CS150 APL: Abstract Interpretation

Guannan Wei
guannan.wei@tufts.edu

Nov 20, 2025

Tufts University

Today

Last time

- Abstract interpretation of conditionals and while loops
- Fixpoint iteration

Today

- More expressive abstract domains
- Widening
- Narrowing

Numeric abstract domains

- Signs
- Intervals
- ...

Interval

- Intervals $I = \{[l, u] \mid l, u \in \mathbb{R} \cup \{-\infty, +\infty\}\} \cup \{\bot\}$
- Example:
  - $[1, 5]$
  - $[-\infty, 3]$
  - $[4, +\infty]$

Interval

- Intervals $I = \{[l, u] \mid l, u \in \mathbb{R} \cup \{-\infty, +\infty\}\} \cup \{\bot\}$
- Example:
  - $[1, 5]$
  - $[-\infty, 3]$
  - $[4, +\infty]$
- Arithmetic operations:
  - Addition: $[l_1, u_1] + [l_2, u_2] = [l_1 + l_2, u_1 + u_2]$
  - Subtraction: $[l_1, u_1] - [l_2, u_2] = [l_1 - u_2, u_1 - l_2]$
  - Multiplication and division can be defined too, see Wikipedia "Interval arithmetic".
- Examples:
  - $[1, 3] + [4, 5] = ?$
  - $[2, 6] - [1, 4] = ?$

Intervals as lattice

- Top element: $\top = [-\infty, +\infty]$
- Bottom element: $\bot$

Intervals as lattice

- Top element: $\top = [-\infty, +\infty]$
- Bottom element: $\bot$
- Partial order: $[l_1, u_1] \sqsubseteq [l_2, u_2]$ iff $l_2 \leq l_1$ and $u_1 \leq u_2$
  - Example: $[2, 4] \sqsubseteq [1, 5]$

Intervals as lattice

- Join:
  - $[l_1, u_1] \sqcup [l_2, u_2] = [\min(l_1, l_2), \max(u_1, u_2)]$
  - If one operand is $\bot$, return the other operand
  - Example: $[2, 4] \sqcup [3, 5] = [2, 5]$, $[2, 4] \sqcup [5, 6] = [2, 6]$

Intervals as lattice

- Join:
  - $[l_1, u_1] \sqcup [l_2, u_2] = [\min(l_1, l_2), \max(u_1, u_2)]$
  - If one operand is $\bot$, return the other operand
  - Example: $[2, 4] \sqcup [3, 5] = [2, 5]$, $[2, 4] \sqcup [5, 6] = [2, 6]$
- Meet:
  - $[l_1, u_1] \sqcap [l_2, u_2] = \bot$, if $\max(l_1, l_2) > \min(u_1, u_2)$
  - $[l_1, u_1] \sqcap [l_2, u_2] = [\max(l_1, l_2), \min(u_1, u_2)]$
  - If one operand is $\bot$, return $\bot$
  - Example: $[2, 4] \sqcap [3, 5] = [3, 4]$, $[2, 3] \sqcap [4, 5] = \bot$

Analyzing loops with intervals

Consider analyzing the following program:

```
x := 0;
while (e) {
  x := x + 1;
}
```

What is the interval for `x` at the end of the program?

Analyzing loops with intervals

Consider analyzing the following program:

```
x := 0;
while (e) {
  x := x + 1;
}
```

What is the interval for `x` at the end of the program?

- Initial state: $\{x \mapsto [0, 0]\}$
- After first iteration: $\{x \mapsto [0, 1]\}$
- After second iteration: $\{x \mapsto [0, 2]\}$
- ...

Why intervals may not converge

- Interval domain does not satisfy the **ascending chain condition** (ACC).
  - As a lattice, interval lattice has an infinite height.
  - $[0,0] \sqsubseteq [0,1] \sqsubseteq [0,2] \sqsubseteq [0,3] \sqsubseteq \ldots$ (infinite steps until convergence) $\cdots \sqsubseteq [0,+\infty]$
- ACC: there is no infinite strictly ascending chain of elements.
- Simply using Kleene's fixpoint iteration does not guarantee termination.

Widening

- Widening is a technique to enforce convergence in abstract interpretation.
- Widening operator $A \nabla B$ is an over-approximation of $A \sqcup B$
  - $A \sqcup B \sqsubseteq A \nabla B$

Widening

- Widening is a technique to enforce convergence in abstract interpretation.
- Widening operator $A \nabla B$ is an over-approximation of $A \sqcup B$
  - $A \sqcup B \sqsubseteq A \nabla B$
- Ensures convergence of ascending chains, and reaches the termination at post-fixed-points.
  - For ascending chain $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq ...$, ascending chain $d_0^\nabla \sqsubseteq d_1^\nabla \sqsubseteq d_2^\nabla ...$ converges in finite steps
  - where $d_0^\nabla = d_0$ and $d_{i+1}^\nabla = d_i^\nabla \nabla d_{i+1}$

Widening for intervals

- Widening operator for intervals:
  - $[l_1, u_1] \nabla \bot = [l_1, u_1]$
  - $\bot \nabla [l_2, u_2] = [l_2, u_2]$
  - $[l_1, u_1] \nabla [l_2, u_2] = [l, u]$, where
    - $l = \begin{cases} -\infty & \text{if } l_2 < l_1 \\ l_1 & \text{otherwise} \end{cases}$
    - $u = \begin{cases} +\infty & \text{if } u_1 < u_2 \\ u_1 & \text{otherwise} \end{cases}$

Widening for intervals

- Widening operator for intervals:
  - $[l_1, u_1] \nabla \bot = [l_1, u_1]$
  - $\bot \nabla [l_2, u_2] = [l_2, u_2]$
  - $[l_1, u_1] \nabla [l_2, u_2] = [l, u]$, where
    - $l = \begin{cases} -\infty & \text{if } l_2 < l_1 \\ l_1 & \text{otherwise} \end{cases}$
    - $u = \begin{cases} +\infty & \text{if } u_1 < u_2 \\ u_1 & \text{otherwise} \end{cases}$
- $[2, 3] \nabla [2, 5] = ?$
- $[2, 3] \nabla [1, 4] = ?$
- $[1, 4] \nabla [2, 3] = ?$

Widening for intervals

- Widening operator for intervals:
  - $[l_1, u_1] \nabla \bot = [l_1, u_1]$
  - $\bot \nabla [l_2, u_2] = [l_2, u_2]$
  - $[l_1, u_1] \nabla [l_2, u_2] = [l, u]$, where
    - $l = \begin{cases} -\infty & \text{if } l_2 < l_1 \\ l_1 & \text{otherwise} \end{cases}$
    - $u = \begin{cases} +\infty & \text{if } u_1 < u_2 \\ u_1 & \text{otherwise} \end{cases}$
- $[2, 3] \nabla [2, 5] = [2, +\infty]$
- $[2, 3] \nabla [1, 4] = [-\infty, +\infty]$
- $[1, 4] \nabla [2, 3] = [1, 4]$

Using widening in fixpoint iteration

Algorithmically:

```
def fix(f: D -> D, init: D) -> D:
  val next = f(init)
  if (next == init) next
  else fix(f, widen(init, next)) // before: fix(f, next)
```

Using widening in fixpoint iteration

Consider analyzing the following program:

```
x := 0;
while (e) {
  x := x + 1;
}
```

Using widening in fixpoint iteration

Consider analyzing the following program:

```
x := 0;
while (e) {
  x := x + 1;
}
```

- Converges at $[0,0] \nabla [0,1] = [0, +\infty]$

Narrowing

- Idea: after reaching a post-fixed-point using widening, we can use narrowing to refine the result.
- $[l, u] \Delta \bot = \bot$
- $\bot \Delta [l, u] = \bot$
- $[l_1, u_1] \Delta [l_2, u_2] = [l, u]$, where
  - $l = \begin{cases} l_2 & \text{if } l_1 = -\infty \\ l_1 & \text{otherwise} \end{cases}$
  - $u = \begin{cases} u_2 & \text{if } u_1 = +\infty \\ u_1 & \text{otherwise} \end{cases}$

Numeric abstract domains

- Non-relational domains (they do not capture relationships between variables)
  - Signs $\{+, -, 0, \top, \bot\}$
  - Intervals $[l, u]$

Numeric abstract domains

- Non-relational domains (they do not capture relationships between variables)
  - Signs $\{+, -, 0, \top, \bot\}$
  - Intervals $[l, u]$
- Relational domains
  - Polyhedra $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \leq c$
  - Octagons $\pm x \pm y \leq c$
  - ...

Numeric abstract domains

- Non-relational domains (they do not capture relationships between variables)
  - Signs $\{+, -, 0, \top, \bot\}$
  - Intervals $[l, u]$
- Relational domains
  - Polyhedra $c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \leq c$
  - Octagons $\pm x \pm y \leq c$
  - ...
- Further reading:
  - Tutorial with Scala code:
    https://continuation.passing.style/blog/writing-abstract-interpreter-in-scala.html
  - Introduction to Neural Network Verification, Aws Albarghouthi:
    https://arxiv.org/abs/2109.10317