A General Static Analysis Framework Based on a Transitional Semantics

Material covered in chapter 4 of
Introduction to Static Analysis: an Abstract Interpretation Perspective

Purpose of this lecture

So far, we have learned

- how to design a sound static analysis (an abstract interpreter) in the compositional semantics style

However,

- **defining a compositional semantics is a burden** for languages with dynamic controls such as function calls or functions/jump-targets/exceptions as values.

By using transitional semantics style we can avoid the difficulty.

Content of the lecture:

- step-by-step framework to design a sound static analysis in **transitional semantics** style

Outline

Transitional semantics (review)

State transition sequence

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

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

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

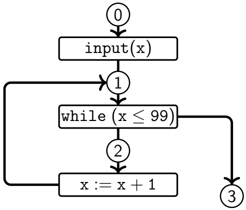
A state $s \in \mathbb{S} = \mathbb{L} \times \mathbb{M}$ of the program is a pair (l, m) of a program label l and the machine state m at that program label during execution.

Concrete transition sequence

Example program:

$$\texttt{input(x);}$$
$$\texttt{while}\ (\mathtt{x} \le 99)$$
$$\{\mathtt{x} := \mathtt{x} + 1\}$$

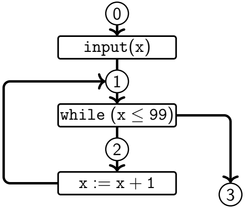
The labeled representation:



From empty memory $\emptyset$, 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



Suppose that the possible inputs are 0, 99, and 100. Then, the set of all reachable states is:

$$\{(0,\emptyset),(1,x\mapsto 100),(3,x\mapsto 100)\}\cup$$
$$\{(0,\emptyset),(1,x\mapsto 99),(2,x\mapsto 99),(1,x\mapsto 100),(3,x\mapsto 100)\}\cup$$
$$\{(0,\emptyset),(1,x\mapsto 0),(2,x\mapsto 0),(1,x\mapsto 1),\cdots,(1,x\mapsto 100),(3,x\mapsto 100)\}.$$

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

$$C_0 = I \text{ and } C_{i+1} = I \cup \mathit{Step}(C_i)$$

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

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

corresponds to *the least fixpoint* of F

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

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 $\hookrightarrow$:

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

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

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

Outline

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":

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

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)

Semantic domain:

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

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

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

Abstraction of the semantic domain $\wp(\mathbb{S})$ (2/2)

$\wp(\mathbb{L} \times \mathbb{M}) \ni$ collection of all states $\left\{ \begin{array}{ll} (0, m_0), (0, m_0'), \cdots, & \text{at } 0 \\ \vdots & \\ (n, m_n), (n, m_n'), \cdots. & \text{at } n \end{array} \right.$

$\mathbb{L} \to \wp(\mathbb{M}) \ni$ label-wise collection $\left\{ \begin{array}{l} (0, \{m_0, m_0', \cdots\}) \\ \vdots \\ (n, \{m_n, m_n', \cdots\}) \end{array} \right.$

$\mathbb{L} \to \mathbb{M}^\sharp \ni$ label-wise abstraction $\left\{ \begin{array}{l} (0, M_0^\sharp) \\ \vdots \\ (n, M_n^\sharp) \end{array} \right.$

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 $\perp$
  - has a least-upper bound for every *chain*
- an abstract domain as $\sqcup$-semilattices also works

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)

For Galois-connection

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

we rely on the following properties:

- 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.

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)

$$(\wp(\mathbb{L} \times \mathbb{M}), \subseteq) \xleftarrow[\alpha]{\gamma} (\mathbb{L} \to \mathbb{M}^\sharp, \sqsubseteq).$$

composes two Galois connections:

$$\begin{array}{ll} (\wp(\mathbb{L} \times \mathbb{M}), \subseteq) & \\ \xleftarrow[\alpha_0]{\gamma_0} \; (\mathbb{L} \to \wp(\mathbb{M}), \sqsubseteq) & (\sqsubseteq \text{ is the label-wise } \subseteq) \\ \quad \xleftarrow[\alpha_2]{\gamma_2} (\mathbb{L} \to \mathbb{M}^\sharp, \sqsubseteq) & (\sqsubseteq \text{ is the label-wise } \sqsubseteq_M) \end{array}$$

$$\alpha_0 \left\{ \begin{array}{l} (0, m_0), (0, m_0'), \cdots, \\ \quad \vdots \\ (n, m_n), (n, m_n'), \cdots \end{array} \right\} = \left\{ \begin{array}{l} (0, \{m_0, m_0', \cdots\}), \\ \quad \vdots \\ (n, \{m_n, m_n', \cdots\}) \end{array} \right\}, \quad \alpha_1 \left\{ \begin{array}{l} (0, \{m_0, m_0', \cdots\}), \\ \quad \vdots \\ (n, \{m_n, m_n', \cdots\}) \end{array} \right\} = \left\{ \begin{array}{l} (0, M_0^\sharp), \\ \quad \vdots \\ (n, M_n^\sharp) \end{array} \right\}$$

Thus, boils down to

$$(\wp(\mathbb{M}), \subseteq) \xleftarrow[\alpha_M]{\gamma_M} (\mathbb{M}^\sharp, \sqsubseteq_M).$$

Abstract semantic functions

Let

$$(\wp(\mathbb{L} \times \mathbb{M}), \subseteq) \xleftarrow[\alpha]{\gamma} (\mathbb{L} \to \mathbb{M}^\sharp, \sqsubseteq).$$

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

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

- abstract transition $\breve{\wp}(\hookrightarrow^\sharp)$:
  - a set $\subseteq \mathbb{L} \times \mathbb{M}^\sharp \quad \mapsto \quad$ a set $\subseteq \mathbb{L} \times \mathbb{M}^\sharp$
- partitioning π:
  - a set $\subseteq \mathbb{L} \times \mathbb{M}^\sharp \quad \mapsto \quad$ a set $\subseteq \mathbb{L} \times \wp(\mathbb{M}^\sharp)$
- joining $\wp(\mathrm{id}, \cup^\sharp_M)$:
  - a set $\subseteq \mathbb{L} \times \wp(\mathbb{M}^\sharp) \quad \mapsto \quad$ 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

$$\{(l_1, {M'}_1^\sharp), (l_2, {M''}_1^\sharp), (l_1, {M'}_2^\sharp)\}.$$

Example

Suppose the program has two labels l_1 and l_2. That is, $\mathbb{L} = \{l_1, l_2\}$. Given an abstract state $\{(l_1, M_1^\sharp), (l_2, M_2^\sharp)\}$, $\mathit{Step}^\sharp$ 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

$$\{(l_1, {M'}_1^\sharp), (l_2, {M''}_1^\sharp), (l_1, {M'}_2^\sharp)\}.$$

By the subsequent partitioning operator π, the result becomes

$$\{(l_1, \{{M'}_1^\sharp, {M'}_2^\sharp\}), (l_2, \{{M''}_1^\sharp\})\}.$$

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

$$\{(l_1, {M'}_1^\sharp), (l_2, {M''}_1^\sharp), (l_1, {M'}_2^\sharp)\}.$$

By the subsequent partitioning operator π, the result becomes

$$\{(l_1, \{{M'}_1^\sharp, {M'}_2^\sharp\}), (l_2, \{{M''}_1^\sharp\})\}.$$

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

$$\{(l_1, {M'}_1^\sharp \cup_M^\sharp {M'}_2^\sharp), (l_2, {M''}_1^\sharp)\}.$$

Conditions for sound $\hookrightarrow^\sharp$ and $\cup_\_^\sharp$

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

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

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

$$\cup \circ (\gamma, \gamma) \;\subseteq\; \gamma \circ \cup_\_^\sharp$$

$$\begin{tikzcd}
X^\sharp \arrow[r, "\breve{\wp}(\hookrightarrow^\sharp)"] \arrow[d, "\gamma"'] \arrow[dr, phantom, "\subseteq"] & Y^\sharp \arrow[d, "\gamma"] \\
X \arrow[r, "\breve{\wp}(\hookrightarrow)"'] & Y
\end{tikzcd}$$

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 a 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 $\mathbf{lfp}F$*:*

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

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 $\mathbf{lfp} F$:

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

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:
$$\left\{ \begin{array}{rcl} a'_0 & = & a_0 \\ a'_{n+1} & = & a'_n \nabla a_n \end{array} \right.$$

Outline

Analysis algorithm based on global iterations: basic version (1/2)

In case that $\mathbb{S}^\sharp$ 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

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

- hence, an algorithm is straightforward:

$$
\left|
\begin{array}{l}
\mathtt{C} \leftarrow \bot \\
\text{repeat} \\
\qquad \mathtt{R} \leftarrow \mathtt{C} \\
\qquad \mathtt{C} \leftarrow F^\sharp(\mathtt{C}) \\
\text{until } \mathtt{C} \sqsubseteq \mathtt{R} \\
\text{return } \mathtt{R}
\end{array}
\right.
$$

Analysis algorithm based on global iterations: basic version (2/2)

In case that $\mathbb{S}^\sharp$ is of infinite-height or $F^\sharp$ is neither monotonic nor extensive:

- use a widening operator ∇

$$
\begin{array}{|l}
\texttt{C} \leftarrow \bot \\
\text{repeat} \\
\qquad \texttt{R} \leftarrow \texttt{C} \\
\qquad \texttt{C} \leftarrow \texttt{C} \nabla F^\sharp(\texttt{C}) \\
\text{until } \texttt{C} \sqsubseteq \texttt{R} \\
\text{return } \texttt{R}
\end{array}
$$

Inefficiency of the basic algorithms

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

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

- $|\mathtt{C}| \sim$ 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

$$
\begin{array}{l}
\mathtt{C} : \mathbb{L} \to \mathbb{M}^\sharp \\
F^\sharp : (\mathbb{L} \to \mathbb{M}^\sharp) \to (\mathbb{L} \to \mathbb{M}^\sharp) \\
\mathtt{WorkList} : \wp(\mathbb{L}) \\
\\
\mathtt{WorkList} \leftarrow \mathbb{L} \\
\mathtt{C} \leftarrow \bot \\
\text{repeat} \\
\quad \mathtt{R} \leftarrow \mathtt{C} \\
\quad \mathtt{C} \leftarrow \mathtt{C} \mathbin{\nabla} F^\sharp(\mathtt{C}|_{\mathtt{WorkList}}) \\
\quad \mathtt{WorkList} \leftarrow \{ l \mid \mathtt{C}(l) \not\sqsubseteq \mathtt{R}(l), l \in \mathbb{L} \} \\
\text{until } \mathtt{WorkList} = \emptyset \\
\text{return } \mathtt{R}
\end{array}
$$

Improvement of the worklist algorithm

Inefficient: $\mathtt{WorkList} \leftarrow \{l \mid \mathtt{C}(l) \not\sqsubseteq \mathtt{R}(l), l \in \mathbb{L}\}$ re-scans all the labels.

- better: at application $\hookrightarrow^\sharp$ to $(l, \mathtt{C}(l))$, if its result $(l', M^\sharp)$ is changed $(M^\sharp \not\sqsubseteq \mathtt{C}(l'))$, add l' to the worklist.

Inefficient: $\mathtt{C} \bigtriangledown F^\sharp(C|_{\mathtt{WorkList}})$ widens at all the labels.

- better: apply $\bigtriangledown$ only at the target of a loop. Use $\cup^\sharp$ at other labels.

Outline

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 **lfp**F where

$$
\begin{array}{rlcl}
\text{concrete domain} & \wp(\mathbb{S}) & = & \wp(\mathbb{L} \times \mathbb{M}) \\
\text{concrete semantic function} & F : \wp(\mathbb{S}) \rightarrow \wp(\mathbb{S}) & & \\
 & F(X) & = & I \cup \mathit{Step}(X) \\
 & \mathit{Step} & = & \breve{\wp}(\hookrightarrow) \\
 & \hookrightarrow & \subseteq & (\mathbb{L} \times \mathbb{M}) \times (\mathbb{L} \times \mathbb{M})
\end{array}
$$

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

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

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:

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

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 $\mathbf{lfp} F$.

Outline

Use example: target language

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

Use example: concrete state transition semantics

Defined as **lfp**F of $F : \wp(\mathbb{S}) \rightarrow \wp(\mathbb{S})$ where

$$F(X) = I \cup Step(X) \quad \text{and} \quad Step(X) = \breve{\wp}(\hookrightarrow).$$

Semantic domains are:

$$\text{states } \mathbb{S} = \mathbb{L} \times \mathbb{M}, \quad \text{memories } \mathbb{M} = \mathbb{X} \rightarrow \mathbb{V}, \quad \text{values } \mathbb{V} = \mathbb{Z} \cup \mathbb{L}.$$

The state transition relation $(l, m) \hookrightarrow (l', m')$ is:

$$\begin{array}{rcll}
\texttt{skip} & : & (l, m) \hookrightarrow (\texttt{next}(l),\ m) & \\
\texttt{input(x)} & : & (l, m) \hookrightarrow (\texttt{next}(l),\ update_{\texttt{x}}(m, z)) & \quad \text{for an input integer } z \\
\texttt{x} := E & : & (l, m) \hookrightarrow (\texttt{next}(l),\ update_{\texttt{x}}(m, eval_E(m))) & \\
C_1; C_2 & : & (l, m) \hookrightarrow (\texttt{next}(l),\ m) & \\
\texttt{if}(B)\{C_1\}\texttt{else}\{C_2\} & : & (l, m) \hookrightarrow (\texttt{nextTrue}(l),\ filter_B(m)) & \\
 & : & (l, m) \hookrightarrow (\texttt{nextFalse}(l),\ filter_{\neg B}(m)) & \\
\texttt{while}(B)\{C\} & : & (l, m) \hookrightarrow (\texttt{nextTrue}(l),\ filter_B(m)) & \\
 & : & (l, m) \hookrightarrow (\texttt{nextFalse}(l),\ filter_{\neg B}(m)) & \\
\texttt{goto } E & : & (l, m) \hookrightarrow (eval_E(m),\ m) &
\end{array}$$

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

Define $\hookrightarrow^\sharp$ as:

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

Use example: abstract state transition semantics

Define $\hookrightarrow^\sharp$ as:

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

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

Define $\hookrightarrow^\sharp$ as:

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

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

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

Let $\mathit{Step}^\sharp$ and $\cup^\sharp_{\_}$ be sound:

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

Use example: abstract state transition semantics

Define $\hookrightarrow^\sharp$ as:

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

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

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

Let $\mathit{Step}^\sharp$ and $\cup^\sharp_{\_}$ be sound:

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

Then we can use $F^\sharp$ to soundly approximates **lfp**F

Use example: defining sound $\hookrightarrow^\sharp$

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

If the semantic operators satisfy the following soundness properties:

$$\begin{array}{rcl}
\wp(eval_E) \circ \gamma_M & \subseteq & \gamma_V \circ eval^\sharp_E \\
\wp(update_{\mathtt{x}}) \circ \times \circ (\gamma_M, \gamma_V) & \subseteq & \gamma_M \circ update^\sharp_{\mathtt{x}} \\
\wp(filter_B) \circ \gamma_M & \subseteq & \gamma_M \circ filter^\sharp_B \\
\wp(filter_{\neg B}) \circ \gamma_M & \subseteq & \gamma_M \circ filter^\sharp_{\neg B}
\end{array}$$

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

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