COSE312: Compilers

Lecture 18 — Interval Analysis

Hakjoo Oh
2015 Fall

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?

$$\mathbf{128 \times 22 + (1920 \times -10) + 4}$$

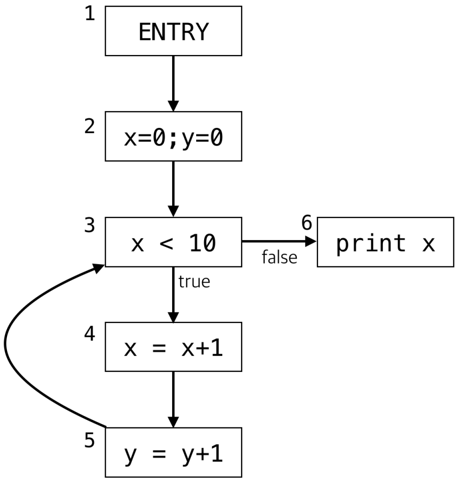
  - static analysis: "an integer"
  - static analysis: "an even number"
  - static analysis: "a number in $[\mathbf{-20000, 20000}]$"

- What value will `x` have?

```
x := 1; repeat x := x + 2 until ...
```

  - static analysis: "an integer"
  - static analysis: "an odd number"
  - static analysis: "$[\mathbf{1, +\infty}]$"

Interval Analysis Example



Node	Result
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	$x \mapsto [10, 10]$ $y \mapsto [0, +\infty]$

Fixed Point Computation Does Not Terminate

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

Node	initial	1	2	3	10	11	k	∞
1	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$
2	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$
3	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,1]$ $y \mapsto [0,1]$	$x \mapsto [0,2]$ $y \mapsto [0,2]$	$x \mapsto [0,9]$ $y \mapsto [0,9]$	$x \mapsto [0,9]$ $y \mapsto [0,10]$	$x \mapsto [0,9]$ $y \mapsto [0,k-1]$	$x \mapsto [0,9]$ $y \mapsto [0,+\infty]$
4	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [1,1]$ $y \mapsto [0,0]$	$x \mapsto [1,2]$ $y \mapsto [0,1]$	$x \mapsto [1,3]$ $y \mapsto [0,2]$	$x \mapsto [1,10]$ $y \mapsto [0,9]$	$x \mapsto [1,10]$ $y \mapsto [0,10]$	$x \mapsto [1,10]$ $y \mapsto [0,k-1]$	$x \mapsto [1,10]$ $y \mapsto [0,+\infty]$
5	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [1,1]$ $y \mapsto [1,1]$	$x \mapsto [1,2]$ $y \mapsto [1,2]$	$x \mapsto [1,3]$ $y \mapsto [1,3]$	$x \mapsto [1,10]$ $y \mapsto [1,10]$	$x \mapsto [1,10]$ $y \mapsto [1,11]$	$x \mapsto [1,10]$ $y \mapsto [1,k]$	$x \mapsto [1,10]$ $y \mapsto [1,+\infty]$
6	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto [0,0]$	$x \mapsto \bot$ $y \mapsto [0,1]$	$x \mapsto \bot$ $y \mapsto [0,2]$	$x \mapsto [10,10]$ $y \mapsto [0,9]$	$x \mapsto [10,10]$ $y \mapsto [0,10]$	$x \mapsto [10,10]$ $y \mapsto [0,k-1]$	$x \mapsto [10,10]$ $y \mapsto [0,+\infty]$

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

Node	initial	1	2	3
1	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$
2	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$
3	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,9]$ $y \mapsto [0,+\infty]$	$x \mapsto [0,9]$ $y \mapsto [0,+\infty]$
4	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [1,1]$ $y \mapsto [0,0]$	$x \mapsto [1,10]$ $y \mapsto [0,+\infty]$	$x \mapsto [1,10]$ $y \mapsto [0,+\infty]$
5	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto [1,1]$ $y \mapsto [1,1]$	$x \mapsto [1,10]$ $y \mapsto [1,+\infty]$	$x \mapsto [1,10]$ $y \mapsto [1,+\infty]$
6	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto [0,0]$	$x \mapsto [10,+\infty]$ $y \mapsto [0,+\infty]$	$x \mapsto [10,+\infty]$ $y \mapsto [0,+\infty]$

2. Fixed Point Computation with Narrowing

Node	initial	1	2
1	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$	$x \mapsto \bot$ $y \mapsto \bot$
2	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$	$x \mapsto [0,0]$ $y \mapsto [0,0]$
3	$x \mapsto [0,9]$ $y \mapsto [0,+\infty]$	$x \mapsto [0,9]$ $y \mapsto [0,+\infty]$	$x \mapsto [0,9]$ $y \mapsto [0,+\infty]$
4	$x \mapsto [1,10]$ $y \mapsto [0,+\infty]$	$x \mapsto [1,10]$ $y \mapsto [0,+\infty]$	$x \mapsto [1,10]$ $y \mapsto [0,+\infty]$
5	$x \mapsto [1,10]$ $y \mapsto [1,+\infty]$	$x \mapsto [1,10]$ $y \mapsto [1,+\infty]$	$x \mapsto [1,10]$ $y \mapsto [1,+\infty]$
6	$x \mapsto [10,+\infty]$ $y \mapsto [0,+\infty]$	$x \mapsto [10,10]$ $y \mapsto [0,+\infty]$	$x \mapsto [10,10]$ $y \mapsto [0,+\infty]$

Programs

Represent a program by a control-flow graph:

$$(\mathbb{C}, \hookrightarrow)$$

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

Commands

A simple set of commands:

$$\begin{array}{rcl} cmd & \rightarrow & skip \mid x := e \mid x < n \\ e & \rightarrow & n \mid x \mid e + e \mid e - e \mid e * e \mid e/e \end{array}$$

Interval Domain

- Definition:

$$\mathbb{I} = \{\bot\} \cup \{[l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty, +\infty\} \ \wedge \ l \leq u\}$$

- An interval is an abstraction of a set of integers:
  - $\gamma([1, 5]) =$
  - $\gamma([3, 3]) =$
  - $\gamma([0, +\infty]) =$
  - $\gamma([-\infty, 7]) =$
  - $\gamma(\bot) =$

Concretization/Abstraction Functions

- $\gamma : \mathbb{I} \to \mathcal{P}(\mathbb{Z})$ is called *concretization function*:

$$\begin{array}{rcl} \gamma(\bot) & = & \emptyset \\ \gamma([a, b]) & = & \{z \in \mathbb{Z} \mid a \leq z \leq b\} \end{array}$$

- $\alpha : \mathcal{P}(\mathbb{Z}) \to \mathbb{I}$ is *abstraction function*:
  - $\alpha(\{2\}) =$
  - $\alpha(\{-1, 0, 1, 2, 3\}) =$
  - $\alpha(\{-1, 3\}) =$
  - $\alpha(\{1, 2, \ldots\}) =$
  - $\alpha(\emptyset) =$
  - $\alpha(\mathbb{Z}) =$

$$\begin{array}{rcl} \alpha(\emptyset) & = & \bot \\ \alpha(S) & = & [\min(S), \max(S)] \end{array}$$

Partial Order ($\sqsubseteq$) $\subseteq \mathbb{I} \times \mathbb{I}$

- $\bot \sqsubseteq i$ for all $i \in \mathbb{I}$
- $i \sqsubseteq [-\infty, +\infty]$ for all $i \in \mathbb{I}$.
- $[1,3] \sqsubseteq [0,4]$
- $[1,3] \not\sqsubseteq [0,2]$

Definition:

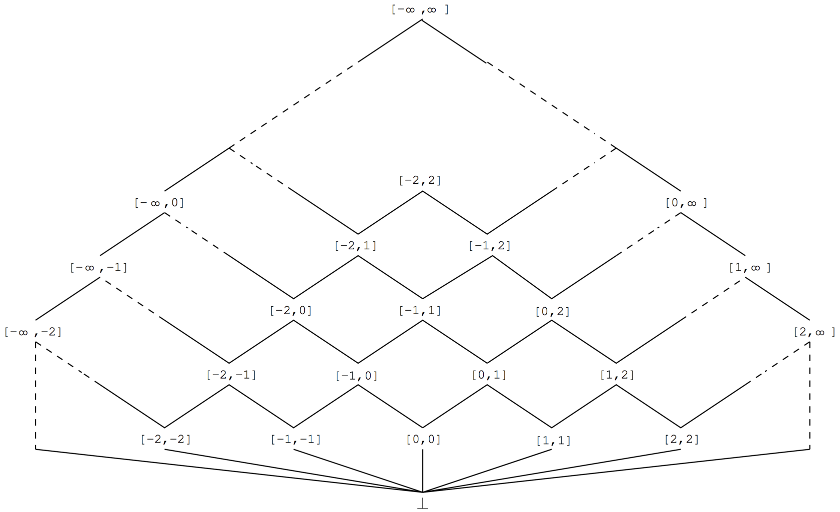
- Mathematical:

$$i_1 \sqsubseteq i_2 \text{ iff } \gamma(i_1) \subseteq \gamma(i_2)$$

- Implementable:

$$i_1 \sqsubseteq i_2 \text{ iff } \begin{cases} i_1 = \bot \lor \\ i_2 = [-\infty, +\infty] \lor \\ (i_1 = [l_1, u_1] \ \land \ i_2 = [l_2, u_2] \ \land \ l_1 \geq l_2 \ \land \ u_1 \leq u_2) \end{cases}$$

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. $i_1 \sqsubseteq i_1 \sqcup i_2 \ \wedge \ i_2 \sqsubseteq i_1 \sqcup i_2$
  2. $\forall i.\ i_1 \sqsubseteq i \ \wedge \ i_2 \sqsubseteq i \implies i_1 \sqcup i_2 \sqsubseteq i$
- Definition:

$$i_1 \sqcup i_2 = \alpha(\gamma(i_1) \cup \gamma(i_2))$$

$$\begin{array}{rcl} \bot \sqcup i & = & i \\ i \sqcup \bot & = & i \\ [l_1, u_1] \sqcup [l_2, u_2] & = & [\min(l_1, l_2), \max(l_1, l_2)] \end{array}$$

Join $\sqcup$ and Meet $\sqcap$ Operators

- The meet operator computes the *greatest lower bound*:
  - $[1, 3] \sqcap [2, 4] = [2, 3]$
  - $[1, 3] \sqcap [7, 9] = \bot$
- The conditions of $i_1 \sqcap i_2$:
  1. $i_1 \sqsubseteq i_1 \sqcup i_2 \ \wedge \ i_2 \sqsubseteq i_1 \sqcup i_2$
  2. $\forall i.\ i \sqsubseteq i_1 \ \wedge \ i \sqsubseteq i_2 \implies i \sqsubseteq i_1 \sqcap i_2$
- Definition:

$$i_1 \sqcap i_2 = \alpha(\gamma(i_1) \cap \gamma(i_2))$$

$$\begin{array}{rcl} \bot \sqcap i & = & \bot \\ i \sqcap \bot & = & \bot \\ [l_1, u_1] \sqcap [l_2, u_2] & = & \begin{cases} \bot & \max(l_1, l_2) > \min(l_1, l_2) \\ [\max(l_1, l_2), \min(l_1, l_2)] & \text{o.w.} \end{cases} \end{array}$$

Widening and Narrowing

A simple widening operator for the Interval domain:

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

A simple narrowing operator:

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

Interval-based Abstract States

$$\mathbb{S} = \boldsymbol{Var} \to \mathbb{I}$$

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

$$s_1 \sqsubseteq s_2 \text{ iff } \forall x \in \boldsymbol{Var}.\ s_1(x) \sqsubseteq s_2(x)$$

$$s_1 \sqcup s_2 = \lambda x.\ s_1(x) \sqcup s_2(x)$$

$$s_1 \sqcap s_2 = \lambda x.\ s_1(x) \sqcap s_2(x)$$

$$s_1 \bigtriangledown s_2 = \lambda x.\ s_1(x) \bigtriangledown s_2(x)$$

$$s_1 \bigtriangleup s_2 = \lambda x.\ s_1(x) \bigtriangleup s_2(x)$$

The Domain of Interval Analysis

$$\mathbb{D} = \mathbb{C} \to \mathbb{S}$$

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

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

$$d_1 \sqcup d_2 = \lambda c.\ d_1(c) \sqcup d_2(c)$$

$$d_1 \sqcap d_2 = \lambda c.\ d_1(c) \sqcap d_2(c)$$

$$d_1 \bigtriangledown d_2 = \lambda c.\ d_1(c) \bigtriangledown d_2(c)$$

$$d_1 \bigtriangleup d_2 = \lambda c.\ d_1(c) \bigtriangleup d_2(c)$$

Abstract Evaluation of Expressions

$$e \rightarrow n \mid x \mid e + e \mid e - e \mid e * e \mid e / e$$

$$\begin{array}{rcl} eval & : & e \times \mathbb{S} \rightarrow \mathbb{I} \\ eval(n, s) & = & [n, n] \\ eval(x, s) & = & s(x) \\ eval(e_1 + e_2, s) & = & eval(e_1, s) \hat{+} eval(e_2, s) \\ eval(e_1 - e_2, s) & = & eval(e_1, s) \hat{-} eval(e_2, s) \\ eval(e_1 * e_2, s) & = & eval(e_1, s) \hat{*} eval(e_2, s) \\ eval(e_1 / e_2, s) & = & eval(e_1, s) \hat{/} eval(e_2, s) \end{array}$$

Abstract Binary Operators

$$
\begin{array}{rcl}
i_1 \hat{+} i_2 & = & \alpha(\{z_1 + z_2 \mid z_1 \in \gamma(i_1) \ \wedge \ z_2 \in \gamma(i_2)\}) \\
i_1 \hat{-} i_2 & = & \alpha(\{z_1 - z_2 \mid z_1 \in \gamma(i_1) \ \wedge \ z_2 \in \gamma(i_2)\}) \\
i_1 \hat{*} i_2 & = & \alpha(\{z_1 * z_2 \mid z_1 \in \gamma(i_1) \ \wedge \ z_2 \in \gamma(i_2)\}) \\
i_1 \hat{/} i_2 & = & \alpha(\{z_1 / z_2 \mid z_1 \in \gamma(i_1) \ \wedge \ z_2 \in \gamma(i_2)\})
\end{array}
$$

Implementable version:

$$
\begin{array}{rcl}
\bot \hat{+} i & = & \\
i \hat{+} \bot & = & \\
[l_1, u_1] \hat{+} [l_2, u_2] & = & \\
[l_1, u_1] \hat{-} [l_2, u_2] & = & \\
[l_1, u_1] \hat{*} [l_2, u_2] & = & \\
[l_1, u_1] \hat{/} [l_2, u_2] & = &
\end{array}
$$

Abstract Execution of Commands

$$f_c : \mathbb{S} \to \mathbb{S}$$

$$f_c(s) = \begin{cases} s & \textbf{cmd}(c) = skip \\ [x \mapsto eval(e, s)]s & \textbf{cmd}(c) = x := e \\ [x \mapsto s(x) \sqcap [-\infty, n-1]]s & \textbf{cmd}(c) = x < n \end{cases}$$

Equation

We aim to compute

$$X : \mathbb{C} \to \mathbb{S}$$

such that

$$X = \lambda c.\ f_c\Big(\bigsqcup_{c' \hookrightarrow c} X(c')\Big)$$

In fixed point form:

$$X = F(X)$$

where

$$F(X) = \lambda c.\ f_c\Big(\bigsqcup_{c' \hookrightarrow c} X(c')\Big)$$

The solution of the equation is a fixed point of

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

Fixed Point Computation

The least fixed point computation may not converge:

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

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

1. widening iteration:

$$\begin{array}{rcll} X_0 & = & \bot & \\ X_i & = & X_{i-1} & \text{if } F(X_{i-1}) \sqsubseteq X_{i-1} \\ & = & X_{i-1} \bigtriangledown F(X_{i-1}) & \text{otherwise} \end{array}$$

2. narrowing iteration:

$$Y_i = \begin{cases} \hat{A} & \text{if } i = 0 \\ Y_{i-1} \bigtriangleup F(Y_{i-1}) & \text{if } i > 0 \end{cases} \quad (1)$$

($\hat{A}$ 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?
- · · ·

Abstract Interpretation Theory