Outline

Transitional Semantics

State transition sequence

$$s_0 \hookrightarrow s_1 \hookrightarrow s_2 \hookrightarrow \cdots$$

where $\hookrightarrow$ is a transition relation between states $\mathbb{S}$

$$\hookrightarrow \subseteq \mathbb{S} \times \mathbb{S}$$

A state $s \in \mathbb{S}$ of the program is a pair (l, m) of a program label l and the machine state m at that program label during execution.

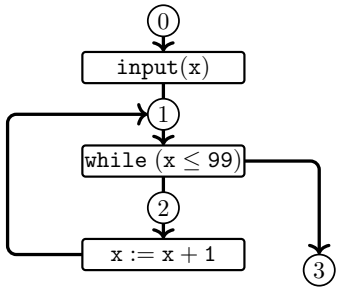
Concrete Transition Sequence

Example

Consider the following program

$$\begin{array}{l}\texttt{input(x);}\\ \texttt{while } (\texttt{x} \leq 99)\\ \qquad \{\texttt{x} := \texttt{x} + 1\}\end{array}$$

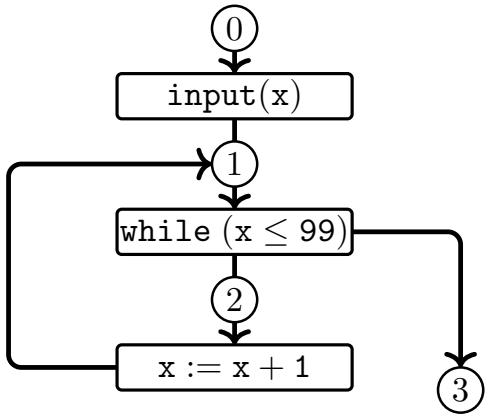
Let labels be "program points."



Let the initial state be $\emptyset$. Some transition sequences are:

For input 100: $(0, \emptyset) \hookrightarrow (1, x \mapsto 100) \hookrightarrow (3, x \mapsto 100)$.
For input 99: $(0, \emptyset) \hookrightarrow (1, x \mapsto 99) \hookrightarrow (2, x \mapsto 99) \hookrightarrow (1, x \mapsto 100) \hookrightarrow (3, x \mapsto 100)$.
For input 0: $(0, \emptyset) \hookrightarrow (1, x \mapsto 0) \hookrightarrow (2, x \mapsto 0) \hookrightarrow (1, x \mapsto 1) \hookrightarrow \cdots \hookrightarrow (3, x \mapsto 100)$.

Reachable States



Assume that the possible inputs are 0, 99, and 100. Then, the set of all reachable states are the set of states occurring in the three transition sequences:

$$
\begin{array}{ll}
 & \{(0,\emptyset),(1,x\mapsto 100),(3,x\mapsto 100)\} \\
\cup & \{(0,\emptyset),(1,x\mapsto 99),(2,x\mapsto 99),(1,x\mapsto 100),(3,x\mapsto 100)\} \\
\cup & \{(0,\emptyset),(1,x\mapsto 0),(2,x\mapsto 0),(1,x\mapsto 1),\cdots,(2,x\mapsto 99),(1,x\mapsto 100),(3,x\mapsto 100)\} \\
= & \{(0,\emptyset),(1,x\mapsto 0),\cdots,(1,x\mapsto 100),(2,x\mapsto 0),\cdots,(2,x\mapsto 99),(3,x\mapsto 100)\}
\end{array}
$$

Concrete Semantics: the Set of Reachable States (1/3)

Given a program, let I be the set of its initial states and *Step* be the powerset-lifted version of $\hookrightarrow$:

$$\begin{array}{l} \mathit{Step} : \wp(\mathbb{S}) \to \wp(\mathbb{S}) \\ \mathit{Step}(X) = \{s' \mid s \hookrightarrow s', s \in X\} \end{array}$$

The set of reachable states is

$$I \cup \mathit{Step}^1(I) \cup \mathit{Step}^2(I) \cup \cdots .$$

which is, equivalently, the limit of C_is

$$\begin{array}{rcl} C_0 & = & I \\ C_{i+1} & = & I \cup \mathit{Step}(C_i) \end{array}$$

which is, the least solution of

$$X = I \cup \mathit{Step}(X).$$

Concrete Semantics: the Set of Reachable States (2/3)

The least solution of

$$X = I \cup \mathit{Step}(X)$$

is also called *the least fixpoint* of F

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

written as

$$\mathbf{lfp} F.$$

Theorem (Least fixpoint)

The least fixpoint $\mathbf{lfp} F$ *of* $F(X) = I \cup \mathit{Step}(X)$ *is*

$$\bigcup_{i \geq 0} F^i(\emptyset)$$

where $F^0(X) = X$ *and* $F^{n+1}(X) = F(F^n(X))$.

Concrete Semantics: the Set of Reachable States (3/3)

Definition (Concrete semantics, the set of reachable states)

Given a program, let $\mathbb{S}$ be the set of states and $\hookrightarrow$ be the one-step transition relation $\subseteq \mathbb{S} \times \mathbb{S}$. Let I be the set of its initial states and *Step* be the powerset-lifted version of $\hookrightarrow$:

$$\begin{array}{l} \mathit{Step} : \wp(\mathbb{S}) \to \wp(\mathbb{S}) \\ \mathit{Step}(X) = \{s' \mid s \hookrightarrow s', s \in X\}. \end{array}$$

Then the concrete semantics of the program, the set of all reachable states from I, is defined as the least fixpoint $\mathbf{lfp}F$ of F

$$F(X) = I \cup \mathit{Step}(X).$$

Analysis Goal

Program-label-wise reachability

For each program label we want to know the set of memories that can occur at that label during executions of the input program.

- labels: "partitioning indices"
- e.g., statement labels as in programs, statement labels after loop unrolling, statement labels after function inlining

Abstract Semantics

Define the abstract semantics "homomorphically":

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

The forthcoming framework will guide us

- conditions for $\mathbb{S}^\sharp$ and $F^\sharp$
- so that the abstract semantics is finitely computable and is an upper-approximation of concrete semantics $\mathbf{lfp}F$.

Abstraction of the Semantic Domain $\wp(\mathbb{S})$ (1/2)

$$\wp(\mathbb{S}) \quad \text{where} \quad \mathbb{S} = \mathbb{L} \times \mathbb{M}$$

Label-wise (two-step) abstraction of states:

set of states $\wp(\mathbb{L} \times \mathbb{M})$ $\xrightarrow{\text{abstraction}}$ label-wise collect $\mathbb{L} \to \wp(\mathbb{M})$ $\xrightarrow{\text{abstraction}}$ label-wise abstraction $\mathbb{L} \to \mathbb{M}^\sharp$.

Abstraction of the Semantic Domain $\wp(\mathbb{S})$ (2/2)

$\wp(\mathbb{L} \times \mathbb{M}) \ni$ collection of all states

$$\begin{cases} (0, m_0), (0, m_0'), \cdots, & \text{at } 0 \\ (1, m_1), (1, m_1'), \cdots, & \text{at } 1 \\ \quad \vdots & \\ (n, m_n), (n, m_n'), \cdots. & \text{at } n \end{cases}$$

$\mathbb{L} \to \wp(\mathbb{M}) \ni$ label-wise collection

$$\begin{cases} (0, \{m_0, m_0', \cdots\}) \\ (1, \{m_1, m_1', \cdots\}) \\ \quad \vdots \\ (n, \{m_n, m_n', \cdots\}) \end{cases}$$

$\mathbb{L} \to \mathbb{M}^\sharp \ni$ label-wise abstraction

$$\begin{cases} (0, M_0^\sharp) \\ (1, M_1^\sharp) \\ \quad \vdots \\ (n, M_n^\sharp) \end{cases}$$

Each $M_l^\sharp$ over-approximates the set $\{m_l, m_l', \cdots\}$ collected at label l.

Preliminary for Abstract Domains (1/3)

- Define an abstract domain as a *CPO*
  - a partial order set
  - has a least element $\bot$
  - has a least-upper bound for every *chain*
- An abstract domain as $\sqcup$-semilattices also work.

Preliminary for Abstract Domains (2/3)

Abstract and concrete domains are structured "consistently".

Definition (Galois connection)

A *Galois connection* is a pair made of a concretization function γ and an abstraction function α such that:

$$\forall c \in \mathbb{C},\ \forall a \in \mathbb{A}, \qquad \alpha(c) \sqsubseteq a \qquad \Longleftrightarrow \qquad c \subseteq \gamma(a)$$

We write such a pair as follows:

$$(\mathbb{C}, \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{A}, \sqsubseteq)$$

Preliminary for Abstract Doamins (3/3)

Galois-connection properties we rely on:
For

$$(\mathbb{C}, \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{A}, \sqsubseteq)$$

- α and γ are monotone functions
- $\forall c \in \mathbb{C},\ c \subseteq \gamma(\alpha(c))$
- $\forall a \in \mathbb{A},\ \alpha(\gamma(a)) \sqsubseteq a$
- If both $\mathbb{C}$ and $\mathbb{A}$ are CPOs, then α is continuous.

(Proofs are in the book.)

Abstract Domains (1/2)

Design an abstract domain as a CPO that is Galois-connected with the concrete domain:

$$(\wp(\mathbb{L} \times \mathbb{M}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{L} \to \mathbb{M}^\sharp, \sqsubseteq).$$

- Abstraction α defines how each concrete elmt (set of concrete states) is abstracted into an abstract elmt.
- Concretization γ defines the set of concrete states implied by each abstract state.
- Partial order $\sqsubseteq$ is the label-wise order:

$$a^\sharp \sqsubseteq b^\sharp \quad \text{iff} \quad \forall l \in \mathbb{L} : a^\sharp(l) \sqsubseteq_M b^\sharp(l)$$

where $\sqsubseteq_M$ is the partial order of $\mathbb{M}^\sharp$.

Abstract Domains (2/2)

The above Galois connection (abstraction)

$$(\wp(\mathbb{L} \times \mathbb{M}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{L} \to \mathbb{M}^\sharp, \sqsubseteq).$$

composes two Galois connections:

$$\begin{array}{l} (\wp(\mathbb{L} \times \mathbb{M}), \subseteq) \\ \underset{\alpha_0}{\overset{\gamma_0}{\leftrightarrows}} \quad (\mathbb{L} \to \wp(\mathbb{M}), \sqsubseteq) \quad (\sqsubseteq \text{ is the label-wise } \subseteq) \\ \qquad \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} (\mathbb{L} \to \mathbb{M}^\sharp, \sqsubseteq) \quad (\sqsubseteq \text{ is the label-wise } \sqsubseteq_M) \end{array}$$

$$\alpha_0 \left\{ \begin{array}{l} (0, m_0), (0, m_0'), \cdots, \\ \quad \vdots \\ (n, m_n), (n, m_n'), \cdots \end{array} \right\} = \left\{ \begin{array}{l} (0, \{m_0, m_0', \cdots\}), \\ \quad \vdots \\ (n, \{m_n, m_n', \cdots\}) \end{array} \right\}$$

$$\alpha_1 \left\{ \begin{array}{l} (0, \{m_0, m_0', \cdots\}), \\ \quad \vdots \\ (n, \{m_n, m_n', \cdots\}) \end{array} \right\} = \left\{ \begin{array}{l} (0, M_0^\sharp), \\ \quad \vdots \\ (n, M_n^\sharp) \end{array} \right\}$$

Thus, boils down to

$$(\wp(\mathbb{M}), \subseteq) \underset{\alpha_M}{\overset{\gamma_M}{\leftrightarrows}} (\mathbb{M}^\sharp, \sqsubseteq_M).$$

Abstract Semantic Functions

Let

$$(\wp(\mathbb{L} \times \mathbb{M}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{L} \to \mathbb{M}^\sharp, \sqsubseteq).$$

A concrete semantic function F	An abstract semantic function $F^\sharp$
$\mathbb{S} = \mathbb{L} \times \mathbb{M}$	$\mathbb{S}^\sharp = \mathbb{L} \to \mathbb{M}^\sharp$
$F : \wp(\mathbb{S}) \to \wp(\mathbb{S})$	$F^\sharp : \mathbb{S}^\sharp \to \mathbb{S}^\sharp$
$F(X) = I \cup \mathit{Step}(X)$	$F^\sharp(X^\sharp) = \alpha(I) \cup^\sharp \mathit{Step}^\sharp(X^\sharp)$
$\mathit{Step} = \breve{\wp}(\hookrightarrow)$	$\mathit{Step}^\sharp = \wp(\mathrm{id}, \cup^\sharp_M) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp)$
$\hookrightarrow \subseteq (\mathbb{L} \times \mathbb{M}) \times (\mathbb{L} \times \mathbb{M})$	$\hookrightarrow^\sharp \subseteq (\mathbb{L} \times \mathbb{M}^\sharp) \times (\mathbb{L} \times \mathbb{M}^\sharp)$

with relations $\hookrightarrow$ and $\hookrightarrow^\sharp$ being functions

As of $\mathit{Step}^\sharp = \wp(\mathrm{id}, \cup^\sharp_M) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp)$

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

- Abstract transition $\breve{\wp}(\hookrightarrow^\sharp)$:
  - a set $\subseteq \mathbb{L} \times \mathbb{M}^\sharp$ $\mapsto$ a set $\subseteq \mathbb{L} \times \mathbb{M}^\sharp$
- Paritioning π:
  - a set $\subseteq \mathbb{L} \times \mathbb{M}^\sharp$ $\mapsto$ a set $\subseteq \mathbb{L} \times \wp(\mathbb{M}^\sharp)$
- Joining $\wp(\mathrm{id}, \cup^\sharp_M)$:
  - a set $\subseteq \mathbb{L} \times \wp(\mathbb{M}^\sharp)$ $\mapsto$ an abstract state $\in \mathbb{L} \to \mathbb{M}^\sharp$

Example

Suppose the program has two labels l_1 and l_2. That is, $\mathbb{L} = \{l_1, l_2\}$. Given an abstract state $\{(l_1, M_1^\sharp), (l_2, M_2^\sharp)\}$, *Step*$^\sharp$ first applies $\breve{\wp}(\hookrightarrow^\sharp)$ to it:

$$\hookrightarrow^\sharp(l_1, M_1^\sharp) \cup \hookrightarrow^\sharp(l_2, M_2^\sharp).$$

Example

Suppose the program has two labels l_1 and l_2. That is, $\mathbb{L} = \{l_1, l_2\}$. Given an abstract state $\{(l_1, M_1^\sharp), (l_2, M_2^\sharp)\}$, *Step*$^\sharp$ first applies $\breve{\wp}(\hookrightarrow^\sharp)$ to it:

$$\hookrightarrow^\sharp(l_1, M_1^\sharp) \cup \hookrightarrow^\sharp(l_2, M_2^\sharp).$$

Suppose the result is

$$\{(l_1, M'^\sharp_1), (l_2, M''^\sharp_1), (l_1, M'^\sharp_2)\}.$$

Example

Suppose the program has two labels l_1 and l_2. That is, $\mathbb{L} = \{l_1, l_2\}$. Given an abstract state $\{(l_1, M_1^\sharp), (l_2, M_2^\sharp)\}$, *Step*$^\sharp$ first applies $\breve{\wp}(\hookrightarrow^\sharp)$ to it:

$$\hookrightarrow^\sharp(l_1, M_1^\sharp) \cup \hookrightarrow^\sharp(l_2, M_2^\sharp).$$

Suppose the result is

$$\{(l_1, M'^\sharp_1), (l_2, M''^\sharp_1), (l_1, M'^\sharp_2)\}.$$

By the subsequent partitioning operator π, the result becomes

$$\{(l_1, \{M'^\sharp_1, M'^\sharp_2\}), (l_2, \{M''^\sharp_1\})\}.$$

Example

Suppose the program has two labels l_1 and l_2. That is, $\mathbb{L} = \{l_1, l_2\}$. Given an abstract state $\{(l_1, M_1^\sharp), (l_2, M_2^\sharp)\}$, *Step*$^\sharp$ first applies $\breve{\wp}(\hookrightarrow^\sharp)$ to it:

$$\hookrightarrow^\sharp(l_1, M_1^\sharp) \cup \hookrightarrow^\sharp(l_2, M_2^\sharp).$$

Suppose the result is

$$\{(l_1, M'^\sharp_1), (l_2, M''^\sharp_1), (l_1, M'^\sharp_2)\}.$$

By the subsequent partitioning operator π, the result becomes

$$\{(l_1, \{M'^\sharp_1, M'^\sharp_2\}), (l_2, \{M''^\sharp_1\})\}.$$

The final organization operation $\wp(\mathrm{id}, \cup^\sharp_M)$ returns the post abstract state $\in \mathbb{L} \to \mathbb{M}^\sharp$:

$$\{(l_1, M'^\sharp_1 \cup^\sharp_M M'^\sharp_2), (l_2, M''^\sharp_1)\}.$$

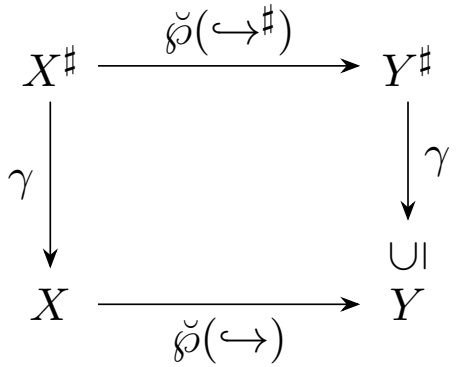
Conditions for Sound $\hookrightarrow^\sharp$ and $\cup\!\!\!\!\cup^\sharp_{\_}$

- sound condition for $\hookrightarrow^\sharp$:

$$\breve{\wp}(\hookrightarrow) \circ \gamma \;\subseteq\; \gamma \circ \breve{\wp}(\hookrightarrow^\sharp)$$

- sound condition for $\cup\!\!\!\!\cup_{\_}^\sharp$:

$$\cup \circ (\gamma, \gamma) \;\subseteq\; \gamma \circ \cup\!\!\!\!\cup^\sharp_{\_}$$



Pattern for the sound condition for each semantic operator $f^\sharp : A^\sharp \to B^\sharp$

$$f \circ \gamma_A \sqsubseteq_B \gamma_B \circ f^\sharp.$$

Then, Follows Sound Static Analysis

- In case $\mathbb{S}^\sharp$ is of finite-height and $F^\sharp$ is monotone or extensive, then

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

  is finitely computable and over-approximates the concrete semantics $\mathbf{lfp}F$.
- Otherwise, find a widening operator $\bigtriangledown$, then the following chain $X_0 \sqsubseteq X_1 \sqsubseteq \cdots$

$$X_0 = \bot \qquad X_{i+1} = X_i \bigtriangledown F^\sharp(X_i)$$

  is finite and its last element over-approximates the concrete semantics $\mathbf{lfp}F$.

Underlying Theorems (1/2)

Theorem (Sound static analysis by $F^\sharp$)

Given a program, let F and $F^\sharp$ be defined as in the framework. If $\mathbb{S}^\sharp$ is of finite-height (every chain $\mathbb{S}^\sharp$ is finite) and $F^\sharp$ is monotone or extensive, then

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

is finitely computable and over-approximates $\mathbf{lfp}F$*:*

$$\mathbf{lfp}F \subseteq \gamma(\bigsqcup_{i \geq 0} F^{\sharp i}(\bot)) \quad \textit{or equivalently} \quad \alpha(\mathbf{lfp}F) \sqsubseteq \bigsqcup_{i \geq 0} F^{\sharp i}(\bot).$$

(Proof is in the book.)

Underlying Theorems (2/2)

Theorem (Sound static analysis by $F^\sharp$ and widening operator ∇)

Given a program, let F and $F^\sharp$ be defined as in the framework. Let ∇ be a widening operator. Then the following chain $Y_0 \sqsubseteq Y_1 \sqsubseteq \cdots$

$$Y_0 = \bot \qquad Y_{i+1} = Y_i \nabla F^\sharp(Y_i)$$

is finite and its last element $Y_{\lim}$ over-approximates $\mathbf{lfp}F$:

$$\mathbf{lfp}F \subseteq \gamma(Y_{\lim}) \quad \textit{or equivalently} \quad \alpha(\mathbf{lfp}F) \sqsubseteq Y_{\lim}.$$

(Proof is in the book.)

Definition (Widening operator)

A *widening* operator over an abstract domain $\mathbb{A}$ is a binary operator ∇, such that:

1. For all abstract elements a_0, a_1, we have

$$\gamma(a_0) \cup \gamma(a_1) \subseteq \gamma(a_0 \nabla a_1)$$

2. For all sequence $(a_n)_{n \in \mathbb{N}}$ of abstract elements, the sequence $(a'_n)_{n \in \mathbb{N}}$ defined below is finitely stationary:

$$\begin{cases} a'_0 & = & a_0 \\ a'_{n+1} & = & a'_n \nabla a_n \end{cases}$$

Analysis Algorithm Based on Global Iterations: Basic Version (1/2)

- Case: $\mathbb{S}^\sharp$ is of finite-height and $F^\sharp$ is monotone or extensive
- Note the increasing chain

$$\bot \sqsubseteq (F^\sharp)^1(\bot) \sqsubseteq (F^\sharp)^2(\bot) \sqsubseteq \cdots$$

  is finite and its biggest element is equal to

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

$\mathtt{C} \leftarrow \bot$
repeat
  $\mathtt{R} \leftarrow \mathtt{C}$
  $\mathtt{C} \leftarrow F^\sharp(\mathtt{C})$
until $\mathtt{C} \sqsubseteq \mathtt{R}$
return $\mathtt{R}$

Analysis Algorithm Based on Global Iterations: Basic Version (2/2)

- Case: $\mathbb{S}^\sharp$ is of infinite-height or $F^\sharp$ is neither monotonic nor extensive
- Use a widening operator $\bigtriangledown$

$\mathtt{C} \leftarrow \bot$
repeat
    $\mathtt{R} \leftarrow \mathtt{C}$
    $\mathtt{C} \leftarrow \mathtt{C} \bigtriangledown F^\sharp(\mathtt{C})$
until $\mathtt{C} \sqsubseteq \mathtt{R}$
return $\mathtt{R}$

Inefficiency of the Basic Algorithms

Recall the algirthm with $F^\sharp(\mathtt{C})$ being inlined:

$$
\begin{array}{l}
\mathtt{C} \leftarrow \bot \\
\text{repeat} \\
\quad \mathtt{R} \leftarrow \mathtt{C} \\
\quad \mathtt{C} \leftarrow \mathtt{C} \bigtriangledown \underbrace{(\wp(\mathrm{id}, \cup^\sharp_M) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp))}_{F^\sharp}(\mathtt{C}) \\
\text{until } \mathtt{C} \sqsubseteq \mathtt{R} \\
\text{return } \mathtt{R}
\end{array}
$$

- $|\mathtt{C}| \sim$ the number of labels in the input program!
- Better apply

$$\breve{\wp}(\hookrightarrow^\sharp)(\mathtt{C})$$

  only to necessary labels

Analysis Algorithm Based on Global Iterations: Worklist Version

- worklist: the set of labels whose input memories are changed in the previous iteration

$\mathtt{C} : \mathbb{L} \to \mathbb{M}^\sharp$
$F^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp)$
$\mathtt{WorkList} : \wp(\mathbb{L})$

$\mathtt{WorkList} \leftarrow \mathbb{L}$
$\mathtt{C} \leftarrow \bot$
repeat
    $\mathtt{R} \leftarrow \mathtt{C}$
    $\mathtt{C} \leftarrow \mathtt{C} \bigtriangledown F^\sharp(\mathtt{C}|_{\mathtt{WorkList}})$
    $\mathtt{WorkList} \leftarrow \{l \mid \mathtt{C}(l) \not\sqsubseteq \mathtt{R}(l), l \in \mathbb{L}\}$
until $\mathtt{WorkList} = \emptyset$
return R

Improvement of the Worklist Algorithm

- Inefficient: $\texttt{WorkList} \leftarrow \{l \mid \texttt{C}(l) \not\sqsubseteq \texttt{R}(l), l \in \mathbb{L}\}$ re-scans all the labels.
  - Better: At application $\hookrightarrow^\sharp$ to $(l, \texttt{C}(l))$, if its result $(l', M^\sharp)$ is changed ($M^\sharp \not\sqsubseteq \texttt{C}(l')$), add l' to the worklist.
- Inefficient: $\texttt{C} \bigtriangledown F^\sharp(C|_{\texttt{WorkList}})$ widens at all the labels.
  - Better: Apply $\bigtriangledown$ only at the target of a loop. Use $\cup^\sharp$ at other labels.

Summary: Recipe for Defining Sound Static Analysis(1/4)

1. Define $\mathbb{M}$ to be the set of memory states that can occur during program executions. Let $\mathbb{L}$ be the finite and fixed set of labels of a given program.
2. Define a concrete semantics as the $\mathbf{lfp}F$ where

concrete domain	$\wp(\mathbb{S})$	$=$	$\wp(\mathbb{L} \times \mathbb{M})$
concrete semantic function	$F : \wp(\mathbb{S}) \to \wp(\mathbb{S})$		
	$F(X)$	$=$	$I \cup \mathit{Step}(X)$
	Step	$=$	$\breve{\wp}(\hookrightarrow)$
	$\hookrightarrow$	$\subseteq$	$(\mathbb{L} \times \mathbb{M}) \times (\mathbb{L} \times \mathbb{M})$

The $\hookrightarrow$ is the one-step transition relation over $\mathbb{L} \times \mathbb{M}$.

Summary: Recipe for Defining Sound Static Analysis(2/4)

3. Define its abstract domain and abstract semantic function as

$$
\begin{array}{rlll}
\text{abstract domain} & \mathbb{S}^\sharp & = & \mathbb{L} \to \mathbb{M}^\sharp \\
\text{abstract semantic function} & F^\sharp : \mathbb{S}^\sharp \to \mathbb{S}^\sharp & & \\
& F^\sharp(X^\sharp) & = & \alpha(I) \cup^\sharp \mathit{Step}^\sharp(X^\sharp) \\
& \mathit{Step}^\sharp & = & \wp(\mathrm{id}, \cup^\sharp_M) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp) \\
& \hookrightarrow^\sharp & \subseteq & (\mathbb{L} \times \mathbb{M}^\sharp) \times (\mathbb{L} \times \mathbb{M}^\sharp)
\end{array}
$$

The $\hookrightarrow^\sharp$ is the one-step abstract transition relation over $\mathbb{L} \times \mathbb{M}^\sharp$. Function π partitions a set $\subseteq \mathbb{L} \times \mathbb{M}^\sharp$ by the labels in $\mathbb{L}$ returning an element in $\mathbb{L} \to \wp(\mathbb{M}^\sharp)$ represented as a set $\subseteq \mathbb{L} \times \wp(\mathbb{M}^\sharp)$.

Summary: Recipe for Defining Sound Static Analysis(3/4)

4. Check the abstract domains $\mathbb{S}^\sharp$ and $\mathbb{M}^\sharp$ are CPOs, and forms a Galois-connection respectively with $\wp(\mathbb{S})$ and $\wp(\mathbb{M})$:

$$(\wp(\mathbb{S}), \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{S}^\sharp, \sqsubseteq) \quad \text{and} \quad (\wp(\mathbb{M}), \subseteq) \underset{\alpha_M}{\overset{\gamma_M}{\leftrightarrows}} (\mathbb{M}^\sharp, \sqsubseteq_M)$$

where the partial order $\sqsubseteq$ of $\mathbb{S}^\sharp$ is label-wise $\sqsubseteq_M$:

$$a^\sharp \sqsubseteq b^\sharp \quad \text{iff} \quad \forall l \in \mathbb{L} : a^\sharp(l) \sqsubseteq_M b^\sharp(l).$$

5. Check the abstract one-step transition $\hookrightarrow^\sharp$ and abstract union $\cup^\sharp_\_$ satisfy:

$$\begin{array}{lcl} \breve{\wp}(\hookrightarrow) \circ \gamma & \subseteq & \gamma \circ \breve{\wp}(\hookrightarrow^\sharp) \\ \cup \circ (\gamma, \gamma) & \subseteq & \gamma \circ \cup^\sharp_\_ \end{array}$$

Summary: Recipe for Defining Sound Static Analysis(4/4)

6. Then, sound static analysis is defined as follows:
   - In case $\mathbb{S}^\sharp$ is of finite-height (every its chain is finite) and $F^\sharp$ is monotone or extensive, then

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

     is finitely computable and over-approximates the concrete semantics $\mathbf{lfp}F$.
   - Otherwise, find a widening operator $\bigtriangledown$, then the following chain $X_0 \sqsubseteq X_1 \sqsubseteq \cdots$

$$X_0 = \bot \qquad X_{i+1} = X_i \bigtriangledown F^\sharp(X_i)$$

     is finite and its last element over-approximates the concrete semantics $\mathbf{lfp}F$.

Use Example: Target Language

x	$\in$	$\mathbb{X}$	program variables
C	::=		statements
	|	`skip`	nop statement
	|	C ; C	sequence of statements
	|	x := E	assignment
	|	`input(x)`	read an integer input
	|	`if(`B`){`C`}else{`C`}`	condition statement
	|	`while(`B`){`C`}`	loop statement
	|	`goto` E	goto with dynamically computed label
E	::=		expression
	|	n	integer
	|	x	variable
	|	$E + E$	addition
B	::=		boolean expression
	|	`true` | `false`	
	|	$E < E$	comparison
	|	$E = E$	equality
P	::=	C	program

Figure: Syntax of a simple imperative language

Use Example: Concrete State Transition Semantics

$$\mathsf{lfp} F$$

of the continuous function

$$\begin{array}{l} F : \wp(\mathbb{S}) \to \wp(\mathbb{S}) \\ F(X) = I \cup \mathit{Step}(X) \\ \mathit{Step}(X) = \breve{\wp}(\hookrightarrow) \end{array}$$

where

$$\mathbb{S} = \mathbb{L} \times \mathbb{M}$$

and

$$\begin{array}{lrcl} \text{memories} & \mathbb{M} & = & \mathbb{X} \to \mathbb{V} \\ \text{values} & \mathbb{V} & = & \mathbb{Z} \cup \mathbb{L} \end{array}$$

The state transition relation $(l, m) \hookrightarrow (l', m')$ is defined as follows.

$$\begin{array}{rcll} \texttt{skip} & : & (l,m) \hookrightarrow (\texttt{next}(l),\ m) & \\ \texttt{input(x)} & : & (l,m) \hookrightarrow (\texttt{next}(l),\ \mathit{update}_{\texttt{x}}(m, z)) & \text{for an input integer } z \\ \texttt{x} := E & : & (l,m) \hookrightarrow (\texttt{next}(l),\ \mathit{update}_{\texttt{x}}(m, \mathit{eval}_E(m))) & \\ C_1; C_2 & : & (l,m) \hookrightarrow (\texttt{next}(l),\ m) & \\ \texttt{if}(B)\{C_1\}\texttt{else}\{C_2\} & : & (l,m) \hookrightarrow (\texttt{nextTrue}(l),\ \mathit{filter}_B(m)) & \\ & : & (l,m) \hookrightarrow (\texttt{nextFalse}(l),\ \mathit{filter}_{\neg B}(m)) & \\ \texttt{while}(B)\{C\} & : & (l,m) \hookrightarrow (\texttt{nextTrue}(l),\ \mathit{filter}_B(m)) & \\ & : & (l,m) \hookrightarrow (\texttt{nextFalse}(l),\ \mathit{filter}_{\neg B}(m)) & \\ \texttt{goto } E & : & (l,m) \hookrightarrow (\mathit{eval}_E(m),\ m) & \end{array}$$

Use Example: Abstract State

An abstract domain $\mathbb{M}^\sharp$ is a CPO such that

$$(\wp(\mathbb{M}), \subseteq) \underset{\alpha_M}{\overset{\gamma_M}{\leftrightarrows}} (\mathbb{M}^\sharp, \sqsubseteq_M)$$

defined as

$$M^\sharp \in \mathbb{M}^\sharp = \mathbb{X} \to \mathbb{V}^\sharp$$

where $\mathbb{V}^\sharp$ is an abstract domain that is a CPO such that

$$(\wp(\mathbb{V}), \subseteq) \underset{\alpha_V}{\overset{\gamma_V}{\leftrightarrows}} (\mathbb{V}^\sharp, \sqsubseteq_V).$$

We design $\mathbb{V}^\sharp$ as

$$\mathbb{V}^\sharp = \mathbb{Z}^\sharp \times \mathbb{L}^\sharp$$

where $\mathbb{Z}^\sharp$ is a CPO that is Galois connected with $\wp(\mathbb{Z})$, and $\mathbb{L}^\sharp$ is the powerset $\wp(\mathbb{L})$ of labels.

Use Example: Abstract State Transition Semantics

Case the l-labeled statement of

$$
\begin{array}{rcl}
\texttt{skip} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{input(x)} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \alpha(\mathbb{Z}))) \\
\texttt{x} := E & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \mathit{eval}^\sharp_E(M^\sharp))) \\
C_1; C_2 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{if}(B)\{C_1\}\texttt{else}\{C_2\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{while}(B)\{C\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{goto}\ E & : & (l, M^\sharp) \hookrightarrow^\sharp (l', M^\sharp) \quad \text{for } l' \in L \text{ of } (z^\sharp, L) = \mathit{eval}^\sharp_E(M^\sharp)
\end{array}
$$

Use Example: Abstract State Transition Semantics

Case the l-labeled statement of

$$
\begin{array}{rcl}
\texttt{skip} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{input(x)} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \alpha(\mathbb{Z}))) \\
\texttt{x} := E & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \mathit{eval}^\sharp_E(M^\sharp))) \\
C_1; C_2 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{if}(B)\{C_1\}\texttt{else}\{C_2\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{while}(B)\{C\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{goto } E & : & (l, M^\sharp) \hookrightarrow^\sharp (l', M^\sharp) \quad \text{for } l' \in L \text{ of } (z^\sharp, L) = \mathit{eval}^\sharp_E(M^\sharp)
\end{array}
$$

Let $F^\sharp$ be defined as the framework:

$$
\begin{array}{l}
F^\sharp : \mathbb{S}^\sharp \to \mathbb{S}^\sharp \\
F^\sharp(S^\sharp) = \alpha(I) \cup^\sharp \mathit{Step}^\sharp(S^\sharp) \\
\mathit{Step}^\sharp = \wp(\mathrm{id}, \cup^\sharp_M) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp).
\end{array}
$$

Use Example: Abstract State Transition Semantics

Case the l-labeled statement of

$$
\begin{array}{rcl}
\texttt{skip} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{input(x)} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \alpha(\mathbb{Z}))) \\
\texttt{x} := E & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \mathit{eval}^\sharp_E(M^\sharp))) \\
C_1; C_2 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{if}(B)\{C_1\}\texttt{else}\{C_2\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
& : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{while}(B)\{C\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
& : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{goto } E & : & (l, M^\sharp) \hookrightarrow^\sharp (l', M^\sharp) \quad \text{for } l' \in L \text{ of } (z^\sharp, L) = \mathit{eval}^\sharp_E(M^\sharp)
\end{array}
$$

Let $F^\sharp$ be defined as the framework:

$$
\begin{array}{l}
F^\sharp : \mathbb{S}^\sharp \to \mathbb{S}^\sharp \\
F^\sharp(S^\sharp) = \alpha(I) \cup^\sharp \mathit{Step}^\sharp(S^\sharp) \\
\mathit{Step}^\sharp = \wp(\mathrm{id}, \cup^\sharp_M) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp).
\end{array}
$$

If the $\mathit{Step}^\sharp$ and $\cup^\sharp_\_$ are sound abstractions of, respectively, Step and $\cup\_$:

$$
\begin{array}{rcl}
\breve{\wp}(\hookrightarrow) \circ \gamma & \subseteq & \gamma \circ \breve{\wp}(\hookrightarrow^\sharp) \\
\cup \circ (\gamma, \gamma) & \subseteq & \gamma \circ \cup^\sharp_\_
\end{array}
$$

Use Example: Abstract State Transition Semantics

Case the l-labeled statement of

$$\begin{array}{rcl}
\texttt{skip} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{input(x)} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \alpha(\mathbb{Z}))) \\
\texttt{x} := E & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), \mathit{update}^\sharp_{\texttt{x}}(M^\sharp, \mathit{eval}^\sharp_E(M^\sharp))) \\
C_1; C_2 & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{next}(l), M^\sharp) \\
\texttt{if}(B)\{C_1\}\texttt{else}\{C_2\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
& : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{while}(B)\{C\} & : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextTrue}(l), \mathit{filter}^\sharp_B(M^\sharp)) \\
& : & (l, M^\sharp) \hookrightarrow^\sharp (\texttt{nextFalse}(l), \mathit{filter}^\sharp_{\neg B}(M^\sharp)) \\
\texttt{goto } E & : & (l, M^\sharp) \hookrightarrow^\sharp (l', M^\sharp) \quad \text{for } l' \in L \text{ of } (z^\sharp, L) = \mathit{eval}^\sharp_E(M^\sharp)
\end{array}$$

Let $F^\sharp$ be defined as the framework:

$$\begin{array}{l}
F^\sharp : \mathbb{S}^\sharp \to \mathbb{S}^\sharp \\
F^\sharp(S^\sharp) = \alpha(I) \cup^\sharp \mathit{Step}^\sharp(S^\sharp) \\
\mathit{Step}^\sharp = \wp(\mathrm{id}, \cup^\sharp_M) \circ \pi \circ \breve{\wp}(\hookrightarrow^\sharp).
\end{array}$$

If the $\mathit{Step}^\sharp$ and $\cup^\sharp_\_$ are sound abstractions of, respectively, Step and $\cup_\_$:

$$\begin{array}{rcl}
\breve{\wp}(\hookrightarrow) \circ \gamma & \subseteq & \gamma \circ \breve{\wp}(\hookrightarrow^\sharp) \\
\cup \circ (\gamma, \gamma) & \subseteq & \gamma \circ \cup^\sharp__
\end{array}$$

then we can use $F^\sharp$ to soundly approximates the concrete semantics $\mathsf{lfp}F$

Use Example: Defining Sound $\hookrightarrow^\sharp$

Theorem (Soundness of $\hookrightarrow^\sharp$)

If the semantic operators satisfy the following soundness properties:

$$\begin{array}{rcl}
\wp(\mathit{eval}_E) \circ \gamma_M & \subseteq & \gamma_V \circ \mathit{eval}^\sharp_E \\
\wp(\mathit{update}_{\mathtt{x}}) \circ \times \circ (\gamma_M, \gamma_V) & \subseteq & \gamma_M \circ \mathit{update}^\sharp_{\mathtt{x}} \\
\wp(\mathit{filter}_B) \circ \gamma_M & \subseteq & \gamma_M \circ \mathit{filter}^\sharp_B \\
\wp(\mathit{filter}_{\neg B}) \circ \gamma_M & \subseteq & \gamma_M \circ \mathit{filter}^\sharp_{\neg B}
\end{array}$$

then $\breve{\wp}(\hookrightarrow) \circ \gamma \sqsubseteq \gamma \circ \breve{\wp}(\hookrightarrow^\sharp)$. *(The* $\times$ *is the Cartesian product operator of two sets.)*

Use Example: Defining Sound $\cup^\sharp_\_$

As of sound $\cup^\sharp_\_$, one candidate is the least upper bound operator $\sqcup$ if $\mathbb{S}^\sharp$ and $\mathbb{M}^\sharp$ are closed by $\sqcup$ (e.g. lattices), since

$$\begin{array}{lcll} (\gamma \circ \sqcup)(a^\sharp, b^\sharp) = \gamma(a^\sharp \sqcup b^\sharp) & \sqsupseteq & \gamma(a^\sharp) \cup \gamma(b^\sharp) & \text{by monotone } \gamma \\ & = & (\cup \circ (\gamma, \gamma))(a^\sharp, b^\sharp). & \end{array}$$