

Introduction to Program Analysis

8. Design and Implementation of Static Analysis

Kihong Heo



Abstract Interpretation of SmaLLVM

- Let us design a **sound** static analysis of SmaLLVM!
 - Transitional style (i.e., operational semantics)
- Road map:
 - Concrete semantics: a set of **reachable states**
 - Abstract semantics: an abstract memory with the **interval** domain **per each label**
 - With widening and narrowing

The SmaLLVM Language



- A program is a tuple $\langle \mathbb{L}, \text{next} \rangle$ where
 - \mathbb{L} is a set of program labels
 - next control-flow relation
- Each program label l is associated with a command denoted by $\text{cmd}(l)$

$L ::=$	l	program label
$E ::=$	n true false	integer boolean
	x	variable
	$E \oplus E$	arithmetic operation
	$E \otimes E$	comparison operation
$\oplus ::=$	$+$ $-$ \times $/$	
$\otimes ::=$	$<$ \leq $>$ \geq $==$ $!=$	
$C ::=$	$x := E$ br $E L L$ goto L $x := \text{input}()$ print(x)	assignment conditional jump unconditional jump input print

Semantic Domains

- The same as before
- For simplicity, boolean values are considered as 0 (false) or 1 (true)

$$\begin{array}{llll} \langle l, m \rangle & \in & \mathbb{S} & = \mathbb{L} \times \mathbb{M} \\ l & \in & \mathbb{L} & \text{program label} \\ m & \in & \mathbb{M} & = \mathbb{X} \rightarrow \mathbb{V} \\ x & \in & \mathbb{X} & \text{variable} \\ & & \mathbb{V} & = \mathbb{Z} \\ n & \in & \mathbb{Z} & \text{integer} \end{array}$$

State Transition

- The semantics is specified by a transition system $(\mathbb{S}, \rightarrow)$
 - $\mathbb{S} = \mathbb{L} \times \mathbb{M}$: the set of states $\langle l, m \rangle$
 - $(\rightarrow) \subseteq \mathbb{S} \times \mathbb{S}$: the transition relation that describes computation steps

$x := E : \langle l, m \rangle \hookrightarrow \langle \text{next}(l), \text{update}_x(m, \text{eval}_E(m)) \rangle$

$\text{br } E \ l_1 \ l_2 : \langle l, m \rangle \hookrightarrow \langle l_1, \text{filter}_E(m) \rangle$

$: \langle l, m \rangle \hookrightarrow \langle l_2, \text{filter}_{\neg E}(m) \rangle$

$\text{goto } l' : \langle l, m \rangle \hookrightarrow \langle l', m \rangle$

$x := \text{input}() : \langle l, m \rangle \hookrightarrow \langle \text{next}(l), \text{update}_x(m, z) \rangle \quad \text{for an input integer } z$

$x := E : \langle l, m \rangle \hookrightarrow \langle \text{next}(l), \text{update}_x(m, \text{eval}_E(m)) \rangle$

Semantic Operators

- The memory update operation

$$\begin{aligned} update_x : \mathbb{M} \times \mathbb{V} &\rightarrow \mathbb{M} \\ update_x(m, n) &= m\{x \mapsto n\} \end{aligned}$$

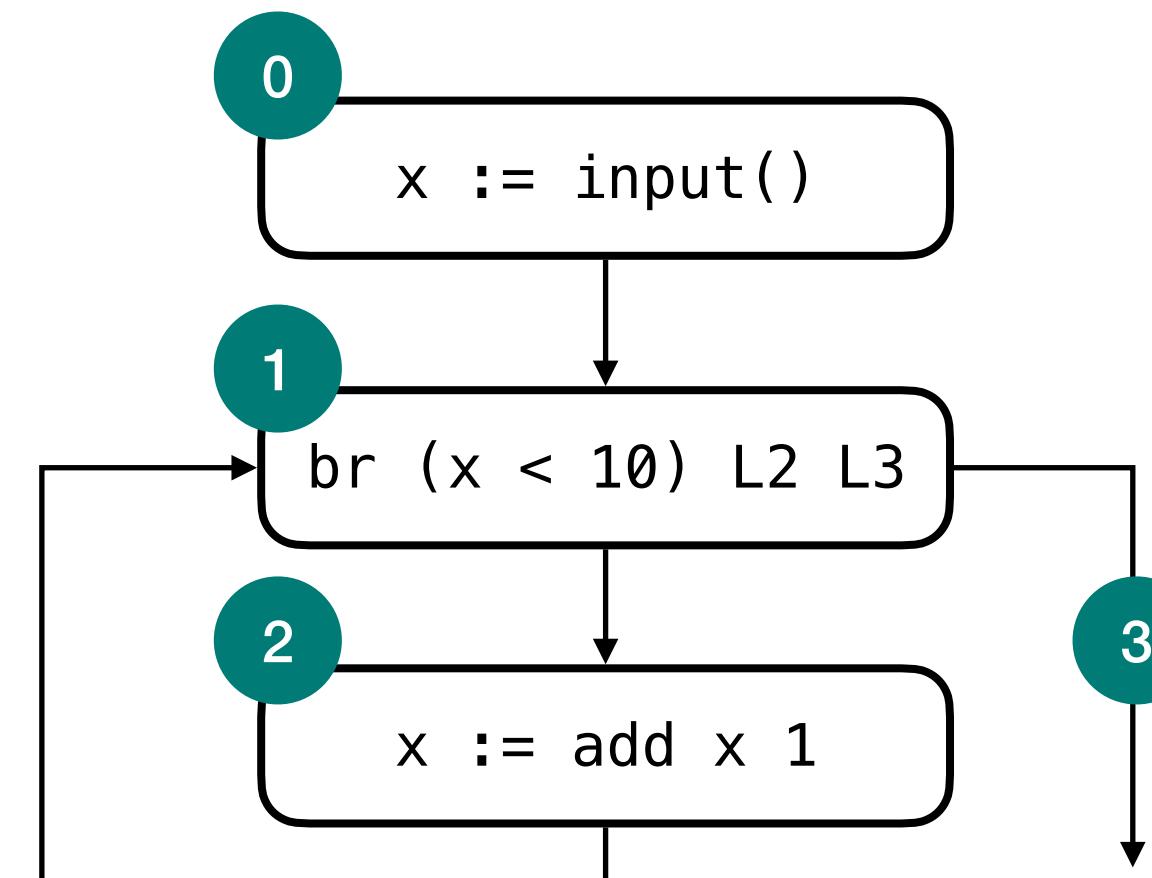
- The expression-evaluation operation

$$\begin{aligned} eval_E : \mathbb{M} &\rightarrow \mathbb{V} \\ eval_n(m) &= n \\ eval_x(m) &= m(x) \\ eval_{E_1 \oplus E_2}(m) &= eval_{E_1}(m) \oplus eval_{E_2}(m) \end{aligned}$$

- The memory filter operation

$$\begin{aligned} filter_E : \mathbb{M} &\rightarrow \mathbb{M} \\ filter_E(m) &= m \quad \text{if } eval_E(m) = \text{true} \end{aligned}$$

Set of Reachable States



...

For input 10 : $\langle 0, \emptyset \rangle \hookrightarrow \langle 1, x \mapsto 10 \rangle \hookrightarrow \langle 3, x \mapsto 10 \rangle$

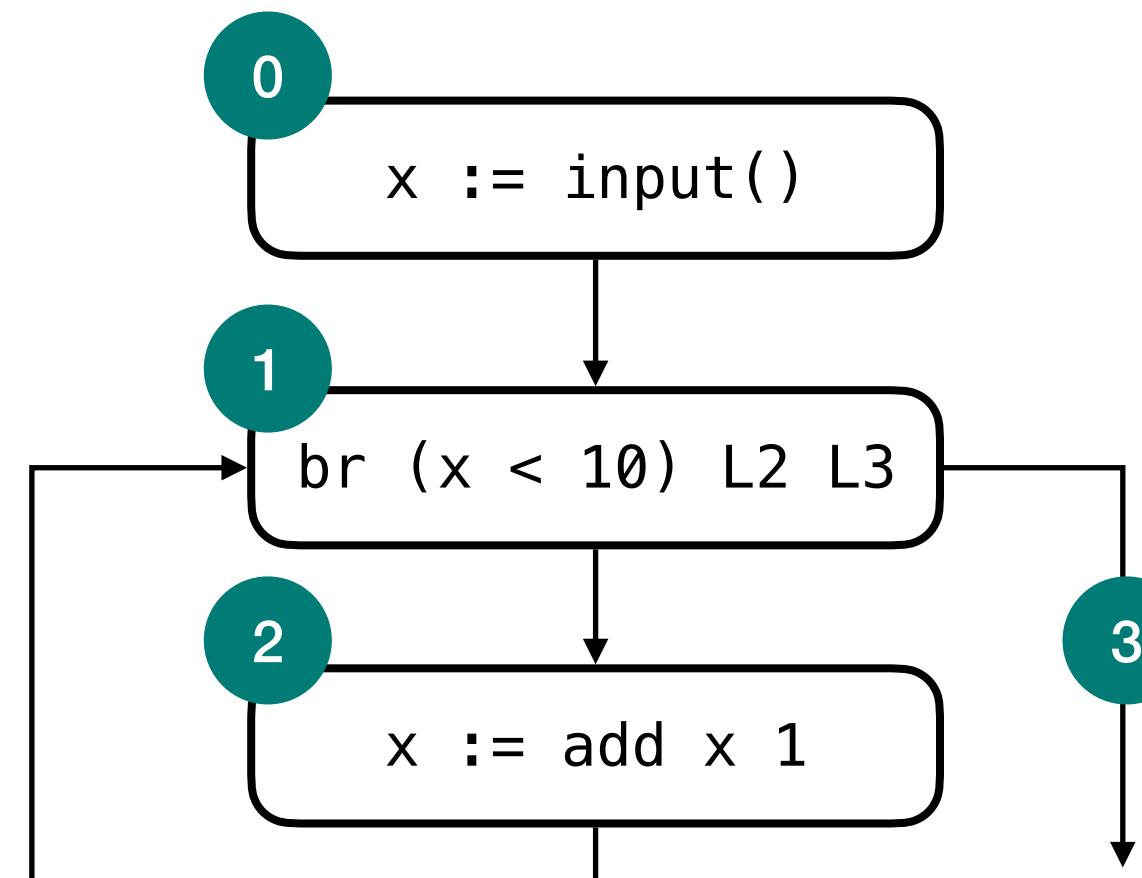
For input 9 : $\langle 0, \emptyset \rangle \hookrightarrow \langle 1, x \mapsto 9 \rangle \hookrightarrow \langle 2, x \mapsto 9 \rangle \hookrightarrow \langle 1, x \mapsto 10 \rangle \hookrightarrow \langle 3, x \mapsto 10 \rangle$

...

For input 0 : $\langle 0, \emptyset \rangle \hookrightarrow \langle 1, x \mapsto 0 \rangle \hookrightarrow \langle 2, x \mapsto 0 \rangle \hookrightarrow \langle 1, x \mapsto 1 \rangle \hookrightarrow \dots \hookrightarrow \langle 3, x \mapsto 10 \rangle$

...

Set of Reachable States



Reachable states = ...

$$\begin{aligned} &\cup \{\langle 0, \emptyset \rangle, \langle 1, x \mapsto 10 \rangle, \langle 2, x \mapsto 10 \rangle, \langle 3, x \mapsto 10 \rangle\} \\ &\cup \{\langle 0, \emptyset \rangle, \langle 1, x \mapsto 9 \rangle, \langle 2, x \mapsto 9 \rangle, \langle 1, x \mapsto 10 \rangle, \langle 3, x \mapsto 10 \rangle\} \\ &\cup \dots \\ &\cup \{\langle 0, \emptyset \rangle, \langle 1, x \mapsto 0 \rangle, \langle 2, x \mapsto 0 \rangle, \langle 1, x \mapsto 1 \rangle, \dots, \langle 3, x \mapsto 10 \rangle\} \\ &\cup \dots \end{aligned}$$

Concrete Domain and Semantics

- Concrete domain: $\mathbb{D} = \wp(\mathbb{S})$ where $\mathbb{S} = \mathbb{L} \times \mathbb{M}$
- Concrete semantic function:

$$F : \wp(\mathbb{S}) \rightarrow \wp(\mathbb{S})$$

$$F(X) = I \cup Step(X)$$

where I is the set of initial states and $Step$ is the powerset-lifted version of \hookrightarrow

$$Step : \wp(\mathbb{S}) \rightarrow \wp(\mathbb{S})$$

$$Step(X) = \{s' \mid s \hookrightarrow s', s \in X\}$$

- Concrete semantic (i.e., reachable states): $\text{lfp } F$

Notations (1)

- An element of $A \rightarrow B$ is interchangeably an element in $\wp(A \times B)$
- A relation $f \subseteq A \times B$ is interchangeably a function $f \in A \rightarrow \wp(B)$: $f(a) = \{b \mid (a, b) \in f\}$
 - For example, $(\rightarrow) \subseteq \mathbb{S} \times \mathbb{S}$ is interchangeably a function $(\rightarrow) \in \mathbb{S} \rightarrow \wp(\mathbb{S})$
- For function $f : A \rightarrow B$, we write $\wp(f)$ is its powers version:

$$\wp(f) : \wp(A) \rightarrow \wp(B), \quad \wp(f)(X) = \{f(x) \mid x \in X\}$$

Notations (2)

- For function $f : A \rightarrow \wp(B)$, we write $\breve{\wp}(f)$ as a shorthand for $\cup \circ \wp(f)$:

$$\breve{\wp}(f) : \wp(A) \rightarrow \wp(B), \quad \breve{\wp}(f)(X) = \bigcup \{f(x) \mid x \in X\}$$

- For example, power-set-lifted function $Step : \wp(\mathbb{S}) \rightarrow \wp(\mathbb{S})$ of relation \hookrightarrow :

$$Step(X) = \{s' \mid s \hookrightarrow s', s \in X\}$$

is equivalently, by regarding \hookrightarrow as a function of $\mathbb{S} \rightarrow \wp(\mathbb{S})$:

$$Step(X) = \bigcup \{(\hookrightarrow)(s) \mid s \in X\} = \cup \circ \wp(\hookrightarrow)(X) = \breve{\wp}(\hookrightarrow)(X)$$

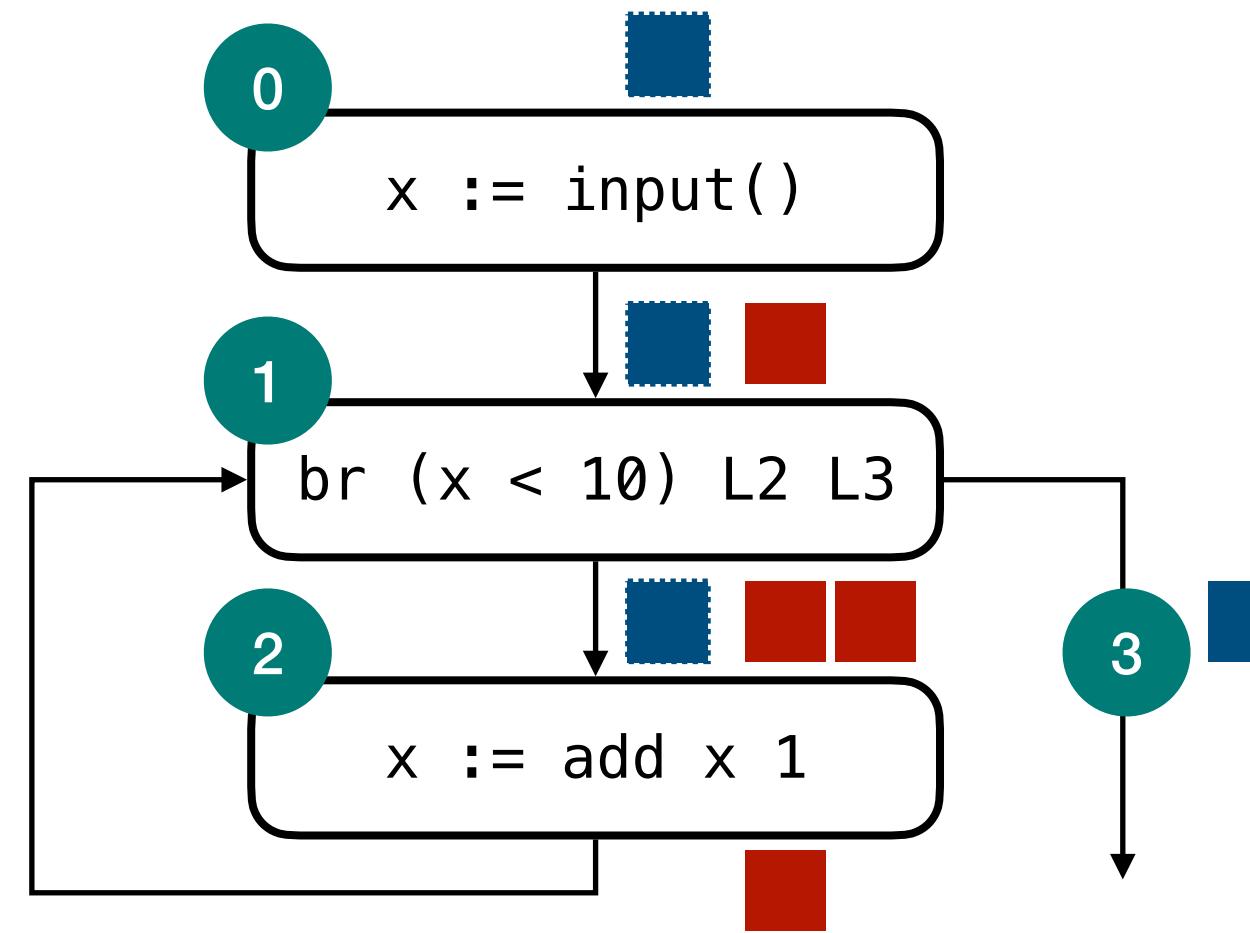
- For function $f : A \rightarrow B$ and $g : A' \rightarrow B'$, we write (f, g) for

$$(f, g) : A \times A' \rightarrow B \times B'$$

$$(f, g)(a, a') = (f(a), g(a'))$$

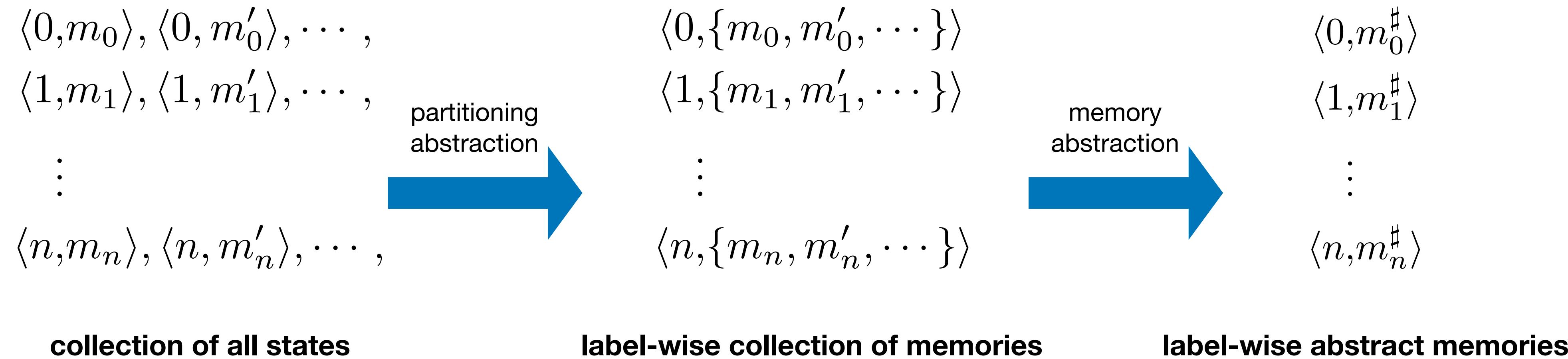
Informal Overview of Abstract Semantics

 : memory at each label
 : memory after transition



- One abstract memory per each label: $D^\sharp = \mathbb{L} \rightarrow M^\sharp$
- Abstract semantics = **transition** + **partitioning** + **join**

Decomposition of Abstraction



Abstraction 1: Partitioning

- In this lecture, program-label-wise collection of reachable memories
 - Collecting the reachable set of memories **for each** program label (so-called flow-sensitive)
 - C.f., **flow-insensitive**: collecting the reachable set of memories at **any** program label

$$\langle 0, \{m_0, m'_0, \dots\} \rangle$$
$$\langle 1, \{m_1, m'_1, \dots\} \rangle$$
$$\vdots$$
$$\langle n, \{m_n, m'_n, \dots\} \rangle$$

**label-wise collection
(i.e., flow-sensitive)**

$$\langle *, \{m_0, m'_0, \dots$$
$$m_1, m'_1, \dots$$
$$\vdots$$
$$m_n, m'_n, \dots\} \rangle$$

vs

**label-agnostic collection
(i.e., flow-insensitive)**

Abstract Domain

- Partitioned set of memories: $\mathbb{D}_1^\sharp = \mathbb{L} \rightarrow \wp(\mathbb{M})$

- Ordered via the point-wise order:

$$\Pi_1 \subseteq_1^\sharp \Pi_2 \iff \forall l \in \mathbb{L}. \Pi_1(l) \subseteq \Pi_2(l)$$

$$\cup_1^\sharp : (\mathbb{L} \rightarrow \wp(\mathbb{M})) \times (\mathbb{L} \rightarrow \wp(\mathbb{M})) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}))$$

$$\Pi_1 \cup_1^\sharp \Pi_2 = \lambda l. \Pi_1(l) \cup \Pi_2(l)$$

- Galois connection: $\wp(\mathbb{S}) \xrightleftharpoons[\alpha_1]{\gamma_1} \mathbb{L} \rightarrow \wp(\mathbb{M})$

$$\alpha_1(S) = \lambda l. \{m \in \mathbb{M} \mid \langle l, m \rangle \in S\}$$

$$\gamma_1(\Pi) = \{\langle l, m \rangle \mid m \in \Pi(l)\}$$

Abstract Semantics

- The abstract semantic function:

$$F_1^\sharp : (\mathbb{L} \rightarrow \wp(\mathbb{M})) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}))$$

$$F_1^\sharp(X) = \alpha_1(I) \cup_1^\sharp Step_1^\sharp(X)$$

- The abstract semantic functions for transition and partition:

$$\boxed{\begin{array}{ll} Step_1^\sharp : (\mathbb{L} \rightarrow \wp(\mathbb{M})) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M})) & \pi_1 : \wp(\mathbb{S}) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M})) \\ Step_1^\sharp(X) = \pi_1 \circ \check{\wp}(\hookrightarrow)(X) & \pi_1(X) = \lambda l. \{m \in \mathbb{M} \mid \langle l, m \rangle \in X\} \end{array}}$$

- Soundness: $\text{lfp } F \subseteq \gamma_1(\bigsqcup_{i \geq 0} F_1^{\sharp i}(\perp))$

Abstract Transition and Partition

$$Step_1^\sharp : (\mathbb{L} \rightarrow \wp(\mathbb{M})) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}))$$

$$Step_1^\sharp(X) = \pi_1 \circ \check{\wp}(\hookrightarrow)(X)$$

$$\pi_1 : \wp(\mathbb{S}) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}))$$

$$\pi_1(X) = \lambda l. \{m \in \mathbb{M} \mid \langle l, m \rangle \in X\}$$

X	$\mathbb{L} \rightarrow \wp(\mathbb{M})$	a partitioned collection of states at a certain step
X	$\wp(\mathbb{L} \times \mathbb{M}) = \wp(\mathbb{S})$	a set of states at a certain step
$\{s' \mid s \hookrightarrow s', s \in X\}$	$\wp(\mathbb{S})$	a set of states at the next step
$\bigcup\{(\hookrightarrow)(s) \mid s \in X\}$	$\wp(\mathbb{S})$	by regarding \hookrightarrow as a function of $\mathbb{S} \rightarrow \wp(\mathbb{S})$
$\cup \circ \wp(\hookrightarrow)(X)$	$\wp(\mathbb{S})$	by definition
$\check{\wp}(\hookrightarrow)(X)$	$\wp(\mathbb{S})$	by definition
$\lambda l. \{m \mid \langle l, m \rangle \in \check{\wp}(\hookrightarrow)(X)\}$	$\mathbb{L} \rightarrow \wp(\mathbb{M})$	partition
$\pi_1 \circ \check{\wp}(\hookrightarrow)(X)$	$\mathbb{L} \rightarrow \wp(\mathbb{M})$	by definition

Composition of Abstract Operators

Theorem. Given sound abstract operators f_1^\sharp and f_2^\sharp with respect to concrete operators f_1 and f_2 :

$$f_1 \circ \gamma \sqsubseteq \gamma \circ f_1^\sharp \quad f_2 \circ \gamma \sqsubseteq \gamma \circ f_2^\sharp$$

then, their composition $f_2^\sharp \circ f_1^\sharp$ is also sound with respect to the concrete version $f_2 \circ f_1$:

$$(f_2 \circ f_1) \circ \gamma \sqsubseteq \gamma \circ (f_2^\sharp \circ f_1^\sharp)$$

Proof.

$$\begin{aligned} f_2 \circ f_1 \circ \gamma &\sqsubseteq f_2 \circ \gamma \circ f_1^\sharp && \text{(by } f_1 \circ \gamma \sqsubseteq \gamma \circ f_1^\sharp \text{ and monotonicity of } f_2\text{)} \\ &\sqsubseteq \gamma \circ f_2^\sharp \circ f_1^\sharp && \text{(by } f_2 \circ \gamma \sqsubseteq \gamma \circ f_2^\sharp\text{)} \end{aligned}$$

Soundness Proof (1)

$\text{lfp } F \subseteq \gamma_1(\bigsqcup_{i \geq 0} F_1^{\sharp i}(\perp))$ if $F \circ \gamma_1 \subseteq \gamma_1 \circ F_1^{\sharp}$ by the fixpoint transfer theorem

$$F : \wp(\mathbb{S}) \rightarrow \wp(\mathbb{S})$$

$$F(X) = I \cup Step(X)$$

$$F_1^{\sharp} : (\mathbb{L} \rightarrow \wp(\mathbb{M})) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}))$$

$$F_1^{\sharp}(X) = \alpha_1(I) \cup_1^{\sharp} Step_1^{\sharp}(X)$$

It is enough to show that $Step_1^{\sharp}$ and \cup_1^{\sharp} are sound abstractions of $Step$ and \cup

Soundness Proof (2)

$$Step \circ \gamma_1 \subseteq \gamma_1 \circ Step_1^\sharp$$

$$\begin{aligned} Step \circ \gamma_1(X) &= \check{\wp}(\hookrightarrow) \circ \gamma_1(X) && \text{(by definition)} \\ &= \check{\wp}(\hookrightarrow)(\{\langle l, m \rangle \mid m \in X(l)\}) && \text{(by definition of } \gamma_1\text{)} \\ &= \bigcup \{(\hookrightarrow)(\langle l, m \rangle) \mid m \in X(l)\} && \text{(by definition)} \\ &= \{\langle l', m' \rangle \mid \langle l, m \rangle \hookrightarrow \langle l', m' \rangle, m \in X(l)\} && \text{(regarding } \hookrightarrow \text{ as a relation)} \\ &= \{\langle l', m' \rangle \mid \langle l, m \rangle \hookrightarrow \langle l', m' \rangle, \langle l, m \rangle \in X\} && \text{(regarding } X \text{ as a set)} \\ &= \{\langle l', m' \rangle \mid s \hookrightarrow \langle l', m' \rangle, s \in X\} && \text{(simplification)} \\ &= \{\langle l', m' \rangle \mid \langle l', m' \rangle \in \{s' \mid s \hookrightarrow s', s \in X\}\} \\ &= \{\langle l', m' \rangle \mid \langle l', m' \rangle \in \bigcup \{(\hookrightarrow)(s) \mid s \in X\}\} && \text{(regarding } \hookrightarrow \text{ as a function)} \\ &= \{\langle l', m' \rangle \mid \langle l', m' \rangle \in \check{\wp}(\hookrightarrow)(X)\} && \text{(by definition)} \\ &= \{\langle l', m' \rangle \mid m' \in \check{\wp}(\hookrightarrow)(X)(l')\} && \text{(regarding } \check{\wp}(\hookrightarrow)(X) \text{ as a function)} \\ &= \gamma_1 \circ (\check{\wp}(\hookrightarrow)(X)) && \text{(by definition of } \gamma\text{)} \\ &= \gamma_1 \circ (\lambda l'. \{m' \mid \langle l', m' \rangle \in \check{\wp}(\hookrightarrow)(X)\}) && \text{(regarding } \check{\wp}(\hookrightarrow)(X) \text{ as a set)} \\ &= \gamma_1 \circ \pi_1 \circ \check{\wp}(\hookrightarrow)(X) && \text{(by definition of } \pi_1\text{)} \\ &= \gamma_1 \circ Step_1^\sharp(X) && \text{(by definition of } Step_1^\sharp\text{)} \end{aligned}$$

Soundness Proof (3)

$$\cup \circ (\gamma_1, \gamma_1) \subseteq \gamma_1 \circ \cup_1^\sharp$$

$$\begin{aligned} \cup \circ (\gamma_1, \gamma_1)(X_1, X_2) &= \gamma_1(X_1) \cup \gamma_1(X_2) && \text{(by definition)} \\ &= \{\langle l_1, m_1 \rangle \mid m_1 \in X_1(l_1)\} \cup \{\langle l_2, m_2 \rangle \mid m_2 \in X_2(l_2)\} && \text{(by definition of } \gamma_1\text{)} \\ &= \{\langle l, m \rangle \mid m \in X_1(l) \cup X_2(l)\} \\ &= \{\langle l, m \rangle \mid m \in (\lambda l. X_1(l) \cup X_2(l))(l)\} \\ &= \gamma_1(\lambda l. X_1(l) \cup X_2(l)) && \text{(by definition of } \gamma_1\text{)} \\ &= \gamma_1 \circ \cup_1^\sharp && \text{(by definition of } \cup_1^\sharp\text{)} \end{aligned}$$

Abstraction 2: Memory

- Variable-wise abstraction of possible values
 - Abstracting the set of possible values for each variable, **individually** (so-called non-relational abstraction)
 - C.f., *Relational* abstraction: computing constraints **between** variables

$$\{x \mapsto 1, y \mapsto 2\}$$
$$\{x \mapsto 2, y \mapsto 3\}$$

set of memories

$$\{x \mapsto [1, 2], y \mapsto [2, 3]\}$$

non-relational abstraction
with interval

$$\begin{aligned} &\{1 \leq x \leq 2, \\ &2 \leq y \leq 3, \\ &y - x = 1\} \end{aligned}$$

relational abstraction

Abstract Domain

- Program-label-wise abstract memories: $\mathbb{D}^\sharp = \mathbb{L} \rightarrow \mathbb{M}^\sharp$
- Ordered via the point-wise order:

$$X_1^\sharp \sqsubseteq_{\mathbb{D}^\sharp} X_2^\sharp \iff \forall l \in \mathbb{L}. X_1^\sharp(l) \sqsubseteq_{\mathbb{M}^\sharp} X_2^\sharp(l)$$

$$X_1^\sharp \sqcup_{\mathbb{D}^\sharp} X_2^\sharp = \lambda l. (X_1^\sharp(l) \sqcup_{\mathbb{M}^\sharp} X_2^\sharp(l))$$

- Galois connection: $\mathbb{L} \rightarrow \wp(\mathbb{M}) \xrightleftharpoons[\alpha_2]{\gamma_2} \mathbb{L} \rightarrow \mathbb{M}^\sharp$

$$\alpha_2(X) = \lambda l. \alpha_{\mathbb{M}}(X(l))$$

$$\gamma_2(X^\sharp) = \lambda l. \gamma_{\mathbb{M}}(X^\sharp(l))$$

Abstract Domain of Memories

- Variable-wise abstract values: $\mathbb{M}^\sharp = \mathbb{X} \rightarrow \mathbb{V}^\sharp$
- Ordered via the point-wise order:

$$m_1^\sharp \sqsubseteq_{\mathbb{M}^\sharp} m_2^\sharp \iff \forall x \in \mathbb{X}. m_1^\sharp(x) \sqsubseteq_{\mathbb{V}^\sharp} m_2^\sharp(x)$$

$$m_1^\sharp \sqcup_{\mathbb{M}^\sharp} m_2^\sharp = \lambda x. (m_1^\sharp(x) \sqcup_{\mathbb{V}^\sharp} m_2^\sharp(x))$$

- Galois connection: $\wp(\mathbb{M}) \xrightleftharpoons[\alpha_{\mathbb{M}}]{\gamma_{\mathbb{M}}} \mathbb{M}^\sharp$

$$\alpha_{\mathbb{M}}(M) = \lambda x \in \mathbb{X}. \alpha_{\mathbb{Z}}(\{m(x) \mid m \in M\})$$

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

Abstract Domain for Values

- Interval abstraction: $\mathbb{Z}^\sharp = \{\perp\} \cup \{[l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty, +\infty\} \wedge l \leq u\}$
- Order:

$$\begin{array}{ll} \perp \sqsubseteq_{\mathbb{Z}^\sharp} X & (\forall X \in \mathbb{Z}^\sharp) \\ [l_1, u_1] \sqsubseteq_{\mathbb{Z}^\sharp} [l_2, u_2] & (\text{if } l_2 \leq l_1 \wedge u_1 \leq u_2) \end{array}$$

$$\begin{array}{ll} \perp \sqcup_{\mathbb{Z}^\sharp} X = X \sqcup_{\mathbb{Z}^\sharp} \perp = X & (\forall X \in \mathbb{Z}^\sharp) \\ [l_1, u_1] \sqcup_{\mathbb{Z}^\sharp} [l_2, u_2] = [\min(l_1, l_2), \max(u_1, u_2)] \end{array}$$

- Galois connection: $\wp(\mathbb{Z}) \xrightleftharpoons[\alpha_{\mathbb{Z}}]{\gamma_{\mathbb{Z}}} \mathbb{Z}^\sharp$

$$\alpha_{\mathbb{Z}}(Z) = [\min Z, \max Z]$$

$$\gamma_{\mathbb{Z}}(n^\sharp) = \begin{cases} \emptyset & \text{if } n^\sharp = \perp \\ \{x \in \mathbb{Z} \mid l \leq x \leq u\} & \text{if } n^\sharp = [l, u] \end{cases}$$

Abstract State Transition

- The abstract semantics is defined using a transition system $(\mathbb{S}^\sharp, \rightarrow^\sharp)$
 - $\mathbb{S}^\sharp = \mathbb{L} \times \mathbb{M}^\sharp$: the set of states $\langle l, m^\sharp \rangle$
 - $(\rightarrow^\sharp) \subseteq \mathbb{S}^\sharp \times \mathbb{S}^\sharp$: the transition relation that describes computation steps

$x := E$	$\vdash \langle l, m^\sharp \rangle \rightarrow^\sharp \langle \text{next}(l), \text{update}_x^\sharp(m, \text{eval}_E^\sharp(m^\sharp)) \rangle$
$\text{br } E \ l_1 \ l_2$	$\vdash \langle l, m^\sharp \rangle \rightarrow^\sharp \langle l_1, \text{filter}_E^\sharp(m^\sharp) \rangle$
	$\vdash \langle l, m^\sharp \rangle \rightarrow^\sharp \langle l_2, \text{filter}_{\neg E}^\sharp(m^\sharp) \rangle$
$\text{goto } l'$	$\vdash \langle l, m^\sharp \rangle \rightarrow^\sharp \langle l', m^\sharp \rangle$
$x := \text{input}()$	$\vdash \langle l, m^\sharp \rangle \rightarrow^\sharp \langle \text{next}(l), \text{update}_x^\sharp(m^\sharp, \alpha_Z(\mathbb{Z})) \rangle$
$x := E$	$\vdash \langle l, m^\sharp \rangle \rightarrow^\sharp \langle \text{next}(l), \text{update}_x^\sharp(m^\sharp, \text{eval}_E^\sharp(m^\sharp)) \rangle$

Question: How to design
abstract semantic operators?

Best Abstraction

Theorem. Given a Galois connection $\wp(\mathbb{D}) \xrightleftharpoons[\alpha]{\gamma} \mathbb{D}^\sharp$ and a concrete semantic function F the best (i.e., most precise) abstraction of the function is defined as follows:

$$F_{best}^\sharp = \alpha \circ F \circ \gamma$$

Proof. Any other sound abstraction function F^\sharp is greater than F_{best}^\sharp :

$$\begin{aligned} F \circ \gamma &\sqsubseteq \gamma \circ F^\sharp && \text{(by the abstract interpretation framework)} \\ \iff \alpha \circ F \circ \gamma &\sqsubseteq \alpha \circ \gamma \circ F^\sharp && \text{(by monotonicity of } \alpha\text{)} \\ \iff \alpha \circ F \circ \gamma &\sqsubseteq F^\sharp && (\alpha \circ \gamma \sqsubseteq id) \end{aligned}$$

Design of Abstract Semantic Functions

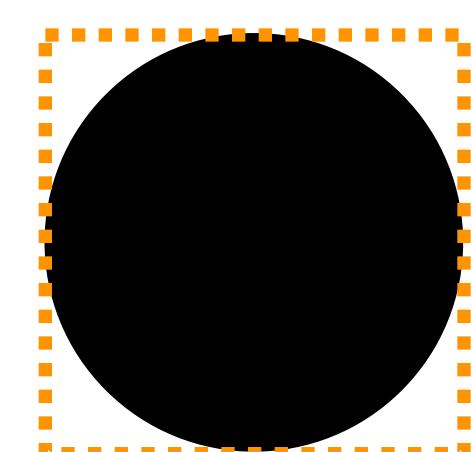
- If the best abstraction can be rewritten using only abstract operators:

$$\begin{aligned} F_{best}^\sharp &= f_n^\sharp \circ \cdots \circ f_1^\sharp \\ &= F^\sharp \end{aligned}$$

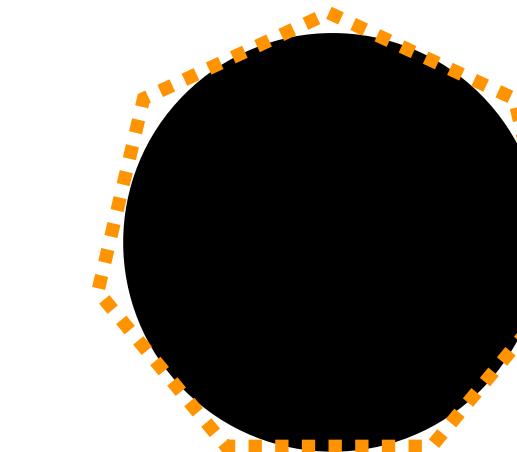
- If the best abstraction cannot be rewritten using only abstract operators:

$$\begin{aligned} F_{best}^\sharp &= f_n^\sharp \circ \cdots \circ \gamma \circ f_m \circ \cdots \circ f_1^\sharp \\ &\sqsubseteq f_n^\sharp \circ \cdots \circ f_{m'}^\sharp \circ \cdots \circ f_1^\sharp \\ &= F^\sharp \end{aligned}$$

- The best abstraction may not exist for some abstract domains



best abstraction
for interval



no best abstraction
for polyhedra (finite set of linear constraints)

Abstract Semantic Operators

- The abstract memory update operation:

$$\begin{aligned} update_x^\# : \mathbb{V}^\# \times \mathbb{M}^\# &\rightarrow \mathbb{M}^\# \\ update_x^\#(n^\#, m^\#) &= m^\# \{x \mapsto n^\#\} \end{aligned}$$

- The abstract expression-evaluation operation:

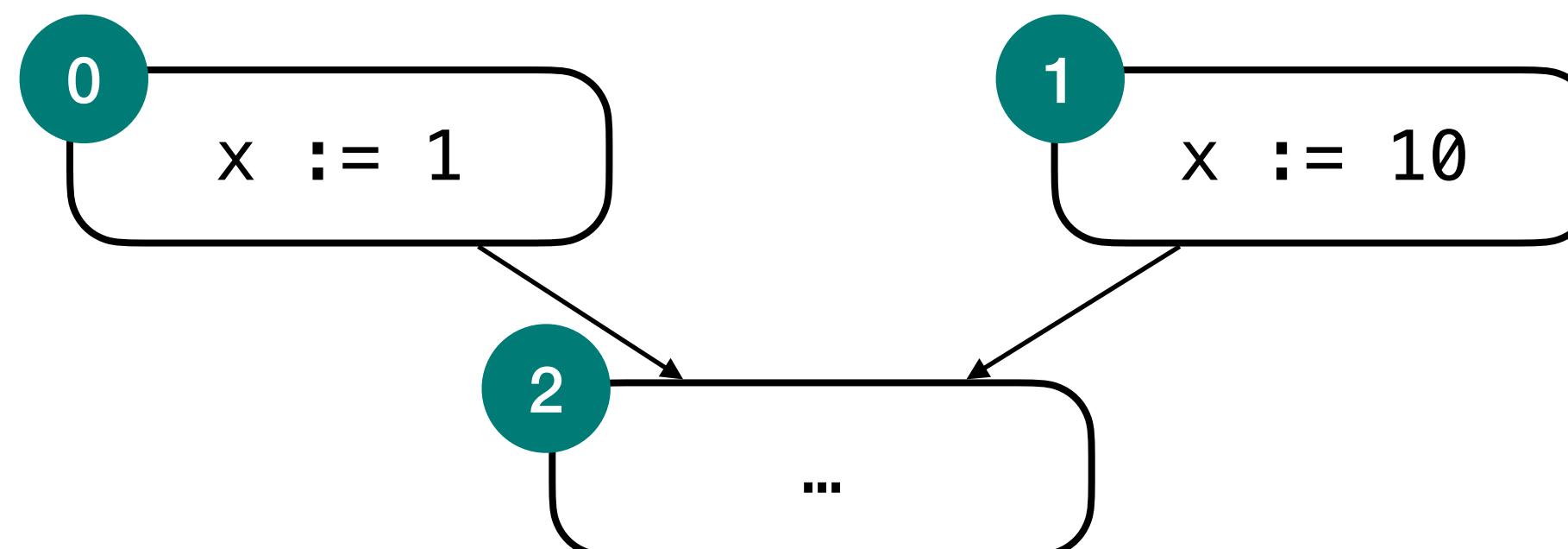
$$\begin{aligned} eval_E^\# : \mathbb{M}^\# &\rightarrow \mathbb{V}^\# \\ eval_n^\#(m) &= \alpha_{\mathbb{Z}}(\{n\}) \\ eval_x^\#(m) &= m^\#(x) \\ eval_{E_1 \oplus E_2}^\#(m) &= eval_{E_1}^\#(m^\#) \oplus^\# eval_{E_2}^\#(m^\#) \end{aligned}$$

- The abstract memory filter operation:

$$\begin{aligned} filter_E^\# : \mathbb{M}^\# &\rightarrow \mathbb{M}^\# \\ filter_E^\#(m^\#) &= \alpha_{\mathbb{M}}(\{m \in \gamma_{\mathbb{M}}(m^\#) \mid eval_E(m) = \text{true}\}) \end{aligned}$$

Practical Aspect 1: Join

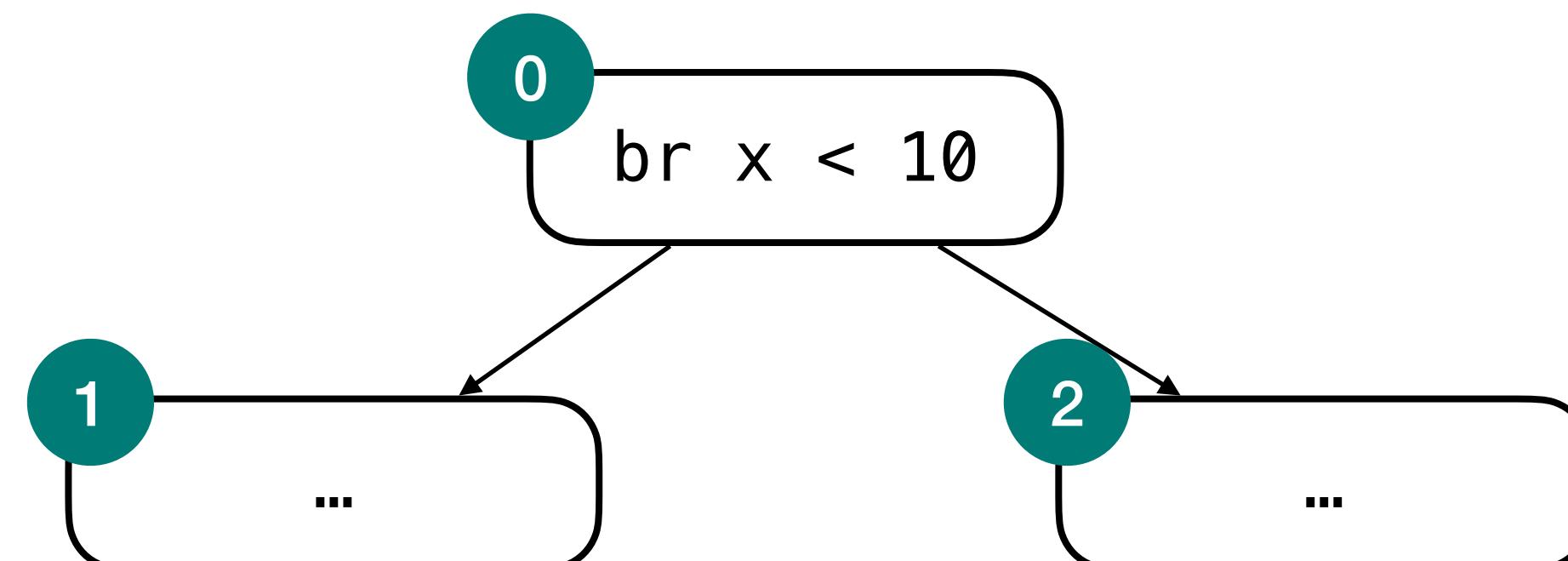
- According to the AI framework, the least upper bound (so-called join) is required to be defined on each chain (i.e., CPO)
- If a join is defined for any non-empty finite set (i.e., join-semilattice), it makes easy to define semantic functions



$$\{2 \mapsto \{x \mapsto [1, 1]\}\} \sqcup \{2 \mapsto \{x \mapsto [10, 10]\}\} = \{2 \mapsto \{x \mapsto [1, 10]\}\}$$

Practical Aspect 2: Meet

- According to the AI framework, no need for the greatest lower bound (so-called meet)
- If a meet is defined for any nonempty finite subset (i.e., meet-semilattice), it makes easy to define precise semantic functions



$$\begin{aligned}\langle 0, m^\# \rangle &\hookrightarrow^\# \langle 1, m^\# \{x \mapsto m^\#(x) \sqcap [-\infty, 9]\} \rangle \\ &\hookrightarrow^\# \langle 2, m^\# \{x \mapsto m^\#(x) \sqcap [10, +\infty]\} \rangle\end{aligned}$$

Example

- Abstract addition:

$$\begin{array}{lllll} n_1^\sharp & +^\sharp & \perp & = & \perp \\ \perp & +^\sharp & n_2^\sharp & = & \perp \\ [l_1, u_1] & +^\sharp & [l_2, u_2] & = & [l_1 + l_2, u_1 + u_2] \end{array}$$

$$\alpha_{\mathbb{Z}}(\{n_1 + n_2 \mid n_1 \in \gamma_{\mathbb{Z}}(n_1^\sharp), n_2 \in \gamma_{\mathbb{Z}}(n_2^\sharp)\})$$

- Abstract filter:

$$filter_{x < k}^\sharp(m^\sharp) = m^\sharp\{x \mapsto m^\sharp(x) \sqcap [-\infty, k - 1]\}$$

$$\alpha_{\mathbb{M}}(\{m \in \gamma_{\mathbb{M}}(m^\sharp) \mid eval_{x < k}(m) = \text{true}\})$$

Abstract Semantics

- The abstract semantic function:

$$F^\sharp : (\mathbb{L} \rightarrow \mathbb{M}^\sharp) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}^\sharp)$$

$$F^\sharp(X) = (\alpha_2 \circ \alpha_1)(I) \sqcup Step^\sharp(X)$$

- The abstract semantic functions for transition and partition:

$$Step^\sharp : (\mathbb{L} \rightarrow \mathbb{M}^\sharp) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}^\sharp)$$

$$Step^\sharp(X) = \wp(id, \sqcup) \circ \pi \circ \wp(\rightarrow^\sharp)(X)$$

$$\pi : \wp(\mathbb{S}^\sharp) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}^\sharp))$$

$$\pi(X) = \lambda l. \{ m^\sharp \in \mathbb{M}^\sharp \mid \langle l, m^\sharp \rangle \in X \}$$

- Soundness: $\text{lfp } F \subseteq \gamma_1 \circ \gamma_2 \left(\bigsqcup_{i \geq 0} F^\sharp(\perp) \right)$

Abstract Transition and Partition

$$Step^\# : (\mathbb{L} \rightarrow \mathbb{M}^\#) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}^\#)$$

$$Step^\#(X) = \wp(id, \sqcup) \circ \pi \circ \check{\wp}(\hookrightarrow^\#)(X)$$

$$\pi : \wp(\mathbb{S}^\#) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}^\#))$$

$$\pi(X) = \lambda l. \{m^\# \in \mathbb{M}^\# \mid \langle l, m^\# \rangle \in X\}$$

$X : \mathbb{L} \rightarrow \mathbb{M}^\#$	a partitioned set of abstract memories
$X : \wp(\mathbb{L} \times \mathbb{M}^\#) = \mathbb{S}^\#$	a set of abstract states at a certain step
$\{s^{\#'} \mid s^\# \hookrightarrow^\# s^{\#'}, s^\# \in X\} : \wp(\mathbb{S}^\#)$	a set of abstract states at the next step
$\bigcup\{(\hookrightarrow^\#)(s^\#) \mid s^\# \in X\} : \wp(\mathbb{S}^\#)$	by regarding $\hookrightarrow^\#$ as a function of $\mathbb{S}^\# \rightarrow \wp(\mathbb{S}^\#)$
$\cup \circ \wp(\hookrightarrow^\#)(X) : \wp(\mathbb{S}^\#)$	by definition
$\check{\wp}(\hookrightarrow^\#)(X) : \wp(\mathbb{S}^\#)$	by definition
$\lambda l. \{m^\# \mid \langle l, m^\# \rangle \in \check{\wp}(\hookrightarrow^\#)(X)\} : \mathbb{L} \rightarrow \wp(\mathbb{M}^\#)$	partition
$\pi \circ \check{\wp}(\hookrightarrow^\#)(X) : \mathbb{L} \rightarrow \wp(\mathbb{M}^\#)$	by definition
$\lambda l. \bigsqcup\{m^\# \mid m^\# \in \pi \circ \check{\wp}(\hookrightarrow^\#)(X)(l)\} : \mathbb{L} \rightarrow \mathbb{M}^\#$	join of multiple abstract states
$\wp(id, \sqcup_{\mathbb{M}}) \circ \pi \circ \check{\wp}(\hookrightarrow^\#)(X) : \mathbb{L} \rightarrow \mathbb{M}^\#$	by definition

Soundness Proof (1)

$\text{lfp } F_1^\sharp \subseteq_1^\sharp \gamma_2(\bigsqcup_{i \geq 0} F^{\sharp i}(\perp))$ if $F_1^\sharp \circ \gamma_2 \subseteq_1^\sharp \gamma_2 \circ F^\sharp$ by the fixpoint transfer theorem

$$F_1^\sharp : (\mathbb{L} \rightarrow \wp(\mathbb{M})) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{M}))$$

$$F_1^\sharp(X) = \alpha_1(I) \cup_1^\sharp Step_1^\sharp(X)$$

$$F^\sharp : (\mathbb{L} \rightarrow \mathbb{M}^\sharp) \rightarrow (\mathbb{L} \rightarrow \mathbb{M}^\sharp)$$

$$F^\sharp(X) = \alpha_2 \circ \alpha_1(I) \sqcup Step^\sharp(X)$$

It is enough to show that $Step^\sharp$ and \sqcup are sound abstractions of $Step_1^\sharp$ and \cup_1^\sharp

Soundness Proof (2)

$$Step_1^\# \circ \gamma_2 \subseteq_1^\# \gamma_2 \circ Step^\#$$

$$Step_1^\# \circ \gamma_2(X^\#) = \pi_1 \circ \wp(\hookrightarrow) \circ \gamma_2(X^\#) \quad (\text{by definition of } Step_1^\#)$$

$$= \wp(\hookrightarrow) \circ \gamma_2(X^\#) \quad (\text{by regarding } \wp(\hookrightarrow) \circ \gamma_2(X^\#) \text{ as a function of } \mathbb{L} \rightarrow \wp(\mathbb{M}))$$

$$\subseteq_1^\# \gamma_2 \circ \wp(\hookrightarrow^\#)(X^\#) \quad (\text{by condition } \wp(\hookrightarrow) \circ \gamma_2 \subseteq_1^\# \gamma_2 \circ \wp(\hookrightarrow^\#) \text{ and new subgoal})$$

by regarding the input of γ_2 as a subset of $\mathbb{L} \times \mathbb{M}^\#$)

$$\subseteq_1^\# \gamma_2 \circ \wp(id, \sqcup_{\mathbb{M}}) \circ \pi \circ \wp(\hookrightarrow^\#)(X^\#) \quad (\text{by monotonicity of } \gamma_2)$$

$$= \gamma_2 \circ Step^\#(X^\#) \quad (\text{by definition of } Step^\#)$$

Soundness Proof (2-1)

$$\check{\wp}(\hookrightarrow) \circ \gamma_2 \subseteq_1^\# \gamma_2 \circ \check{\wp}(\hookrightarrow^\#)$$

Proof sketch: We prove the soundness of the abstract transition relation by case analysis for the command at a label. The soundness of each case will also require the soundness of each abstract operator used in the definition.

Soundness Proof (3)

$$\cup_1^\# \circ (\gamma_2, \gamma_2) \subseteq_1^\# \gamma_2 \circ \sqcup_{\mathbb{D}^\#}$$

$$\begin{aligned}\cup_1^\# \circ (\gamma_2, \gamma_2)(X_1^\#, X_2^\#) &= \gamma_2(X_1^\#) \cup_1^\# \gamma_2(X_2^\#) \\&= \lambda l. \gamma_2(X_1^\#)(l) \cup \gamma_2(X_2^\#)(l) && \text{(by definition of } \cup_1^\text{)} \\&= \lambda l. (\lambda l_1. \gamma_{\mathbb{M}}(X_1^\#(l_1))(l)) \cup (\lambda l_2. \gamma_{\mathbb{M}}(X_2^\#(l_2))(l)) && \text{(by definition of } \gamma_2\text{)} \\&= \lambda l. \gamma_{\mathbb{M}}(X_1^\#(l)) \cup \gamma_{\mathbb{M}}(X_2^\#(l)) && \text{(lambda application)} \\&\subseteq_1^\# \lambda l. \gamma_{\mathbb{M}}(X_1^\#(l) \sqcup_{\mathbb{M}} X_2^\#(l)) && \text{(by monotonicity of } \gamma_{\mathbb{M}}\text{)} \\&= \lambda l. \gamma_{\mathbb{M}}(\lambda l'. (X_1^\#(l') \sqcup_{\mathbb{M}^\#} X_2^\#(l'))(l)) && \text{(lambda abstraction)} \\&= \lambda l. \gamma_{\mathbb{M}}((X_1^\# \sqcup_{\mathbb{D}^\#} X_2^\#)(l)) && \text{(by definition of } \sqcup_{\mathbb{D}^\#}\text{)} \\&= \gamma_2 \circ (X_1^\# \sqcup_{\mathbb{D}^\#} X_2^\#) && \text{(by definition of } \gamma_2\text{)} \\&= \gamma_2 \circ \sqcup_{\mathbb{D}^\#}(X_1^\#, X_2^\#)\end{aligned}$$

Widening and Narrowing

- Widening and narrowing for domains \mathbb{D}^\sharp and \mathbb{M}^\sharp are defined as point-wise lifted ones (label-wise and variable-wise)
- Essence: widening and narrowing for the interval domain \mathbb{Z}^\sharp
 - A simple widening operator for the interval domain:

$$\begin{aligned}[a, b] \nabla \perp &= [a, b] \\ \perp \nabla [c, d] &= [c, d] \\ [a, b] \nabla [c, d] &= [(c < a? -\infty : a), (b < d? +\infty : b)]\end{aligned}$$

Check: Safety conditions
for widening

- A simple narrowing operator for the interval domain:

$$\begin{aligned}[a, b] \Delta \perp &= \perp \\ \perp \Delta [c, d] &= \perp \\ [a, b] \Delta [c, d] &= [(a = -\infty? c : a), (b = +\infty? d : b)]\end{aligned}$$

Check: Safety conditions
for narrowing

Basic Fixpoint Algorithm

- For finite-height abstract domains

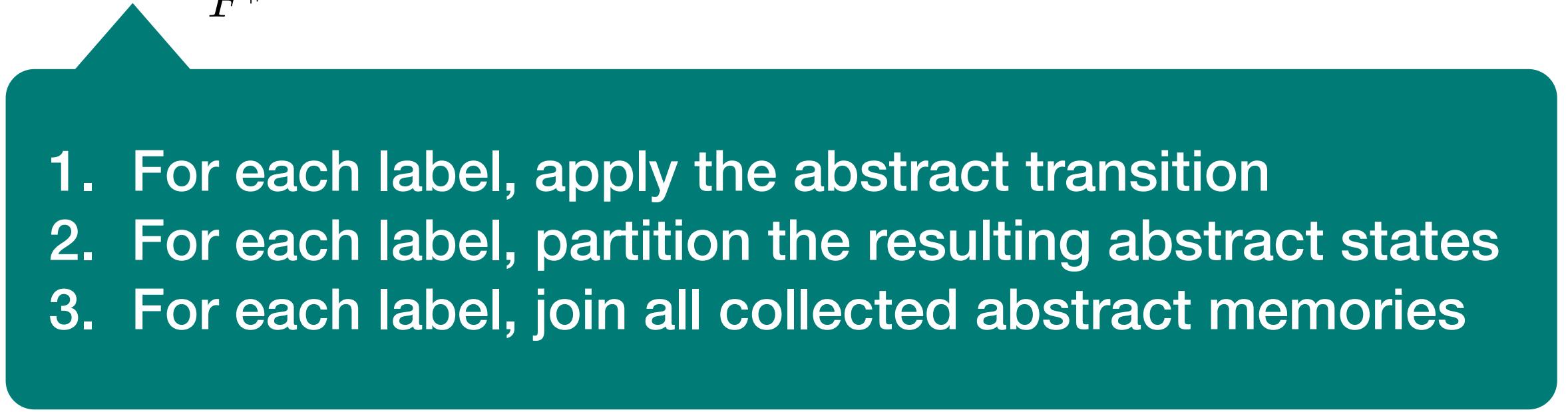
```
X, R : D#
F# : D# → D#
begin
    X ← ⊥
repeat
    R ← X
    X ← F#(X)
until X ⊑ R
return R
end
```

- For infinite-height abstract domains

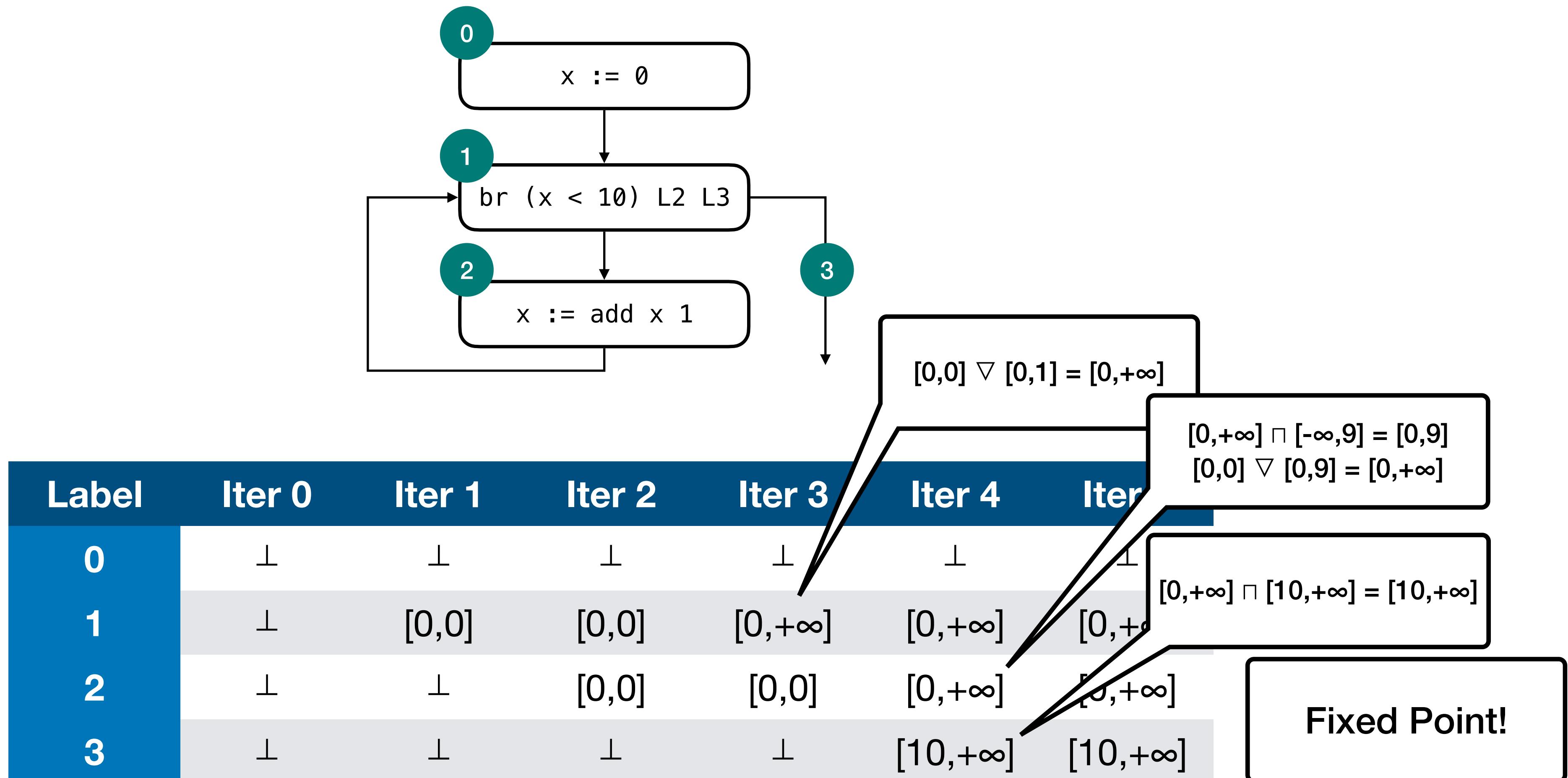
```
X, R : D#
F# : D# → D#
begin
    X ← ⊥
repeat
    R ← X
    X ← X ∇ F#(X)
until X ⊑ R
return R
end
```

Basic Fixpoint Algorithm (Inlined)

```
X, R : L → M#
F# : (L → M#) → (L → M#)
begin
    X ← ⊥
    repeat
        R ← X
        X ← X ∇ (φ(id, ⊑M) ∘ π ∘ φ(→#))(X)
    until X ⊑ R
    return R
end
```

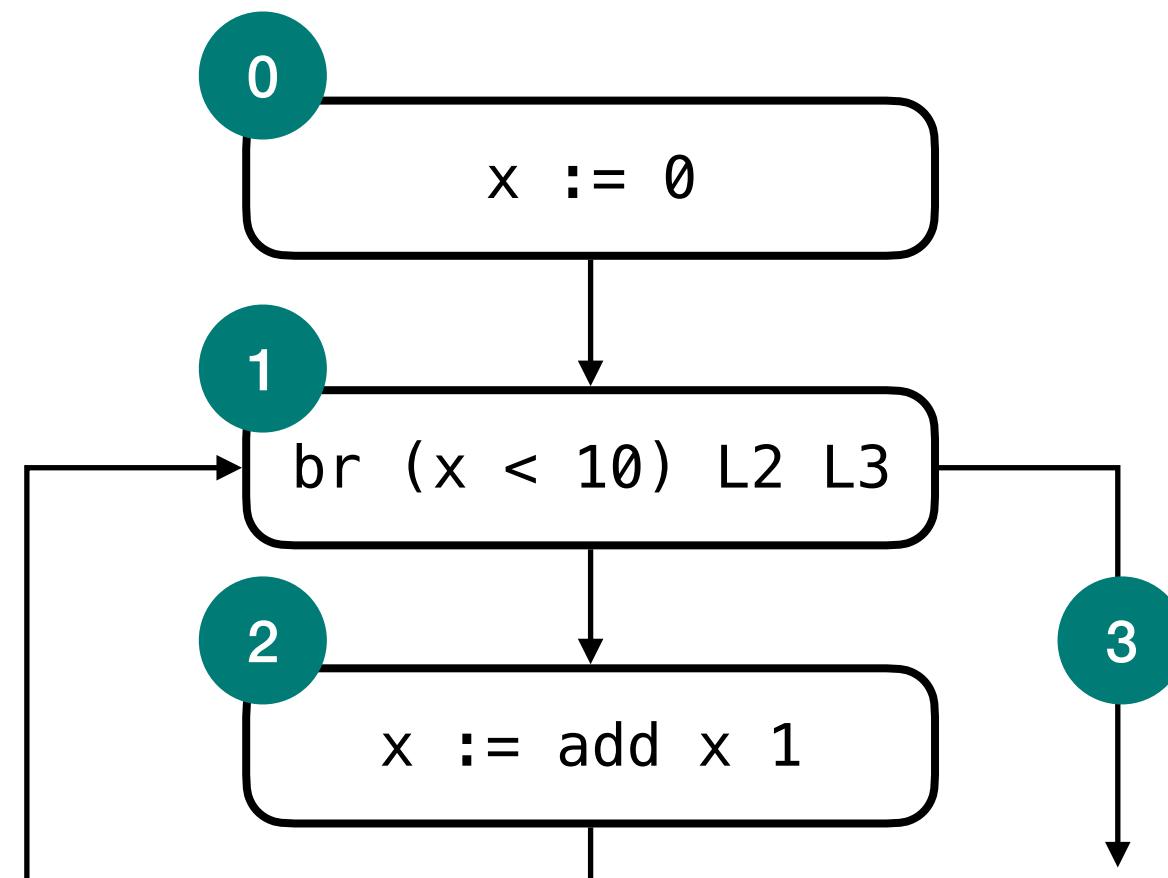
- 
1. For each label, apply the abstract transition
 2. For each label, partition the resulting abstract states
 3. For each label, join all collected abstract memories

Example



Questions

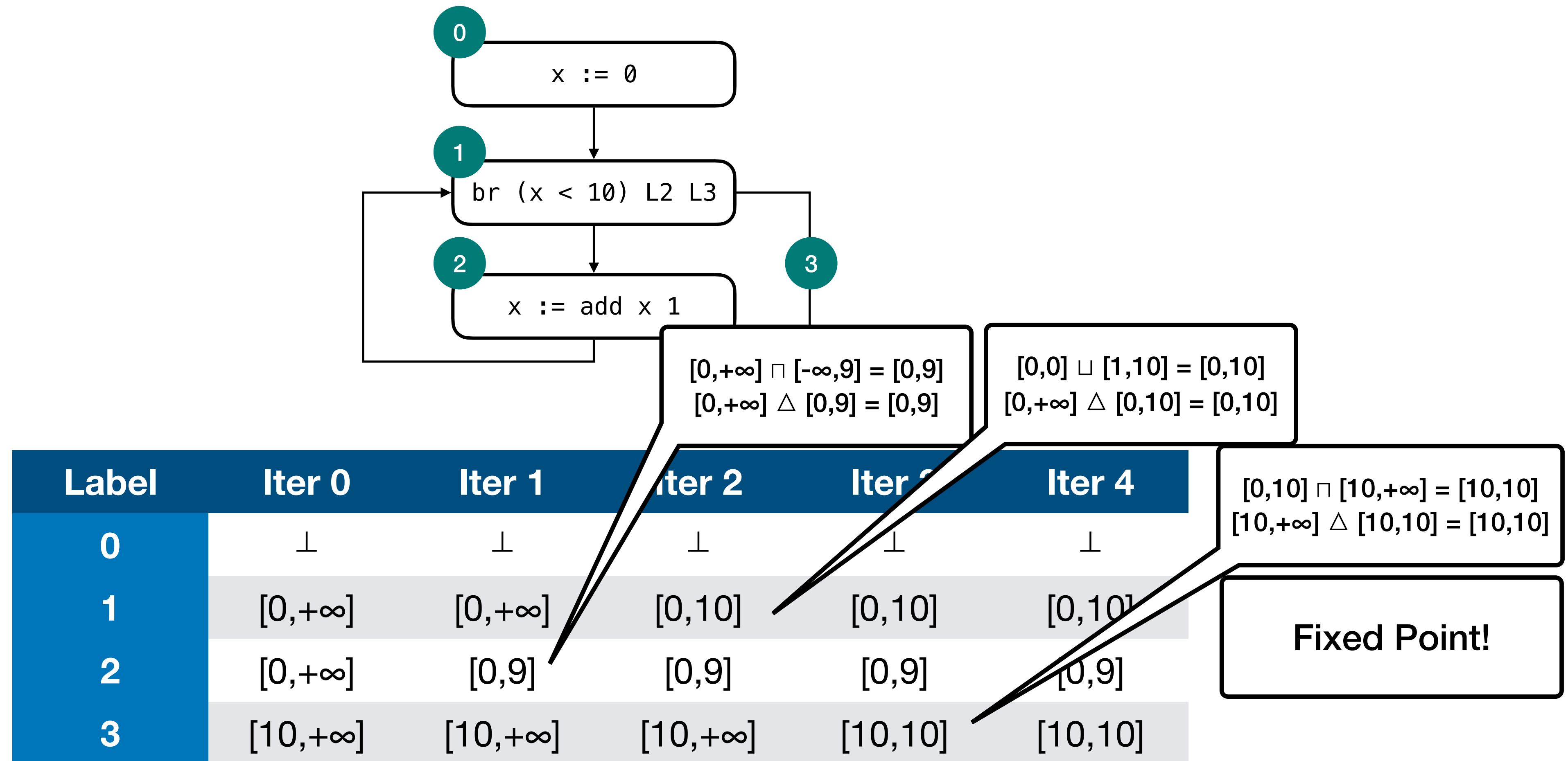
2. Inefficiency?
(#labels / iter ?)



Label	Iter 0	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5
0	⊥	⊥	⊥	⊥	⊥	⊥
1	⊥	[0,0]	[0,0]	[0,∞]	[0,∞]	[0,∞]
2	⊥	⊥	[0,0]	[0,0]	[0,∞]	[0,∞]
3	⊥	⊥	⊥	⊥	[10,∞]	[10,∞]

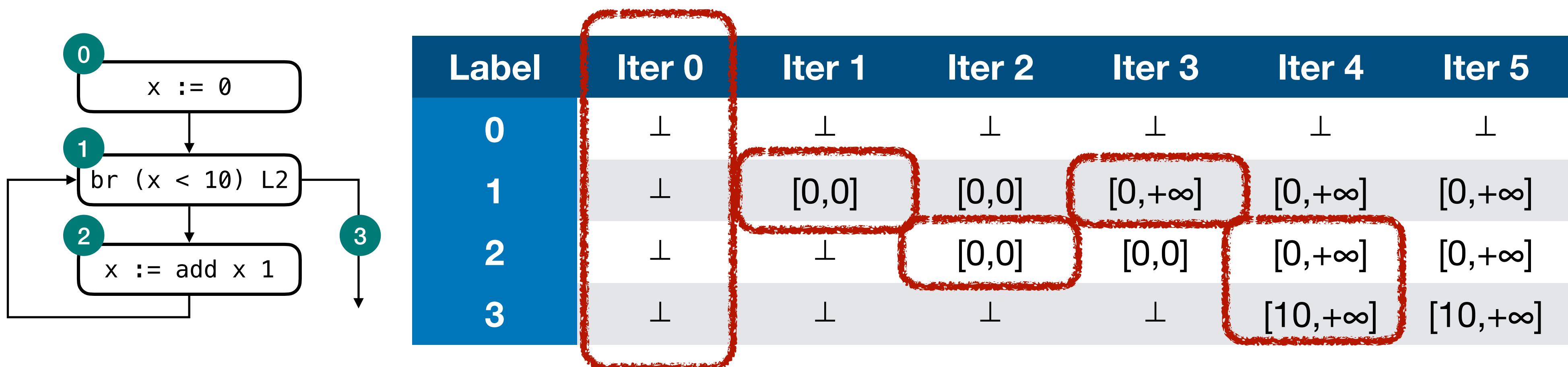
1. Why so imprecise?

Refinement by Narrowing



Worklist Algorithm

- Do only necessary “work” by maintaining a worklist
 - Work: a label whose abstract transition should be recomputed
 - Why recompute? because of updated input



Final Algorithm

- Worklist algorithm specialized for widening

```
X, R : L → M#
F# : (L → M#) → (L → M#)
Worklist : ϕ(L)
begin
    Worklist ← L
    X ← ⊥
    repeat
        R ← X
        X ← X ∇ F#(X|Worklist)
        Worklist ← {l ∈ L | X(l) ⊉ R(l)}
    until Worklist = ∅
    return R
end
```

Final Algorithm

- Worklist algorithm with widening

```
X : L → M#
F# : (L → M#) → (L → M#)
Worklist : ϕ(L)
begin
    Worklist ← L
    X ← ⊥
    repeat
        (w, Worklist) ← pop(Worklist)
        m#old ← X(w)
        m#new ← ⋃{m#out | ⟨l, X(l)⟩ ↦# ⟨w, m#out⟩}
        if m#new ⊈ m#old then
            m#new ← m#old ▽ m#new
            X(w) ← m#new
            Worklist ← Worklist ∪ {l | ⟨w, m#new⟩ ↦# ⟨l, -⟩}
        endif
    until Worklist = ∅
    return X
end
```

- Worklist algorithm with narrowing

```
X : L → M#
F# : (L → M#) → (L → M#)
Worklist : ϕ(L)
begin
    Worklist ← L
    X ← ⊥
    repeat
        (w, Worklist) ← pop(Worklist)
        m#old ← X(w)
        m#new ← ⋃{m#out | ⟨l, X(l)⟩ ↦# ⟨w, m#out⟩}
        if m#new ⊇ m#old then
            m#new ← m#old △ m#new
            X(w) ← m#new
            Worklist ← Worklist ∪ {l | ⟨w, m#new⟩ ↦# ⟨l, -⟩}
        endif
    until Worklist = ∅
    return X
end
```

Summary

- End-to-end design and implementation of a static analysis
 - Systematic and step-by-step construction of abstract semantics (composition, composition, and **composition!**)
 - Efficient fixed point computation algorithm based on worklist
- In practice, many design choices for precision and scalability
 - E.g., partitioning, relational abstraction, widening/narrowing, etc