[OPLSS: Static Analysis]

Abstract Interpretation

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**

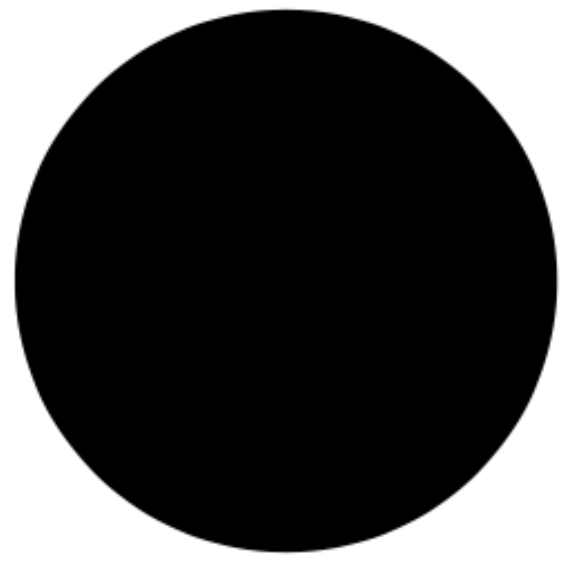
(4) **Automatic Derivation of Static Analysis**

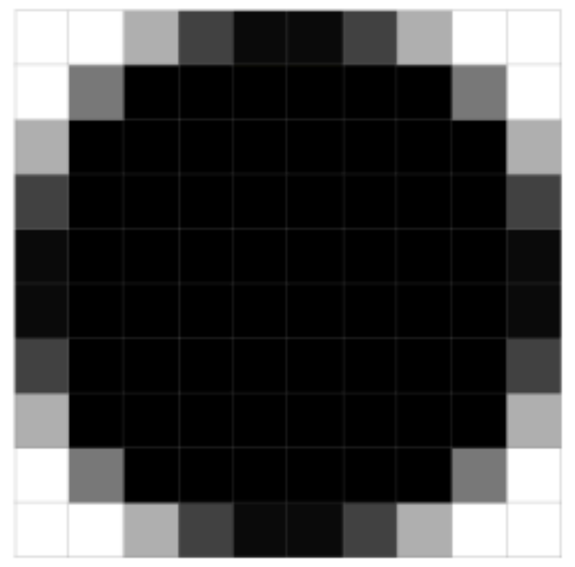
Abstract Interpretation

- A **powerful framework** for designing **correct** static analysis
  - Framework: given some inputs, a static analysis comes out
  - Powerful: all static analyses are understood in this framework (e.g., type systems, data-flow analysis, etc)
  - Correct: mathematically proven
- Estcabilished by Patrick and Radhia Cousot
  - *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints*, 1977
  - *Systematic Design of Program Analysis Frameworks*, 1979

Abstract

- Concrete (execution, dynamic) vs Abstract (analysis, static)
- Without abstraction, it is undecidable to subsume all possible behavior of SW
  - Recall the Rice’s theorem


Concrete


Abstract

Example

```
x = 3;
while (*) {
  x += 2;
}
x -= 1;
print(x);
```

Q: What are the possible output values?

- Concrete interpretation : 2, 4, …, uncomputable (infinitely many possibilities)
- Abstract interpretation 1 : “integers” (good)
- Abstract interpretation 2 : “positive integers” (better)
- Abstract interpretation 3 : “positive even integers” (best)

How to analyze?

- Interpret the target program
  - with abstract semantics (= analyzer’s concern)
  - not concrete semantics (= interpreter’s and compiler’s concern)
- Example

```
x = 3;
while (*) {
  x += 2;
}
x -= 1;
print(x);
```

Concrete	Abstract 1	Abstract 2	Abstract 3
{3}	Int	Pos	PosOdd
{3, 5, 7, …}	Int	Pos	PosOdd
{2, 4, 6, …}	Int	Pos	PosEven

Principles

Concrete Semantics $\overset{?}{\approx}$ **Abstract Semantics**

- How to guarantee soundness?
- How to guarantee termination?
- How to design more precise abstraction?
- How to compute abstract semantics?

Practices

Concrete Semantics $\overset{?}{\approx}$ **Abstract Semantics**

- Guidance for a lot of design choices in practice such as
  - Soundness vs Scalability vs Precision vs Usability vs …
  - Characteristics of target programs and properties
  - Optimizations of program analyzers

Abstract Interpretation Framework

- Abstract interpretation concerns
  - Concrete semantics: $[\![C]\!] = \mathsf{lfp} F \in \mathbb{D}$
  - Abstract semantics: $[\![C]\!]^\sharp = \bigsqcup_{i \geq 0} F^{\sharp i}(\bot) \in \mathbb{D}^\sharp$
- Requirements:
  - Relationship between $\mathbb{D}$ and $\mathbb{D}^\sharp$
  - Relationship between $F \in \mathbb{D} \to \mathbb{D}$ and $F^\sharp \in \mathbb{D}^\sharp \to \mathbb{D}^\sharp$
- Guarantees:
  - Correctness (soundness): $[\![C]\!] \approx [\![C]\!]^\sharp$
  - Computability: $[\![C]\!]^\sharp$ is computable within finite time

Design of Static Analysis

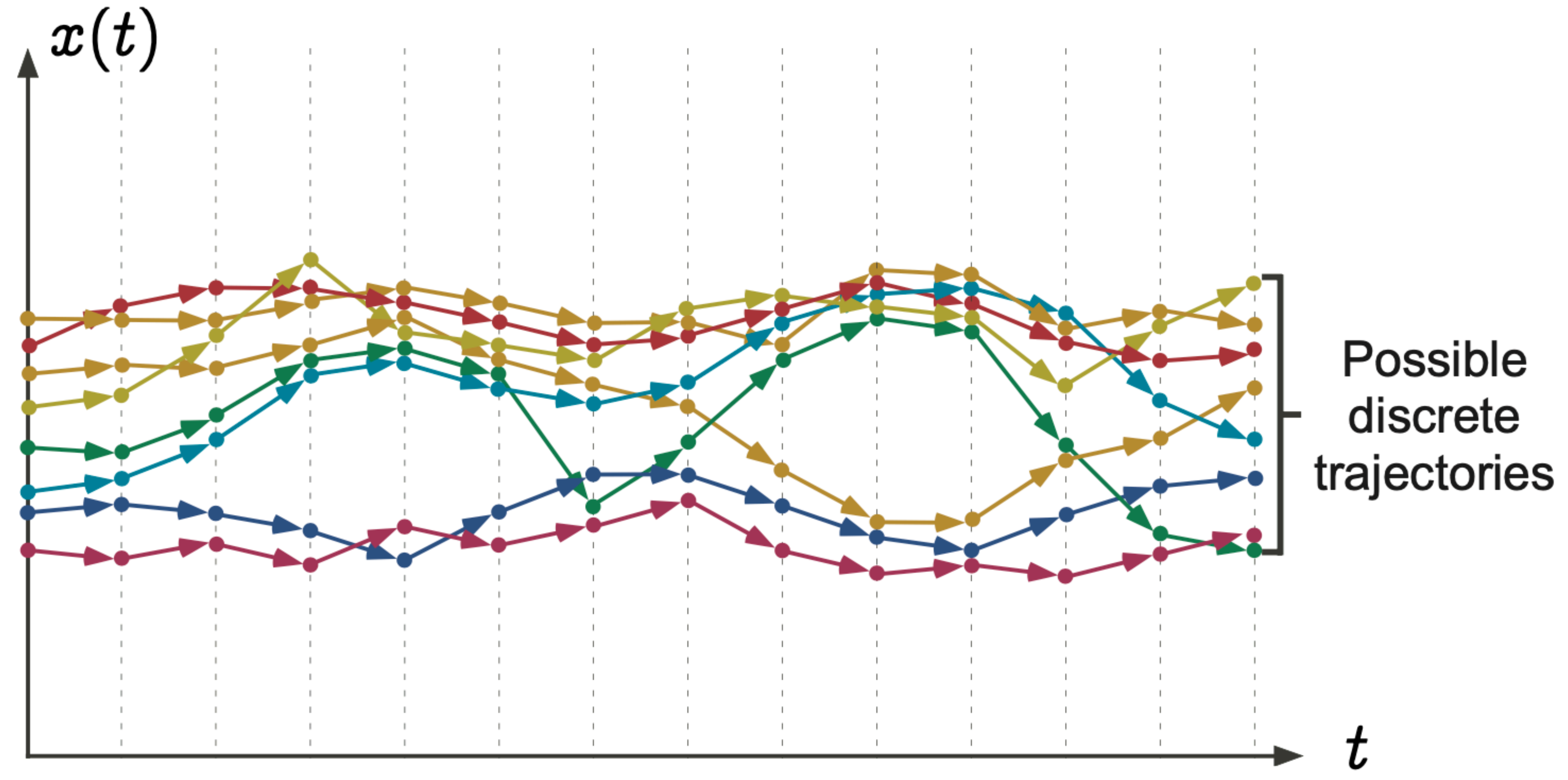
- Goal: conservative and terminating static analysis
- Design principles:
  - Define concrete semantics
  - Define abstract semantics (sound w.r.t the concrete semantics)
- Computation & implementation:
  - Abstract semantics of a program: the least fixed point of the semantic function
  - Static analyzer: compute the least fixed point within finite time

Step 0: Define Standard Semantics

- Formalization of a **single program execution**
  - Recall Lecture 2 (denotation semantics)
- What to describe: different choices depending on the purpose
  - E.g., denotational, operational, etc
- In this lecture, we will use denotational semantics
  - Recall the denotational semantics the simple imperative language

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

Step 0: Define Standard Semantics



*from Patrick Cousot's slides

Standard Semantics

- Define a semantic domain $\mathbb{D} = \mathbb{M} \to \mathbb{M}$ (CPO)
- Define a semantic function $F : \mathbb{D} \to \mathbb{D}$ (continuous)
- Semantics of a program: the least fixed point of F

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

Standard Semantics of Commands

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

Step 1: Define Concrete Semantics

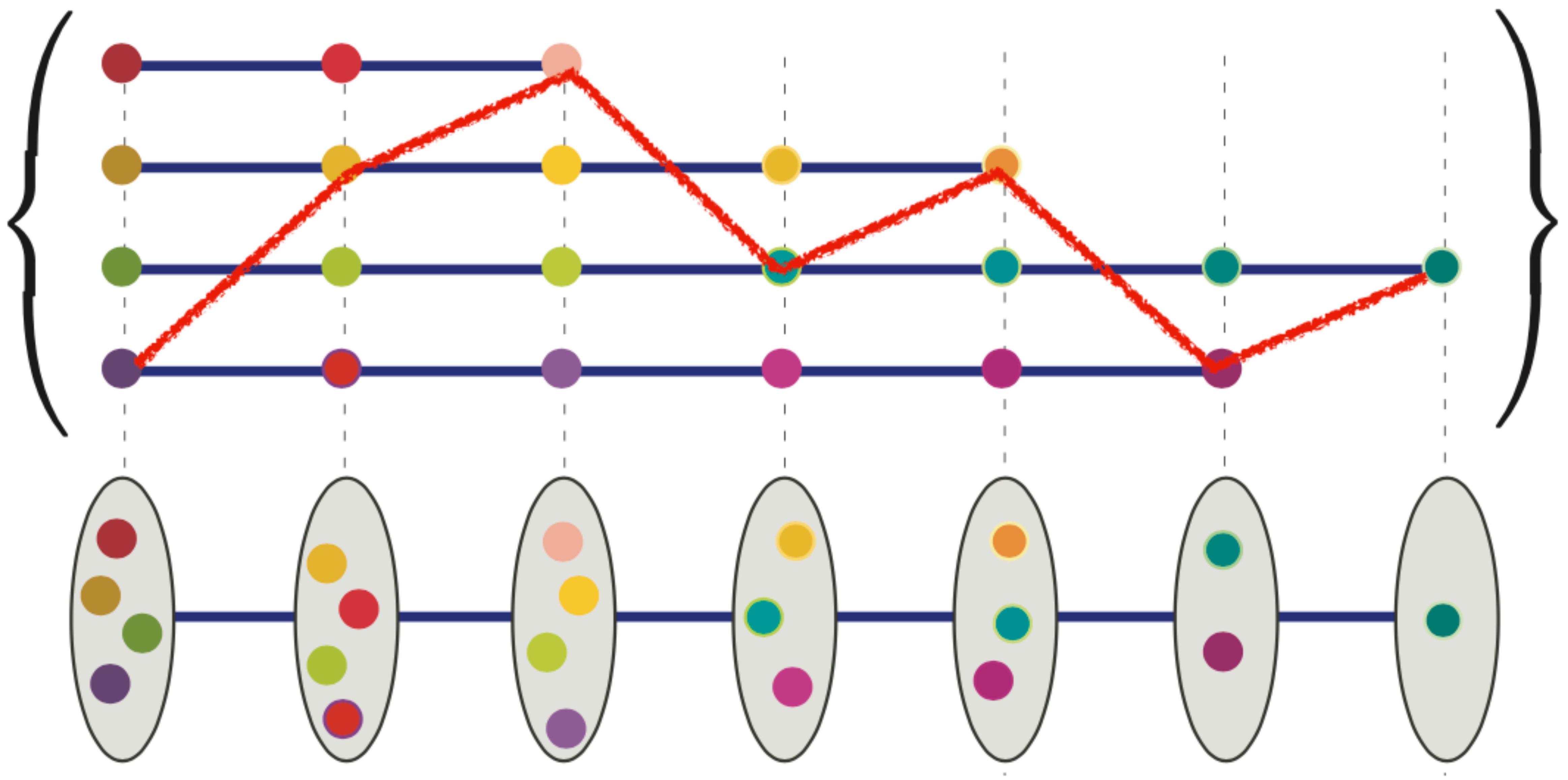
- Formalization of **all possible** program executions
  - So-called collecting semantics
  - Usually a simple extension of the standard semantics
- What to describe: different choices depending on the purposes (recall, property)
  - Some are more expressive than others
  - E.g., traces (sequence of states), reachable states (set of states), etc
- In this lecture, we will use reachable states for concrete semantics

$$\llbracket C \rrbracket : \mathbb{M} \to \mathbb{M} \quad \xrightarrow{\textbf{collecting}} \quad \llbracket C \rrbracket_{\wp} : \wp(\mathbb{M}) \to \wp(\mathbb{M})$$

Traces vs Reachable States

Traces

Reachable States



Can Answer:

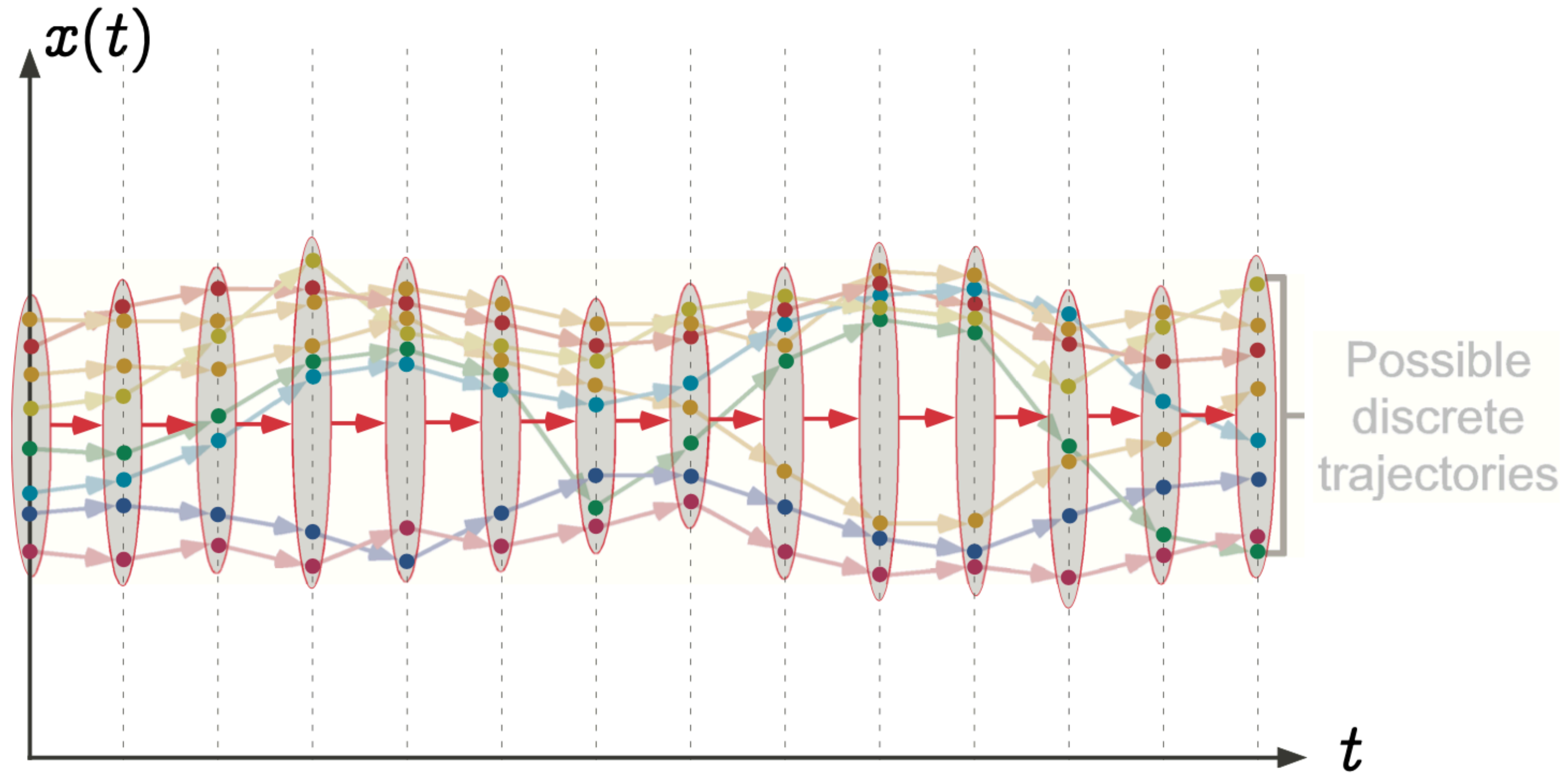
- Can variable `p` be `NULL` at line 10?
- Can buffer index `i` be larger than size `s`?
- ...

Can't Answer:

- Does the red trace exist?
- ...

*from Patrick Cousot's slides

Transitions of Sets of States



*from Patrick Cousot's slides

Concrete Semantics

- Define a concrete domain $\mathbb{D}$ (CPO)
- Define a semantic function $F : \mathbb{D} \to \mathbb{D}$ (continuous)
- Then the concrete semantics is defined as the least fixed point of the semantic function F:

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

Example

- Define a concrete semantics of the simple language using denotational semantics
    - Concrete domain $\mathbb{D} = \wp(\mathbb{M}) \rightarrow \wp(\mathbb{M})$
    - Define a semantic function $F : \mathbb{D} \rightarrow \mathbb{D}$
    - Concrete semantics $\mathbf{lfp}F \in \mathbb{D}$
- Q: How to define F?

Concrete Semantics of Expressions

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

$$
\begin{aligned}
[\![B]\!]_{\wp} &: \wp(\mathbb{M}) \to \wp(\mathbb{M}) \\
[\![\mathsf{true}]\!]_{\wp} &= \lambda M.M \\
[\![\mathsf{false}]\!]_{\wp} &= \lambda M.\emptyset \\
[\![E_1 \otimes E_2]\!]_{\wp} &= \lambda M.\{m \in M \mid [\![E_1]\!](m) \otimes [\![E_2]\!](m) = \mathsf{true}\}
\end{aligned}
$$

Concrete Semantics of Commands

$$[\![C]\!]_{\wp} \ : \ \wp(\mathbb{M}) \to \ \wp(\mathbb{M})$$

$$\begin{aligned}
[\![\texttt{skip}]\!]_{\wp} &= \lambda M.M \\
[\![C_0\texttt{;}C_1]\!]_{\wp} &= \lambda M.[\![C_1]\!]_{\wp} \circ [\![C_0]\!]_{\wp}(M) \\
[\![x\texttt{:=}E]\!]_{\wp} &= \lambda M.\{m\{x \mapsto [\![E]\!](m)\} \mid m \in M\} \\
[\![\texttt{input}(x)]\!]_{\wp} &= \lambda M.\{m\{x \mapsto n\} \mid m \in M, n \in \mathbb{Z}\} \\
[\![\texttt{if } B \texttt{ then } C_1 \texttt{ else } C_2]\!]_{\wp} &= \lambda M.[\![C_1]\!]_{\wp} \circ [\![B]\!]_{\wp}(M) \cup [\![C_2]\!]_{\wp} \circ [\![\neg B]\!]_{\wp}(M) \\
[\![\texttt{while } B\ C]\!]_{\wp} &= \lambda M.[\![\neg B]\!]_{\wp}\big(\mathbf{lfp}\lambda X.M \cup [\![C]\!]_{\wp} \circ [\![B]\!]_{\wp}(X)\big)
\end{aligned}$$

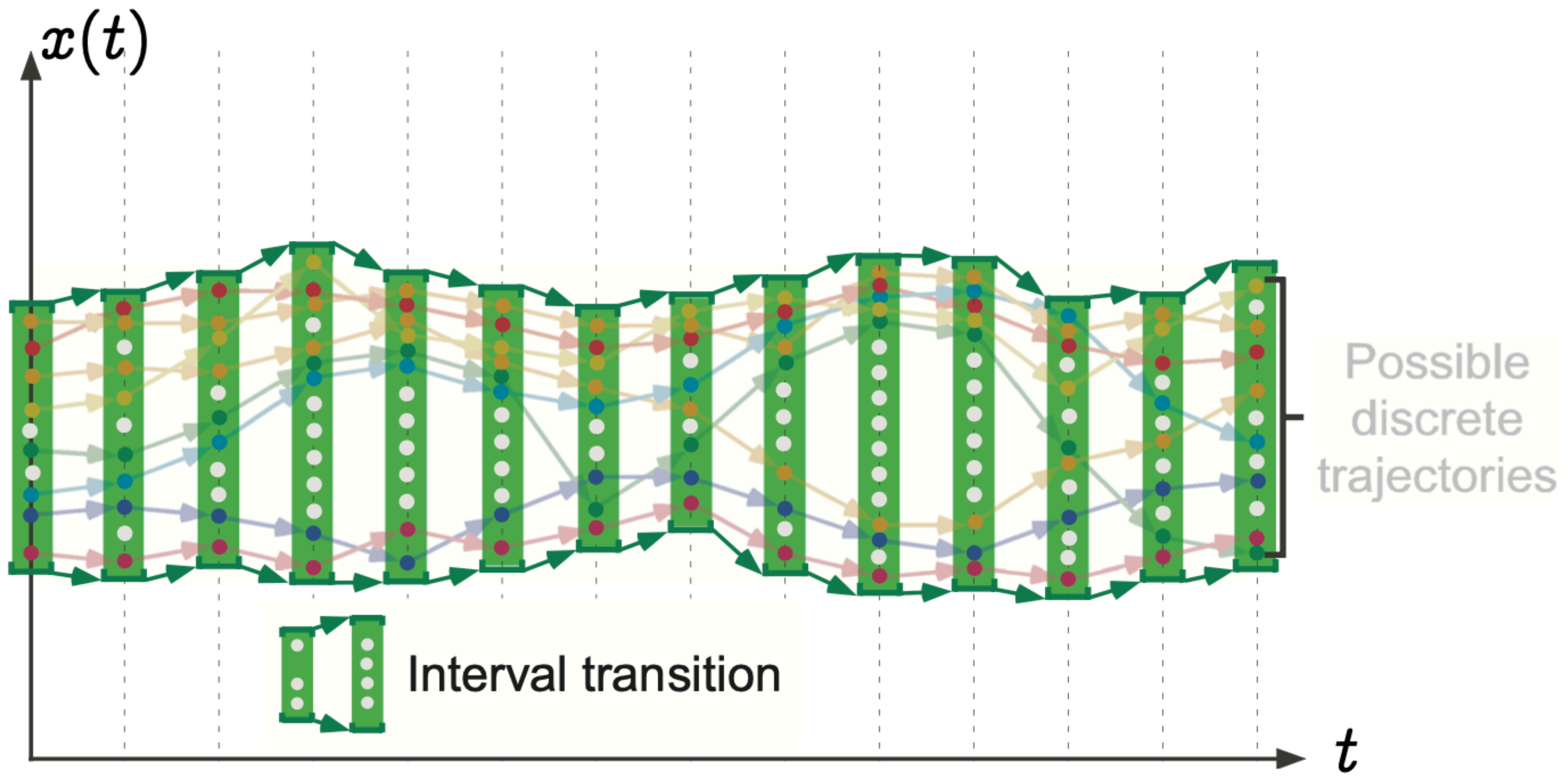
Design of Static Analysis

- Goal: conservative and terminating static analysis
- Design principles:
  - Define concrete semantics
  - Define abstract semantics (sound w.r.t the concrete semantics)
- Computation & implementation:
  - Abstract semantics of a program: the least fixed point of the semantic function
  - Static analyzer: compute the least fixed point within finite time

Step 2: Design Abstract Semantics

- Formalization of **abstract** program executions
  - Soundly subsume concrete executions
- How to subsume: different choices depending on the purposes (some are more expressive than others)
- Example: abstraction of {1, 3, 5, 7}
  - Integer, Positive, Odd, [1, 7], etc

Transitions of Abstract States



*from Patrick Cousot's slides

Abstract Semantics

- Define an abstract domain $\mathbb{D}^\sharp$ (CPO)
- Define an abstract semantic function $F^\sharp : \mathbb{D}^\sharp \rightarrow \mathbb{D}^\sharp$ (monotone or extensive)

(Monotone) $\forall x^\sharp, y^\sharp \in \mathbb{D}^\sharp.\ x^\sharp \sqsubseteq y^\sharp \implies F^\sharp(x^\sharp) \sqsubseteq F^\sharp(y^\sharp)$

(Extensive) $\forall x^\sharp \in \mathbb{D}.\ x^\sharp \sqsubseteq F^\sharp(x^\sharp)$

- Static analysis is to compute an upper bound of the chain:

$$\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^\sharp)$$

Q. How to ensure that the abstract semantics soundly subsume the concrete semantics?

Requirement 1: Galois Connection

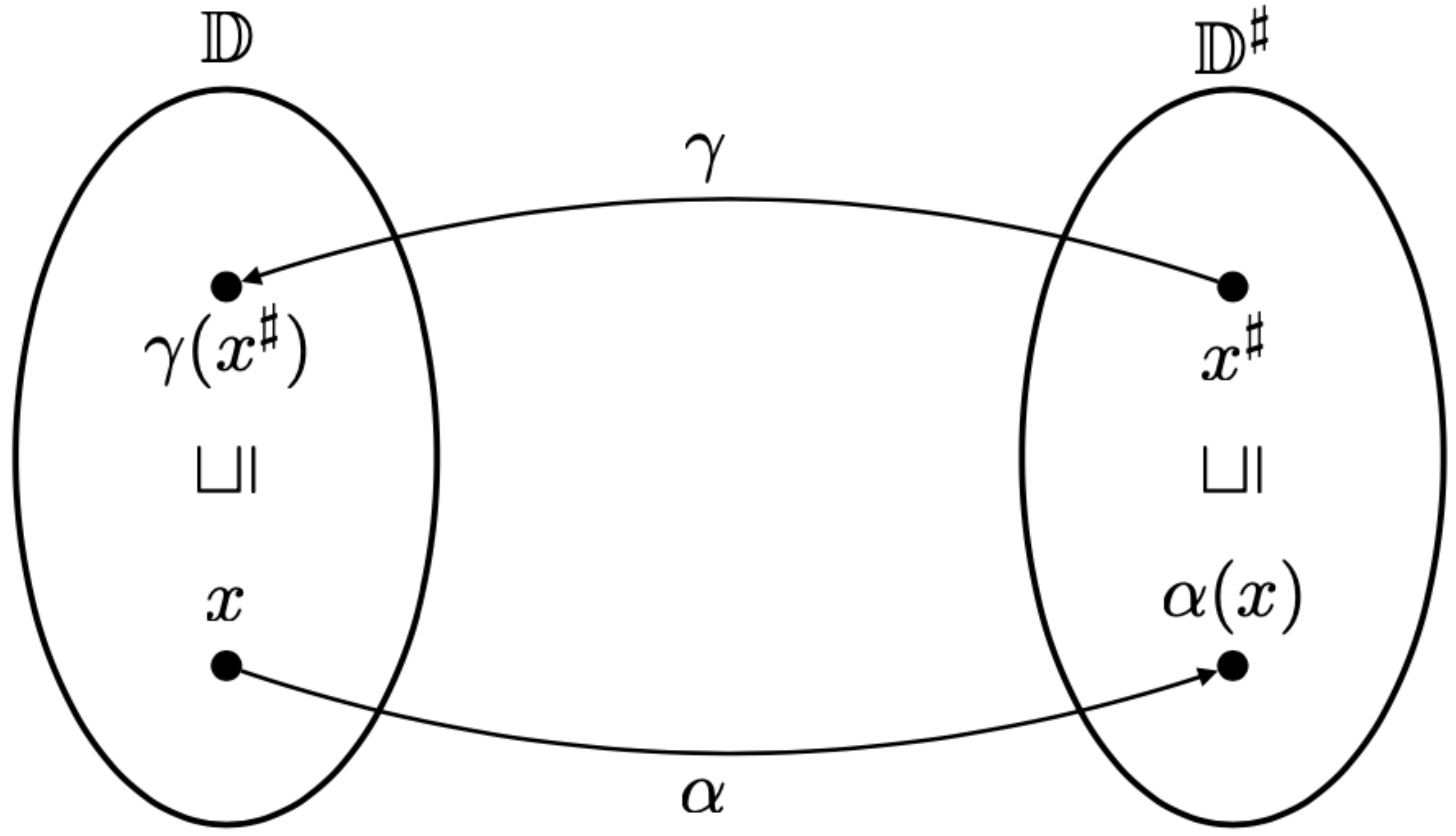
$$\mathbb{D} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{D}^\sharp$$

- $\mathbb{D}$ and $\mathbb{D}^\sharp$ must me related with a Galois connection where
  - Abstraction function: $\alpha \in \mathbb{D} \to \mathbb{D}^\sharp$
  - Concretization function: $\gamma \in \mathbb{D}^\sharp \to \mathbb{D}$

$$\forall x \in \mathbb{D}, x^\sharp \in \mathbb{D}^\sharp.\ \alpha(x) \sqsubseteq x^\sharp \iff x \sqsubseteq \gamma(x^\sharp)$$

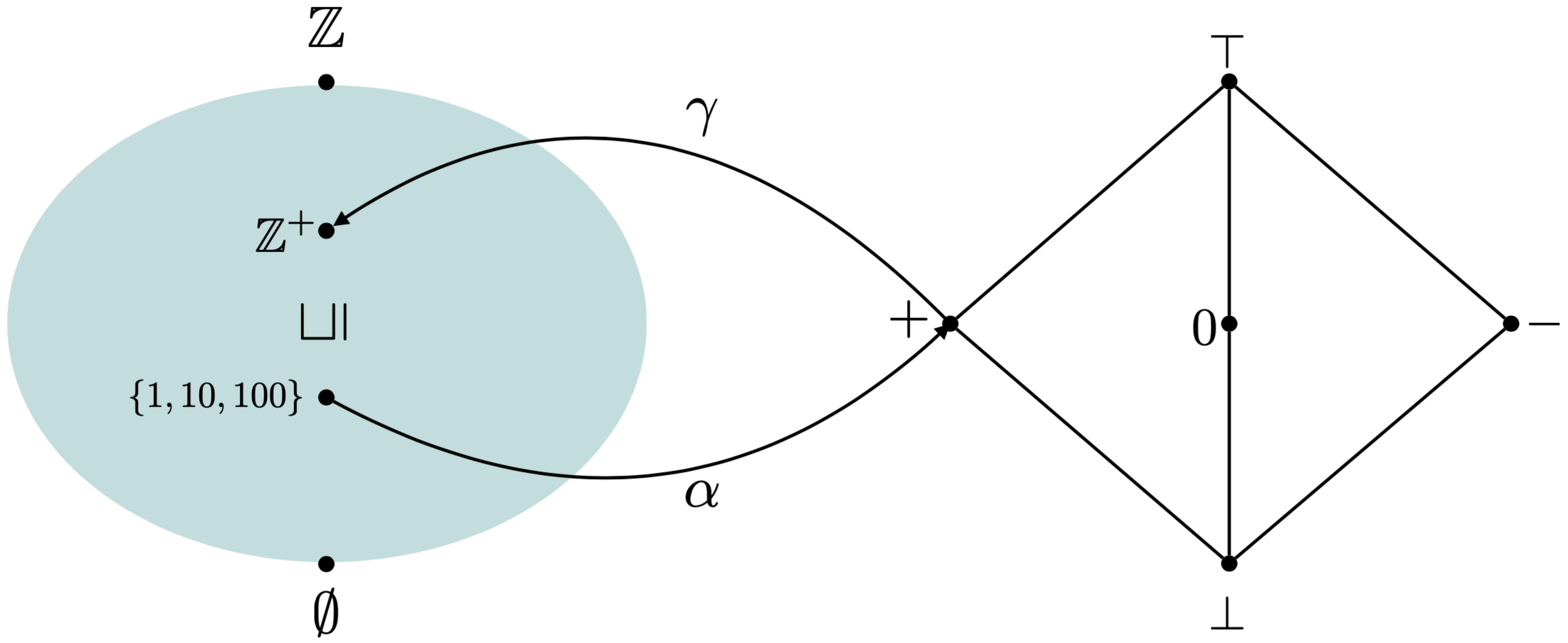
Requirement 1: Galois Connection

- Intuition: order preservation between two semantic domains



Example: Sign Abstraction

$$\wp(\mathbb{Z}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \{\bot, -, 0, +, \top\}$$



Example: Sign Abstraction

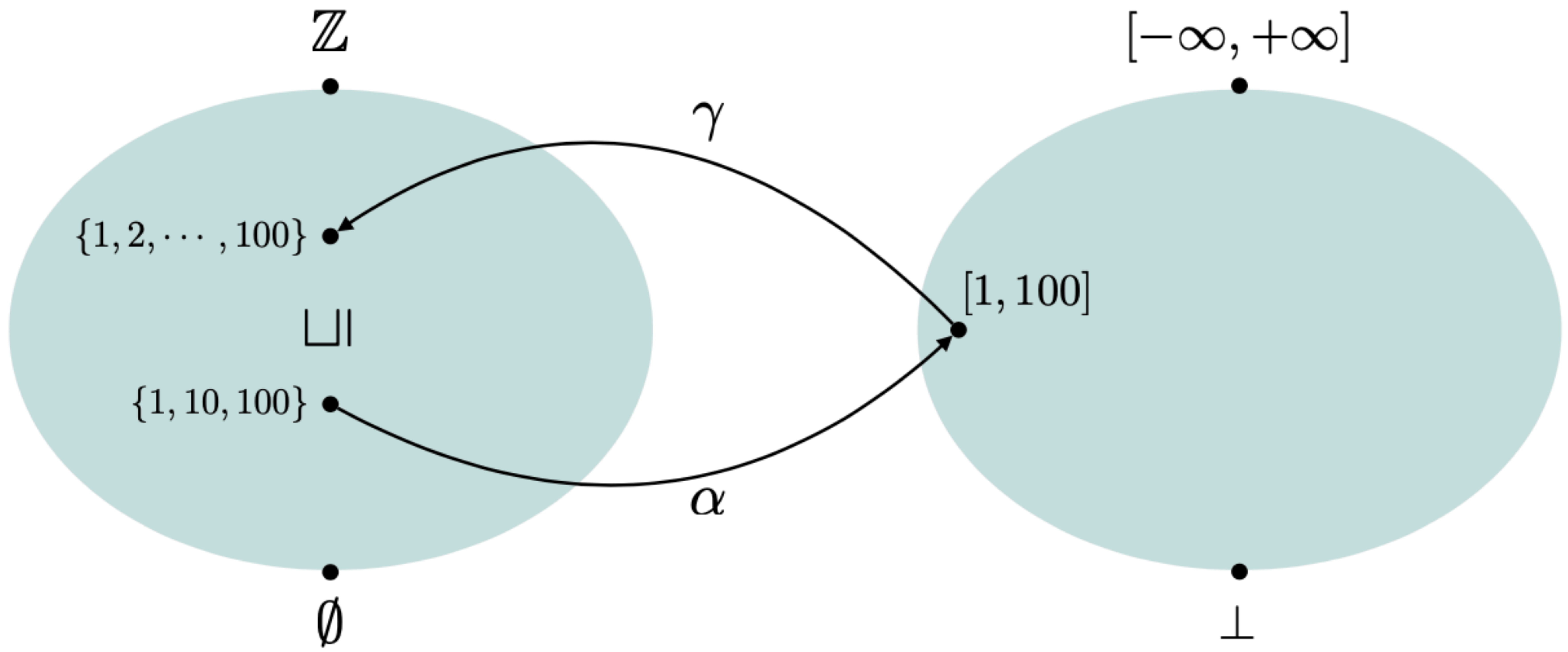
$$\wp(\mathbb{Z}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \{\bot, -, 0, +, \top\}$$

$$\alpha(Z) = \begin{cases} \bot & Z = \emptyset \\ + & \forall z \in Z.\ z > 0 \\ 0 & Z = \{0\} \\ - & \forall z \in Z.\ z < 0 \\ \top & \text{otherwise} \end{cases}$$

$$\gamma(\bot) = \emptyset$$
$$\gamma(+) = \{z \in \mathbb{Z} \mid z > 0\}$$
$$\gamma(0) = \{0\}$$
$$\gamma(-) = \{z \in \mathbb{Z} \mid z < 0\}$$
$$\gamma(\top) = \mathbb{Z}$$

Example: Interval Abstraction

$$\wp(\mathbb{Z}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \{\bot\} \cup \{[a,b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\}\}$$



Example: Interval Abstraction

$$\wp(\mathbb{Z}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \{\bot\} \cup \{[a,b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\}\}$$

$$\begin{aligned} \alpha(\emptyset) &= \bot \\ \alpha(X) &= [\min X, \max X] \end{aligned}$$

$$\begin{aligned} \gamma(\bot) &= \emptyset \\ \gamma([a,b]) &= \{x \in \mathbb{Z} \mid a \leq x \leq b\} \end{aligned}$$

Properties of Galois Connection

$$\forall x \in \mathbb{D}, x^\sharp \in \mathbb{D}^\sharp.\ \alpha(x) \sqsubseteq x^\sharp \iff x \sqsubseteq \gamma(x^\sharp)$$

Properties of Galois Connection

$$\forall x \in \mathbb{D}, x^\sharp \in \mathbb{D}^\sharp.\ \alpha(x) \sqsubseteq x^\sharp \iff x \sqsubseteq \gamma(x^\sharp)$$

- $id \sqsubseteq \gamma \circ \alpha$

$$\begin{array}{lrcll} & \alpha(x) & \sqsubseteq & \alpha(x) & \\ \iff & x & \sqsubseteq & \gamma(\alpha(x)) & \text{(by Galois connection)} \end{array}$$

Properties of Galois Connection

$$\forall x \in \mathbb{D}, x^\sharp \in \mathbb{D}^\sharp.\ \alpha(x) \sqsubseteq x^\sharp \iff x \sqsubseteq \gamma(x^\sharp)$$

- $id \sqsubseteq \gamma \circ \alpha$

$$\begin{array}{rrcll} & \alpha(x) & \sqsubseteq & \alpha(x) & \\ \iff & x & \sqsubseteq & \gamma(\alpha(x)) & \text{(by Galois connection)} \end{array}$$

- $\alpha \circ \gamma \sqsubseteq id$

$$\begin{array}{rrcll} & \gamma(x^\sharp) & \sqsubseteq & \gamma(x^\sharp) & \\ \iff & \alpha(\gamma(x^\sharp)) & \sqsubseteq & x^\sharp & \text{(by Galois connection)} \end{array}$$

Properties of Galois Connection

$$\forall x \in \mathbb{D}, x^\sharp \in \mathbb{D}^\sharp.\ \alpha(x) \sqsubseteq x^\sharp \iff x \sqsubseteq \gamma(x^\sharp)$$

- $id \sqsubseteq \gamma \circ \alpha$

$$\begin{array}{llll} & \alpha(x) & \sqsubseteq & \alpha(x) \\ \iff & x & \sqsubseteq & \gamma(\alpha(x)) \quad \text{(by Galois connection)} \end{array}$$

- $\alpha \circ \gamma \sqsubseteq id$

$$\begin{array}{llll} & \gamma(x^\sharp) & \sqsubseteq & \gamma(x^\sharp) \\ \iff & \alpha(\gamma(x^\sharp)) & \sqsubseteq & x^\sharp \quad \text{(by Galois connection)} \end{array}$$

- α is monotone

$$\begin{array}{llll} & x & \sqsubseteq & y \\ \implies & x & \sqsubseteq & \gamma(\alpha(y)) \quad (id \sqsubseteq \gamma \circ \alpha) \\ \iff & \alpha(x) & \sqsubseteq & \alpha(y) \quad \text{(by Galois connection)} \end{array}$$

Properties of Galois Connection

$$\forall x \in \mathbb{D}, x^\sharp \in \mathbb{D}^\sharp.\ \alpha(x) \sqsubseteq x^\sharp \iff x \sqsubseteq \gamma(x^\sharp)$$

- $id \sqsubseteq \gamma \circ \alpha$

$$\begin{array}{lrcll} & \alpha(x) & \sqsubseteq & \alpha(x) & \\ \iff & x & \sqsubseteq & \gamma(\alpha(x)) & \text{(by Galois connection)} \end{array}$$

- $\alpha \circ \gamma \sqsubseteq id$

$$\begin{array}{lrcll} & \gamma(x^\sharp) & \sqsubseteq & \gamma(x^\sharp) & \\ \iff & \alpha(\gamma(x^\sharp)) & \sqsubseteq & x^\sharp & \text{(by Galois connection)} \end{array}$$

- α is monotone

$$\begin{array}{lrcll} & x & \sqsubseteq & y & \\ \Longrightarrow & x & \sqsubseteq & \gamma(\alpha(y)) & (id \sqsubseteq \gamma \circ \alpha) \\ \iff & \alpha(x) & \sqsubseteq & \alpha(y) & \text{(by Galois connection)} \end{array}$$

- γ is monotone

$$\begin{array}{lrcll} & x^\sharp & \sqsubseteq & y^\sharp & \\ \Longrightarrow & \alpha(\gamma(x^\sharp)) & \sqsubseteq & y^\sharp & (\alpha \circ \gamma \sqsubseteq id) \\ \iff & \gamma(x^\sharp) & \sqsubseteq & \gamma(y^\sharp) & \text{(by Galois connection)} \end{array}$$

Deriving Galois Connections

- Pointwise lifting:

Given a Galois connection $\mathbb{D} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{D}^\sharp$ and a set $\mathbb{S}$

$$\mathbb{S} \to \mathbb{D} \underset{\alpha'}{\overset{\gamma'}{\leftrightarrows}} \mathbb{S} \to \mathbb{D}^\sharp$$

where $\alpha'(f) = \lambda x \in \mathbb{S}.\ \alpha(f(x))$ and $\gamma'(f^\sharp) = \lambda x \in \mathbb{S}.\ \gamma(f^\sharp(x))$

- Composition:

Given two galois connections $\mathbb{D}_1 \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \mathbb{D}_2 \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \mathbb{D}_3$

$$\mathbb{D}_1 \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} \mathbb{D}_3$$

Example

- Concrete domain: $\mathbb{D} = \wp(\mathbb{M}) \to \wp(\mathbb{M})$ where $\mathbb{M} = \mathbb{X} \to \mathbb{Z}$
- Abstract domain: $\mathbb{D}^\sharp = \mathbb{M}^\sharp \to \mathbb{M}^\sharp$ where $\mathbb{M}^\sharp = \mathbb{X} \to \mathbb{Z}^\sharp$
- Galois connection: $\mathbb{D} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{D}^\sharp$
  - Memory abstraction: $\wp(\mathbb{M}) \underset{\alpha_\mathbb{M}}{\overset{\gamma_\mathbb{M}}{\leftrightarrows}} \mathbb{M}^\sharp$ via $\wp(\mathbb{X} \to \mathbb{Z}) \underset{\alpha_\mathbb{M}}{\overset{\gamma_\mathbb{M}}{\leftrightarrows}} \mathbb{X} \to \mathbb{Z}^\sharp$

$$\gamma_\mathbb{M} : \mathbb{M}^\sharp \to \wp(\mathbb{M})$$

$$\gamma_\mathbb{M} = \lambda m^\sharp.\{m \mid \forall x.m(x) \in \gamma_\mathbb{Z}(m^\sharp(x))\}$$

  - Value abstraction: $\wp(\mathbb{Z}) \underset{\alpha_\mathbb{Z}}{\overset{\gamma_\mathbb{Z}}{\leftrightarrows}} \mathbb{Z}^\sharp$

Requirement 2: F and $F^\sharp$

- $F^\sharp$ is a sound abstraction of F (option 1)

$$F \circ \gamma \sqsubseteq \gamma \circ F^\sharp$$

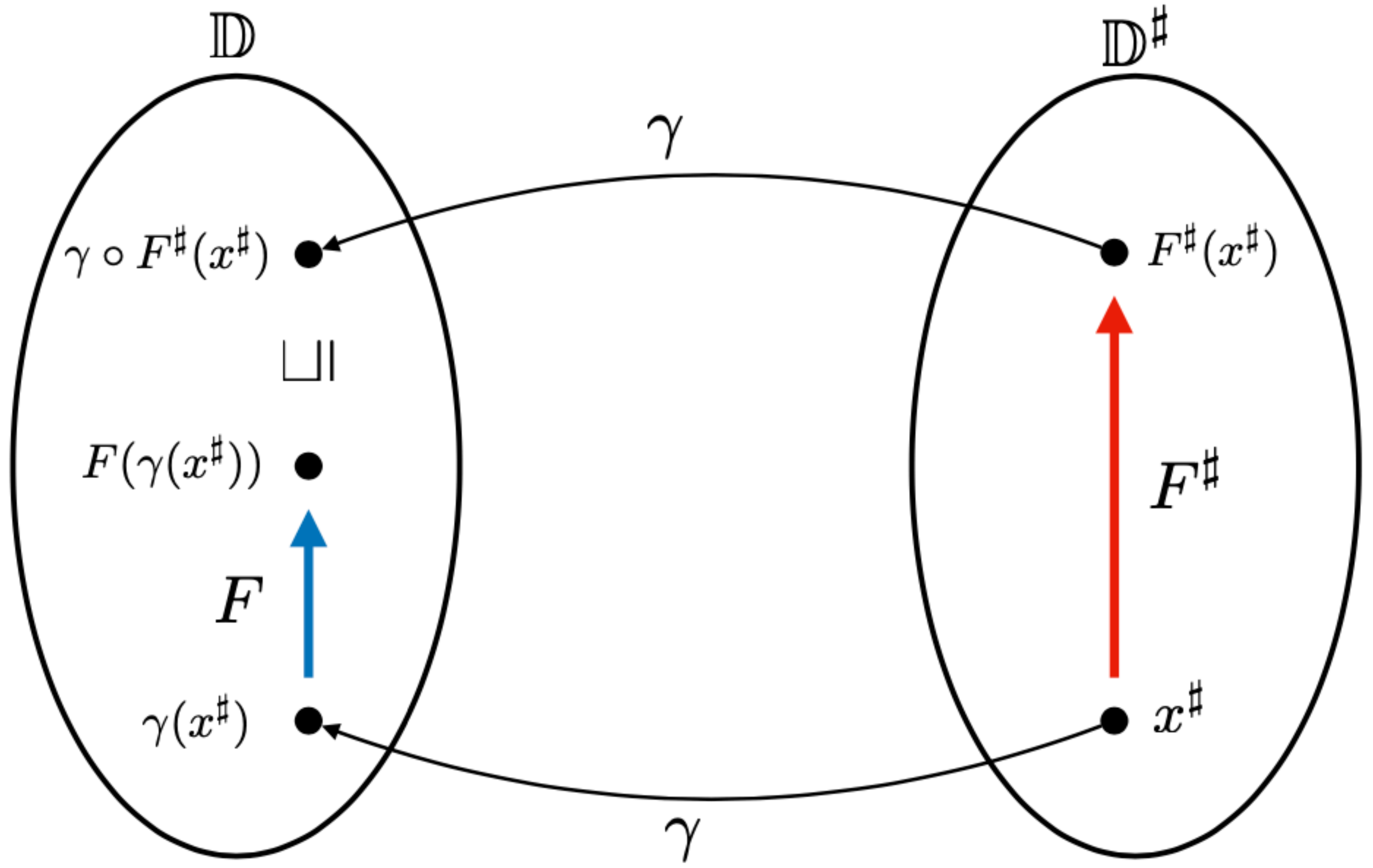
- $F^\sharp$ is a sound abstraction of F (option 2)

$$x \sqsubseteq \gamma(x^\sharp) \implies F(x) \sqsubseteq \gamma(F^\sharp(x^\sharp))$$

Intuition: the result of one-step abstract execution ($F^\sharp$) subsumes that of one-step concrete execution (F)

Sound Abstract Semantics (1)

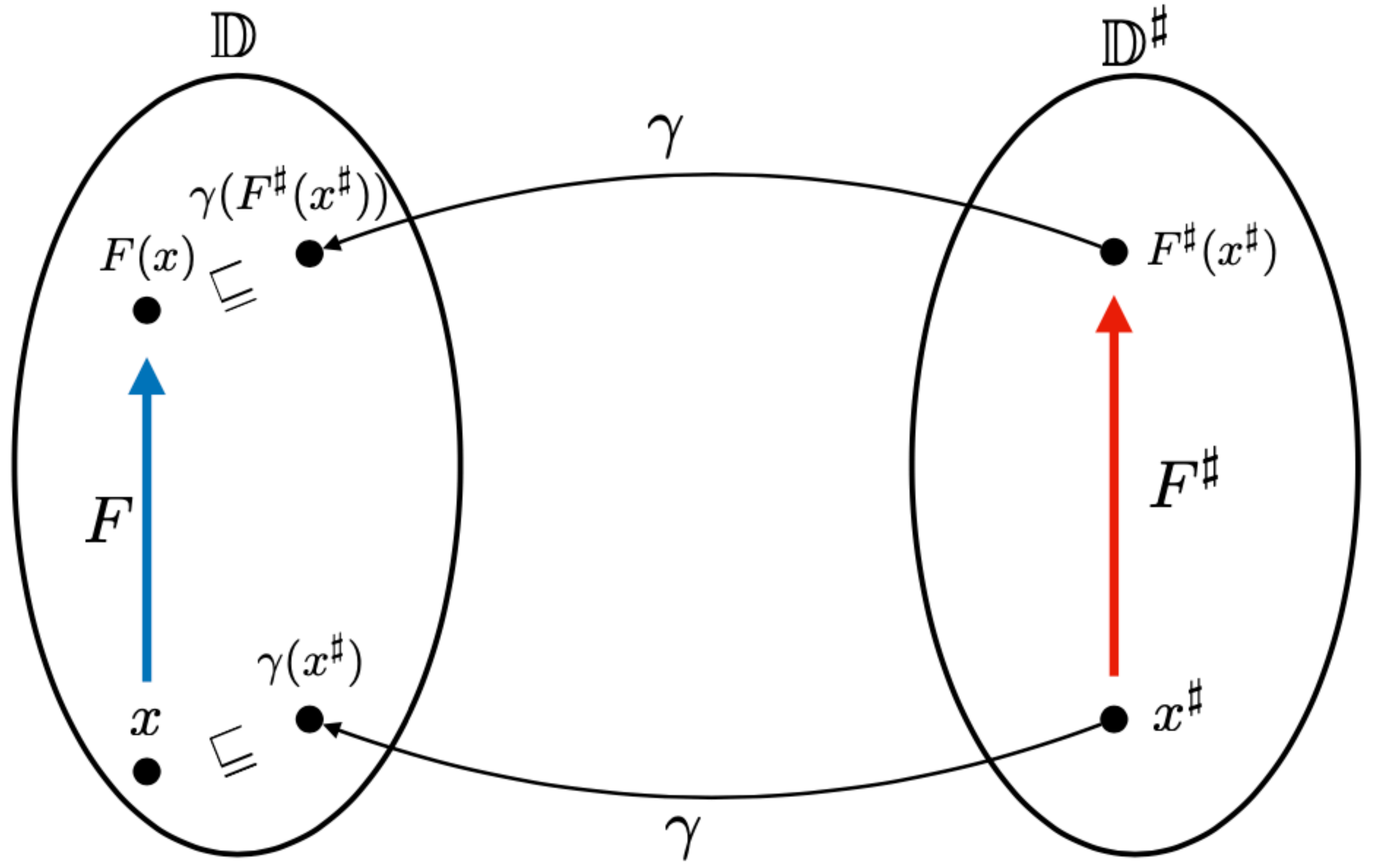
$$F \circ \gamma \sqsubseteq \gamma \circ F^\sharp$$



Intuition: the result of one-step abstract execution ($F^\sharp$) subsumes that of one-step concrete execution (F)

Sound Abstract Semantics (2)

$$x \sqsubseteq \gamma(x^\sharp) \implies F(x) \sqsubseteq \gamma(F^\sharp(x^\sharp))$$



Intuition: the result of one-step abstract execution ($F^\sharp$) subsumes that of one-step concrete execution (F)

Soundness

- Static analysis result $\bigsqcup_{i \geq 0} F^{\sharp i}(\bot)$ soundly subsumes all possible executions

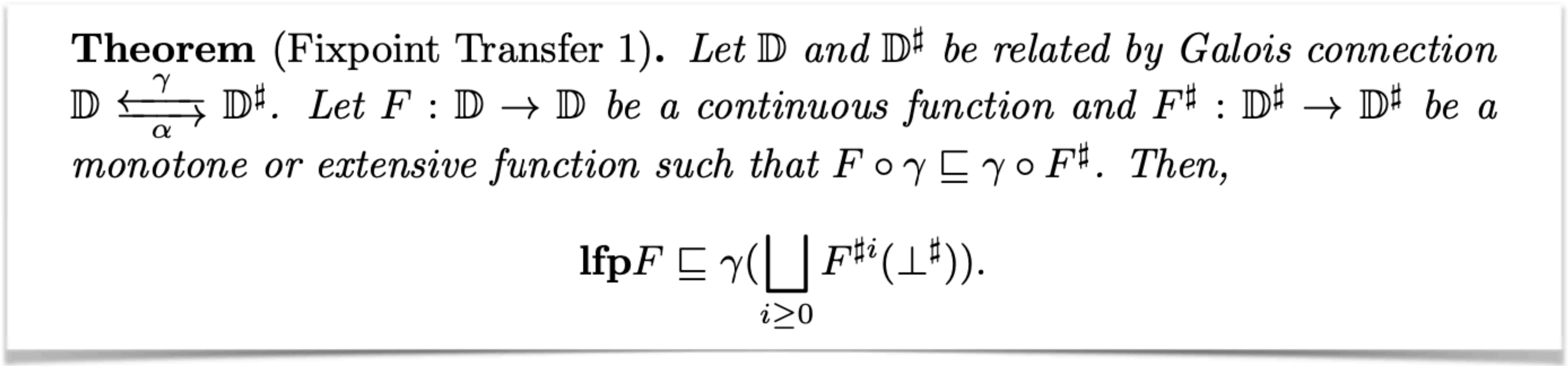
$$\mathbf{lfp}F \sqsubseteq \gamma(\bigsqcup_{i \geq 0} F^{\sharp i}(\bot))$$

- How to guarantee the soundness?

- How to compute the sound result within finite time?

Fixpoint Transfer Theorems

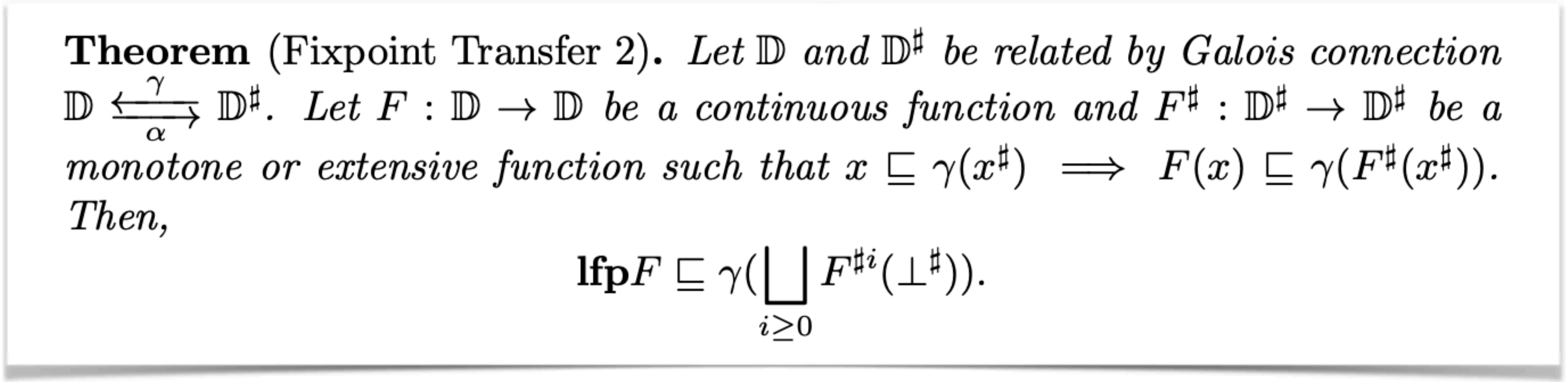
- With option 1



Theorem (Fixpoint Transfer 1). *Let $\mathbb{D}$ and $\mathbb{D}^\sharp$ be related by Galois connection $\mathbb{D} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{D}^\sharp$. Let $F : \mathbb{D} \to \mathbb{D}$ be a continuous function and $F^\sharp : \mathbb{D}^\sharp \to \mathbb{D}^\sharp$ be a monotone or extensive function such that $F \circ \gamma \sqsubseteq \gamma \circ F^\sharp$. Then,*

$$\mathbf{lfp} F \sqsubseteq \gamma(\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^\sharp)).$$

- With option 2



Theorem (Fixpoint Transfer 2). *Let $\mathbb{D}$ and $\mathbb{D}^\sharp$ be related by Galois connection $\mathbb{D} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{D}^\sharp$. Let $F : \mathbb{D} \to \mathbb{D}$ be a continuous function and $F^\sharp : \mathbb{D}^\sharp \to \mathbb{D}^\sharp$ be a monotone or extensive function such that $x \sqsubseteq \gamma(x^\sharp) \implies F(x) \sqsubseteq \gamma(F^\sharp(x^\sharp))$. Then,*

$$\mathbf{lfp} F \sqsubseteq \gamma(\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^\sharp)).$$

Design of Static Analysis

- Goal: conservative and terminating static analysis
- Design principles:
    - Define concrete semantics
    - Define abstract semantics (sound w.r.t the concrete semantics)
- Computation & implementation:
    - Abstract semantics of a program: the least fixed point of the semantic function
    - Static analyzer: compute the least fixed point within finite time

Computing Abstract Semantics

- If the abstract domain $\mathbb{D}^\sharp$ has **finite** height (i.e., all chains are finite)

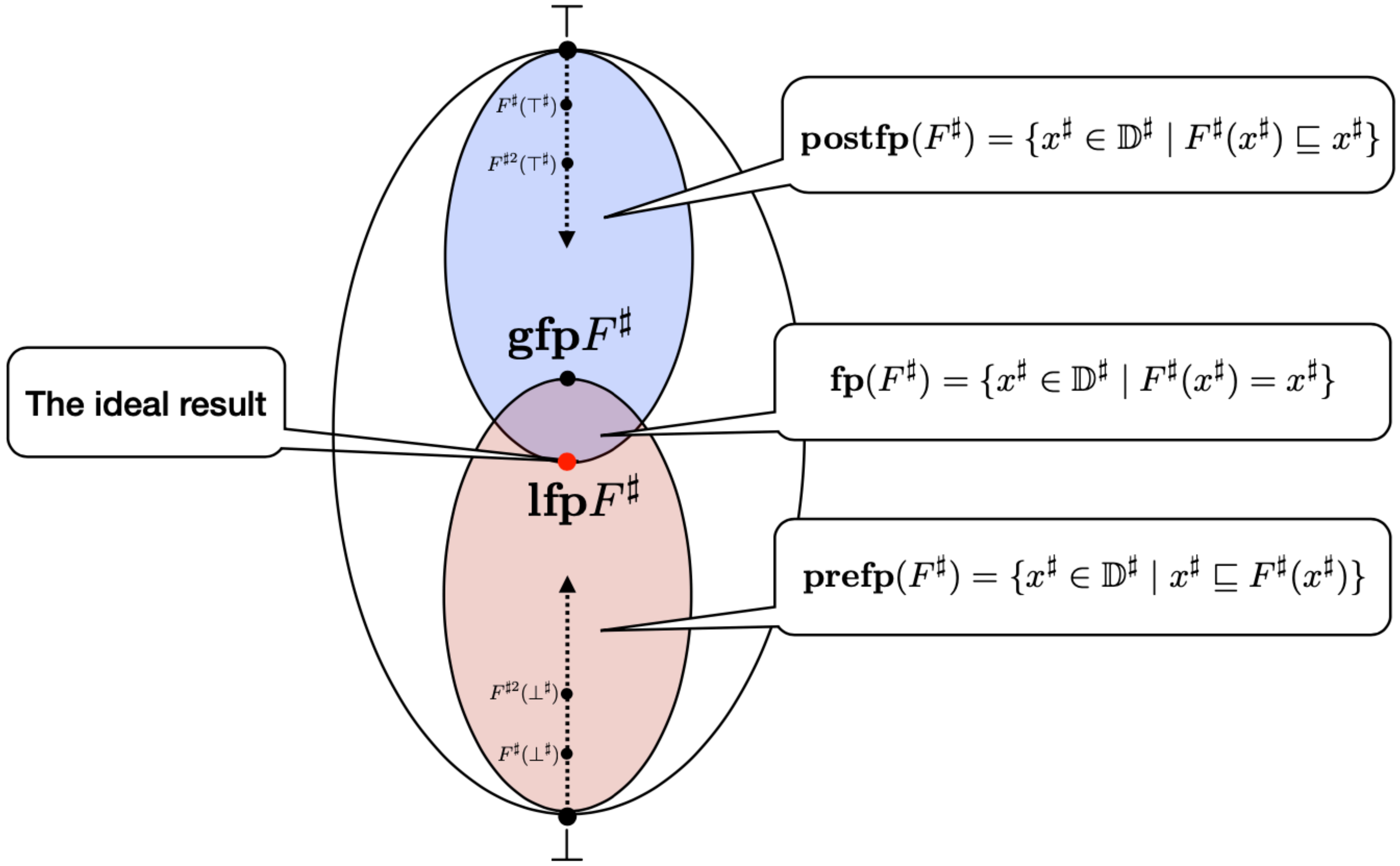
$$\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^\sharp)$$

- If the abstract domain $\mathbb{D}^\sharp$ has **infinite** height, we compute a finite chain $Y_0^\sharp \sqsubseteq Y_1^\sharp \sqsubseteq Y_2^\sharp \sqsubseteq \ldots \sqsubseteq Y_{\text{lim}}^\sharp$ such that

$$\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^\sharp) \sqsubseteq Y_{\text{lim}}^\sharp$$

where $F^\sharp$ is monotone

Fixed Points

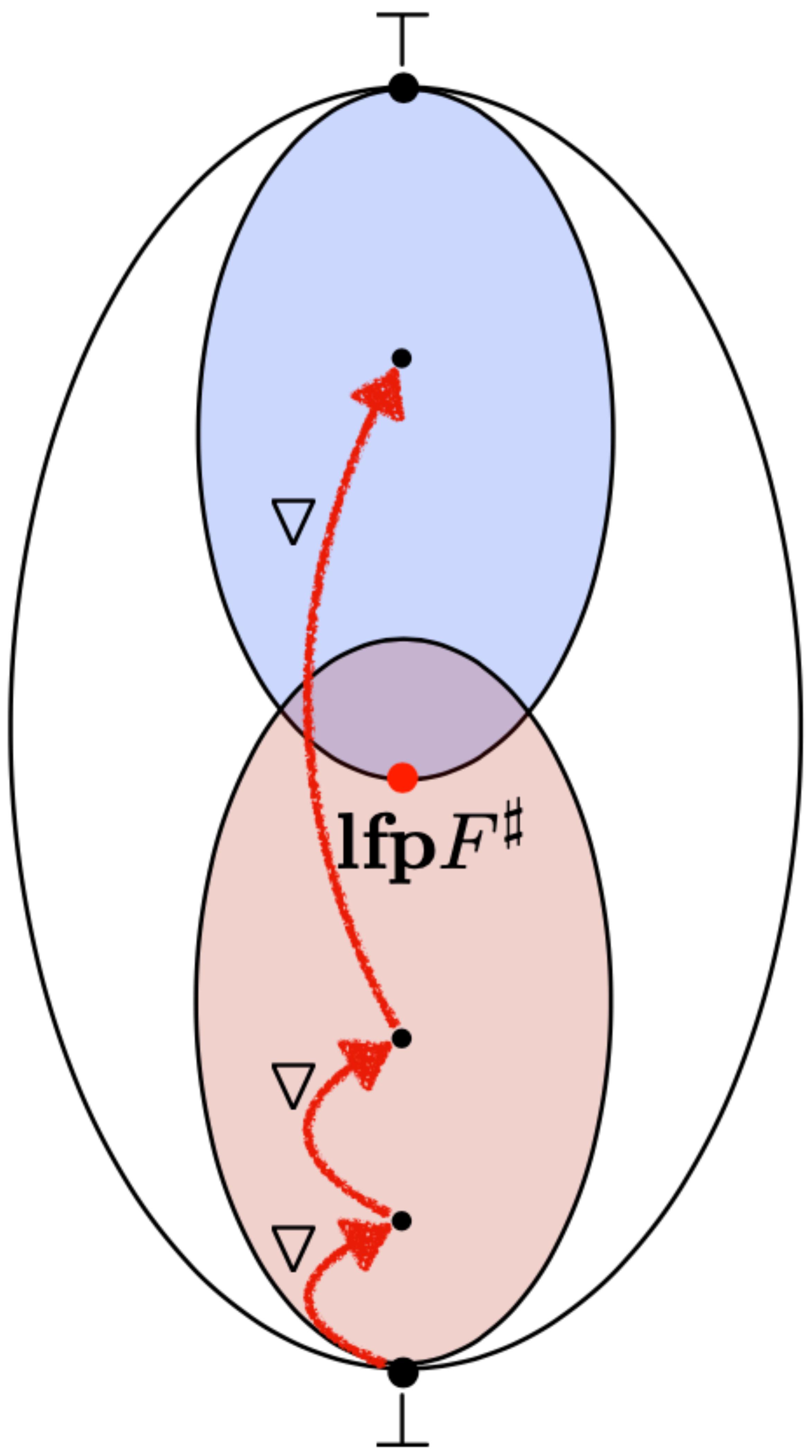


$\mathbf{postfp}(F^\sharp) = \{x^\sharp \in \mathbb{D}^\sharp \mid F^\sharp(x^\sharp) \sqsubseteq x^\sharp\}$

$\mathbf{fp}(F^\sharp) = \{x^\sharp \in \mathbb{D}^\sharp \mid F^\sharp(x^\sharp) = x^\sharp\}$

$\mathbf{prefp}(F^\sharp) = \{x^\sharp \in \mathbb{D}^\sharp \mid x^\sharp \sqsubseteq F^\sharp(x^\sharp)\}$

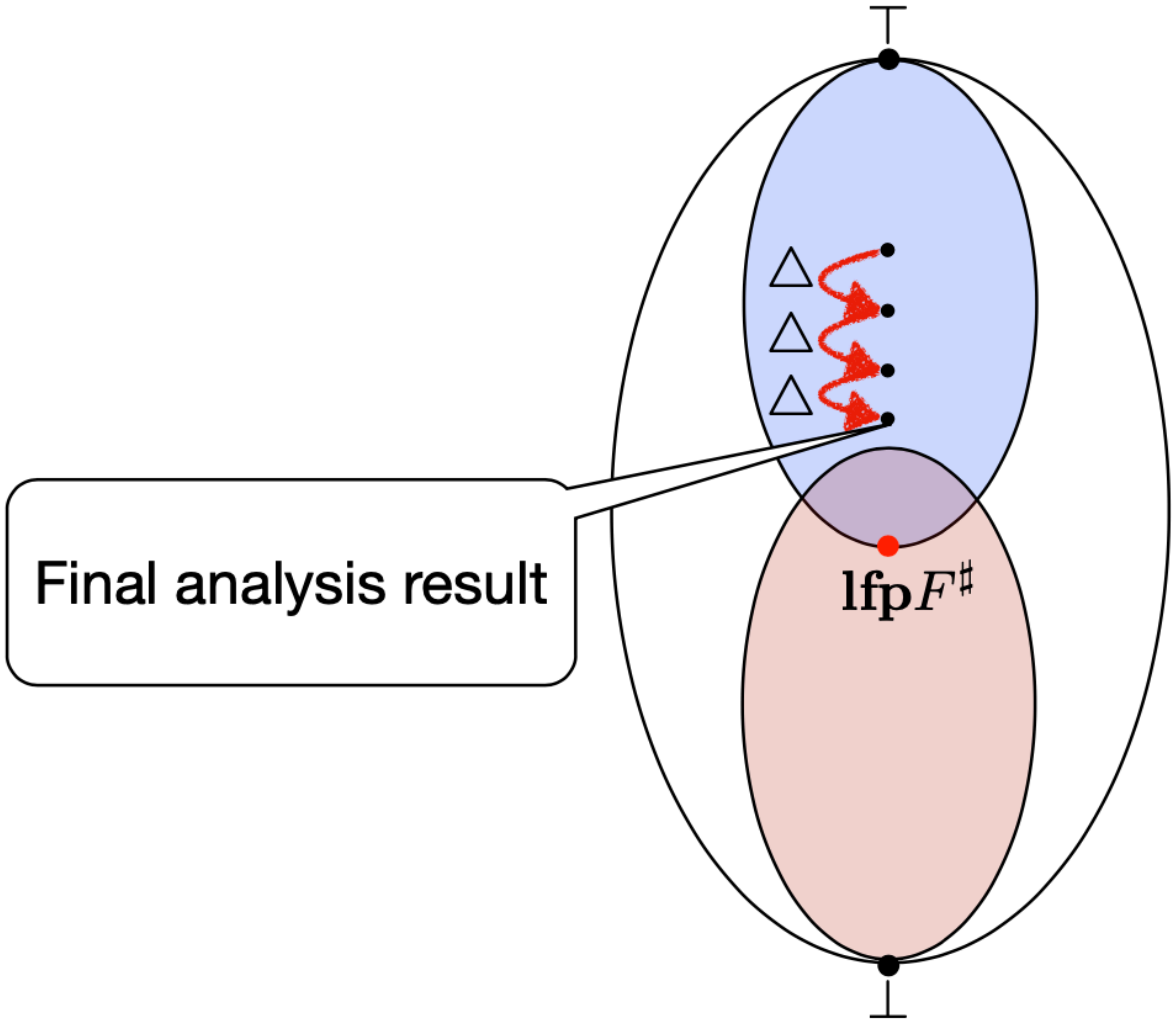
Widening



$$\nabla : \mathbb{D}^\sharp \times \mathbb{D}^\sharp \to \mathbb{D}^\sharp$$

Widening: enforcing the convergence of fix point iterations

Narrowing



$$\triangle : \mathbb{D}^\sharp \times \mathbb{D}^\sharp \to \mathbb{D}^\sharp$$

Narrowing: refining the analysis results with widening

Overshooting by Widening

$$\bigsqcup_{i\geq 0} F^{\sharp i}(\bot^\sharp) \sqsubseteq Y^\sharp_{\lim}$$

- Define finite chain $\{Y^\sharp_i\}_i$ by an widening operator $\triangledown \in \mathbb{D}^\sharp \times \mathbb{D}^\sharp \to \mathbb{D}^\sharp$:

$$\begin{array}{rcl} Y^\sharp_0 & = & \bot^\sharp \\ Y^\sharp_{i+1} & = & \begin{cases} Y^\sharp_i & \text{if } F^\sharp(Y^\sharp_i) \sqsubseteq Y^\sharp_i \\ Y^\sharp_i \;\triangledown\; F^\sharp(Y^\sharp_i) & \text{otherwise} \end{cases} \end{array}$$

Finite Increasing Chain with Widening

$$\begin{tikzcd}
\vdots \arrow[d, dash] \\
F^{\sharp 2}(\perp^\sharp) \arrow[d, dash] \\
F^{\sharp}(\perp^\sharp) \arrow[d, dash] \\
\perp^\sharp
\end{tikzcd}$$

Original Chain

$$\begin{tikzcd}
Y^\sharp_{N+1} = Y^\sharp_N \arrow[d, dash] \\
Y^\sharp_N = Y^\sharp_{N-1} \ \triangledown\ F^\sharp(Y^\sharp_{N-1}) \\
\vdots \\
Y^\sharp_2 = Y^\sharp_1 \ \triangledown\ F^\sharp(Y^\sharp_1) \arrow[d, dash] \\
Y^\sharp_1 = Y^\sharp_0 \ \triangledown\ F^\sharp(Y^\sharp_0) \arrow[d, dash] \\
Y^\sharp_0 = \perp^\sharp
\end{tikzcd}$$

New Chain with Widening

Q. What conditions are required to ensure

$$\bigsqcup_{i \geq 0} F^{\sharp i}(\perp^\sharp) \sqsubseteq Y^\sharp_{\lim}$$

Safety of Widening Operator

- Conditions on widening operator:
  - $\forall a, b \in \mathbb{D}^\sharp.\ (a \sqsubseteq a \triangledown b) \ \wedge \ (b \sqsubseteq a \triangledown b)$
  - $\forall$increasing chain $\{x_i\}_i$, the following increasing chain $\{y_i\}_i$ is finite:

$$y_0 = x_0$$
$$y_{i+1} = y_i \triangledown x_{i+1}$$

- Then,
  - Chain $\{Y_i^\sharp\}_i$ is finite
  - $\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^\sharp) \sqsubseteq Y^\sharp_{\text{lim}}$

Refinement by Narrowing

$$\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^{\sharp}) \sqsubseteq Z^{\sharp}_{\text{lim}}$$

- Define finite chain $\{Z^{\sharp}_i\}_i$ by an narrowing operator $\triangle \in \mathbb{D}^{\sharp} \times \mathbb{D}^{\sharp} \to \mathbb{D}^{\sharp}$:

$$\begin{array}{rcl} Z^{\sharp}_0 & = & Y^{\sharp}_{\text{lim}} \\ Z^{\sharp}_{i+1} & = & Z^{\sharp}_i \ \triangle \ F^{\sharp}(Z^{\sharp}_i) \end{array}$$

Finite Decreasing Chain with Narrowing

$$\begin{tikzcd}
Y^\sharp_{\text{lim}} \arrow[d, dash] \\
F^\sharp(Y^\sharp_{\text{lim}}) \arrow[d, dash] \\
F^{\sharp 2}(Y^\sharp_{\text{lim}}) \\
\vdots
\end{tikzcd}$$

Original Chain

$$\begin{tikzcd}
Z^\sharp_0 = Y^\sharp_{\text{lim}} \arrow[d, dash] \\
Z^\sharp_1 = Z^\sharp_0 \bigtriangleup F^\sharp(Z^\sharp_0) \arrow[d, dash] \\
Z^\sharp_2 = Z^\sharp_1 \bigtriangleup F^\sharp(Z^\sharp_1) \\
\vdots \\
Z^\sharp_N = Z^\sharp_{N-1} \bigtriangleup F^\sharp(Z^\sharp_{N-1}) \arrow[d, dash] \\
Z^\sharp_{N+1} = Z^\sharp_N
\end{tikzcd}$$

New Chain with Narrowing

Q. What conditions are required to ensure

$$\bigsqcup_{i \geq 0} F^{\sharp i}(\perp^\sharp) \sqsubseteq Z^\sharp_{\text{lim}}$$

Safety of Narrowing Operator

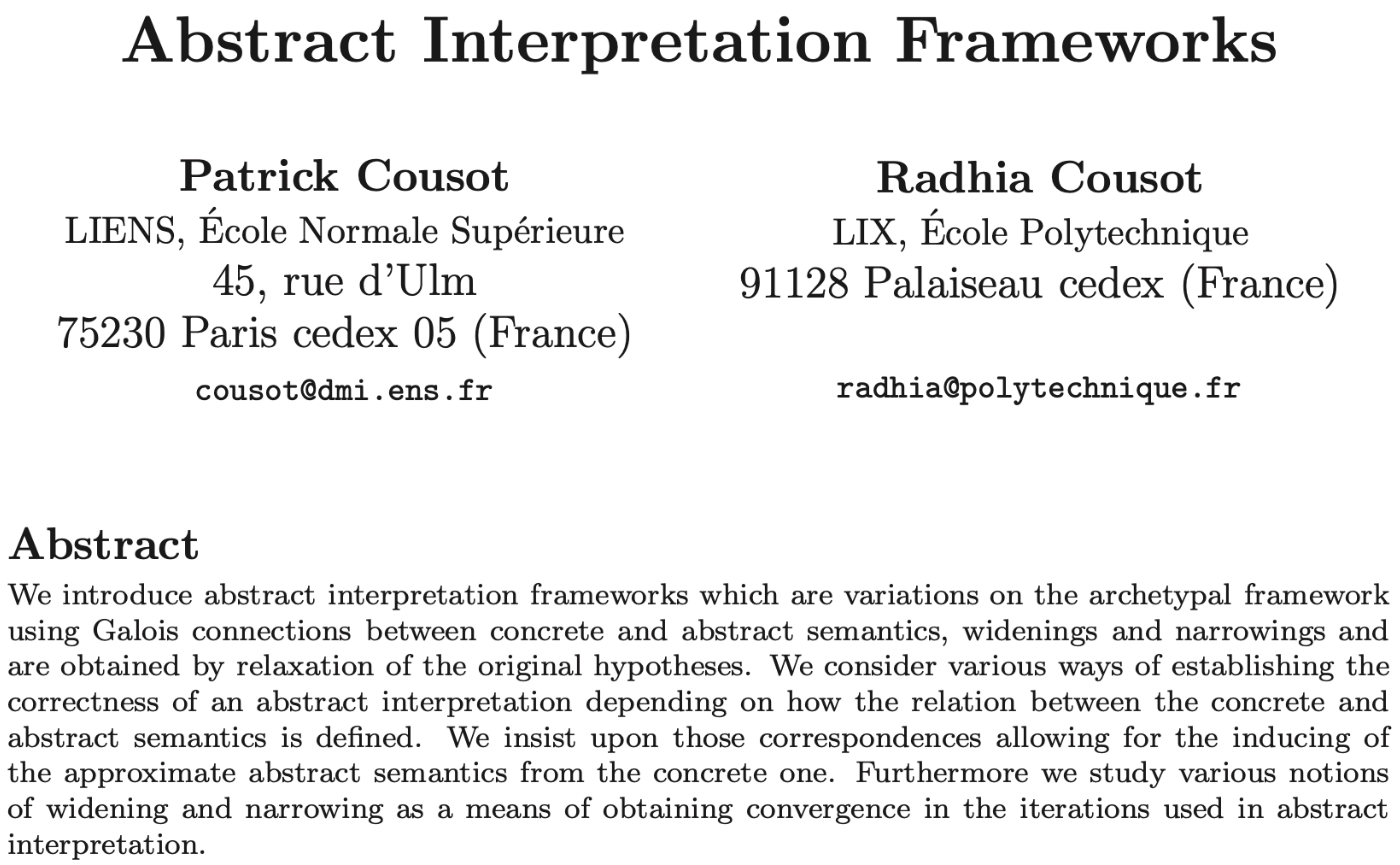
- Conditions on narrowing operator:
  - $\forall a, b \in \mathbb{D}^\sharp.\ a \sqsupseteq b \implies a \sqsupseteq (a \bigtriangleup b) \sqsupseteq b$
  - For all decreasing chain $\{x_i\}_i$, the following decreasing chain $\{y_i\}_i$ is finite

$$\begin{aligned} y_0 &= x_0 \\ y_{i+1} &= y_i \bigtriangleup x_{i+1} \end{aligned}$$

- Then,
  - Decreasing chain $\{Z_i^\sharp\}_i$ is finite
  - $\bigsqcup_{i \geq 0} F^{\sharp i}(\bot^\sharp) \sqsubseteq Z_{\lim}$

Summary

- Abstract interpretation: a **framework** for designing correct static analysis
- Galois connection, sound abstract semantic function: **soundness** guarantee
  - Fixpoint transfer theorem
- Widening: **termination** guarantee
- Narrowing: **refinement** of widening results



Abstract Interpretation Frameworks

Patrick Cousot
LIENS, École Normale Supérieure
45, rue d'Ulm
75230 Paris cedex 05 (France)
`cousot@dmi.ens.fr`

Radhia Cousot
LIX, École Polytechnique
91128 Palaiseau cedex (France)
`radhia@polytechnique.fr`

Abstract

We introduce abstract interpretation frameworks which are variations on the archetypal framework using Galois connections between concrete and abstract semantics, widenings and narrowings and are obtained by relaxation of the original hypotheses. We consider various ways of establishing the correctness of an abstract interpretation depending on how the relation between the concrete and abstract semantics is defined. We insist upon those correspondences allowing for the inducing of the approximate abstract semantics from the concrete one. Furthermore we study various notions of widening and narrowing as a means of obtaining convergence in the iterations used in abstract interpretation.

- Patrick Cousot & Radhia Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation,* 2(4):511—547, August 1992.

Concepts	
concrete	abstract
concretization	abstraction
more precise	less precise
is approximated by	approximates
widening	narrowing
minimal	maximal
monotonic	monotonic
increasing	decreasing
least fixpoint	greatest fixpoint
reductive	extensive

Galois connections	
$\langle S;\ \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle S';\ \sqsubseteq'\rangle$	$\langle S';\ \sqsupseteq'\rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle S;\ \sqsupseteq\rangle$

Sets	
$\mathcal{P}^\natural$	$\mathcal{P}^\sharp$
C	A

Set elements	
c	a
$\underline{\bot}^\natural$	$\underline{\bot}^\sharp$
$\bot^\natural$	$\top^\sharp$
$\top^\natural$	$\bot^\sharp$
$\mathit{lfp}_{\bot^\natural}^{\sqsubseteq^\natural} F^\natural$	$\mathit{gfp}_{\top^\sharp}^{\sqsubseteq^\sharp} F^\sharp$

Functions	
α	γ
$F^\natural$	$F^\sharp$
$\amalg^\natural$	$\amalg^\sharp$

$\triangledown^\natural$	$\vartriangle^\sharp$
$\vartriangle^\natural$	$\triangledown^\sharp$
$\sqcup^\natural$	$\sqcap^\sharp$
$\sqcap^\natural$	$\sqcup^\sharp$

Relations	
ρ	ρ^{-1}
σ	σ^{-1}
α	γ
γ	α
$\preceq^\natural$	$\succeq^\sharp$
$\succeq^\natural$	$\preceq^\sharp$
$\approx^\natural$	$\approx^\sharp$
$\sqsubseteq^\natural$	$\sqsupseteq^\sharp$
$\sqsupseteq^\natural$	$\sqsubseteq^\sharp$

Figure 1: Dual abstract interpretations

9 In conclusion, which framework to use?

Starting from the definition of the concrete semantics $\langle \mathcal{P}^\natural;\ \bot^\natural, F^\natural, \amalg^\natural \rangle$ of programs, the design of an abstract interpretation consists in choosing:

1. an abstract semantic domain $\mathcal{P}^\sharp$ which is an approximate version of the concrete semantic domain $\mathcal{P}^\natural$; and
2. a method for defining an abstract semantics $\langle \bot^\sharp, F^\sharp, \amalg^\sharp \rangle$ of programs; and
3. the specification of the soundness correspondence between the concrete and abstract properties; and
4. a convergence criterion of the abstract iteration sequence, ensuring the best possible precision; and
5. a convergence acceleration method ensuring rapid termination of the abstract interpreter.

We have discussed several abstract interpretation frameworks, obtained by weakening the hypotheses made in [6, 7, 10, 12, 19]. Each one has many variants, most of them have not been explicitly formulated for short (such as for example the use of an abstraction function together with a concrete approximation relation). To simplify, the principal alternatives are:

1. using a soundness relation σ; or
2. using an abstraction relation $\alpha \in \wp(\mathcal{P}^\natural \times \mathcal{P}^\sharp)$ and an abstract approximation relation $\preceq^\sharp$ so that the soundness relation is $\langle c,\ a\rangle \in \sigma \Leftrightarrow \exists a' \in \mathcal{P}^\sharp : \langle c,\ a'\rangle \in \alpha \wedge a' \preceq^\sharp a$; or
3. using a concretization relation $\gamma \in \wp(\mathcal{P}^\sharp \times \mathcal{P}^\natural)$ and a concrete approximation relation $\preceq^\natural$ so that the soundness relation is $\langle c,\ a\rangle \in \sigma \Leftrightarrow \exists c' \in \mathcal{P}^\natural : c \preceq^\natural c' \wedge \langle a,\ c'\rangle \in \gamma$; or
4. using an abstraction function $\alpha \in \mathcal{P}^\natural \mapsto \mathcal{P}^\sharp$ and an abstract approximation relation $\preceq^\sharp$ so that the soundness relation is $\langle c,\ a\rangle \in \sigma \ \Leftrightarrow\ \alpha(c) \preceq^\sharp a$; or
5. using a concretization function $\gamma \in \mathcal{P}^\sharp \mapsto \mathcal{P}^\natural$ and a concrete approximation relation $\preceq^\natural$ so that the soundness relation is $\langle c,\ a\rangle \in \sigma \ \Leftrightarrow\ c \preceq^\natural \gamma(a)$; or
6. using an Galois connection $\langle \mathcal{P}^\natural;\ \preceq^\natural\rangle \ \underset{\alpha}{\overset{\gamma}{\leftrightarrows}}\ \langle \mathcal{P}^\sharp;\ \preceq^\sharp\rangle$ so that the soundness relation is $\langle c,\ a\rangle \in \sigma \ \Leftrightarrow\ \alpha(c) \preceq^\sharp a \ \Leftrightarrow\ c \preceq^\natural \gamma(a)$.

For a given application, the more powerful applicable framework should be chosen so as to benefit from the best possible guidelines for designing that application. This choice should be guided by the following principles:

1. Preference should be given to the inducing of the abstract interpretation from the starting semantics over empirical designs followed by *a posteriori* soundness verifications; and

2. Efficiency of the implementation should be taken into account during the design of the abstract interpretation (that is in the choice of $\mathcal{P}^\sharp$ but, in addition, in that of $F^\sharp$, $\triangledown^\sharp$ and $\vartriangle^\sharp$).

Numerous abstract interpretation frameworks exist and many more are to come in order to take into account the peculiarities of each practical situation. We give our preference to language and semantics independent formulations and hope that this will lead to a cross-fertilization of the various domains of application of abstract interpretation.

[OPLSS: Static Analysis]

Static Analysis for JavaScript Specification

Sukyoung Ryu

with PLRG@KAIST and friends

July 4, 2023

Intermediate Representation for ECMAScript

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

Semantic Domains

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	$s \in \mathbb{V}_s$	

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

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

- 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

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

Semantics of Expressions

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

- Records:

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

where

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

- Lists:

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

where

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

- Type Checks:

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

where

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

- Variable Existence Checks:

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

where

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

- Field Existence Checks:

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

where

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

- 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:

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

Abstract Semantic Domains

Abstract States	$d^\sharp \in \mathbb{S}^\sharp = \mathbb{M} \times \mathbb{R}$
Result Maps	$m \in \mathbb{M} = \mathbb{L} \times \mathbb{T}^* \to \mathbb{E}^\sharp$
Return Point Maps	$r \in \mathbb{R} = \mathbb{F} \times \mathbb{T}^* \to \mathcal{P}(\mathbb{L} \times \mathbb{T}^* \times \mathbb{X})$
Abstract Environments	$\sigma^\sharp \in \mathbb{E}^\sharp = \mathbb{X} \to \mathbb{T}^\sharp$
Abstract Types	$\tau^\sharp \in \mathbb{T}^\sharp = \mathcal{P}(\mathbb{T})$

Abstract Semantic Functions

Then, we define the abstract semantics $[\![P]\!]^\sharp$ of a program P as the least fixpoint of the abstract transfer $F^\sharp : \mathbb{S}^\sharp \to \mathbb{S}^\sharp$:

$$
\begin{aligned}
[\![P]\!]^\sharp &= \lim_{n\to\infty} (F^\sharp)^n(d^\sharp_\iota) \\
F^\sharp(d^\sharp) &= d^\sharp \sqcup \left(\bigsqcup_{(\ell,\overline{\tau}) \in \text{Domain}(m)} [\![\texttt{inst}(\ell)]\!]^\sharp_i(\ell,\overline{\tau})(d^\sharp) \right)
\end{aligned}
$$

where $d^\sharp = (m, \_)$ and $d^\sharp_\iota$ denotes the initial abstract state.

B. Instructions: $[\![i]\!]^\sharp_i : (\mathbb{L} \times \mathbb{T}^*) \to \mathbb{S}^\sharp \to \mathbb{S}^\sharp$

- Variable Declarations:

$$
[\![\texttt{let x} = e]\!]^\sharp_i(\ell,\overline{\tau})(d^\sharp) = (\{(\texttt{next}(\ell),\overline{\tau}) \mapsto \sigma^\sharp_{\mathrm{x}}\}, \varnothing)
$$

where

$$
\begin{aligned}
&d^\sharp = (m, \_) \wedge \sigma^\sharp = m(\ell,\overline{\tau}) \wedge \\
&\sigma^\sharp_{\mathrm{x}} = \sigma^\sharp[\mathrm{x} \mapsto [\![e]\!]^\sharp_e(\sigma^\sharp)]
\end{aligned}
$$