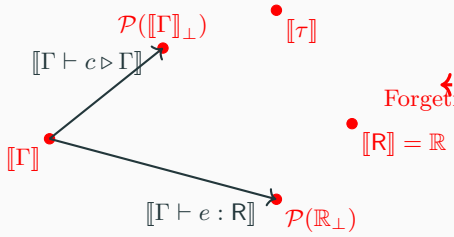
Interpretation

In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



Denotational Semantics

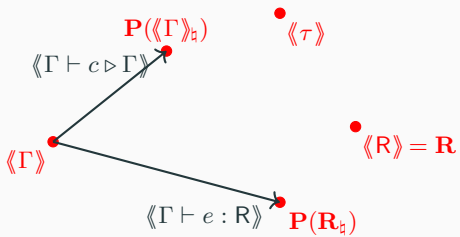
In Set:



Forgetful Functor

Interpretation

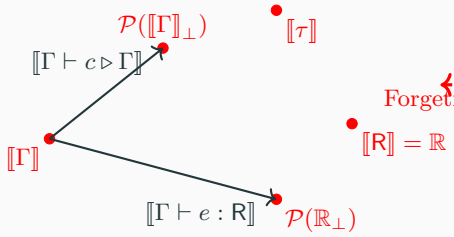
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Add a new data type τ' and a new expression construct $\mathsf{f}(t_1, \cdots, t_n) : \tau'$

Denotational Semantics

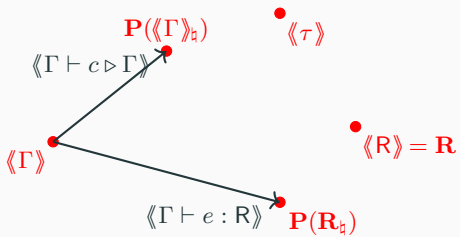
In Set:



Forgetful Functor

Interpretation

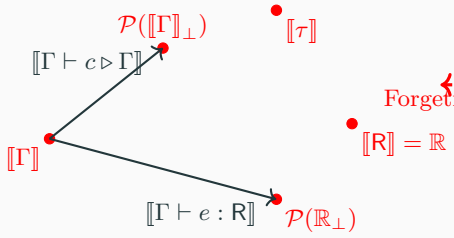
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Add a new data type τ' and a new expression construct $\mathsf{f}(t_1, \cdots, t_n) : \tau'$
- Assign $\langle\!\langle\tau'\rangle\!\rangle$ an assembly and $\langle\!\langle\mathsf{f}\rangle\!\rangle$ a morphism to $\mathbf{P}(\langle\!\langle\tau'\rangle\!\rangle_\natural)$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

Denotational Semantics

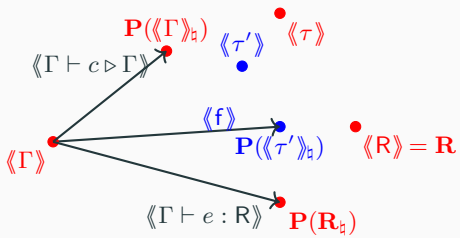
In Set:



Forgetful Functor

Interpretation

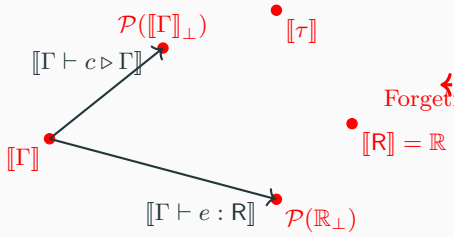
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Add a new data type τ' and a new expression construct $\mathsf{f}(t_1, \cdots, t_n) : \tau'$
- Assign $\langle\!\langle\tau'\rangle\!\rangle$ an assembly and $\langle\!\langle\mathsf{f}\rangle\!\rangle$ a morphism to $\mathbf{P}(\langle\!\langle\tau'\rangle\!\rangle_\natural)$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$

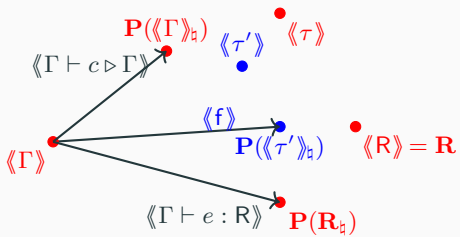
Denotational Semantics

In Set:



Interpretation

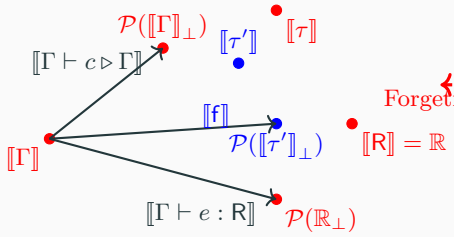
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Add a new data type τ' and a new expression construct $\mathsf{f}(t_1, \cdots, t_n) : \tau'$
- Assign $\langle\!\langle\tau'\rangle\!\rangle$ an assembly and $\langle\!\langle\mathsf{f}\rangle\!\rangle$ a morphism to $\mathbf{P}(\langle\!\langle\tau'\rangle\!\rangle_\natural)$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Define the denotations by the forgetful functor

Denotational Semantics

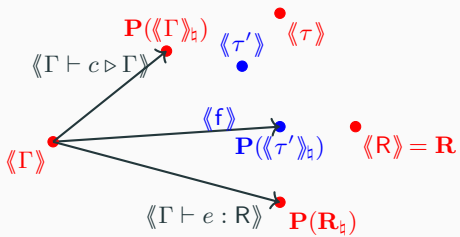
In Set:



Forgetful Functor

Interpretation

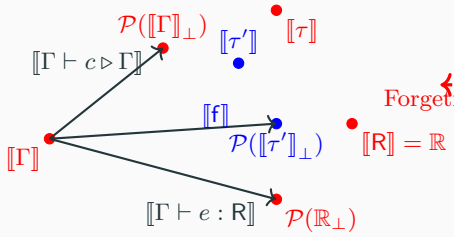
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Add a new data type τ' and a new expression construct $\mathsf{f}(t_1, \cdots, t_n) : \tau'$
- Assign $\langle\!\langle\tau'\rangle\!\rangle$ an assembly and $\langle\!\langle\mathsf{f}\rangle\!\rangle$ a morphism to $\mathbf{P}(\langle\!\langle\tau'\rangle\!\rangle_\natural)$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Define the denotations by the forgetful functor

Denotational Semantics

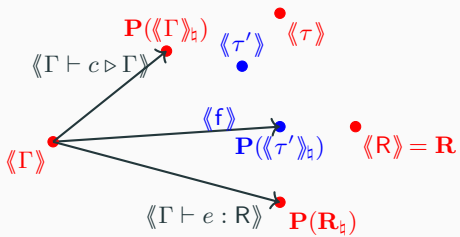
In Set:



Forgetful Functor

Interpretation

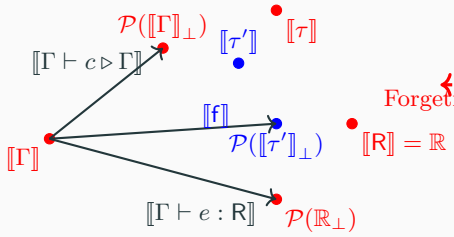
In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Add a new data type τ' and a new expression construct $\mathsf{f}(t_1, \cdots, t_n) : \tau'$
- Assign $\langle\!\langle\tau'\rangle\!\rangle$ an assembly and $\langle\!\langle\mathsf{f}\rangle\!\rangle$ a morphism to $\mathbf{P}(\langle\!\langle\tau'\rangle\!\rangle_\natural)$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Define the denotations by the forgetful functor
- Commands remain the same, for a specification language that is expressive for the expression language, the proof rule remains sound

Denotational Semantics

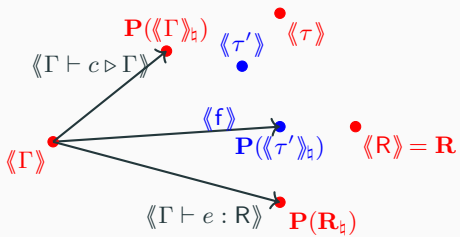
In Set:



Forgetful Functor

Interpretation

In $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$:



- Add a new data type τ' and a new expression construct $\mathsf{f}(t_1, \cdots, t_n) : \tau'$
- Assign $\langle\!\langle\tau'\rangle\!\rangle$ an assembly and $\langle\!\langle\mathsf{f}\rangle\!\rangle$ a morphism to $\mathbf{P}(\langle\!\langle\tau'\rangle\!\rangle_\natural)$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Define the denotations by the forgetful functor
- Commands remain the same, for a specification language that is expressive for the expression language, the proof rule remains sound
- per collection of assemblies and morphisms $\mathcal{E}$ (with consistency property), we define a while language over $\mathcal{E}$
  - with laziness
  - with matrices
  - with continuous real functions

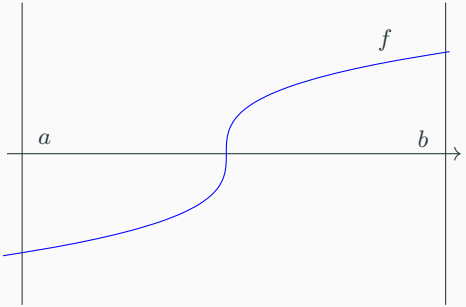
Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

When $a, b : \mathsf{R}, f : \mathsf{C}$

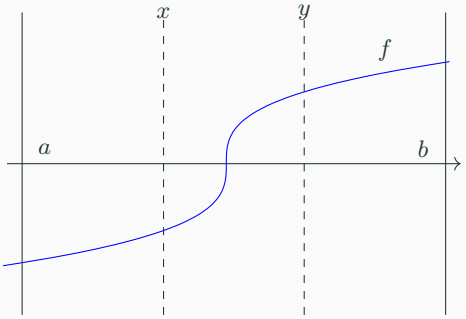


Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

When $x, y, a, b : \mathsf{R}, f : \mathsf{C}$

$x := (2 \times a + b)/3;$
$y := (a + 2 \times b)/3;$

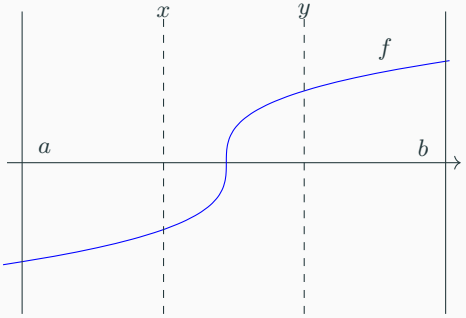


Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

When $x, y, a, b : \mathsf{R}, f : \mathsf{C}$

$x := (2 \times a + b)/3;$
$y := (a + 2 \times b)/3;$
$\mathtt{if}\ \mathsf{choose}(\mathsf{eval}(f, x) \lesssim 0, 0 \lesssim \mathsf{eval}(f, y)) = 1$

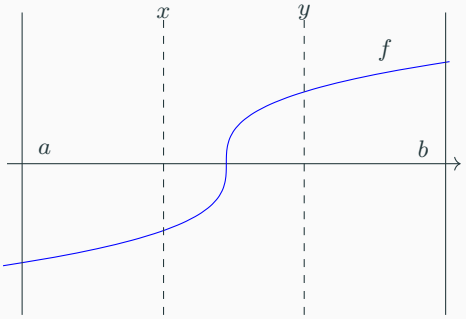


Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

When $x, y, a, b : \mathsf{R}, f : \mathsf{C}$

$x := (2 \times a + b)/3;$
$y := (a + 2 \times b)/3;$
if $\mathsf{choose}(\mathsf{eval}(f, x) \lesssim 0, 0 \lesssim \mathsf{eval}(f, y)) = 1$
  then $a := x$
  else $b := y$
end

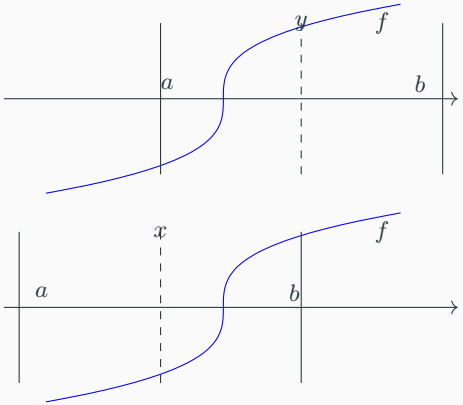


Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

When $x, y, a, b : \mathsf{R}, f : \mathsf{C}$

$x := (2 \times a + b)/3;$
$y := (a + 2 \times b)/3;$
if $\mathsf{choose}(\mathsf{eval}(f, x) \lesssim 0, 0 \lesssim \mathsf{eval}(f, y)) = 1$
  then $a := x$
  else $b := y$
end

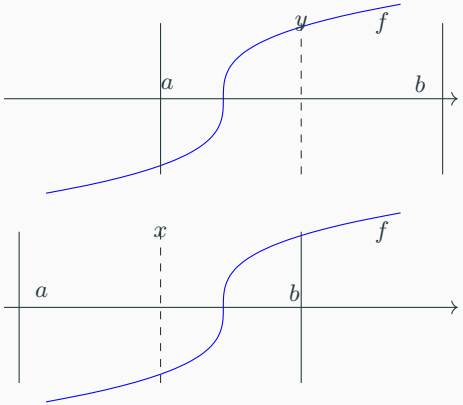


Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

When $p : \mathsf{Z}, x, y, a, b : \mathsf{R}, f : \mathsf{C}$

```
while b − a ≲ 2^p do
    x := (2 × a + b)/3;
    y := (a + 2 × b)/3;
    if choose(eval(f, x) ≲ 0, 0 ≲ eval(f, y)) = 1
      then a := x
      else b := y
    end
end
```

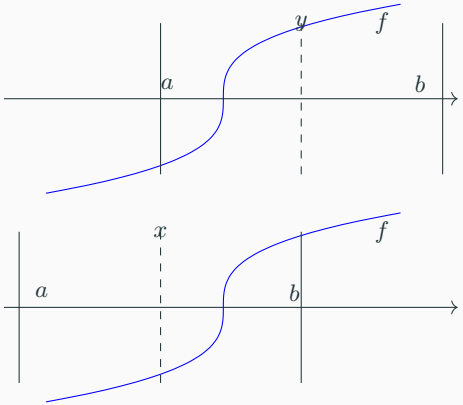


Extension with Continuous Real Functions

- Add a new data type C and a term construct $\mathsf{eval} : \mathsf{C} \times \mathsf{R} \to \mathsf{R}$
- Interpret $\langle\!\langle \mathsf{C} \rangle\!\rangle = \mathbf{R}^{\mathbf{R}}$ and $\langle\!\langle \mathsf{eval} \rangle\!\rangle$ the evaluation map of $\mathbf{R}^{\mathbf{R}}$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$
- Derived $[\![\mathsf{C}]\!] = \mathcal{C}(\mathbb{R}, \mathbb{R})$ and $[\![\mathsf{eval}]\!]$ is the evaluation map in Set

When $p : \mathsf{Z}, x, y, a, b : \mathsf{R}, f : \mathsf{C}$

$\{f$ has unique root in $(a, b) \wedge f(a) < 0 < f(b)\}$
`while` $b - a \lesssim 2^p$ `do`
  $x := (2 \times a + b)/3;$
  $y := (a + 2 \times b)/3;$
  `if` $\mathsf{choose}(\mathsf{eval}(f, x) \lesssim 0, 0 \lesssim \mathsf{eval}(f, y)) = 1$
    `then` $a := x$
    `else` $b := y$
  `end`
`end`
$\{a$ approximates the root of f by $2^p\}$



With explicit limits and multi-limits

Imperative Language with explicit Limit

state

x_1	$=$	$\pi\sqrt{2}$
x_2	$=$	π
x_3	$=$	$\sqrt{2}$
	$\vdots$	
x_d	$=$	0

Imperative Language with explicit Limit

(initial) state			command	(final) state		
x_1	$=$	42		x_1	$=$	$\pi\sqrt{2}$
x_2	$=$	0		x_2	$=$	π
x_3	$=$	$\sqrt{2}$	$x_2 := \texttt{lim}(n \to \infty, e)$	x_3	$=$	$\sqrt{2}$
	$\vdots$				$\vdots$	
x_d	$=$	0		x_d	$=$	0

if e is an expression that evaluates to 2^{-n}-approximation of π.

Imperative Language with explicit Limit

(initial) state			command	(final) state		
x_1	$=$	42		x_1	$=$	$\pi\sqrt{2}$
x_2	$=$	0		x_2	$=$	π
x_3	$=$	$\sqrt{2}$	$x_2 := \texttt{lim}(n \to \infty, e)$	x_3	$=$	$\sqrt{2}$
	$\vdots$				$\vdots$	
x_d	$=$	0		x_d	$=$	0

if e is an expression that evaluates to 2^{-n}-approximation of π.

$$\left| \pi/4 - \sum_{k=0}^{2^n} 1/(4k+1) - 1/(4k+3) \right| \leq 2^{-n}$$

(initial) state		
x_1	$=$	42
x_2	$=$	0
x_3	$=$	$\sqrt{2}$
	$\vdots$	
x_d	$=$	0

command

$$
\begin{aligned}
x_2 := \texttt{lim}\big(& n \to \infty, \\
& \texttt{new var } x := 0 \texttt{ in} \\
& \texttt{new var } k := 0 \texttt{ in} \\
& \quad \texttt{while } k < 2^{-n} \texttt{ do} \\
& \quad x = x + \tfrac{1}{4k+1} - \tfrac{1}{4k+3}; \\
& \quad k = k + 1 \texttt{ end}; 4 \times x\big)
\end{aligned}
$$

(final) state		
x_1	$=$	$\pi\sqrt{2}$
x_2	$=$	π
x_3	$=$	$\sqrt{2}$
	$\vdots$	
x_d	$=$	0

expression should be expressive enough to define a sequence of approximation of π.

$$\left|\pi/4 - \sum_{k=0}^{2^n} 1/(4k+1) - 1/(4k+3)\right| \leq 2^{-n}$$

Clerical

Clerical

- Remove the expression v.s. command distinction

Clerical

- Remove the expression v.s. command distinction
- Add `lim` $n\,.\,e$ as an expression construct

Clerical

- Remove the expression v.s. command distinction
- Add $\texttt{lim}\, n \,.\, e$ as an expression construct
- Distinguish two failures: $\pi \lesssim \pi = \flat$ and $\texttt{lim}\, n \,.\, n = \sharp$

Clerical

- Remove the expression v.s. command distinction
- Add $\texttt{lim}\, n \,.\, e$ as an expression construct
- Distinguish two failures: $\pi \lesssim \pi = \flat$ and $\texttt{lim}\, n \,.\, n = \sharp$
- Powerdomain on $S^{\texttt{e}}_{\perp} = S \cup \{\perp, \texttt{e}\}$ where $\perp$ for $\flat$ and $\texttt{e}$ for $\sharp$

Clerical

- Remove the expression v.s. command distinction
- Add $\texttt{lim}\, n \,.\, e$ as an expression construct
- Distinguish two failures: $\pi \lesssim \pi = \flat$ and $\texttt{lim}\, n \,.\, n = \sharp$
- Powerdomain on $S^{\mathsf{e}}_{\perp} = S \cup \{\perp, \mathsf{e}\}$ where $\perp$ for $\flat$ and e for $\sharp$
- Denotational semantics using the powerdomain

Clerical

- Remove the expression v.s. command distinction
- Add $\texttt{lim}\, n \,.\, e$ as an expression construct
- Distinguish two failures: $\pi \lesssim \pi = \natural$ and $\texttt{lim}\, n \,.\, n = \sharp$
- Powerdomain on $S^{\mathsf{e}}_{\perp} = S \cup \{\perp, \mathsf{e}\}$ where $\perp$ for $\natural$ and e for $\sharp$
- Denotational semantics using the powerdomain
- Sound and relatively complete proof rules for each expression; e.g.,

$$\frac{\left\{\phi'\right\}\ e\ \left\{y : \mathsf{R} \mid \psi'\right\}}{\left\{\phi\right\}\ \texttt{lim}\, x \,.\, e\ \left\{z : \mathsf{R} \mid \psi\right\}} \quad \begin{array}{l} \phi \Rightarrow \\ \exists z{:}\mathsf{R} \,.\, (\forall x{:}\mathsf{Z} \,.\, x \geq 0 \Rightarrow \phi' \wedge (\forall y.\ \psi' \Rightarrow |y - z| \leq 2^{-x})) \wedge \psi \end{array}$$

Clerical

- Remove the expression v.s. command distinction
- Add $\texttt{lim}\, n \,.\, e$ as an expression construct
- Distinguish two failures: $\pi \lesssim \pi = \flat$ and $\texttt{lim}\, n \,.\, n = \sharp$
- Powerdomain on $S^{\mathsf{e}}_{\perp} = S \cup \{\perp, \mathsf{e}\}$ where $\perp$ for $\flat$ and e for $\sharp$
- Denotational semantics using the powerdomain
- Sound and relatively complete proof rules for each expression; e.g.,

$$\frac{\left\{\phi'\right\}\ e\ \left\{y : \mathsf{R} \mid \psi'\right\}}{\left\{\phi\right\}\ \texttt{lim}\, x \,.\, e\ \left\{z : \mathsf{R} \mid \psi\right\}} \quad \begin{array}{l} \phi \Rightarrow \\ \exists z{:}\mathsf{R} \,.\, (\forall x{:}\mathsf{Z} \,.\, x \geq 0 \Rightarrow \phi' \wedge (\forall y.\ \psi' \Rightarrow |y - z| \leq 2^{-x})) \wedge \psi \end{array}$$

- Interpretation using $\sharp\mathsf{M}\flat$ in $\mathsf{Asm}(\mathbb{N}^{\mathbb{N}})$ confirming that the denotational semantics is computable

Conclusion

Conclusion and Future Works

Summary:

- Designed a simple imperative languages for verified computation over continuous data
- together with formal computable semantics dealing with partiality and nondeterminism
- equipped with sound proof rules for verification

Future works:

- Case study with further continuous structures: e.g., matrices
- Higher type objects: e.g., functions and operators
- Use the result to really verify fragments of existing exact computation software including `iRRAM`