Static Analysis

A general method for automatic and sound approximation of sw run-time behaviors before the execution

- “before”: statically, without running sw
- “automatic”: sw analyzes sw
- “sound”: all possibilities into account
- “approximation”: cannot be exact
- “general”: for any source language and property
  - C, C++, C#, F#, Java, JavaScript, ML, Scala, Python, JVM, Dalvik, x86, Excel, etc
  - “buffer-overrun?”, “memory leak?”, “type errors?”, “x = y at line 2?”, “memory use \( \leq 2K \)”, etc
Static Analysis: “Abstract Interpretation” of Programs

- What is the value of the expression?

\[128 \times 22 + (1920 \times -10) + 4\]

- static analysis: “an integer”
- static analysis: “an even number”
- static analysis: “a number in \([-20000, 20000]\)”

- What value will \(x\) have?

\[x := 1; \text{repeat } x := x + 2 \text{ until } ...\]

- static analysis: “an integer”
- static analysis: “an odd number”
- static analysis: “[1, +\infty]”
Interval Analysis Example

\[\begin{array}{c|c|c|c|c}
\text{Node} & \text{Result} \\
\hline
1 & x \mapsto \bot \\
& y \mapsto \bot \\
2 & x \mapsto [0, 0] \\
& y \mapsto [0, 0] \\
3 & x \mapsto [0, 9] \\
& y \mapsto [0, +\infty] \\
4 & x \mapsto [1, 10] \\
& y \mapsto [0, +\infty] \\
5 & x \mapsto [1, 10] \\
& y \mapsto [1, +\infty] \\
6 & x \mapsto [10, 10] \\
& y \mapsto [0, +\infty] \\
\end{array}\]

Diagram:

1. ENTRY
2. \(x=0; y=0\)
3. \(x < 10\)
4. \(x = x+1\)
5. \(y = y+1\)
6. Print x

\[\begin{array}{c}
\text{Node} \\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
\end{array}\]

Decision:

\[\begin{array}{c}
\text{false} \\
\text{true} \\
\end{array}\]
Fixed Point Computation Does Not Terminate

The conventional fixed point computation requires an infinite number of iterations to converge:

<table>
<thead>
<tr>
<th>Node</th>
<th>initial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>10</th>
<th>11</th>
<th>$k$</th>
<th>$\infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
</tr>
<tr>
<td>2</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
</tr>
<tr>
<td>3</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 1]$</td>
<td>$x \mapsto [0, 2]$</td>
<td>$x \mapsto [0, 9]$</td>
<td>$x \mapsto [0, 9]$</td>
<td>$x \mapsto [0, 9]$</td>
<td>$x \mapsto [0, 9]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 1]$</td>
<td>$y \mapsto [0, 2]$</td>
<td>$y \mapsto [0, 9]$</td>
<td>$y \mapsto [0, 10]$</td>
<td>$y \mapsto [0, k-1]$</td>
<td>$y \mapsto [0, +\infty]$</td>
</tr>
<tr>
<td>4</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto [1, 1]$</td>
<td>$x \mapsto [1, 2]$</td>
<td>$x \mapsto [1, 3]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 1]$</td>
<td>$y \mapsto [0, 2]$</td>
<td>$y \mapsto [0, 9]$</td>
<td>$y \mapsto [0, 10]$</td>
<td>$y \mapsto [0, k-1]$</td>
<td>$y \mapsto [0, +\infty]$</td>
</tr>
<tr>
<td>5</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto [1, 1]$</td>
<td>$x \mapsto [1, 2]$</td>
<td>$x \mapsto [1, 3]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto [1, 1]$</td>
<td>$y \mapsto [1, 2]$</td>
<td>$y \mapsto [1, 3]$</td>
<td>$y \mapsto [1, 10]$</td>
<td>$y \mapsto [1, 10]$</td>
<td>$y \mapsto [1, 10]$</td>
<td>$y \mapsto [1, +\infty]$</td>
</tr>
<tr>
<td>6</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto [10, 10]$</td>
<td>$x \mapsto [10, 10]$</td>
<td>$x \mapsto [10, 10]$</td>
<td>$x \mapsto [10, 10]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 1]$</td>
<td>$y \mapsto [0, 2]$</td>
<td>$y \mapsto [0, 9]$</td>
<td>$y \mapsto [0, 10]$</td>
<td>$y \mapsto [0, k-1]$</td>
<td>$y \mapsto [0, +\infty]$</td>
</tr>
</tbody>
</table>
Fixed Point Computation with Widening and Narrowing

Two staged fixed point computation:
1. increasing widening sequence
2. decreasing narrowing sequence
1. Fixed Point Computation with Widening

<table>
<thead>
<tr>
<th>Node</th>
<th>initial</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>
| 1    | $x \mapsto \bot$  
      | $y \mapsto \bot$  | $x \mapsto \bot$  
      | $y \mapsto \bot$  | $x \mapsto \bot$  
      | $y \mapsto \bot$  |
| 2    | $x \mapsto \bot$  
      | $y \mapsto \bot$  | $x \mapsto [0, 0]$  
      | $y \mapsto [0, 0]$  | $x \mapsto [0, 0]$  
      | $y \mapsto [0, 0]$  |
| 3    | $x \mapsto \bot$  
      | $y \mapsto \bot$  | $x \mapsto [0, 0]$  
      | $y \mapsto [0, 0]$  | $x \mapsto [0, 9]$  
      | $y \mapsto [0, +\infty]$  |
| 4    | $x \mapsto \bot$  
      | $y \mapsto \bot$  | $x \mapsto [1, 1]$  
      | $y \mapsto [0, 0]$  | $x \mapsto [1, 10]$  
      | $y \mapsto [0, +\infty]$  |
| 5    | $x \mapsto \bot$  
      | $y \mapsto \bot$  | $x \mapsto [1, 1]$  
      | $y \mapsto [1, 1]$  | $x \mapsto [1, 10]$  
      | $y \mapsto [1, +\infty]$  |
| 6    | $x \mapsto \bot$  
      | $y \mapsto \bot$  | $x \mapsto \bot$  
      | $y \mapsto [0, 0]$  | $x \mapsto [10, +\infty]$  
      | $y \mapsto [0, +\infty]$  |
2. Fixed Point Computation with Narrowing

<table>
<thead>
<tr>
<th>Node</th>
<th>initial</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
<td>$x \mapsto \bot$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
<td>$y \mapsto \bot$</td>
</tr>
<tr>
<td>2</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
<td>$x \mapsto [0, 0]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
<td>$y \mapsto [0, 0]$</td>
</tr>
<tr>
<td>3</td>
<td>$x \mapsto [0, 9]$</td>
<td>$x \mapsto [0, 9]$</td>
<td>$x \mapsto [0, 9]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto [0, +\infty]$</td>
<td>$y \mapsto [0, +\infty]$</td>
<td>$y \mapsto [0, +\infty]$</td>
</tr>
<tr>
<td>4</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto [0, +\infty]$</td>
<td>$y \mapsto [0, +\infty]$</td>
<td>$y \mapsto [0, +\infty]$</td>
</tr>
<tr>
<td>5</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
<td>$x \mapsto [1, 10]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto [1, +\infty]$</td>
<td>$y \mapsto [1, +\infty]$</td>
<td>$y \mapsto [1, +\infty]$</td>
</tr>
<tr>
<td>6</td>
<td>$x \mapsto [10, +\infty]$</td>
<td>$x \mapsto [10, 10]$</td>
<td>$x \mapsto [10, 10]$</td>
</tr>
<tr>
<td></td>
<td>$y \mapsto [0, +\infty]$</td>
<td>$y \mapsto [0, +\infty]$</td>
<td>$y \mapsto [0, +\infty]$</td>
</tr>
</tbody>
</table>
Programs

Represent a program by a control-flow graph:

\[(C, \rightarrow)\]

- \(C\): the set of program points (i.e., nodes) in the program
- \(\rightarrow \subseteq C \times C\): the control-flow relation
  - \(c \rightarrow c'\): \(c\) is a predecessor of \(c'\)
- Each program point \(c\) is associated with a command, denoted \(\text{cmd}(c)\)
Commands

A simple set of commands:

\[
\begin{align*}
\text{cmd} & \rightarrow \text{skip} \mid x := e \mid x < n \\
\text{e} & \rightarrow n \mid x \mid e + e \mid e - e \mid e \ast e \mid e/e
\end{align*}
\]
Interval Domain

Definition:

\[ \mathbb{I} = \{ \perp \} \cup \{ [l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty, +\infty\} \land l \leq u \} \]

An interval is an abstraction of a set of integers:

- \( \gamma([1, 5]) = \)
- \( \gamma([3, 3]) = \)
- \( \gamma([0, +\infty]) = \)
- \( \gamma([\neg\infty, 7]) = \)
- \( \gamma(\perp) = \)
Concretization/Abstraction Functions

- $\gamma: \mathbb{I} \to \mathcal{P}(\mathbb{Z})$ is called concretization function:
  \[
  \gamma(\bot) = \emptyset \\
  \gamma([a, b]) = \{ z \in \mathbb{Z} | a \leq z \leq b \}
  \]

- $\alpha: \mathcal{P}(\mathbb{Z}) \to \mathbb{I}$ is abstraction function:
  \[
  \alpha(\{2\}) = \]
  \[
  \alpha(\{-1, 0, 1, 2, 3\}) = \]
  \[
  \alpha(\{-1, 3\}) = \]
  \[
  \alpha(\{1, 2, \ldots\}) = \]
  \[
  \alpha(\emptyset) = \]
  \[
  \alpha(\mathbb{Z}) = \]
  \[
  \alpha(\emptyset) = \bot \]
  \[
  \alpha(S) = [\min(S), \max(S)]
  \]
Partial Order (⊆) ⊆ \(\mathbb{I} \times \mathbb{I}\)

- \(\bot \subseteq i\) for all \(i \in \mathbb{I}\)
- \(i \subseteq [\infty, \infty]\) for all \(i \in \mathbb{I}\).
- \([1, 3] \subseteq [0, 4]\)
- \([1, 3] \not\subseteq [0, 2]\)

Definition:

- Mathematical:

\[i_1 \subseteq i_2 \iff \gamma(i_1) \subseteq \gamma(i_2)\]

- Implementable:

\[i_1 \subseteq i_2 \iff \begin{cases} i_1 = \bot \lor \\
\quad i_2 = [\infty, \infty] \lor \\
\quad (i_1 = [l_1, u_1] \land i_2 = [l_2, u_2] \land l_1 \geq l_2 \land u_1 \leq u_2)\end{cases}\]
Partial Order
Join $\sqcup$ and Meet $\sqcap$ Operators

- The join operator computes the *least upper bound*:
  - $[1, 3] \sqcup [2, 4] = [1, 4]$
  - $[1, 3] \sqcup [7, 9] = [1, 9]$

- The conditions of $i_1 \sqcup i_2$:
  1. $i_1 \subseteq i_1 \sqcup i_2$ $\land$ $i_2 \subseteq i_1 \sqcup i_2$
  2. $\forall i. i_1 \subseteq i$ $\land$ $i_2 \subseteq i$ $\implies$ $i_1 \sqcup i_2 \subseteq i$

- Definition:
  \[
  i_1 \sqcup i_2 = \alpha(\gamma(i_1) \cup \gamma(i_2))
  \]
  \[
  \bot \sqcup i = i
  \]
  \[
  i \sqcup \bot = i
  \]
  \[
  [l_1, u_1] \sqcup [l_2, u_2] = [\min(l_1, l_2), \max(l_1, l_2)]
  \]
Join $\sqcap$ and Meet $\sqcap$ Operators

- The meet operator computes the *greatest lower bound*:
  - $[1, 3] \cap [2, 4] = [2, 3]$
  - $[1, 3] \cap [7, 9] = \bot$

- The conditions of $i_1 \sqcap i_2$:
  1. $i_1 \subseteq i_1 \sqcup i_2 \land i_2 \subseteq i_1 \sqcup i_2$
  2. $\forall i. i \subseteq i_1 \land i \subseteq i_2 \implies i \subseteq i_1 \sqcap i_2$

- Definition:
  $$i_1 \cap i_2 = \alpha(\gamma(i_1) \cap \gamma(i_2))$$

\[\begin{align*}
\bot \cap i &= \bot \\
i \cap \bot &= \bot \\
[l_1, u_1] \cap [l_2, u_2] &= \begin{cases}
\bot & \text{max}(l_1, l_2) > \min(l_1, l_2) \\
\left[\max(l_1, l_2), \min(l_1, l_2)\right] & \text{o.w.}
\end{cases}
\end{align*}\]
Widening and Narrowing

A simple widening operator for the Interval domain:

\[
[a, b] \uparrow \bot = [a, b] \\
\bot \uparrow [c, d] = [c, d] \\
[a, b] \uparrow [c, d] = [(c < a? -\infty : a), (b < d? +\infty : b)]
\]

A simple narrowing operator:

\[
[a, b] \triangle \bot = \bot \\
\bot \triangle [c, d] = \bot \\
[a, b] \triangle [c, d] = [(a = -\infty?c : a), (b = +\infty?d : b)]
\]
Interval-based Abstract States

\[ \mathcal{S} = \text{Var} \rightarrow \mathbb{I} \]

Partial order, join, meet, widening, and narrowing are lifted pointwise:

\[ s_1 \sqsubseteq s_2 \text{ iff } \forall x \in \text{Var}. \; s_1(x) \sqsubseteq s_2(x) \]

\[ s_1 \sqcup s_2 = \lambda x. \; s_1(x) \sqcup s_2(x) \]

\[ s_1 \sqcap s_2 = \lambda x. \; s_1(x) \sqcap s_2(x) \]

\[ s_1 \triangledown s_2 = \lambda x. \; s_1(x) \triangledown s_2(x) \]

\[ s_1 \triangledown s_2 = \lambda x. \; s_1(x) \triangle s_2(x) \]
The Domain of Interval Analysis

\[ \mathbb{D} = \mathbb{C} \to \mathbb{S} \]

Partial order, join, meet, widening, and narrowing are lifted pointwise:

\[ d_1 \sqsubseteq d_2 \text{ iff } \forall c \in \mathbb{C}. \ d_1(x) \sqsubseteq d_2(x) \]

\[ d_1 \sqcup d_2 = \lambda c. \ d_1(c) \sqcup d_2(c) \]

\[ d_1 \sqcap d_2 = \lambda c. \ d_1(c) \sqcap d_2(c) \]

\[ d_1 \triangledown d_2 = \lambda c. \ d_1(c) \triangledown d_2(c) \]

\[ d_1 \triangle d_2 = \lambda c. \ d_1(c) \triangle d_2(c) \]
Abstract Evaluation of Expressions

\[ e \rightarrow n \mid x \mid e + e \mid e - e \mid e \times e \mid e/e \]

\[
\begin{align*}
\text{eval} & : \quad e \times S \rightarrow I \\
\text{eval}(n, s) &= [n, n] \\
\text{eval}(x, s) &= s(x) \\
\text{eval}(e_1 + e_2, s) &= \text{eval}(e_1, s) \hat{+} \text{eval}(e_2, s) \\
\text{eval}(e_1 - e_2, s) &= \text{eval}(e_1, s) \hat{-} \text{eval}(e_2, s) \\
\text{eval}(e_1 \times e_2, s) &= \text{eval}(e_1, s) \hat{\times} \text{eval}(e_2, s) \\
\text{eval}(e_1/e_2, s) &= \text{eval}(e_1, s) \hat{\div} \text{eval}(e_2, s)
\end{align*}
\]
Abstract Binary Operators

\[ i_1 \hat{+} i_2 = \alpha(\{z_1 + z_2 \mid z_1 \in \gamma(i_1) \land z_2 \in \gamma(i_2)\}) \]
\[ i_1 \hat{-} i_2 = \alpha(\{z_1 - z_2 \mid z_1 \in \gamma(i_1) \land z_2 \in \gamma(i_2)\}) \]
\[ i_1 \hat{\ast} i_2 = \alpha(\{z_1 \ast z_2 \mid z_1 \in \gamma(i_1) \land z_2 \in \gamma(i_2)\}) \]
\[ i_1 \hat{/} i_2 = \alpha(\{z_1/z_2 \mid z_1 \in \gamma(i_1) \land z_2 \in \gamma(i_2)\}) \]

Implementable version:

\[ \bot \hat{+} i = \]
\[ i \hat{+} \bot = \]
\[ [l_1, u_1] \hat{+} [l_2, u_2] = \]
\[ [l_1, u_1] \hat{-} [l_2, u_2] = \]
\[ [l_1, u_1] \hat{\ast} [l_2, u_2] = \]
\[ [l_1, u_1] \hat{/} [l_2, u_2] = \]
Abstract Execution of Commands

$$f_c : \mathcal{S} \rightarrow \mathcal{S}$$

$$f_c(s) = \begin{cases} 
    s & \text{cmd}(c) = \text{skip} \\
    [x \mapsto \text{eval}(e, s)]s & \text{cmd}(c) = x := e \\
    [x \mapsto s(x) \cap [-\infty, n - 1]]s & \text{cmd}(c) = x < n 
\end{cases}$$
We aim to compute

\[ X : \mathbb{C} \rightarrow \mathbb{S} \]

such that

\[ X = \lambda c. f_c( \bigsqcup_{c' \rightarrow c} X(c')) \]

In fixed point form:

\[ X = F(X) \]

where

\[ F(X) = \lambda c. f_c( \bigsqcup_{c' \rightarrow c} X(c')) \]

The solution of the equation is a fixed point of

\[ F : (\mathbb{C} \rightarrow \mathbb{S}) \rightarrow (\mathbb{C} \rightarrow \mathbb{S}) \]
Fixed Point Computation

The least fixed point computation may not converge:

\[ \text{fix } F = \bigsqcup_{i \in \mathbb{N}} F^i(\bot) = F^0(\bot) \sqcup F^1(\bot) F^2(\bot) \sqcup \cdots \]

Instead, we aim to find a (not necessarily least) fixed point with widening and narrowing:

1. **widening iteration:**

   \[
   X_0 = \bot \\
   X_i = X_{i-1} \quad \text{if } F(X_{i-1}) \sqsubseteq X_{i-1} \\
   = X_{i-1} \nabla F(X_{i-1}) \quad \text{otherwise}
   \]

2. **narrowing iteration:**

   \[
   Y_i = \begin{cases} 
   \hat{A} & \text{if } i = 0 \\
   Y_{i-1} \triangledown F(Y_{i-1}) & \text{if } i > 0
   \end{cases}
   \]

   \((\hat{A} \text{ is the result from the widening iteration, i.e., } \lim_i X_i)\)
Need of Static Analysis Theory

- How to design or choose an abstract domain?
- How to ensure that the abstract execution is sound?
- How to design widening and narrowing?
- How to ensure the termination of widening and narrowing?
- ...