Introduction to Program Analysis

3. Denotational Semantics

Kihong Heo

KAIST

Different Styles of Semantics (Revisited)

- **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

Denotation 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: $[\![P]\!] : \mathbb{M} \rightarrow \mathbb{M}$

A Simple Imperative Language

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

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: $[\![\texttt{x:=7;y:=3}]\!]\{\} = \{\mathrm{x} \mapsto 7, \mathrm{y} \mapsto 3\}$

Semantics of Expressions

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

$$
\begin{aligned}
[\![B]\!] : \mathbb{M} &\rightarrow \mathbb{B} \\
[\![\mathsf{true}]\!](m) &= \mathsf{true} \\
[\![\mathsf{false}]\!](m) &= \mathsf{false} \\
[\![B_1 \otimes B_2]\!](m) &= [\![B_1]\!](m) \otimes [\![B_2]\!](m)
\end{aligned}
$$

Semantics of Commands (1)

$$\llbracket C \rrbracket : \mathbb{M} \rightarrow \mathbb{M}$$

$$\begin{aligned}
\llbracket \texttt{skip} \rrbracket(m) &= m \\
\llbracket C_0\,\texttt{;}\,C_1 \rrbracket(m) &= \llbracket C_1 \rrbracket(\llbracket C_0 \rrbracket(m)) \\
\llbracket x \,\texttt{:=}\, E \rrbracket(m) &= m\{x \mapsto \llbracket E \rrbracket(m)\} \\
\llbracket \texttt{input}(x) \rrbracket(m) &= m\{x \mapsto n\} \\
\llbracket \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2 \rrbracket(m) &= \begin{cases} \llbracket C_1 \rrbracket(m) & \text{if } \llbracket B \rrbracket(m) = \texttt{true} \\ \llbracket C_2 \rrbracket(m) & \text{if } \llbracket B \rrbracket(m) = \texttt{false} \end{cases}
\end{aligned}$$

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

$$\mathit{fibo}(n) = \mathit{fibo}(n-1) + \mathit{fibo}(n-2) \qquad \textbf{vs} \qquad \mathit{fibo}(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Semantics as Fixed Point (1)

- Consider this inductive definition as an equation

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

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

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

$$\boxed{\begin{aligned} & F(X) = \cdots \\ \iff & F = \lambda X. \cdots \end{aligned}}$$

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

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

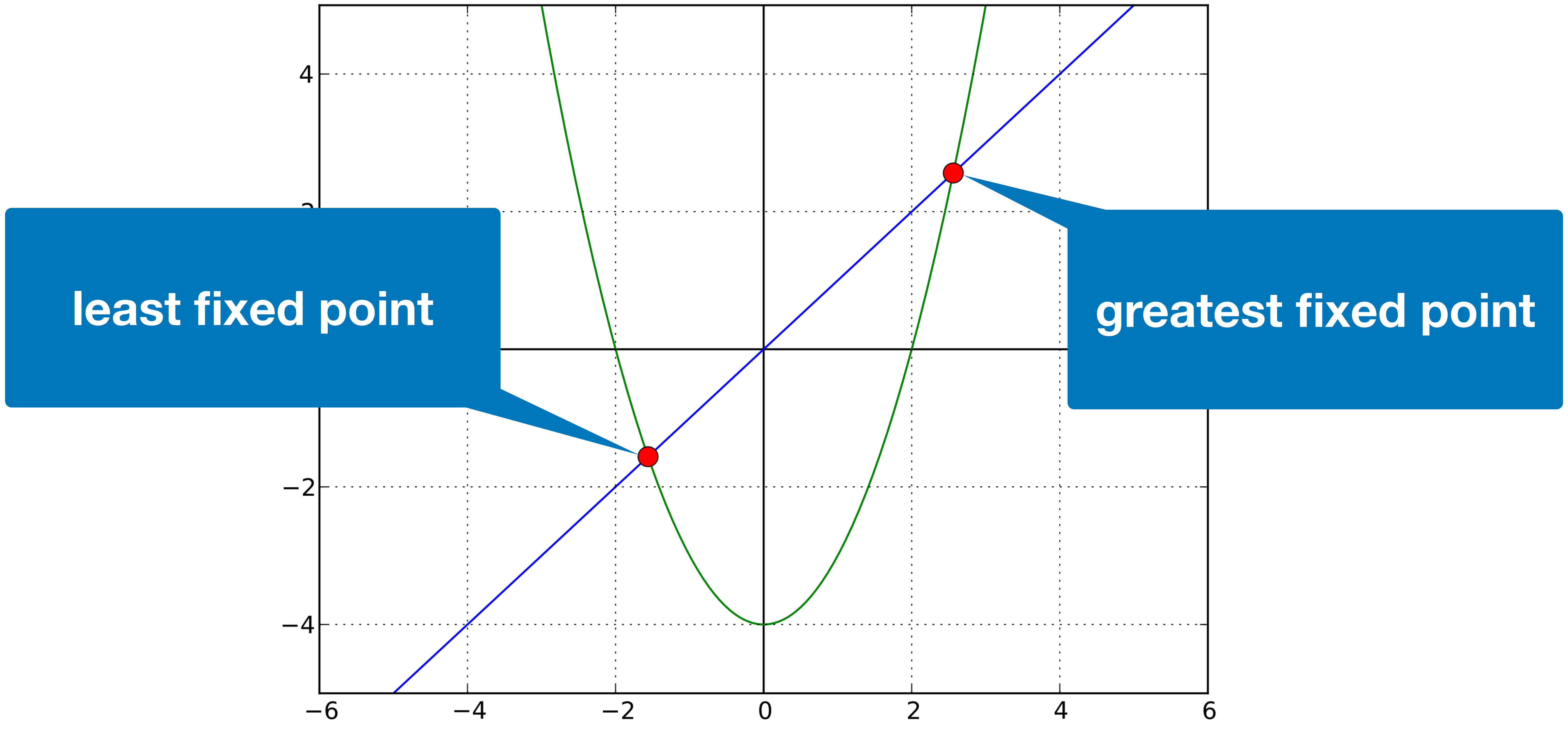
$$\text{where } \mathcal{F}_{B,C}(X)(m) = \begin{cases} X([\![C]\!](m)) & \text{if } [\![B]\!](m) = \texttt{true} \\ m & \text{if } [\![B]\!](m) = \texttt{false} \end{cases}$$

Compositional? **Yes!**

Exercise: Fixed Point

$$\boxed{f(x) = x^2 - 4}$$

•



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

Exercise: Fixed Point

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

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

-

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

Exercise: Fixed Point

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

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

-

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

Exercise: Fixed Point

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

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

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

$$\therefore \texttt{reach} = \mathit{fix}\big(\lambda X.(\lambda N.N \cup X(\texttt{next}(N))\big)$$

Exercise: Fixed Point

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

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

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

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

Semantics as Fixed Point (Revisited)

- 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!

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

Partial Order

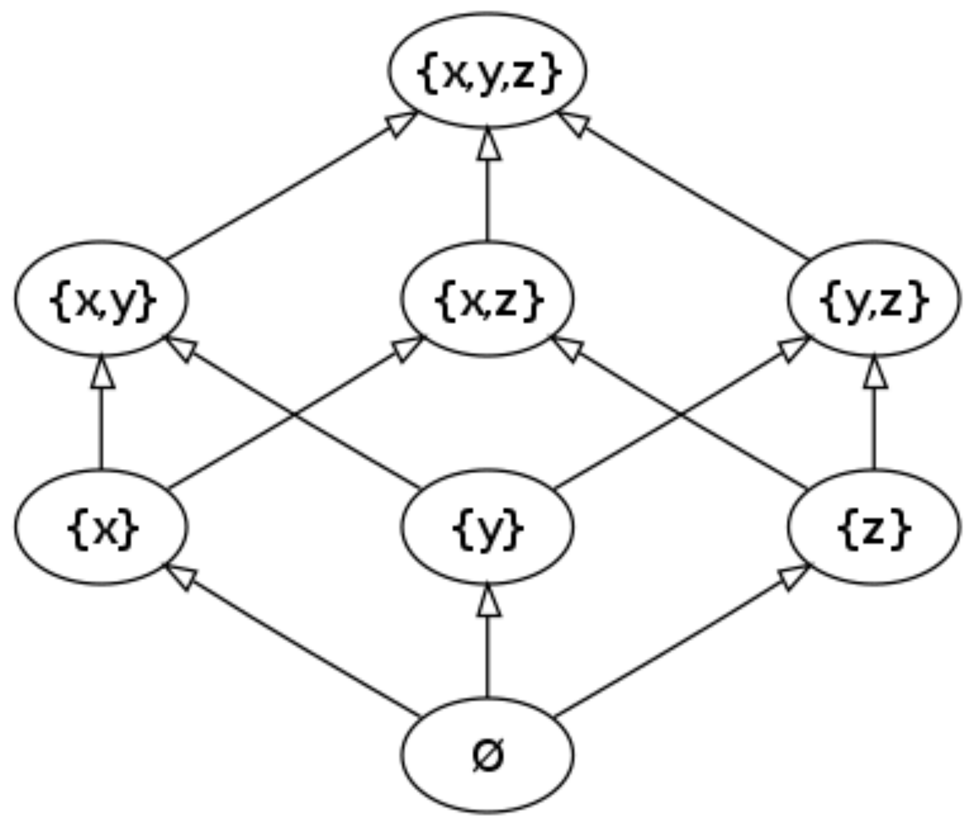
Definition (Partial Order). A binary relation $\sqsubseteq$ is a **partial order** on a set D if it has:

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 $a \sqsubseteq c$

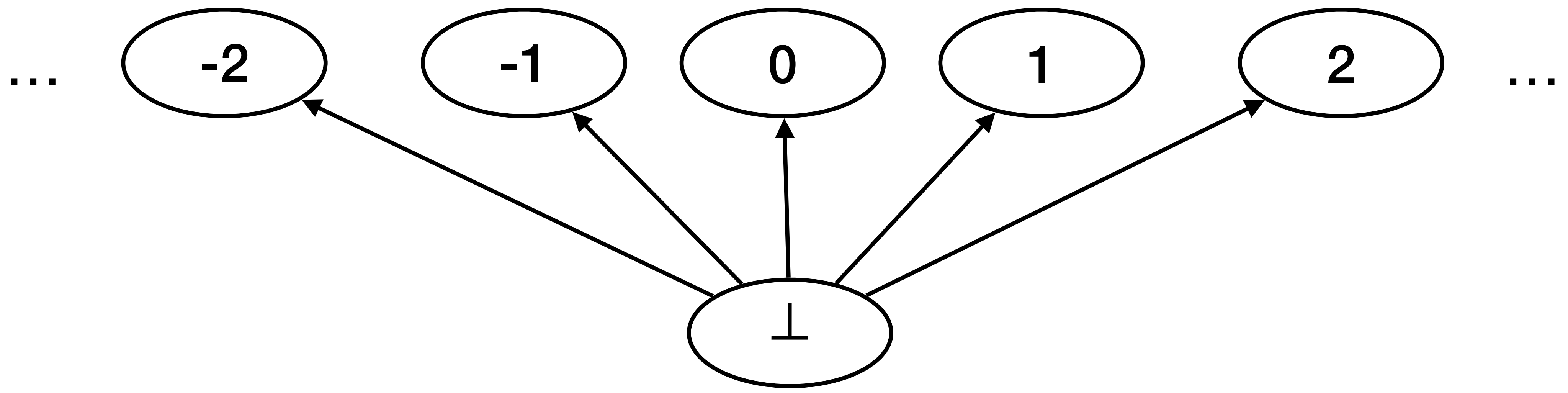
A set D with a partial order $\sqsubseteq$ is called a **partially ordered set** $(D, \sqsubseteq)$, or simply **poset**.

Partial Order

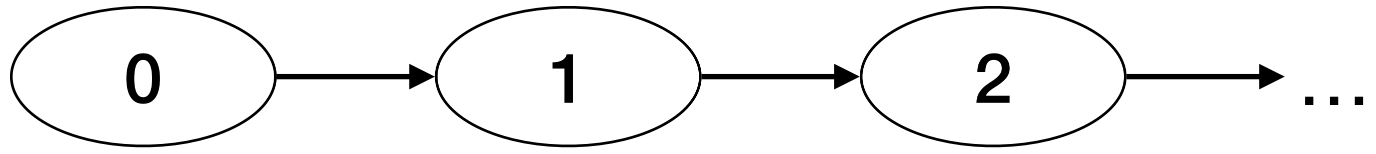
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



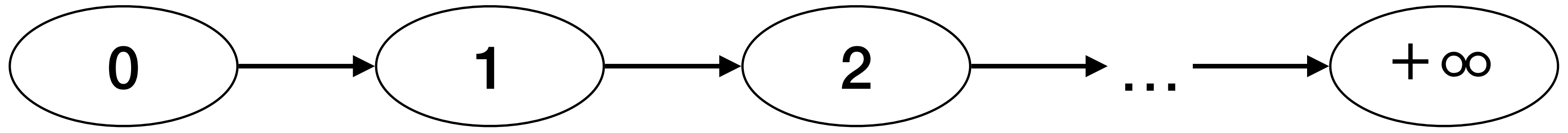
- Example 2: $(\mathbb{Z}_\perp, \sqsubseteq)$



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



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



Least Upper Bound (Join)

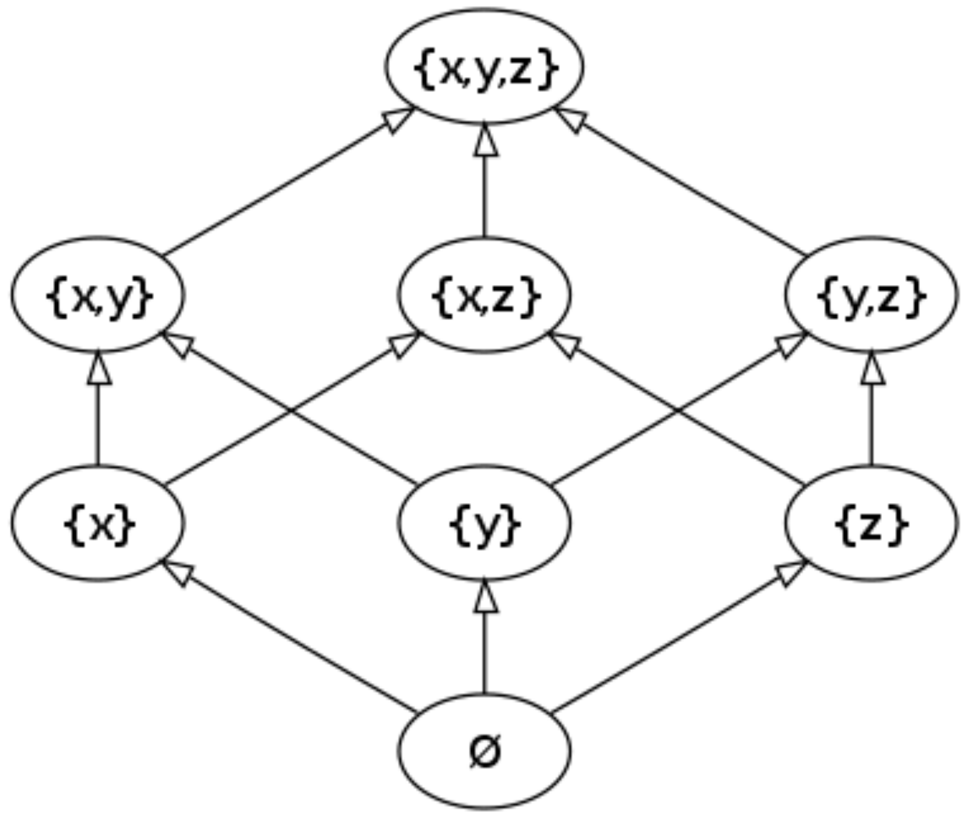
Definition (Least Upper Bound). For a partial ordered set $(D, \sqsubseteq)$ and subset $X \subseteq D$, $d \in X$ is an **upper bound** of X iff

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

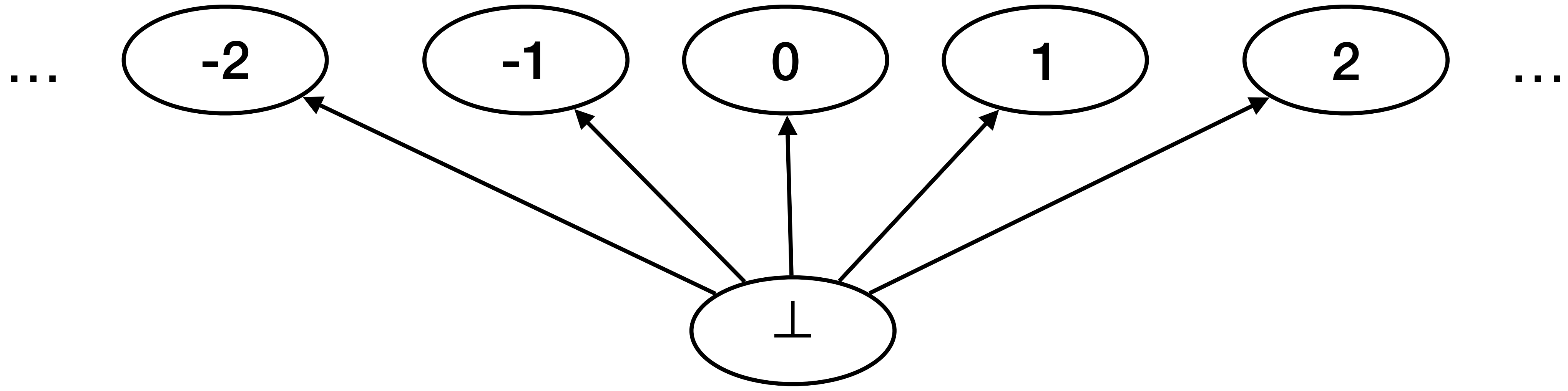
An upper bound d is the **least upper bound** of X iff for all upper bounds y of X, $d \sqsubseteq y$.
The least upper bound of X is denoted by $\bigsqcup X$.

Least Upper Bound (Join)

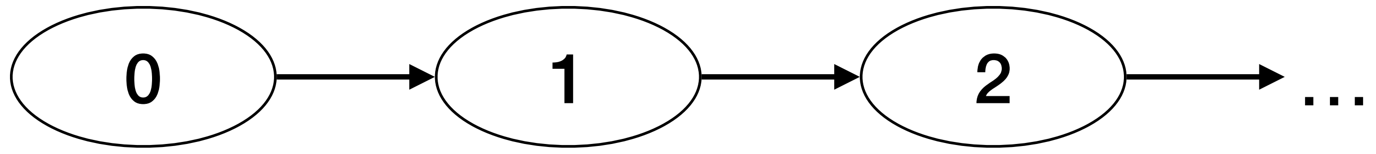
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



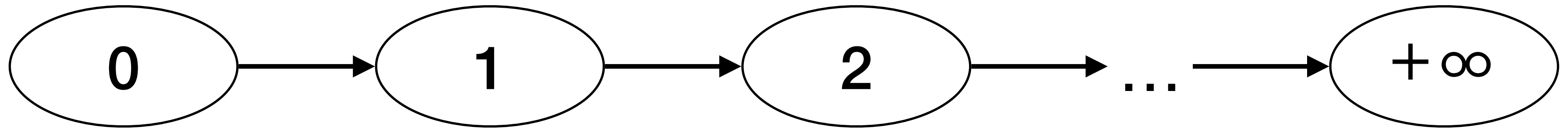
- Example 2: $(\mathbb{Z}_\bot, \sqsubseteq)$



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



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



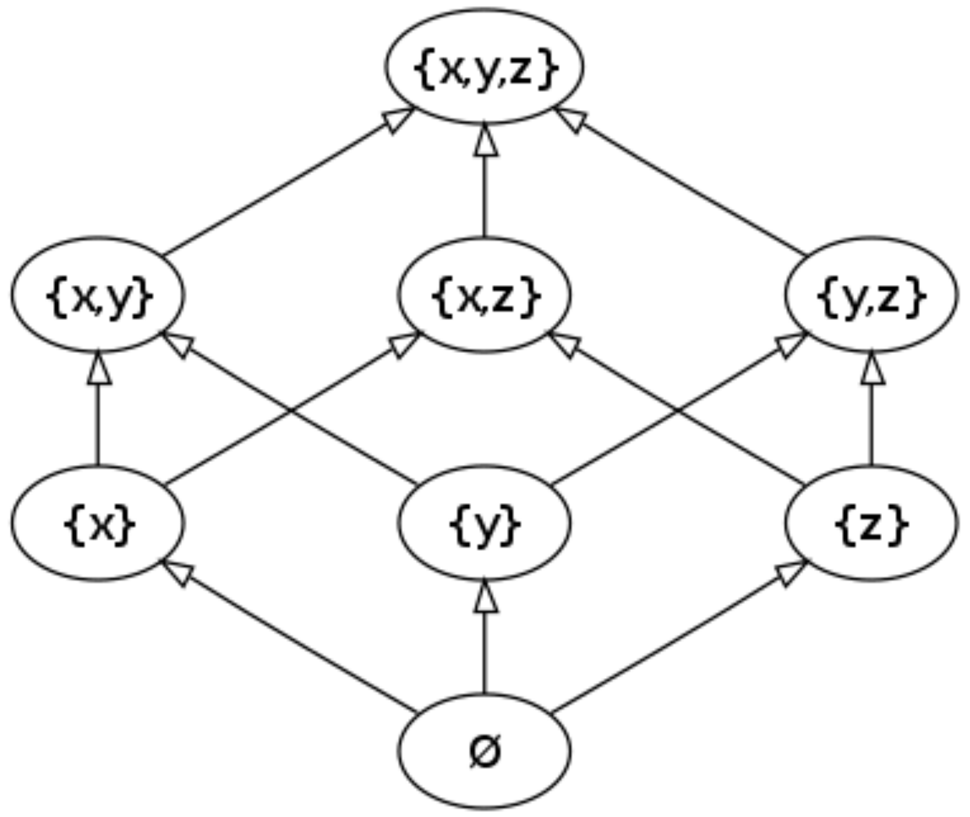
Chain

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

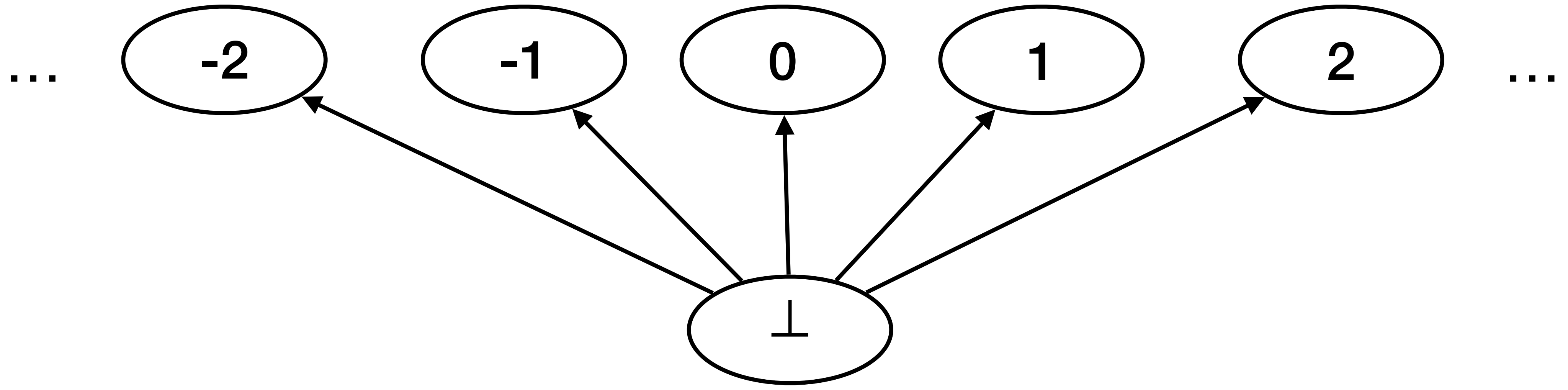
$$\forall x_1, x_2 \in X . x_1 \sqsubseteq x_2 \text{ or } x_2 \sqsubseteq x_1$$

Chain

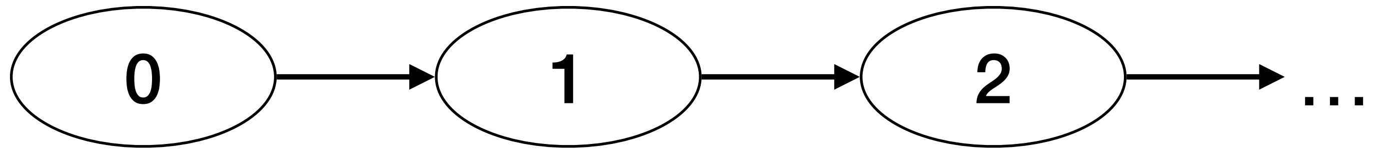
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



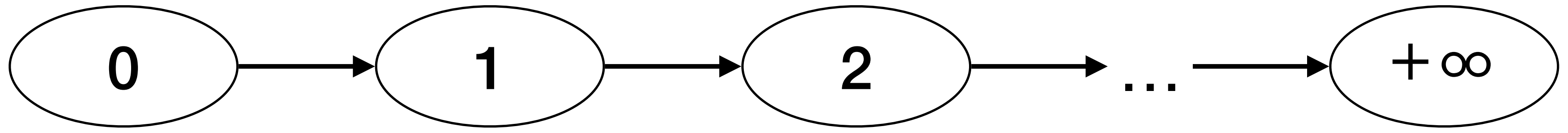
- Example 2: $(\mathbb{Z}_\perp, \sqsubseteq)$



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



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



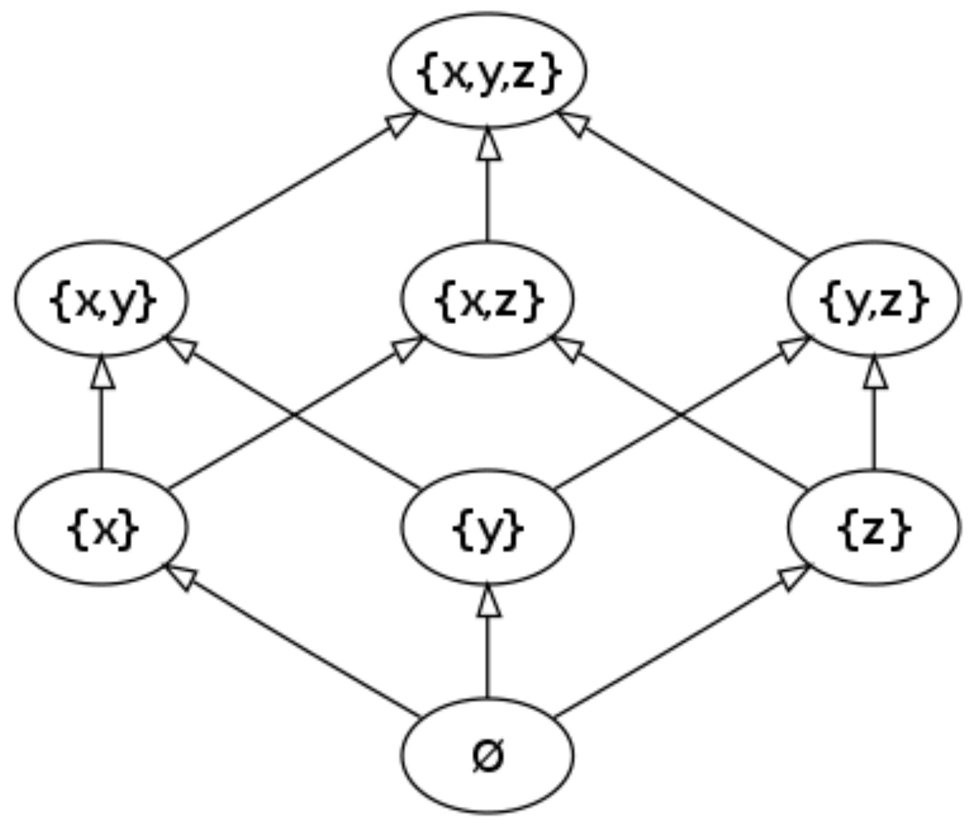
CPO

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

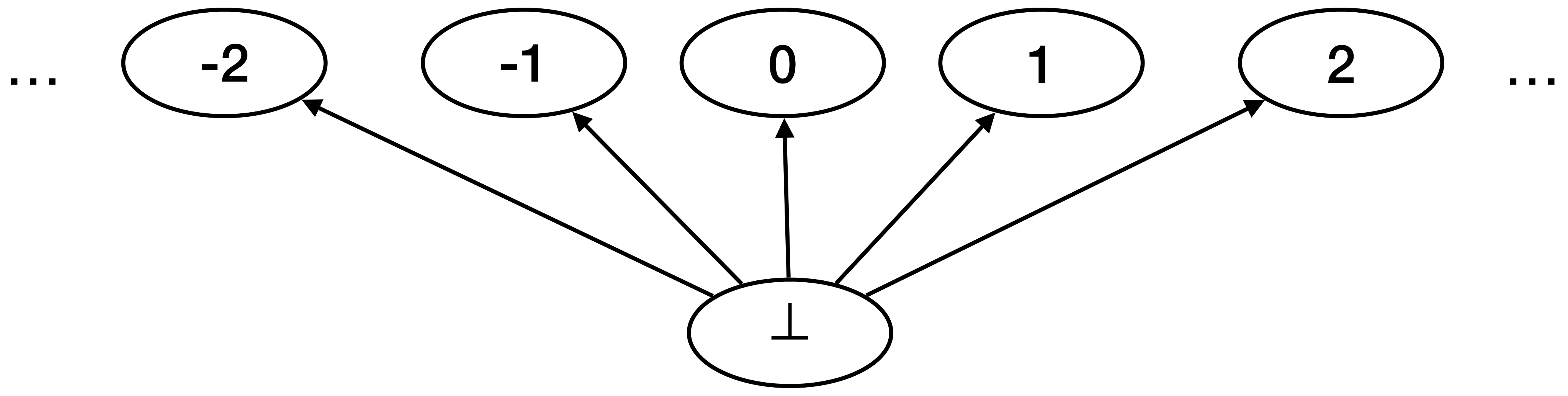
Lemma. If poset $(D, \sqsubseteq)$ is a CPO, it has the **least element** $\bot = \bigsqcup \emptyset$.

CPO

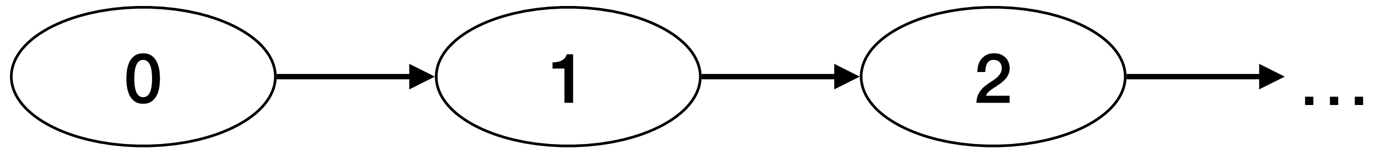
- Example 1: $(\wp(\{x, y, z\}), \subseteq)$



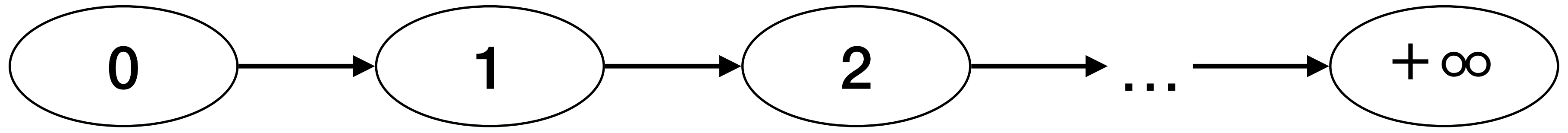
- Example 2: $(\mathbb{Z}_\perp, \sqsubseteq)$



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



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

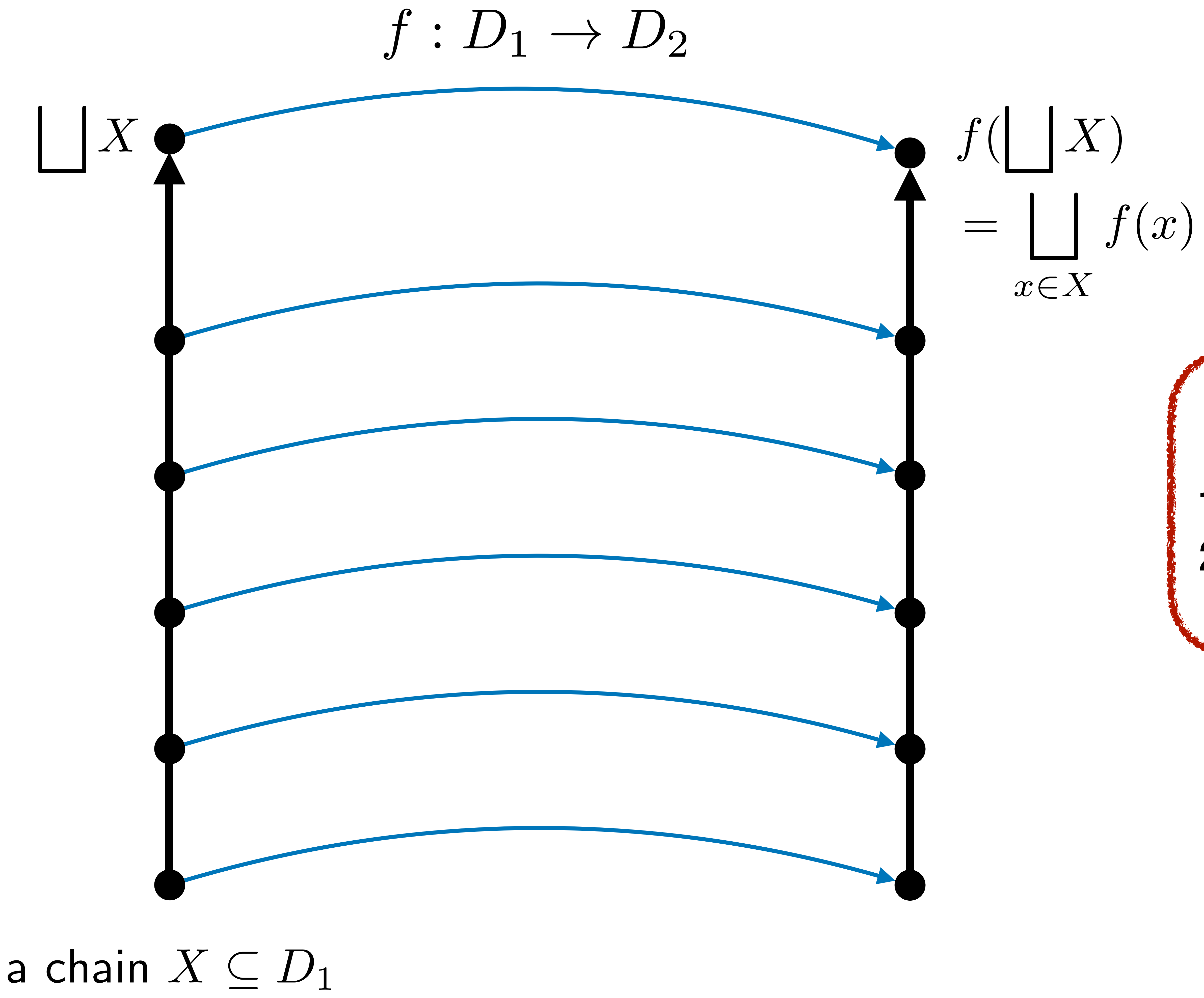


Continuous Function

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:

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

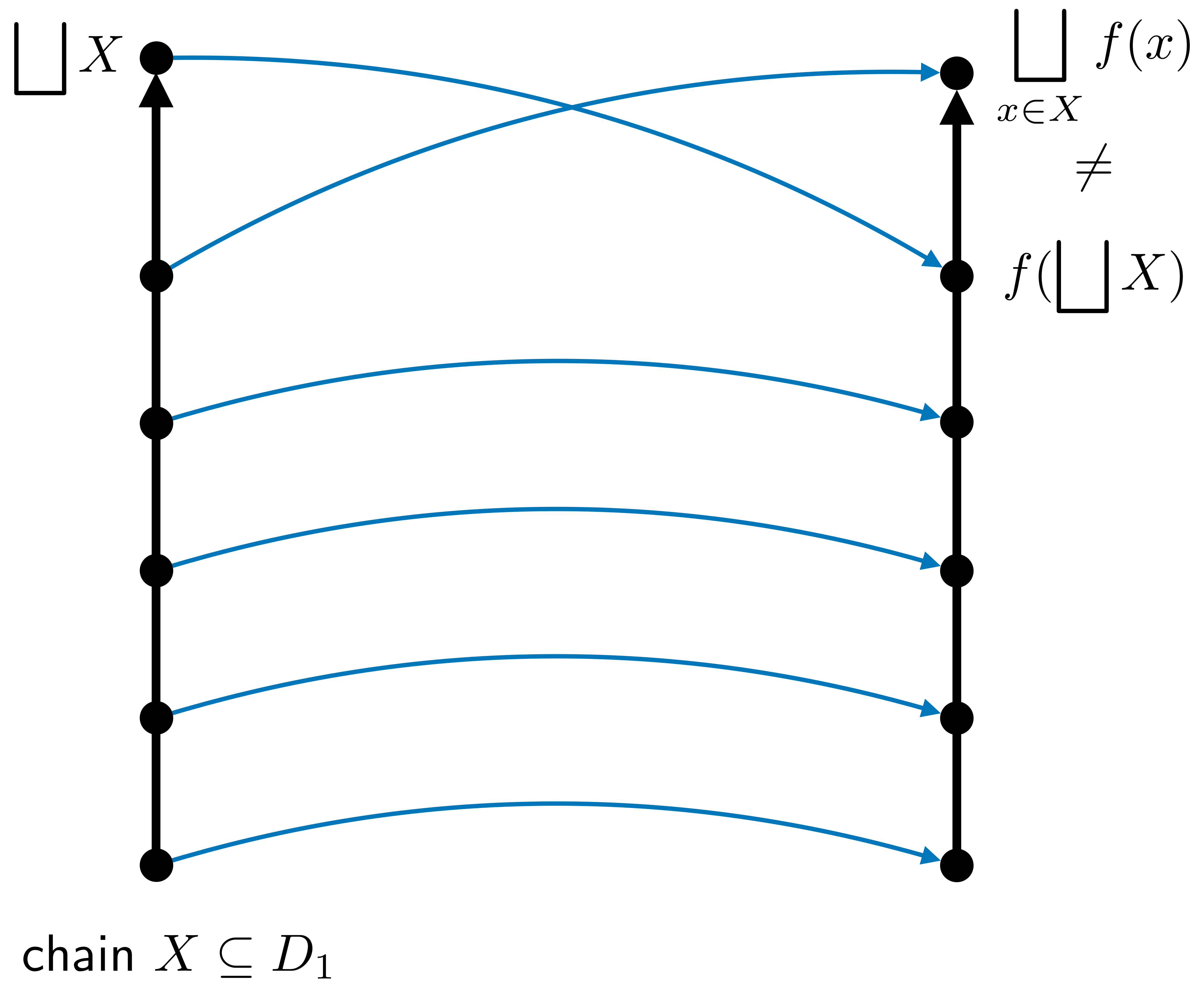
Continuous Function



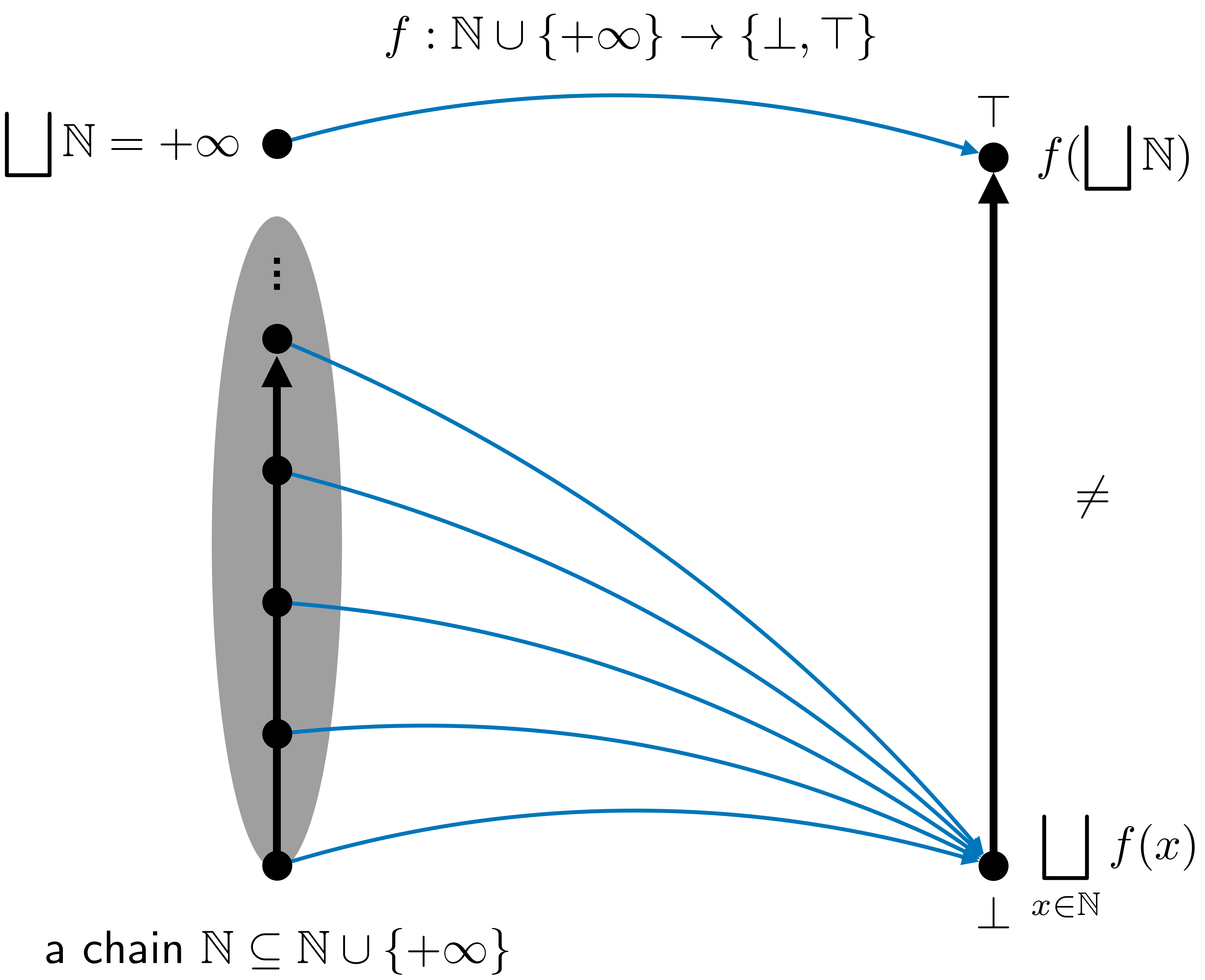
Continuous =
1. Monotone +
2. LUB preservation

Non-Continuous Function (1)

$f : D_1 \to D_2$



Non-Continuous Function (2)



Continuous Function

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

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

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

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 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 $\forall n \in \mathbb{N}.\ f^n(\bot) \sqsubseteq f^{n+1}(\bot)$:

- $\bot \sqsubseteq f(\bot)$ $\quad$ ($\bot$ is the least element of the CPO)
- $f^n(\bot) \sqsubseteq f^{n+1}(\bot) \implies f^{n+1}(\bot) \sqsubseteq f^{n+2}(\bot)$ $\quad$ (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 $\bigsqcup_{i \geq 0} f^i(\perp)$ is the least fixed point of f

The proof consists of two parts:

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

(2-2) $\bigsqcup_{i \geq 0} f^i(\perp)$ 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) $\bigsqcup_{i \geq 0} f^i(\perp)$ 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 $f^i(\perp)$ is smaller than d by induction:

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

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

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

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

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?

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?

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?

Semantic Domains (Revisited)

- Domains of memories, variables, and values: CPOs

$$\begin{aligned}\mathbb{M} &= \mathbb{X} \to \mathbb{V} \\ \mathbb{X} &= \mathit{Var}_\perp \\ \mathbb{V} &= \mathbb{Z}_\perp + \mathbb{B}_\perp\end{aligned}$$

- Domain of commands: function CPO

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

- Domain of expressions: function CPO

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

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`

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

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

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

Example

$$\mathcal{F}(X) = \lambda m. \begin{cases} X(\llbracket x := x + 1 \rrbracket(m)) & \text{if } \llbracket \mathtt{x} < \mathtt{10} \rrbracket(m) = \mathtt{true} \\ m & \text{if } \llbracket \mathtt{x} < \mathtt{10} \rrbracket(m) = \mathtt{false} \end{cases}$$

$$\bot$$

0 iter

$$\mathcal{F}(\bot) = \lambda m. \begin{cases} \bot(\llbracket \mathrm{x} := \mathrm{x} + 1 \rrbracket(m)) & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(m) = \mathtt{true} \\ m & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(m) = \mathtt{false} \end{cases}$$

0, 1 iter

$$\mathcal{F}^2(\bot) = \lambda m. \begin{cases} \mathcal{F}(\bot)(\llbracket \mathrm{x} := \mathrm{x} + 1 \rrbracket(m)) & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(m) = \mathtt{true} \\ m & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(m) = \mathtt{false} \end{cases}$$

$$= \lambda m. \begin{cases} \begin{cases} \bot(\llbracket \mathrm{x} := \mathrm{x} + 1 \rrbracket^2(m)) & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(\llbracket \mathrm{x} := \mathrm{x} + 1 \rrbracket(m)) = \mathtt{true} \\ \llbracket \mathrm{x} := \mathrm{x} + 1 \rrbracket(m) & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(\llbracket \mathrm{x} := \mathrm{x} + 1 \rrbracket(m)) = \mathtt{false} \end{cases} & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(m) = \mathtt{true} \\ m & \text{if } \llbracket \mathrm{x} < \mathtt{10} \rrbracket(m) = \mathtt{false} \end{cases}$$

0,1,2 iters

$$\mathcal{F}^3(\bot) = \cdots$$

Semantics of Commands (Revisited)

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

Summary

- Denotational semantics describes mathematical meaning of programs
  - semantics of a program is an element of a **CPO**: $\llbracket P \rrbracket \in D$
  - 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:

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