Program Analysis

8. Design and Implementation of Static Analysis

Kihong Heo

KAIST

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

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

Semantic Domains

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

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

State Transition

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

$$
\begin{array}{rcll}
x \texttt{ := } E & : & \langle l, m \rangle \hookrightarrow \langle \texttt{next}(l), \mathit{update}_x(m, \mathit{eval}_E(m)) \rangle & \\
\texttt{br } E\ l_1\ l_2 & : & \langle l, m \rangle \hookrightarrow \langle l_1, \mathit{filter}_E(m) \rangle & \\
 & : & \langle l, m \rangle \hookrightarrow \langle l_2, \mathit{filter}_{\neg E}(m) \rangle & \\
\texttt{goto } l' & : & \langle l, m \rangle \hookrightarrow \langle l', m \rangle & \\
x \texttt{ := input()} & : & \langle l, m \rangle \hookrightarrow \langle \texttt{next}(l), \mathit{update}_x(m, z) \rangle & \text{for an input integer } z \\
\texttt{print}(x) & : & \langle l, m \rangle \hookrightarrow \langle \texttt{next}(l), m \rangle &
\end{array}
$$

Semantic Operators

- The memory update operation

$$update_x : \mathbb{M} \times \mathbb{V} \to \mathbb{M}$$
$$update_x(m, n) = m\{x \mapsto n\}$$

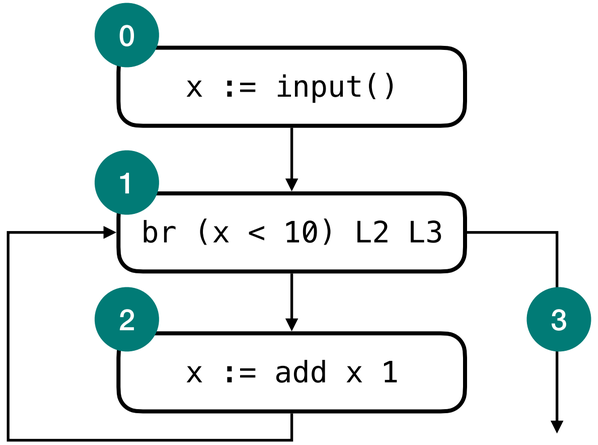
- The expression-evaluation operation

$$eval_E : \mathbb{M} \to \mathbb{V}$$
$$eval_n(m) = n$$
$$eval_x(m) = m(x)$$
$$eval_{E_1 \oplus E_2}(m) = eval_{E_1}(m) \oplus eval_{E_2}(m)$$

- The memory filter operation

$$filter_E : \mathbb{M} \to \mathbb{M}$$
$$filter_E(m) = m \quad \text{if } eval_E(m) = \mathsf{true}$$

Set of Reachable States



$\ldots$

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

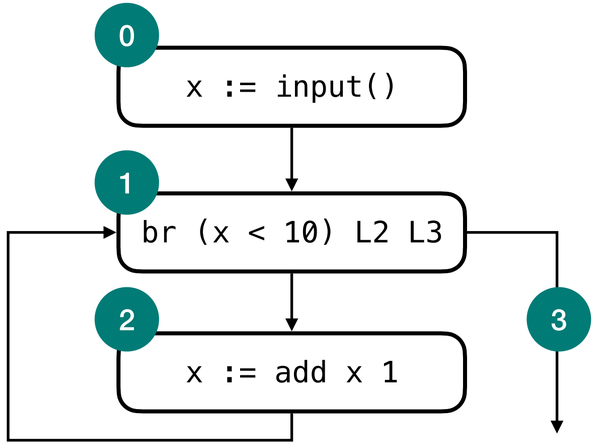
For input 9 : $\langle 0, \emptyset \rangle \hookrightarrow \langle 1, x \mapsto 9 \rangle \hookrightarrow \langle 2, x \mapsto 9 \rangle \hookrightarrow \langle 1, x \mapsto 10 \rangle \hookrightarrow \langle 3, x \mapsto 10 \rangle$

$\ldots$

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

$\ldots$

Set of Reachable States



Reachable states $= \cdots$

$$\cup \{\langle 0, \emptyset \rangle, \langle 1, x \mapsto 10 \rangle, \langle 2, x \mapsto 10 \rangle, \langle 3, x \mapsto 10 \rangle\}$$
$$\cup \{\langle 0, \emptyset \rangle, \langle 1, x \mapsto 9 \rangle, \langle 2, x \mapsto 9 \rangle, \langle 1, x \mapsto 10 \rangle, \langle 3, x \mapsto 10 \rangle\}$$
$$\cup \cdots$$
$$\cup \{\langle 0, \emptyset \rangle, \langle 1, x \mapsto 0 \rangle, \langle 2, x \mapsto 0 \rangle, \langle 1, x \mapsto 1 \rangle, \cdots, \langle 3, x \mapsto 10 \rangle\}$$
$$\cup \cdots$$

Concrete Domain and Semantics

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

- Concrete semantic function:

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

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

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

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

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

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

Notations (1)

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

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

Notations (2)

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

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

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

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

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

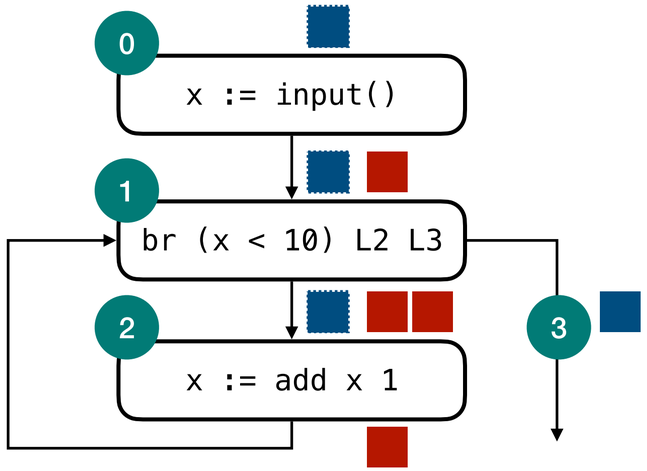
$$Step(X) = \bigcup \{(\hookrightarrow)(s) \mid s \in X\} = \cup \circ \wp(\hookrightarrow)(X) = \breve{\wp}(\hookrightarrow)(X)$$

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

$$(f, g) : A \times A' \to B \times B'$$
$$(f, g)(a, a') = (f(a), g(a'))$$

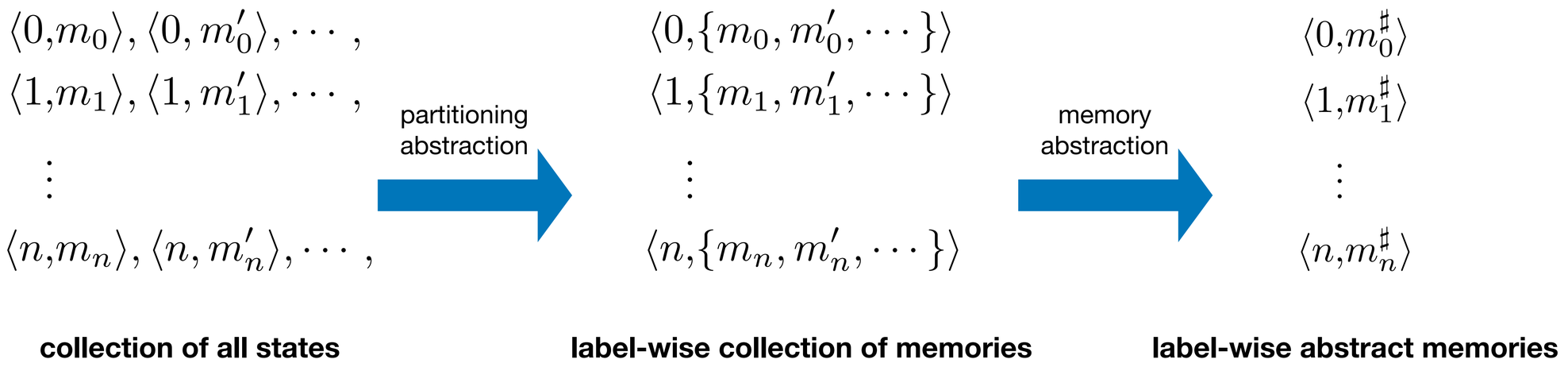
Informal Overview of Abstract Semantics

: memory at each label
: memory after transition



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

Sequence of Abstractions



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

$$\langle 0, \{m_0, m_0', \cdots\}\rangle$$
$$\langle 1, \{m_1, m_1', \cdots\}\rangle$$
$$\vdots$$
$$\langle n, \{m_n, m_n', \cdots\}\rangle$$

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

vs

$$\langle *, \{m_0, m_0', \cdots$$
$$m_1, m_1', \cdots$$
$$\vdots$$
$$m_n, m_n', \cdots\}\rangle$$

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

Abstract Domain

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

- Ordered via the point-wise order:

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

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

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

- Galois connection: $\wp(\mathbb{S}) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \mathbb{L} \to \wp(\mathbb{M})$

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

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

Abstract Semantics

- The abstract semantic function:

$$F_1^\sharp : (\mathbb{L} \to \wp(\mathbb{M})) \to (\mathbb{L} \to \wp(\mathbb{M}))$$
$$F_1^\sharp(X) = \alpha_1(I) \cup_1^\sharp Step_1^\sharp(X)$$

- The abstract semantic functions for transition and partition:

$$Step_1^\sharp : (\mathbb{L} \to \wp(\mathbb{M})) \to (\mathbb{L} \to \wp(\mathbb{M})) \qquad \pi_1 : \wp(\mathbb{S}) \to (\mathbb{L} \to \wp(\mathbb{M}))$$
$$Step_1^\sharp(X) = \pi_1 \circ \breve{\wp}(\hookrightarrow)(X) \qquad \pi_1(X) = \lambda l.\{m \in \mathbb{M} \mid \langle l, m \rangle \in X\}$$

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

Abstract Transition and Partition

$$Step_1^\sharp : (\mathbb{L} \to \wp(\mathbb{M})) \to (\mathbb{L} \to \wp(\mathbb{M}))$$
$$Step_1^\sharp(X) = \pi_1 \circ \breve{\wp}(\hookrightarrow)(X)$$

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

X	:	$\mathbb{L} \to \wp(\mathbb{M})$	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 $\mathbb{S} \to \wp(\mathbb{S})$
$\cup \circ \wp(\hookrightarrow)(X)$	:	$\wp(\mathbb{S})$	by definition
$\breve{\wp}(\hookrightarrow)(X)$	:	$\wp(\mathbb{S})$	by definition
$\lambda l.\{m \mid \langle l, m \rangle \in \breve{\wp}(\hookrightarrow)(X)\}$	:	$\mathbb{L} \to \wp(\mathbb{M})$	partition
$\pi_1 \circ \breve{\wp}(\hookrightarrow)(X)$	:	$\mathbb{L} \to \wp(\mathbb{M})$	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:

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

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.

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

Soundness Proof (1)

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

$$F : \wp(\mathbb{S}) \to \wp(\mathbb{S})$$
$$F(X) = I \cup Step(X)$$

$$F_1^\sharp : (\mathbb{L} \to \wp(\mathbb{M})) \to (\mathbb{L} \to \wp(\mathbb{M}))$$
$$F_1^\sharp(X) = \alpha_1(I) \cup_1^\sharp Step_1^\sharp(X)$$

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

Soundness Proof (2)

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

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

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

$$\{1 \leq x \leq 2,\ 2 \leq y \leq 3,\ y - x = 1\}$$

relational abstraction

Abstract Domain

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

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

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

- Galois connection: $\mathbb{L} \to \wp(\mathbb{M}) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \mathbb{L} \to \mathbb{M}^\sharp$

$$\alpha_2(X) = \lambda l.\alpha_{\mathbb{M}}(X(l))$$
$$\gamma_2(X^\sharp) = \lambda l.\gamma_{\mathbb{M}}(X^\sharp(l))$$

Abstract Domain of Memories

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

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

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

- Galois connection: $\wp(\mathbb{M}) \underset{\alpha_{\mathbb{M}}}{\overset{\gamma_{\mathbb{M}}}{\leftrightarrows}} \mathbb{M}^\sharp$

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

Abstract Domain for Values

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

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

$$\bot \sqcup_{\mathbb{Z}^\sharp} X = X \sqcup_{\mathbb{Z}^\sharp} \bot = X \qquad (\forall X \in \mathbb{Z}^\sharp)$$
$$[l_1, u_1] \sqcup_{\mathbb{Z}^\sharp} [l_2, u_2] = [\min(l_1, l_2), \max(u_1, u_2)]$$

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

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

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

Abstract State Transition

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

$$
\begin{array}{rcl}
x \texttt{ := } E & : & \langle l, m^\sharp \rangle \hookrightarrow^\sharp \langle \texttt{next}(l), update^\sharp_x(m, eval^\sharp_E(m^\sharp)) \rangle \\
\texttt{br } E\ l_1\ l_2 & : & \langle l, m^\sharp \rangle \hookrightarrow^\sharp \langle l_1, filter^\sharp_E(m^\sharp) \rangle \\
 & : & \langle l, m^\sharp \rangle \hookrightarrow^\sharp \langle l_2, filter^\sharp_{\neg E}(m^\sharp) \rangle \\
\texttt{goto } l' & : & \langle l, m^\sharp \rangle \hookrightarrow^\sharp \langle l', m^\sharp \rangle \\
x \texttt{ := input()} & : & \langle l, m^\sharp \rangle \hookrightarrow^\sharp \langle \texttt{next}(l), update^\sharp_x(m^\sharp, \alpha_Z(\mathbb{Z})) \rangle \\
\texttt{print}(x) & : & \langle l, m^\sharp \rangle \hookrightarrow \langle \texttt{next}(l), m^\sharp \rangle
\end{array}
$$

Question: How to design abstract semantic operators?

Best Abstraction

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

$$F^\sharp_{best} = \alpha \circ F \circ \gamma$$

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

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

Design of Abstract Semantic Functions

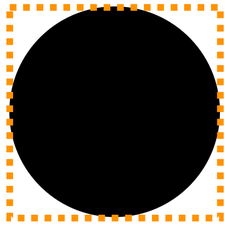
- If the best abstraction can be rewritten using only abstract operators:

$$\begin{aligned} F^\sharp_{best} &= f^\sharp_n \circ \cdots \circ f^\sharp_1 \\ &= F^\sharp \end{aligned}$$

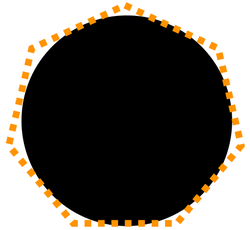
- If the best abstraction cannot be rewritten using only abstract operators:

$$\begin{aligned} F^\sharp_{best} &= f^\sharp_n \circ \cdots \circ \alpha \circ f_m \circ \gamma \circ \cdots \circ f^\sharp_1 \\ &\sqsubseteq f^\sharp_n \circ \cdots \circ f^\sharp_{m'} \circ \cdots \circ f^\sharp_1 \\ &= F^\sharp \end{aligned}$$

- 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:

$$update_x^\sharp : \mathbb{V}^\sharp \times \mathbb{M}^\sharp \to \mathbb{M}^\sharp$$
$$update_x^\sharp(n^\sharp, m^\sharp) = m^\sharp\{x \mapsto n^\sharp\}$$

- The abstract expression-evaluation operation:

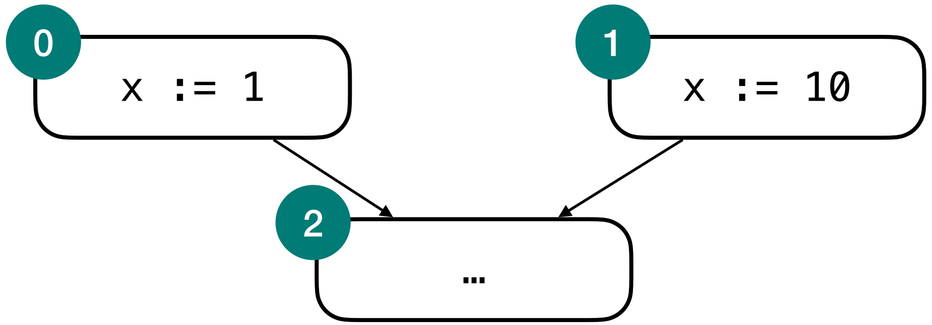
$$eval_E^\sharp : \mathbb{M}^\sharp \to \mathbb{V}^\sharp$$
$$eval_n^\sharp(m) = \alpha_{\mathbb{Z}}(\{n\})$$
$$eval_x^\sharp(m) = m^\sharp(x)$$
$$eval_{E_1 \oplus E_2}^\sharp(m) = eval_{E_1}^\sharp(m^\sharp) \oplus^\sharp eval_{E_2}^\sharp(m^\sharp)$$

- The abstract memory filter operation:

$$filter_E^\sharp : \mathbb{M}^\sharp \to \mathbb{M}^\sharp$$
$$filter_E^\sharp(m^\sharp) = \alpha_{\mathbb{M}}(\{m \in \gamma_{\mathbb{M}}(m^\sharp) \mid eval_E(m) = \mathsf{true}\})$$

Practical Aspect 1: Join

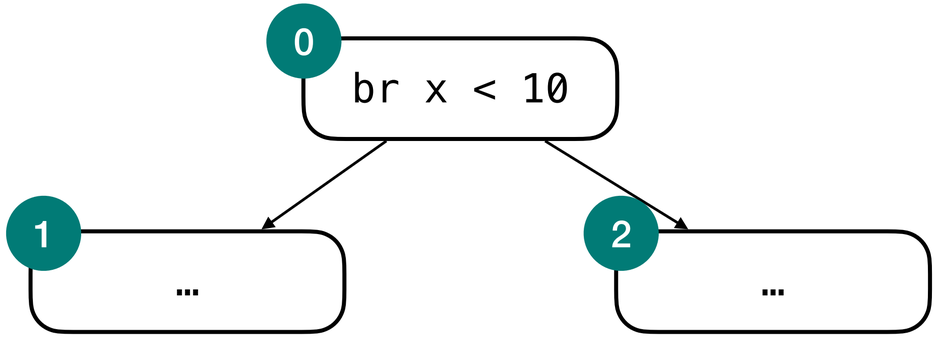
- According to the AI framework, the least upper bound (so-called join) is required to be defined on each chain (i.e., CPO)
- If a join is defined for any non-empty finite set (i.e., join-semilattice), it makes easy to define semantic functions



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

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



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

Example

- Abstract addition:

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

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

- Abstract filter:

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

$$\alpha_{\mathbb{M}}\big(\{m \in \gamma_{\mathbb{M}}(m^\sharp) \mid \mathit{eval}_{x<k}(m) = \mathsf{true}\}\big)$$

Abstract Semantics

- The abstract semantic function:

$$F^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp)$$
$$F^\sharp(X) = (\alpha_2 \circ \alpha_1)(I) \sqcup Step^\sharp(X)$$

- The abstract semantic functions for transition and partition:

$$Step^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp) \qquad \pi : \wp(\mathbb{S}^\sharp) \to (\mathbb{L} \to \wp(\mathbb{M}^\sharp))$$
$$Step^\sharp(X) = \wp(id, \sqcup) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp)(X) \qquad \pi(X) = \lambda l.\{m^\sharp \in \mathbb{M}^\sharp \mid \langle l, m^\sharp \rangle \in X\}$$

- Soundness: $\mathbf{lfp}F \subseteq \gamma_1 \circ \gamma_2(\bigsqcup_{i \geq 0} F^\sharp(\bot))$

Abstract Transition and Partition

$$Step^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp)$$
$$Step^\sharp(X) = \wp(id, \sqcup) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp)(X)$$

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

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

Soundness Proof (1)

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

$$F_1^\sharp : (\mathbb{L} \to \wp(\mathbb{M})) \to (\mathbb{L} \to \wp(\mathbb{M}))$$
$$F_1^\sharp(X) = \alpha_1(I) \cup_1^\sharp Step_1^\sharp(X)$$

$$F^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp)$$
$$F^\sharp(X) = \alpha_2 \circ \alpha_1(I) \sqcup Step^\sharp(X)$$

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

Soundness Proof (2)

$$Step^\sharp_1 \circ \gamma_2 \sqsubseteq^\sharp_1 \gamma_2 \circ Step^\sharp$$

$$
\begin{aligned}
Step^\sharp_1 \circ \gamma_2(X^\sharp) &= \pi_1 \circ \breve{\wp}(\hookrightarrow) \circ \gamma_2(X^\sharp) && \text{(by definition of } Step^\sharp_1) \\
&= \breve{\wp}(\hookrightarrow) \circ \gamma_2(X^\sharp) && \text{(by regarding } \breve{\wp}(\hookrightarrow) \circ \gamma_2(X^\sharp) \text{ as a function of } \mathbb{L} \to \wp(\mathbb{M})) \\
&\sqsubseteq^\sharp_1 \gamma_2 \circ \breve{\wp}(\hookrightarrow^\sharp)(X^\sharp) && \text{(by condition } \breve{\wp}(\hookrightarrow) \circ \gamma_2 \sqsubseteq^\sharp_1 \gamma_2 \circ \breve{\wp}(\hookrightarrow^\sharp) \text{ and} \\
& && \text{by regarding the input of } \gamma_2 \text{ as a subset of } \mathbb{L} \times \mathbb{M}^\sharp) \\
&\sqsubseteq^\sharp_1 \gamma_2 \circ \wp(id, \sqcup_{\mathbb{M}}) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp)(X^\sharp) && \text{(by monotonicity of } \gamma_2) \\
&= \gamma_2 \circ Step^\sharp(X^\sharp) && \text{(by definition of } Step^\sharp)
\end{aligned}
$$

new subgoal

Soundness Proof (2-1)

$$\breve{\wp}(\hookrightarrow) \circ \gamma_2 \sqsubseteq_1^\sharp \gamma_2 \circ \breve{\wp}(\hookrightarrow^\sharp)$$

Proof sketch: The abstract transition relation $\hookrightarrow^\sharp$ is the same as the concrete transition $\hookrightarrow$ except that it uses the abstract correspondents for semantic operators:

$$\begin{array}{rcl}
x \texttt{ := } E & : & \langle l, m\rangle \hookrightarrow \langle \texttt{next}(l), update_x(m, eval_E(m))\rangle \\
\texttt{br } E\ l_1\ l_2 & : & \langle l, m\rangle \hookrightarrow \langle l_1, filter_E(m)\rangle \\
 & : & \langle l, m\rangle \hookrightarrow \langle l_2, filter_{\neg E}(m)\rangle \\
\texttt{goto } l' & : & \langle l, m\rangle \hookrightarrow \langle l', m\rangle \\
x \texttt{ := input()} & : & \langle l, m\rangle \hookrightarrow \langle \texttt{next}(l), update_x(m, z)\rangle \\
\texttt{print}(x) & : & \langle l, m\rangle \hookrightarrow \langle \texttt{next}(l), m\rangle
\end{array}$$

$$\begin{array}{rcl}
x \texttt{ := } E & : & \langle l, m^\sharp\rangle \hookrightarrow^\sharp \langle \texttt{next}(l), update^\sharp_x(m, eval^\sharp_E(m^\sharp))\rangle \\
\texttt{br } E\ l_1\ l_2 & : & \langle l, m^\sharp\rangle \hookrightarrow^\sharp \langle l_1, filter^\sharp_E(m^\sharp)\rangle \\
 & : & \langle l, m^\sharp\rangle \hookrightarrow^\sharp \langle l_2, filter^\sharp_{\neg E}(m^\sharp)\rangle \\
\texttt{goto } l' & : & \langle l, m^\sharp\rangle \hookrightarrow^\sharp \langle l', m^\sharp\rangle \\
x \texttt{ := input()} & : & \langle l, m^\sharp\rangle \hookrightarrow^\sharp \langle \texttt{next}(l), update^\sharp_x(m^\sharp, \alpha_Z(\mathbb{Z}))\rangle \\
\texttt{print}(x) & : & \langle l, m^\sharp\rangle \hookrightarrow^\sharp \langle \texttt{next}(l), m^\sharp\rangle
\end{array}$$

Therefore, it is enough to prove the soundness of each semantic operator:

$$\begin{array}{rcl}
\wp(eval_E) \circ \gamma_{\mathbb{M}} & \sqsubseteq & \gamma_{\mathbb{V}} \circ eval^\sharp_E \\
\wp(update_x) \circ \times \circ (\gamma_{\mathbb{M}}, \gamma_{\mathbb{V}}) & \sqsubseteq & \gamma_{\mathbb{M}} \circ update^\sharp_x \\
\wp(filter_E) \circ \gamma_{\mathbb{M}} & \sqsubseteq & \gamma_{\mathbb{M}} \circ filter^\sharp_E \\
\wp(filter_{\neg E}) \circ \gamma_{\mathbb{M}} & \sqsubseteq & \gamma_{\mathbb{M}} \circ filter^\sharp_{\neg E}
\end{array}$$

Soundness Proof (3)

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

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

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:

$$\begin{array}{rcll} [a,b] & \triangledown & \bot & = & [a,b] \\ \bot & \triangledown & [c,d] & = & [c,d] \\ [a,b] & \triangledown & [c,d] & = & [(c < a ? -\infty : a), (b < d ? +\infty : b)] \end{array}$$

Check: Safety conditions for widening

  - A simple narrowing operator for the interval domain:

$$\begin{array}{rcll} [a,b] & \triangle & \bot & = & \bot \\ \bot & \triangle & [c,d] & = & \bot \\ [a,b] & \triangle & [c,d] & = & [(a = -\infty ? c : a), (b = +\infty ? d : b)] \end{array}$$

Check: Safety conditions for narrowing

Basic Fixpoint Algorithm

- For finite-height abstract domains

$X, R : \mathbb{D}^\sharp$
$F^\sharp : \mathbb{D}^\sharp \to \mathbb{D}^\sharp$
begin
  $X \leftarrow \bot$
  repeat
    $R \leftarrow X$
    $X \leftarrow F^\sharp(X)$
  until $X \sqsubseteq R$
  return R
end

- For infinite-height abstract domains

$X, R : \mathbb{D}^\sharp$
$F^\sharp : \mathbb{D}^\sharp \to \mathbb{D}^\sharp$
begin
  $X \leftarrow \bot$
  repeat
    $R \leftarrow X$
    $X \leftarrow X \triangledown F^\sharp(X)$
  until $X \sqsubseteq R$
  return R
end

Basic Fixpoint Algorithm (Inlined)

$X, R : \mathbb{L} \to \mathbb{M}^\sharp$

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

begin

  $X \leftarrow \bot$

  repeat

    $R \leftarrow X$

    $X \leftarrow X \triangledown \big(\underbrace{\wp(id, \sqcup_{\mathbb{M}}) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp)}_{F^\sharp}\big)(X)$

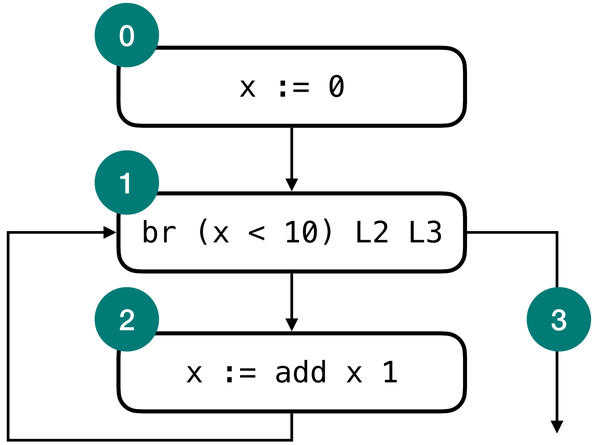
  until $X \sqsubseteq R$

  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



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

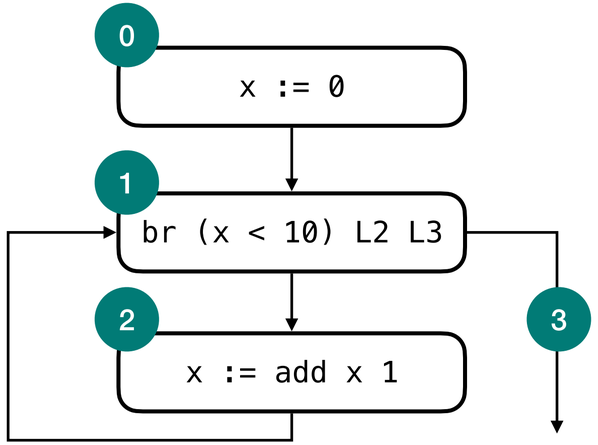
[0,0] ▽ [0,1] = [0,+∞]

[0,+∞] ⊓ [-∞,9] = [0,9]
[0,0] ▽ [0,9] = [0,+∞]

[0,+∞] ⊓ [10,+∞] = [10,+∞]

Fixed Point!

Questions

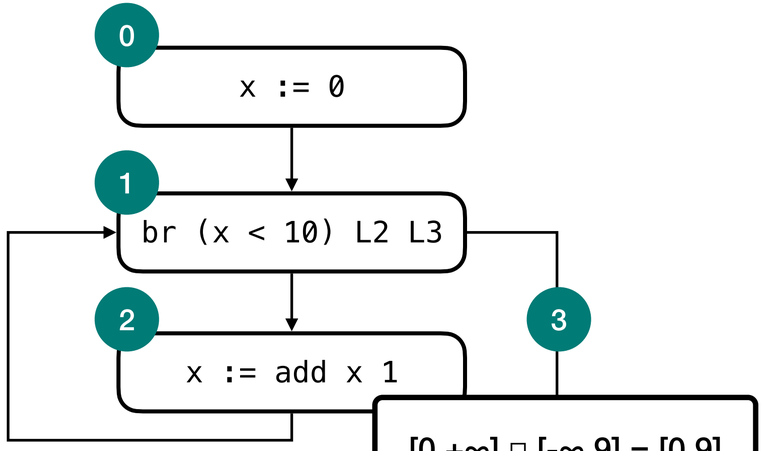


2. Inefficiency? (#labels / iter ?)

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



[0,+∞] ⊓ [-∞,9] = [0,9]
[0,+∞] △ [0,9] = [0,9]

[0,0] ⊔ [1,10] = [0,10]
[0,+∞] △ [0,10] = [0,10]

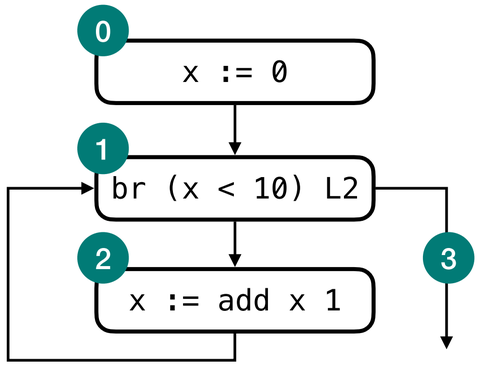
[0,10] ⊓ [10,+∞] = [10,10]
[10,+∞] △ [10,10] = [10,10]

Label	Iter 0	Iter 1	Iter 2	Iter 3	Iter 4
0	⊥	⊥	⊥	⊥	⊥
1	[0,+∞]	[0,+∞]	[0,10]	[0,10]	[0,10]
2	[0,+∞]	[0,9]	[0,9]	[0,9]	[0,9]
3	[10,+∞]	[10,+∞]	[10,+∞]	[10,10]	[10,10]

Fixed Point!

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



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,+∞]

No more work!
= Fixed Point!

Final Algorithm

- Worklist algorithm with widening

$X : \mathbb{L} \to \mathbb{M}^\sharp$
$F^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp)$
$Worklist : \wp(\mathbb{L})$
begin
  $Worklist \leftarrow \mathbb{L}$
  $X \leftarrow \bot$
  repeat
    $(w, Worklist) \leftarrow \mathsf{pop}(Worklist)$
    $m^\sharp_{old} \leftarrow X(w)$
    $m^\sharp_{new} \leftarrow \bigsqcup\{m^\sharp_{in} \mid \langle l, X(l)\rangle \hookrightarrow^\sharp \langle w, m^\sharp_{in}\rangle\}$
    $m^\sharp_{new} \leftarrow m^\sharp_{old} \triangledown m^\sharp_{new}$
    if $m^\sharp_{new} \not\sqsubseteq m^\sharp_{old}$ then
      $X(w) \leftarrow m^\sharp_{new}$
      $Worklist \leftarrow Worklist \cup \{l \mid \langle w, m^\sharp_{new}\rangle \hookrightarrow^\sharp \langle l, \_\rangle\}$
    endif
  until $Worklist = \emptyset$
  return X
end

- Worklist algorithm with narrowing

$X : \mathbb{L} \to \mathbb{M}^\sharp$
$F^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp)$
$Worklist : \wp(\mathbb{L})$
begin
  $Worklist \leftarrow \mathbb{L}$
  $X \leftarrow \bot$
  repeat
    $(w, Worklist) \leftarrow \mathsf{pop}(Worklist)$
    $m^\sharp_{old} \leftarrow X(w)$
    $m^\sharp_{new} \leftarrow \bigsqcup\{m^\sharp_{in} \mid \langle l, X(l)\rangle \hookrightarrow^\sharp \langle w, m^\sharp_{in}\rangle\}$
    $m^\sharp_{new} \leftarrow m^\sharp_{old} \triangle m^\sharp_{new}$
    if $m^\sharp_{new} \not\sqsupseteq m^\sharp_{old}$ then
      $X(w) \leftarrow m^\sharp_{new}$
      $Worklist \leftarrow Worklist \cup \{l \mid \langle w, m^\sharp_{new}\rangle \hookrightarrow^\sharp \langle l, \_\rangle\}$
    endif
  until $Worklist = \emptyset$
  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