[OPLSS: Static Analysis]

Operational / Denotational Semantics

Sukyoung Ryu

[Courtesy by Prof. Kihong Heo]

July 4, 2023

OPLSS: Static Analysis

(1) Concepts in Static Analysis

(2) **Operational / Denotational Semantics**

(3) Abstract Interpretation

(4) Automatic Derivation of Static Analysis

A Simple Imperative Language

E	$::=$		arithmetic expressions
	$\mid$	n	integer constants
	$\mid$	x	variable
	$\mid$	$E \odot E$	binary operation
B	$::=$		boolean expression
	$\mid$	true $\mid$ false	boolean constants
	$\mid$	$E \circledcirc E$	comparison expressions
C	$::=$		commands
	$\mid$	`skip`	command that does nothing
	$\mid$	$C; C$	sequence
	$\mid$	$x{:=}E$	assignment
	$\mid$	`input`(x)	command reading of a value
	$\mid$	`if` B `then` C `else` C	conditional command
	$\mid$	`while` B C	loop command

Different Styles of Semantics

- **Operational Semantics**: "**How** to compute the execution result"
  - so-called transitional style
  - `3 * (2 + 1)` : **3 * (2 + 1) → 3 * 3 → 9**
- **Denotational Semantics**: "**What** the execution result is"
  - so-called compositional style
  - `3 * (2 + 1)` **:** **9**
- **...**

Different approaches for different purposes and languages

Denotational Semantics

- Mathematical meaning of a program (i.e., no state or transition)
- Program semantics is a function from input states to output states
- The semantics of a program is determined by that of each subcomponents (i.e., **compositional**)
- Notation: $[\![P]\!] : \mathbb{M} \to \mathbb{M}$

Semantic Domains

- Sets of semantic objects
- Memory is a mapping $\mathbb{M} = \mathbb{X} \to \mathbb{V}$
  - $\mathbb{X}$: the set of variables
  - $\mathbb{V}$: the set of integers ($\mathbb{Z}$) and booleans ($\mathbb{B}$)
- Example: $[\![\mathtt{x:=7;y:=3}]\!]\{\} = \{\mathtt{x} \mapsto 7, \mathtt{y} \mapsto 3\}$

Semantics of Expressions

$$
\begin{aligned}
[\![E]\!] : \mathbb{M} &\to \mathbb{Z} \\
[\![n]\!](m) &= n \\
[\![x]\!](m) &= m(x) \\
[\![E_1 \odot E_2]\!](m) &= [\![E_1]\!](m) \odot [\![E_2]\!](m)
\end{aligned}
$$

$$
\begin{aligned}
[\![B]\!] : \mathbb{M} &\to \mathbb{B} \\
[\![\mathsf{true}]\!](m) &= \mathsf{true} \\
[\![\mathsf{false}]\!](m) &= \mathsf{false} \\
[\![E_1 \oslash E_2]\!](m) &= [\![E_1]\!](m) \oslash [\![E_2]\!](m)
\end{aligned}
$$

Semantics of Commands (1)

$$
\begin{aligned}
[\![C]\!] : \mathbb{M} &\to \mathbb{M} \\
[\![\texttt{skip}]\!](m) &= m \\
[\![C_0 \texttt{;} C_1]\!](m) &= [\![C_1]\!]([\![C_0]\!](m)) \\
[\![x \texttt{:=} E]\!](m) &= m\{x \mapsto [\![E]\!](m)\} \\
[\![\texttt{input}(x)]\!](m) &= m\{x \mapsto n\}
\end{aligned}
$$

$$
[\![\texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2]\!](m) = \begin{cases} [\![C_1]\!](m) & \text{if } [\![B]\!](m) = \texttt{true} \\ [\![C_2]\!](m) & \text{if } [\![B]\!](m) = \texttt{false} \end{cases}
$$

Compositional? **Yes!**

Semantics of Commands (2)

- Semantics of While loop

$$[\![\texttt{while}\ B\ C]\!](m) = \begin{cases} [\![\texttt{while}\ B\ C]\!]([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases}$$

Compositional? **NO!**

- C.f) inductive vs compositional definition of the Fibonacci function

$$fib(n) = fib(n-1) + fib(n-2) \text{ where } n > 1$$
$$fib(0) = fib(1) = 1$$

vs

$$fib(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Semantics as Fixed Point (1)

- Consider this inductive definition as an equation

$$[\![\texttt{while}\ B\ C]\!](m) = \begin{cases} [\![\texttt{while}\ B\ C]\!]([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases}$$

$$[\![\texttt{while}\ B\ C]\!] = \mathcal{F}_{B,C}([\![\texttt{while}\ B\ C\}]\!])$$

where $\mathcal{F}_{B,C}(X) = \lambda m. \begin{cases} X([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases}$

$$F(X) = \cdots$$
$$\iff F = \lambda X. \cdots$$

Semantics as Fixed Point (2)

- What is the semantics of loops? A solution of this equation!

$$[\![\texttt{while}\ B\ C]\!] = \mathcal{F}_{B,C}([\![\texttt{while}\ B\ C]\!]) \quad \Longleftrightarrow \quad X = \mathcal{F}_{B,C}(X) \text{ where } X = [\![\texttt{while}\ B\ C]\!]$$

- Solution: a fixed point of $\mathcal{F}_{B,C}$
- Especially, we define the semantics as the least fixed point

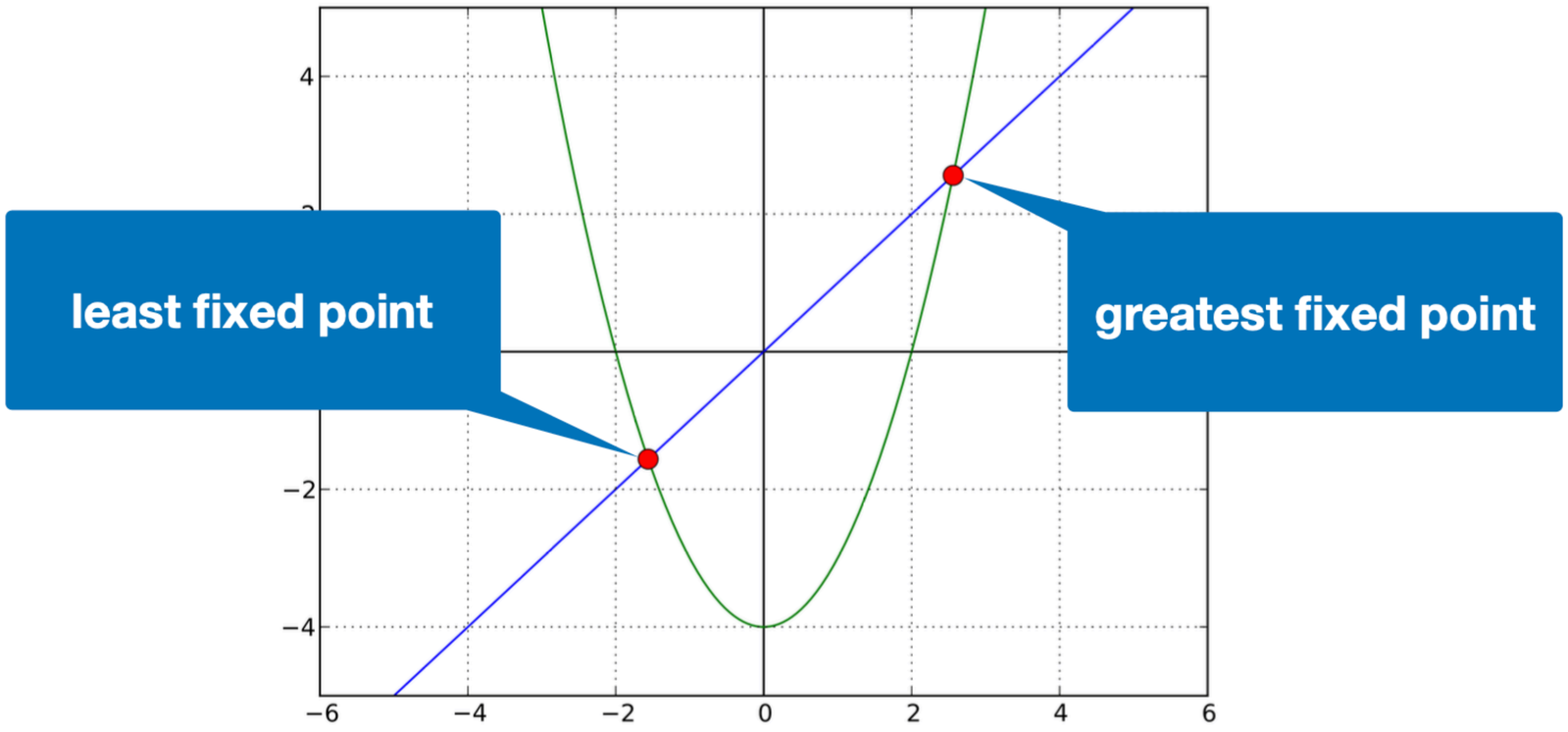
$$[\![\texttt{while}\ B\ C]\!] = \mathbf{lfp}\mathcal{F}_{B,C}$$

$$\text{where } \mathcal{F}_{B,C}(X)(m) = \begin{cases} X([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases}$$

Compositional? **Yes!**

Exercise

- $f(x) = x^2 - 4$



*https://en.wikipedia.org/wiki/Least_fixed_point

Exercise

- $\mathbb{N} = \{0\} \cup \{n + 1 \mid n \in \mathbb{N}\}$

$\mathbb{N}$ is the least fixed point of F where $F(X) = \{0\} \cup \{n + 1 \mid n \in X\}$

$$\therefore \mathbb{N} = \mathbf{lfp}\big(\lambda X.\{0\} \cup \{n + 1 \mid n \in X\}\big)$$

Exercise

- $\mathbb{L} = \{\mathsf{nil}\} \cup \{0 :: l \mid l \in \mathbb{L}\}$

$\mathbb{L}$ is the least fixed point of F where $F(X) = \{\mathsf{nil}\} \cup \{0 :: l \mid l \in X\}$

$$\therefore \mathbb{L} = \mathbf{lfp}\big(\lambda X.\{\mathsf{nil}\} \cup \{0 :: l \mid l \in X\}\big)$$

Exercise

- $\texttt{reach}(N) = N \cup \texttt{reach}(\texttt{next}(N))$

$$\texttt{reach} = \lambda N.N \cup \texttt{reach}(\texttt{next}(N))$$

$\texttt{reach}$ is the least fixed point of F where $F(X) = \lambda N.N \cup X(\texttt{next}(N))$

$$\therefore \texttt{reach} = \textbf{lfp}\left(\lambda X.(\lambda N.N \cup X(\texttt{next}(N)))\right)$$

Exercise

- $\texttt{fact}(N) = \texttt{if}\ N = 0\ ?\ 1\ :\ N * \texttt{fact}(N-1)$

$$\texttt{fact} = \lambda N.\texttt{if}\ N = 0\ ?\ 1\ :\ N * \texttt{fact}(N-1)$$

$\texttt{fact}$ is the least fixed point of F where $F(X) = \lambda N.\texttt{if}\ N = 0\ ?\ 1\ :\ N * X(N-1)$

$$\therefore \texttt{fact} = \mathbf{lfp}\big(\lambda X.(\lambda N.\texttt{if}\ N = 0\ ?\ 1\ :\ N * X(N-1))\big)$$

Questions

- Why does the semantic equation have a solution?
- Does the equation have a unique solution?
- How to compute the solution?

Domain Theory

- Semantics of a program is an element of a domain called **CPO** (complete partial order set)
  - Example: $[\![C]\!] \in \mathbb{M} \to \mathbb{M}$
- Semantics of a program is the **least fixed point** of a **continuous function**
  - Example: $\mathcal{F}_{B,C} : (\mathbb{M} \to \mathbb{M}) \to (\mathbb{M} \to \mathbb{M})$
- Established by Dana Scott
  - *Outline of a Mathematical Theory of Computation,* 1970
  - *Mathematical Concepts in Programming Language Semantics*, 1972

Partial Order

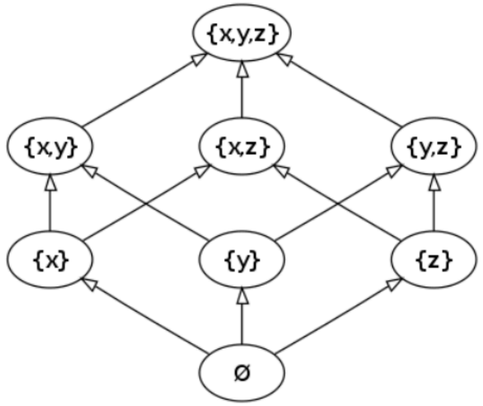
Definition (Partial Order). A binary relation $\sqsubseteq$ is a **partial order** on a set D if it has:

1. reflexivity: $a \sqsubseteq a$ for all $a \in D$
2. Antisymmetry: $a \sqsubseteq b$ and $b \sqsubseteq a$ implies $a = b$
3. Transitivity: $a \sqsubseteq b$ and $b \sqsubseteq c$ implies $a \sqsubseteq c$

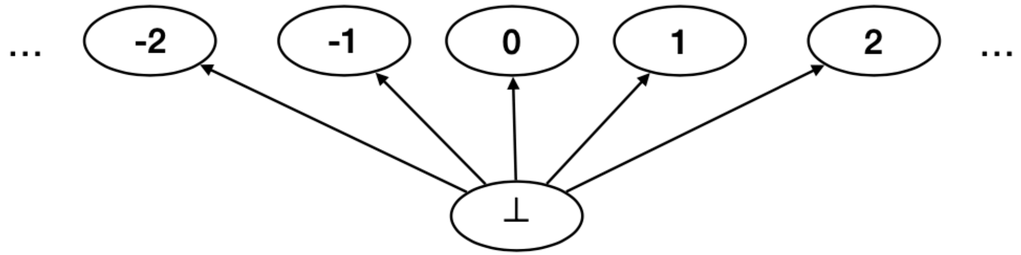
A set D with a partial order $\sqsubseteq$ is called a **partially ordered set** $(D, \sqsubseteq)$, or simply **poset**.

Partial Order

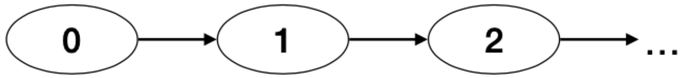
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



- Example 2: $(\mathbb{Z}_\perp, \sqsubseteq)$



- Example 3: $(\mathbb{N}, \leq)$



- Example 4: $(\mathbb{N} + \{+\infty\}, \leq)$

Least Upper Bound

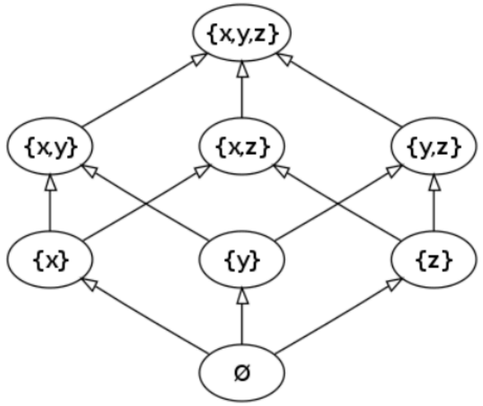
Definition (Least Upper Bound). For a partial ordered set $(D, \sqsubseteq)$ and subset $X \subseteq D$, $d \in D$ is an **upper bound** of X iff

$$\forall x \in X.\ x \sqsubseteq d.$$

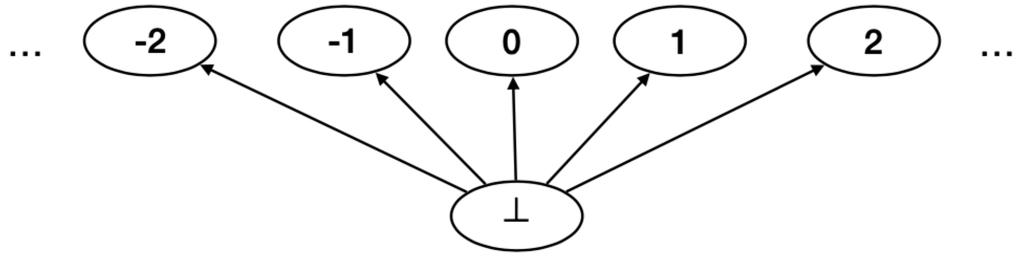
An upper bound d is the **least upper bound** of X iff for all upper bounds y of X, $d \sqsubseteq y$. The least upper bound of X is denoted by $\bigsqcup X$.

Least Upper Bound

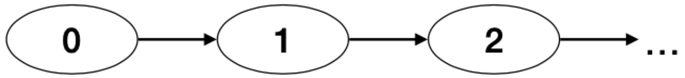
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



- Example 2: $(\mathbb{Z}_\perp, \sqsubseteq)$



- Example 3: $(\mathbb{N}, \leq)$



- Example 4: $(\mathbb{N} + \{+\infty\}, \leq)$

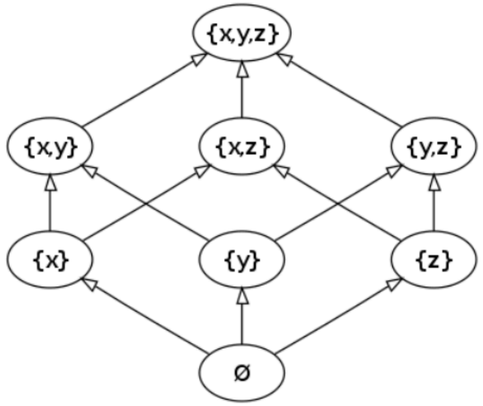
Chain

Definition (Chain). Let $(D, \sqsubseteq)$ be a partial ordered set. A subset $X \subseteq D$ is called **chain** if X is totally ordered:

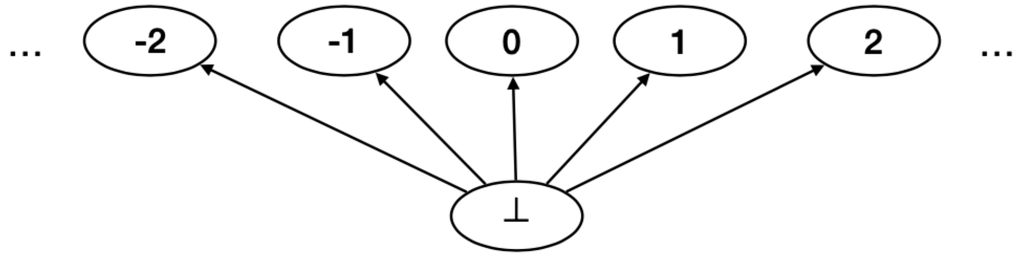
$$\forall x_1, x_2 \in X.\ x_1 \sqsubseteq x_2 \text{ or } x_2 \sqsubseteq x_1.$$

Chain

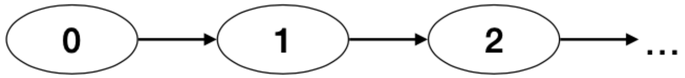
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



- Example 2: $(\mathbb{Z}_{\perp}, \sqsubseteq)$



- Example 3: $(\mathbb{N}, \leq)$



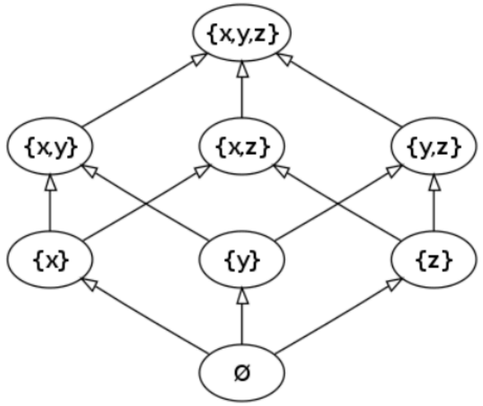
- Example 4: $(\mathbb{N} + \{+\infty\}, \leq)$

CPO

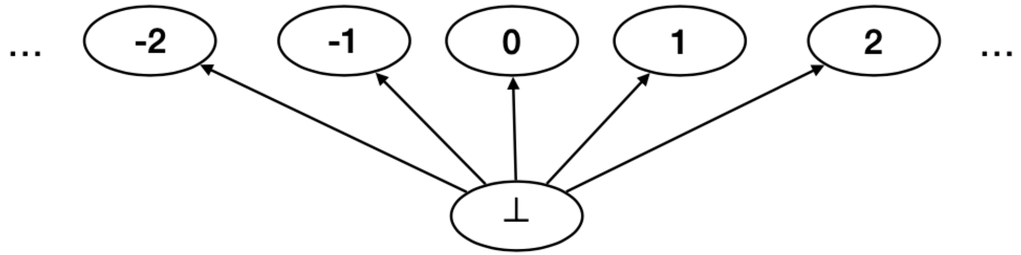
Definition (CPO). A poset $(D, \sqsubseteq)$ is a **CPO** (complete partial order) if every chain X of D has $\bigsqcup X \in D$.

CPO

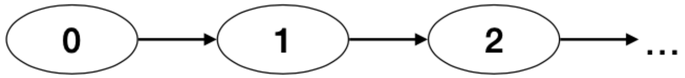
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



- Example 2: $(\mathbb{Z}_{\perp}, \sqsubseteq)$



- Example 3: $(\mathbb{N}, \leq)$



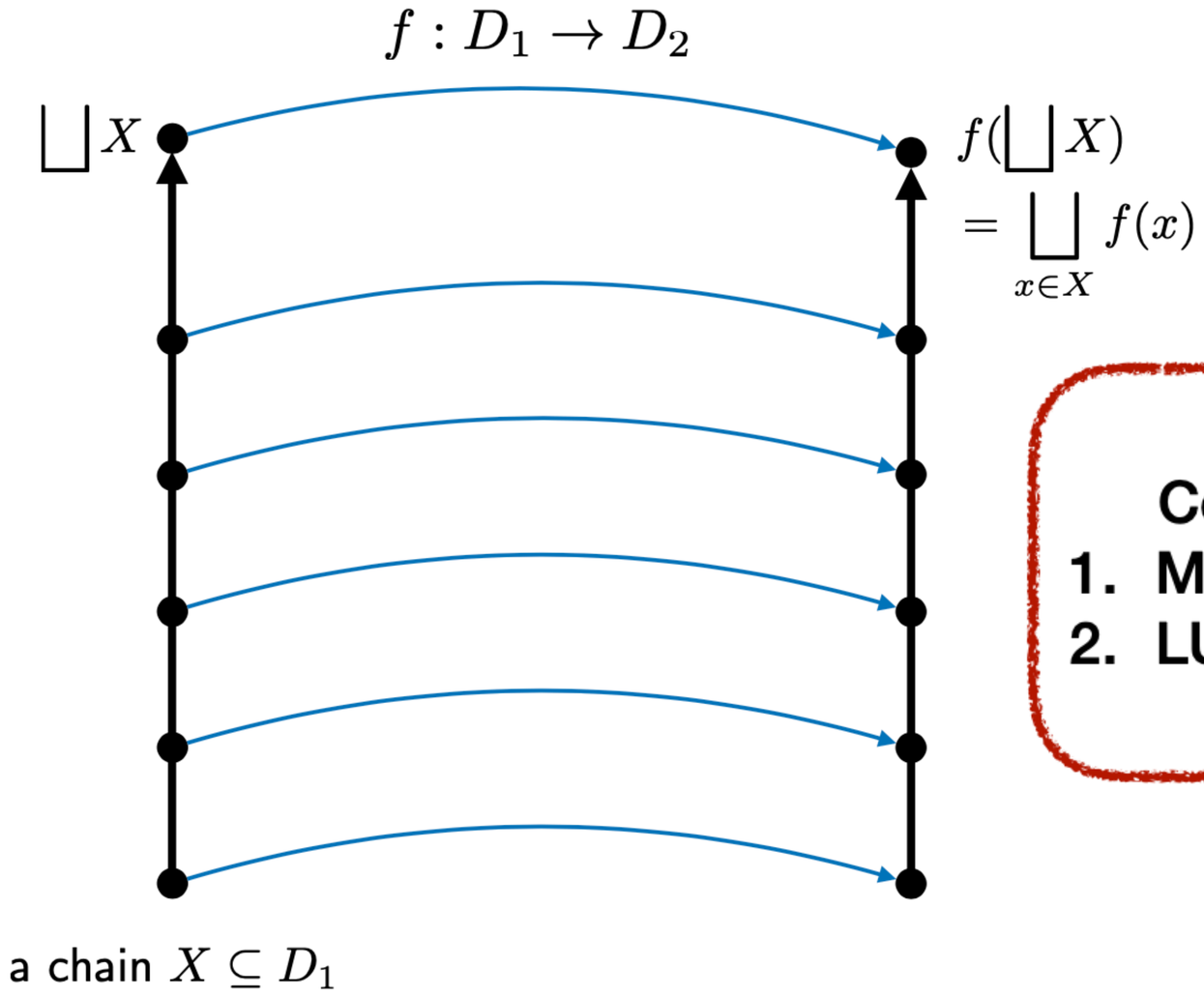
- Example 4: $(\mathbb{N} + \{+\infty\}, \leq)$

Continuous Function

Definition (Continuous Function). Given two partially ordered sets D_1 and D_2, a function $f : D_1 \to D_2$ is **continuous** if it preserves least upper bounds of chains:

$$\forall chain\ X \subseteq D_1.\ \bigsqcup_{x \in X} f(x) = f(\bigsqcup X).$$

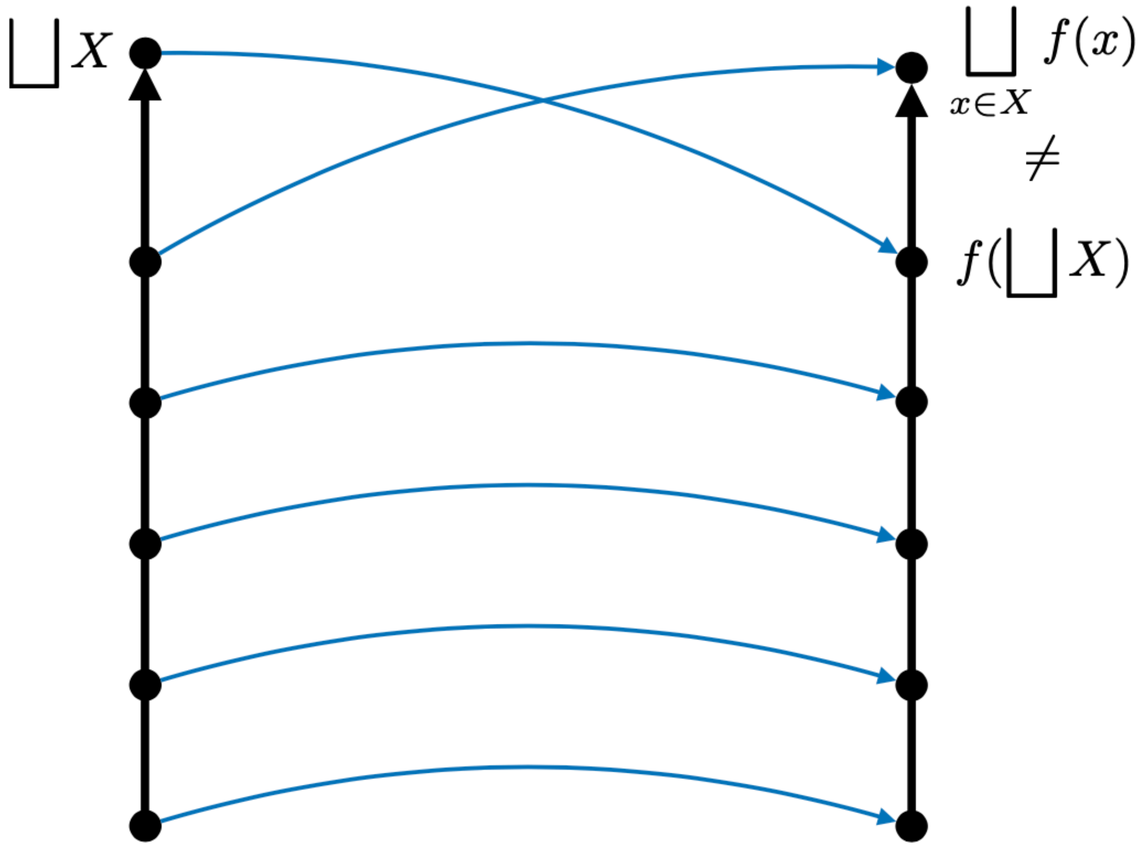
Continuous Function



Continuous =
1. **Monotone +**
2. **LUB preservation**

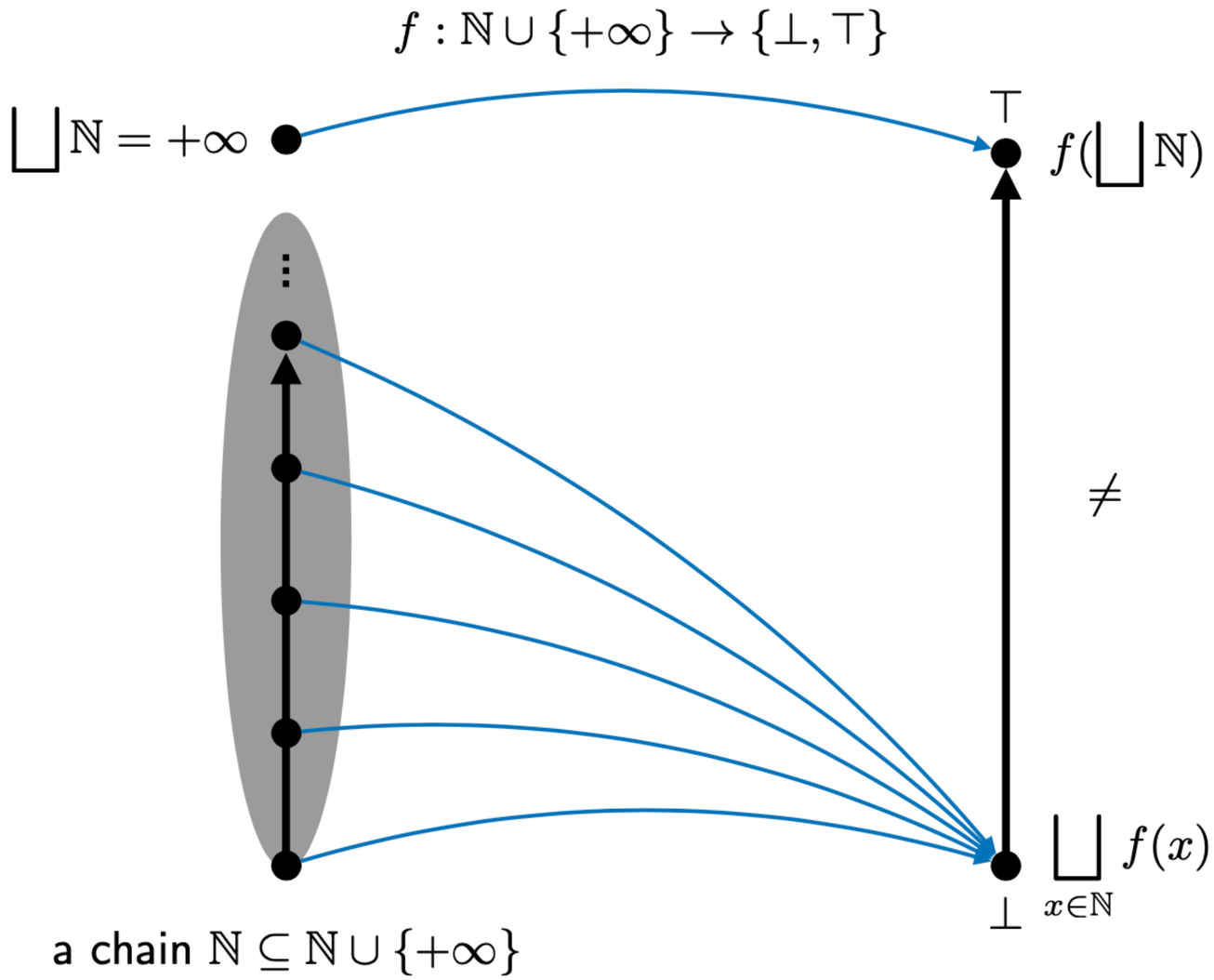
Non-Continuous Function (1)

$$f : D_1 \to D_2$$



a chain $X \subseteq D_1$

Non-Continuous Function (2)



Continuous Function

Lemma. If a function f is continuous, f is monotone.

Proof. Suppose that f is not monotone, i.e., for some $a \sqsubseteq b$, $f(a) \sqsupset f(b)$. Then,

$$\begin{array}{lll} f(a) & \sqsubseteq & f(a) \sqcup f(b) \quad (\text{by definition of } \sqcup) \\ & = & f(a \sqcup b) \quad (\text{by continuity of } f) \\ & = & f(b) \quad (\text{because } a \sqsubseteq b) \end{array}$$

This is a contradiction. Therefore, $f(a) \sqsubseteq f(b)$.

Fixed Point

Definition (Fixed Point). Let $(D, \sqsubseteq)$ be a partial ordered set. A **fixed point** of a function $f : D \to D$ is an element x such that $f(x) = x$. We write $\mathbf{lfp} f$ for the **least fixed point** of f such that

$$f(\mathbf{lfp} f) = \mathbf{lfp} f \quad \text{and} \quad \forall d \in D.\ f(d) = d \implies \mathbf{lfp} f \sqsubseteq d$$

Theorem (Kleene Fixed Point). Let $f : D \to D$ be a continuous function on a CPO D. Then f has the **least fixed point** $\mathbf{lfp} f$ and

$$\mathbf{lfp} f = \bigsqcup_{i \geq 0} f^i(\bot)$$

Proof

$$\mathbf{lfp} f = \bigsqcup_{i \geq 0} f^i(\bot)$$

- Plans: It is enough to show the following two things:

  (1) There exists the chain $\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \cdots$ and its least upper bound $\bigsqcup_{i \geq 0} f^i(\bot)$ in D

  (2) The least upper bound $\bigsqcup_{i \geq 0} f^i(\bot)$ is the least fixed point of f

Proof of (1)

(1) There exists the chain $\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \cdots$ and its least upper bound $\bigsqcup_{i \geq 0} f^i(\bot)$ in D

Proof. We show by induction that $\forall n \in \mathbb{N}.\ f^n(\bot) \sqsubseteq f^{n+1}(\bot)$:

- $\bot \sqsubseteq f(\bot)$ ($\bot$ is the least element of the CPO)
- $f^n(\bot) \sqsubseteq f^{n+1}(\bot) \implies f^{n+1}(\bot) \sqsubseteq f^{n+2}(\bot)$ (by monotonicity of f)

By definition of CPO, least upper bounds of all chains are also in the CPO. Therefore, the least upper bound $\bigsqcup_{i \geq 0} f^i(\bot)$ of the above chain is in D.

Proof of (2)

(2) The least upper bound $\bigsqcup_{i \geq 0} f^i(\bot)$ is the least fixed point of f

The proof consists of two parts:

(2-1) $\bigsqcup_{i \geq 0} f^i(\bot)$ is a fixed point of f

(2-2) $\bigsqcup_{i \geq 0} f^i(\bot)$ is smaller than all the other fixed points

Proof of (2-1)

(2-1) $\bigsqcup_{i \geq 0} f^i(\bot)$ is a fixed point of f

Proof.

$$
\begin{aligned}
f(\bigsqcup_{n \geq 0} f^n(\bot)) &= \bigsqcup_{n \geq 0} f(f^n(\bot)) && \text{(by continuity of } f\text{)} \\
&= \bigsqcup_{n \geq 0} f^{n+1}(\bot) \\
&= \bigsqcup_{n \geq 0} f^n(\bot)
\end{aligned}
$$

Proof of (2-2)

(2-2) $\bigsqcup_{i \geq 0} f^i(\bot)$ is smaller than all the other fixed points

Proof. Suppose d is a fixed point, i.e., $d = f(d)$. We show that any element $f^i(\bot)$ is smaller than d by induction:

$$\forall n \in \mathbb{N}.\ f^n(\bot) \sqsubseteq d.$$

- $\bot \sqsubseteq d$ ($\bot$ is the least element of the CPO)
- $f^n(\bot) \sqsubseteq d \implies f^{n+1}(\bot) \sqsubseteq f(d) = d$ (by monotonicity of f)

Because all the elements $f^i(\bot)$ are smaller than d, their least upper bound $\bigsqcup_{i \geq 0} f^i(\bot)$ is also smaller than d. Therefore

$$\bigsqcup_{i \geq 0} f^i(\bot) = \mathbf{lfp}\, f$$

Constructions on CPOs

- If S is a set, and D_1 and D_2 are CPOs, then the followings are CPOs
  - Lifted set : $D = S_\perp$
  - Cartesian product : $D = D_1 \times D_2$
  - Separated sum : $D = D_1 + D_2$
  - Function : $D = D_1 \to D_2$

Lifted Set

- $D = S_\perp$

> For any set S, let $D = S + \{\perp\}$ where $\perp$ is an element not in S. Then $(D, \sqsubseteq)$ is a CPO where
>
> $$d \sqsubseteq d' \iff (d = d') \vee (d = \perp)$$

- Why CPO?

Cartesian Product

- $D = D_1 \times D_2$

  Given two CPOs $(D_1, \sqsubseteq_1)$ and $(D_2, \sqsubseteq_2)$, $(D, \sqsubseteq)$ is a CPO where

  $$D = D_1 \times D_2 = \{(d_1, d_2) \mid d_1 \in D_1 \ \wedge \ d_2 \in D_2\}$$

  $$(d_1, d_2) \sqsubseteq (d_1', d_2') \iff (d_1 \sqsubseteq_1 d_1') \wedge (d_2 \sqsubseteq_2 d_2')$$

- Why CPO?

  Exercise!

Separated Sum

- $D = D_1 + D_2$

  Given two CPOs $(D_1, \sqsubseteq_1)$ and $(D_2, \sqsubseteq_2)$, $(D, \sqsubseteq)$ is a CPO where

$$D = D_1 + D_2 = \{(d_1, 1) \mid d_1 \in D_1\} \cup \{(d_2, 2) \mid d_2 \in D_2\} \cup \{\bot\}$$

$$(d_1, 1) \sqsubseteq (d_1', 1) \iff d_1 \sqsubseteq_1 d_1'$$

$$(d_2, 2) \sqsubseteq (d_2', 2) \iff d_2 \sqsubseteq_2 d_2'$$

- Why CPO?

  Exercise!

Function

- $D = D_1 \to D_2$

  Given two CPOs $(D_1, \sqsubseteq_1)$ and $(D_2, \sqsubseteq_2)$, $(D, \sqsubseteq)$ is a CPO where

  $$D = D_1 \to D_2 = \{f \mid f : D_1 \to D_2 \text{ is a continuous function}\}$$

  $$f \sqsubseteq f' \iff \forall d_1 \in D_1.\ f(d_1) \sqsubseteq_2 f'(d_1)$$

- Why CPO?

  Exercise!

Semantic Domains (Revisited)

- Domains of memories, variables, and values: CPOs

$$\mathbb{M} = \mathbb{X} \to \mathbb{V}$$

$$\mathbb{X} = Var_{\perp}$$

$$\mathbb{V} = \mathbb{Z}_{\perp} + \mathbb{B}_{\perp}$$

- Domain of commands: function CPO

$$[\![C]\!] \in \mathbb{M} \to \mathbb{M}$$

- Domain of expressions: function CPO

$$[\![E]\!] \in \mathbb{M} \to \mathbb{Z}_{\perp} \qquad\qquad [\![B]\!] \in \mathbb{M} \to \mathbb{B}_{\perp}$$

Semantics of `while` (Revisited)

- Semantics of `while` is defined as the least fixed point as follows: (i.e., the lfp exists)
  - All the sets are CPOs
  - All the functions are continuous

$$[\![\texttt{while}\ B\ C]\!] = \mathbf{lfp}\mathcal{F}_{B,C}$$

$$\text{where } \mathcal{F}_{B,C}(X)(m) = \begin{cases} X([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases}$$

Example

- `while (x < 10) x := x + 1`

$$[\![\texttt{while (x < 10) x := x + 1}]\!] = \lambda m. \begin{cases} [\![\texttt{while (x < 10) x := x + 1}]\!]([\![x := x + 1]\!](m)) & \text{if } [\![\texttt{x < 10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x < 10}]\!](m) = \texttt{false} \end{cases}$$

$$[\![\texttt{while (x < 10) x := x + 1}]\!] = \mathbf{lfp}\mathcal{F} \text{ where } \mathcal{F}(X) = \lambda m. \begin{cases} X([\![x := x + 1]\!](m)) & \text{if } [\![\texttt{x < 10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x < 10}]\!](m) = \texttt{false} \end{cases}$$

$$\mathbf{lfp}\mathcal{F} = \bot \sqcup \mathcal{F}(\bot) \sqcup \mathcal{F}^2(\bot) \sqcup \cdots$$

Example

$$\mathcal{F}(X) = \lambda m. \begin{cases} X([\![x := x + 1]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

Example

$$\mathcal{F}(X) = \lambda m. \begin{cases} X([\![x := x + 1]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

$\bot$

Example

$$\mathcal{F}(X) = \lambda m. \begin{cases} X([\![x := x + 1]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

$\bot$

0 iter

$$\mathcal{F}(\bot) = \lambda m. \begin{cases} \bot([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

x >= 10

Example

$$\mathcal{F}(X) = \lambda m. \begin{cases} X([\![x := x + 1]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

$\bot$

0 iter

$$\mathcal{F}(\bot) = \lambda m. \begin{cases} \bot([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

x >= 10

0, 1 iter

$$\mathcal{F}^2(\bot) = \lambda m. \begin{cases} \mathcal{F}(\bot)([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

x >= 9

$$= \lambda m. \begin{cases} \begin{cases} \bot([\![\texttt{x} := \texttt{x} + \texttt{1}]\!]^2(m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!]([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) = \texttt{true} \\ [\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m) & \text{if } [\![\texttt{x} < \texttt{10}]\!]([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) = \texttt{false} \end{cases} & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

Example

$$\mathcal{F}(X) = \lambda m. \begin{cases} X([\![x := x + 1]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

$\bot$

0 iter $\mathcal{F}(\bot) = \lambda m. \begin{cases} \bot([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$

x >= 10

0, 1 iter $\mathcal{F}^2(\bot) = \lambda m. \begin{cases} \mathcal{F}(\bot)([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$

x >= 9

$$= \lambda m. \begin{cases} \begin{cases} \bot([\![\texttt{x} := \texttt{x} + \texttt{1}]\!]^2(m)) & \text{if } [\![\texttt{x} < \texttt{10}]\!]([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) = \texttt{true} \\ [\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m) & \text{if } [\![\texttt{x} < \texttt{10}]\!]([\![\texttt{x} := \texttt{x} + \texttt{1}]\!](m)) = \texttt{false} \end{cases} & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{true} \\ m & \text{if } [\![\texttt{x} < \texttt{10}]\!](m) = \texttt{false} \end{cases}$$

0,1,2 iters $\mathcal{F}^3(\bot) = \cdots$

x >= 8

Semantic of Commands (Revisited)

$$
\begin{aligned}
[\![C]\!] &: \mathbb{M} \to \mathbb{M} \\
[\![\texttt{skip}]\!] &= \lambda m.m \\
[\![C_0\,;C_1]\!] &= \lambda m.[\![C_1]\!]([\![C_0]\!](m)) \\
[\![x := E]\!] &= \lambda m.m\{x \mapsto [\![E]\!](m)\} \\
[\![\texttt{input}(x)]\!] &= \lambda m.m\{x \mapsto n\} \\
[\![\texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2]\!] &= \lambda m. \begin{cases} [\![C_1]\!](m) & \text{if } [\![B]\!](m) = \texttt{true} \\ [\![C_2]\!](m) & \text{if } [\![B]\!](m) = \texttt{false} \end{cases} \\
[\![\texttt{while } B\ C]\!] &= \mathbf{lfp}\lambda X. \left(\lambda m. \begin{cases} X([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases} \right)
\end{aligned}
$$

Generality of Fixed Point

- Consider the following function H that has two components F and G

$$H(x, y) = (F(x), G(y))$$

$$\text{where } F(x) = 1$$
$$\text{and } G(y) = G(y-1) \times y$$

- The least fixed point of H :

$$\begin{aligned} \mathbf{lfp}H &= (1, \mathbf{lfp}G) \\ &= (\mathbf{lfp}F, \mathbf{lfp}G) \end{aligned}$$

Semantics of Programs

- Generally, semantics of a program is defined as the least fixed point
- A program P consists of subcomponents C_1, …, C_n

$$P = (C_1, \ldots, C_i, \ldots C_n)$$

- The semantics of P consists of that of C_1, …, C_n

$$\begin{aligned}
[\![P]\!] &= ([\![C_1]\!], \ldots, [\![C_i]\!], \ldots [\![C_n]\!]) \\
&= (A_1, \ldots, \cdots [\![C_i]\!] \cdots, \ldots A_n) \\
&= \mathcal{F}([\![C_1]\!], \ldots, [\![C_i]\!], \ldots [\![C_n]\!]) \\
&= \mathbf{lfp}\mathcal{F}
\end{aligned}$$

Summary

- Denotational semantics describes mathematical meaning of programs
  - semantics of a program is an element of a **CPO**: $[\![P]\!] \in D$
  - semantics is the **least fixed point** of a **continuous** function: $\mathcal{F} \in D \to D$
  - **compositionally** defined by the semantics of subcomponents
  - the least fixed point is the **least upper bound** of the following chain:

$$
\begin{aligned}
[\![P]\!] &= \mathcal{F}([\![P]\!]) \\
&= \mathbf{lfp}\mathcal{F} \\
&= \bigsqcup_{i \geq 0} \mathcal{F}^i(\bot)
\end{aligned}
$$

[OPLSS: Static Analysis]

Static Analysis for JavaScript Specification

Sukyoung Ryu

with PLRG@KAIST and friends

July 4, 2023

A Simple Imperative Language

E	$::=$		arithmetic expressions
	$\mid$	n	integer constants
	$\mid$	x	variable
	$\mid$	$E \odot E$	binary operation
B	$::=$		boolean expression
	$\mid$	$\mathsf{true} \mid \mathsf{false}$	boolean constants
	$\mid$	$E \circledcirc E$	comparison expressions
C	$::=$		commands
	$\mid$	$\texttt{skip}$	command that does nothing
	$\mid$	$C; C$	sequence
	$\mid$	$x{:=}E$	assignment
	$\mid$	$\texttt{input}(x)$	command reading of a value
	$\mid$	$\texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ C$	conditional command
	$\mid$	$\texttt{while}\ B\ C$	loop command

Intermediate Representation for ECMAScript

$$
\begin{array}{lrcl}
\textbf{Functions} & \mathbb{F} \ni f & ::= & \texttt{def x(x}^*\texttt{, [x}^*\texttt{])}\, l \\
\textbf{Instructions} & \mathbb{I} \ni i & ::= & \texttt{let x} = e \mid \texttt{x} = \texttt{(}e\ e^*\texttt{)} \mid \texttt{assert}\ e \\
 & & & \mid \texttt{if}\ e\ l\ l \mid \texttt{return}\ e \mid r = e \\
\textbf{References} & & r & ::= \texttt{x} \mid r\texttt{[}e\texttt{]} \\
\textbf{Expressions} & & e & ::= t\ \texttt{\{}\left[\texttt{x} : e\right]^*\texttt{\}} \mid \texttt{[}e^*\texttt{]} \mid e : \tau \mid r\texttt{?} \\
 & & & \mid e \oplus e \mid \ominus e \mid r \mid c \mid p \\
\textbf{Primitives} & \mathbb{P} \ni p & ::= & \texttt{undefined} \mid \texttt{null} \mid b \mid n \mid j \mid s \mid \texttt{@}s \\
\textbf{Types} & \mathbb{T} \ni \tau & ::= & t \mid \texttt{[]} \mid \texttt{[}\tau\texttt{]} \mid \texttt{js} \mid \texttt{prim} \\
 & & & \mid \texttt{undefined} \mid \texttt{null} \mid \texttt{bool} \mid \texttt{numeric} \\
 & & & \mid \texttt{num} \mid \texttt{bigint} \mid \texttt{str} \mid \texttt{symbol}
\end{array}
$$

Semantic Domains (Revisited)

- Domains of memories, variables, and values: CPOs

$$\mathbb{M} = \mathbb{X} \to \mathbb{V}$$

$$\mathbb{X} = Var_{\perp}$$

$$\mathbb{V} = \mathbb{Z}_{\perp} + \mathbb{B}_{\perp}$$

- Domain of commands: function CPO

$$[\![C]\!] \in \mathbb{M} \to \mathbb{M}$$

- Domain of expressions: function CPO

$$[\![E]\!] \in \mathbb{M} \to \mathbb{Z}_{\perp} \qquad\qquad [\![B]\!] \in \mathbb{M} \to \mathbb{B}_{\perp}$$

Semantic Domains

States	$d \in \mathbb{S} = \mathbb{L} \times \mathbb{C}^* \times \mathbb{H} \times \mathbb{E}$
Contexts	$\kappa \in \mathbb{C} = \mathbb{L} \times \mathbb{E} \times \mathbb{X}$
Heaps	$h \in \mathbb{H} = \mathbb{A} \to \mathbb{O}$
Addresses	$a \in \mathbb{A}$
Objects	$o \in \mathbb{O} = (\mathbb{T}_t \times (\mathbb{V}_s \to \mathbb{V})) \uplus \mathbb{V}^*$
Nominal Types	$t \in \mathbb{T}_t$
Environments	$\sigma \in \mathbb{E} = \mathbb{X} \times \mathbb{V}$
Values	$v \in \mathbb{V} = \mathbb{F} \uplus \mathbb{A} \uplus \mathbb{V}_c \uplus \mathbb{P}$
Constants	$c \in \mathbb{V}_c$
Strings	$s \in \mathbb{V}_s$

Semantic of Commands (Revisited)

$$
\begin{aligned}
[\![C]\!] &: \mathbb{M} \to \mathbb{M} \\
[\![\texttt{skip}]\!] &= \lambda m.m \\
[\![C_0\,;C_1]\!] &= \lambda m.[\![C_1]\!]([\![C_0]\!](m)) \\
[\![x := E]\!] &= \lambda m.m\{x \mapsto [\![E]\!](m)\} \\
[\![\texttt{input}(x)]\!] &= \lambda m.m\{x \mapsto n\} \\
[\![\texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2]\!] &= \lambda m. \begin{cases} [\![C_1]\!](m) & \text{if } [\![B]\!](m) = \texttt{true} \\ [\![C_2]\!](m) & \text{if } [\![B]\!](m) = \texttt{false} \end{cases} \\
[\![\texttt{while } B\ C]\!] &= \mathbf{lfp}\lambda X. \left(\lambda m. \begin{cases} X([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases} \right)
\end{aligned}
$$

Semantics of Instructions

B. Instructions: $[\![i]\!]_i : \mathbb{S} \to \mathbb{S}$

- Variable Declarations:

$$[\![\texttt{let x} = e]\!]_i(d) = (\texttt{next}(\ell), \overline{\kappa}, h, \sigma[\texttt{x} \mapsto v])$$

where

$$[\![e]\!]_e(d) = ((\ell, \overline{\kappa}, h, \sigma), v)$$

- Function Calls:

$$[\![\texttt{x} = \texttt{(}e_0\ e_1 \cdots e_n\texttt{)}]\!]_i(d) = (\ell_{\texttt{f}}, \kappa :: \overline{\kappa}, h, \sigma')$$

where

$$\begin{array}{l}[\![e_0]\!]_e(d) = (d_0, \texttt{def f(p}_1, \cdots, \texttt{p}_m\texttt{)}\ \ell_{\texttt{f}} \wedge \\ [\![e_1]\!]_e(d_0) = (d_1, v_1) \wedge \cdots \wedge [\![e_n]\!]_e(d_{n-1}) = (d_n, v_n) \wedge \\ d_n = (\ell, \overline{\kappa}, h, \sigma) \wedge k = \min(n, m) \wedge \\ \sigma' = [\texttt{p}_1 \mapsto v_1, \cdots, \texttt{p}_k \mapsto v_k] \wedge \kappa = (\texttt{next}(\ell), \sigma, \texttt{x})\end{array}$$

- Assertions:

$$[\![\texttt{assert}\ e]\!]_i(d) = d' \quad \text{if } [\![e]\!]_e(d) = (d', \texttt{\#t})$$

- Branches:

$$[\![\texttt{if}\ e\ \ell_{\texttt{t}}\ \ell_{\texttt{f}}]\!]_i(d) = \begin{cases} (\ell_{\texttt{t}}, \overline{\kappa}, h, \sigma) & \text{if } v = \texttt{\#t} \\ (\ell_{\texttt{f}}, \overline{\kappa}, h, \sigma) & \text{if } v = \texttt{\#f} \end{cases}$$

where

$$[\![e]\!]_e(d) = ((\ell_{\texttt{t}}, \overline{\kappa}, h, \sigma), v)$$

- Returns:

$$[\![\texttt{return}\ e]\!]_i(d) = (\ell, \overline{\kappa}, h, \sigma[\texttt{x} \mapsto v])$$

where

$$[\![e]\!]_e(d) = ((\_, (\ell, \sigma, \texttt{x}) :: \overline{\kappa}, h, \_), v)$$

- Variable Updates:

$$[\![\texttt{x} = e]\!]_i(d) = (\texttt{next}(\ell), \overline{\kappa}, h, \sigma[\texttt{x} \mapsto v])$$

where

$$[\![e]\!]_e(d) = ((\ell, \overline{\kappa}, h, \sigma), v)$$

- Field Updates:

$$[\![r\texttt{[}e_0\texttt{]} = e_1]\!]_i(d) = (\texttt{next}(\ell), \overline{\kappa}, h[a \mapsto o'], \sigma)$$

where

$$\begin{array}{l}[\![r]\!]_e(d) = (d', a) \wedge [\![e_0]\!]_e(d') = (d_0, v_0) \wedge \\ [\![e_1]\!]_e(d_0) = ((\ell, \overline{\kappa}, h, \sigma), v_1) \wedge o = h(a) \wedge \\ o' = \begin{cases} o_r & \text{if } o = (t, \texttt{fs}) \wedge v_0 = s \\ o_l & \text{if } o = [v'_1, \cdots, v'_m] \wedge v_0 = n \end{cases} \wedge \\ o_r = (t, \texttt{fs}[s \mapsto v_1]) \wedge o_l = [\cdots, v'_{n-1}, v_1, v'_{n+1}, \cdots]\end{array}$$

Semantics of Expressions

D. Expressions: $[\![e]\!]_e : \mathbb{S} \to \mathbb{S} \times \mathbb{V}$

- Records:

$$[\![t\ \{\mathtt{x}_1 : e_1, \cdots, \mathtt{x}_n : e_n\}]\!]_e(d) = (d', a)$$

where

$$[\![e_1]\!]_e(d) = (d_1, v_1) \wedge \cdots \wedge [\![e_n]\!]_e(d_{n-1}) = (d_n, v_n) \wedge$$
$$d_n = (\ell, \overline{\kappa}, h, \sigma) \wedge \mathtt{fs} = [\mathtt{x}_1 \mapsto v_1, \cdots, \mathtt{x}_n \mapsto v_n]$$
$$a \notin \text{Domain}(h) \wedge d' = (\ell, \overline{\kappa}, h[a \mapsto (t, \mathtt{fs})], \sigma)$$

- Lists:

$$[\![\mathtt{[}e_1, \cdots, e_n\mathtt{]}]\!]_e(d) = (d', a)$$

where

$$[\![e_1]\!]_e(d) = (d_1, v_1) \wedge \cdots \wedge [\![e_n]\!]_e(d_{n-1}) = (d_n, v_n) \wedge$$
$$d_n = (\ell, \overline{\kappa}, h, \sigma) \wedge a \notin \text{Domain}(h) \wedge$$
$$d' = (\ell, \overline{\kappa}, h[a \mapsto [v_1, \cdots, v_n]], \sigma)$$

- Type Checks:

$$[\![e : \tau]\!]_e(d) = (d', b)$$

where

$$[\![e]\!]_e(d) = (d', v) \wedge b = \begin{cases} \mathtt{\#t} & \text{if } v \text{ is a value of } \tau \\ \mathtt{\#f} & \text{otherwise} \end{cases}$$

- Variable Existence Checks:

$$[\![\mathtt{x?}]\!]_e(d) = (d, b)$$

where

$$d = (\_, \_, \_, \sigma) \wedge b = \begin{cases} \mathtt{\#t} & \text{if } \mathtt{x} \in \text{Domain}(\sigma) \\ \mathtt{\#f} & \text{otherwise} \end{cases}$$

- Field Existence Checks:

$$[\![r\mathtt{[}e\mathtt{]?}]\!]_e(d) = (d'', b)$$

where

$$[\![r]\!]_e(d) = (d', a) \wedge [\![e]\!]_e(d') = (d'', v) \wedge$$
$$d'' = (\ell, \overline{\kappa}, h, \sigma) \wedge o = h(a) \wedge$$
$$b = \begin{cases} \mathtt{\#t} & \text{if } o = (t, \mathtt{fs}) \wedge v = s \wedge s \in \text{Domain}(\mathtt{fs}) \\ \mathtt{\#t} & \text{if } o = [v'_1, \cdots, v'_m] \wedge v = n \wedge 1 \leq n \leq m \\ \mathtt{\#f} & \text{otherwise} \end{cases}$$

- Binary Operations:

$$[\![e \oplus e]\!]_e(d) = (d'', v_0 \oplus v_1)$$

where

$$[\![e_0]\!]_e(d) = (d', v_0) \wedge [\![e_1]\!]_e(d') = (d'', v_1)$$

- Unary Operations:

$$[\![\ominus\ e]\!]_e(d) = (d', \ominus\ v)$$

where

$$[\![e]\!]_e(d) = (d', v)$$

- References:

$$[\![r]\!]_e(d) = [\![r]\!]_r(d)$$

- Constants:

$$[\![c]\!]_e(d) = (d, c)$$

- Primitives:

$$[\![p]\!]_e(d) = (d, p)$$