

Lecture 16
Computing Interval Bounds
Chaotic Iteration
Widening and Narrowing

Exercise: Abstract Postcondition for Intervals

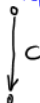
$$\begin{aligned} A &= \{\perp\} \cup \{[p, q] \mid p \in \{-\infty\} \cup \mathbb{Z}, q \in \mathbb{Z} \cup \{\infty\}, p \leq q\} \\ \gamma([p, q]) &= \{x \in \mathbb{Z} \mid p \leq x \leq q\}, \quad \gamma(\perp) = \emptyset \\ a_1 \sqsubseteq a_2 &\stackrel{\text{def}}{\iff} \gamma(a_1) \subseteq \gamma(a_2) \\ \alpha(c) &= [\inf(c), \sup(c)], \text{ if } c \neq \emptyset \quad \alpha(\emptyset) = \perp \end{aligned}$$

Consider program with two variables x, y . We track interval for each.
Domain for one program point: $\bar{A} = A \times A$ of intervals for x and for y
 $\bar{\gamma}([p_1, q_1], [p_2, q_2]) = \gamma([p_1, q_1]) \times \gamma([p_2, q_2])$, $\bar{\alpha}$ analogous
Define effect of statement c on two intervals using $(\bar{\alpha}, \bar{\gamma})$ and sp :

$$sp^\#([p_1, q_1], [p_2, q_2], c) = \bar{\alpha}(sp(\bar{\gamma}([p_1, q_1], [p_2, q_2]), \rho(c)))$$

Let c be the assignment $x = 2 * x - 5 * y$. Compute $sp^\#$ for this case.

$x: [p_1, q_1]$
 $y: [p_2, q_2]$



Automatic Computation of $sp^\#$ on Intervals

Describe an algorithm that computes $sp^\#(a, c)$ for the domain of two intervals, where c is given as an input in terms of its defining formula $R(c) = T(x, y, x', y')$. Assume that the formula is in Presburger arithmetic. For example, the previous case is when $\mathcal{R}(x, y, x', y')$ is $x' = 2 * x - 5 * y \wedge y' = y$

$$sp^\#([p_1, q_1], [p_2, q_2]) = ([p_1', q_1'], [p_2', q_2'])$$

$$p_1' = \overset{(\text{inf})}{\min} \{ x' \mid \exists x, y \in [p_1, q_1] \times [p_2, q_2]. \exists y'. T(x, y, x', y') \}$$

Maximization using Quantifier Elimination

(Usually not the best way, but possible.)

$$\max\{t(\bar{x}) \mid F(\bar{x})\}$$

Maximization using Quantifier Elimination

(Usually not the best way, but possible.)

$$\bar{a} = \max\{t(\bar{x}) \mid F(\bar{x})\}$$

The result is \bar{a} such that:

$$F(\bar{a}) \wedge \forall \bar{x}. (F(\bar{x}) \rightarrow t(\bar{x}) \leq t(\bar{a}))$$

How to find such \bar{a} ?

Maximization using Quantifier Elimination

(Usually not the best way, but possible.)

$$\max\{t(\bar{x}) \mid F(\bar{x})\}$$

The result is \bar{a} such that:

$$F(\bar{a}) \wedge \forall \bar{x}. (F(\bar{x}) \rightarrow t(\bar{x}) \leq t(\bar{a}))$$

How to find such \bar{a} ? Check satisfiability of the above formula.

Maximization using Quantifier Elimination

(Usually not the best way, but possible.)

$$\max\{t(\bar{x}) \mid F(\bar{x})\}$$

The result is \bar{a} such that:

$$F(\bar{a}) \wedge \forall \bar{x}. (F(\bar{x}) \rightarrow t(\bar{x}) \leq t(\bar{a})) \quad F(\bar{a}, b)$$

How to find such \bar{a} ? Check satisfiability of the above formula.

What if F has some parameters, say b

$$x' = 2 * x - 5 * y + b$$

Maximization using Quantifier Elimination

(Usually not the best way, but possible.)

SP#

$$\max\{t(\bar{x}) \mid F(\bar{x})\}$$

The result is \bar{a} such that:

$$F(\bar{a}) \wedge \forall \bar{x}. (F(\bar{x}) \rightarrow t(\bar{x}) \leq t(\bar{a}))$$

How to find such \bar{a} ? Check satisfiability of the above formula.

What if F has some parameters, say b

$$x' = 2 * x - 5 * y + b$$

The result is described by

$F'(\bar{a}, b)$

choose \bar{a} . $F(\bar{a}) \wedge \forall \bar{x}. (F(\bar{x}) \rightarrow t(\bar{x}) \leq t(\bar{a}))$

We can use synthesis to recover such \bar{a}

Exercise

Draw the control-flow graph for the following program.

Run abstract interpretation that maintains an interval for x at each program point, until you reach a fixpoint.

What are the fixpoint values at program points v_4 and v_5 ?

```
// v0
x := 0;
// v1
while (x < 10) {
  // v2
  x := x + 3;
}
// v3
if (x >= 0) {
  if (x <= 15) {
    a[x]=7; // index in range
  } else {
    // v4
    error;
  }
} else {
  // v5
  error;
}
```

More about Fixed Point Iterations

Comparing Limits of Sequences

partial order

Lemma: Let (L, \sqsubseteq) be a lattice and let $x_i, y_i \in L$ and for $i \geq 0$ be sequences such that for each i there exists j such that $x_i \sqsubseteq y_j$. Suppose that there exist x_* and y_* (e.g. if L is a complete lattice) such that

$$x_* = \bigsqcup_{i \geq 0} x_i$$

$$y_* = \bigsqcup_{j \geq 0} y_j$$

Then $x_* \sqsubseteq y_*$.

Proof: Take any x_i . Then there is y_j such

$$x_i \sqsubseteq y_j \sqsubseteq y_*$$

Thus, y_* is an upper bound on the set $\{x_i \mid i \geq 0\}$. Because x_* is the least upper bound, $x_* \sqsubseteq y_*$.

Parallel versus “Imperative” Updates

Consider a function $H : \mathbb{A}^2 \rightarrow \mathbb{A}^2$ (e.g. $A = \mathbb{R}$)

This function can be given by two component functions $H_x, H_y : \mathbb{R}^2 \rightarrow \mathbb{R}$, one for each result component of the pair:

$$H(x_1, x_2) = (H_1(x_1, x_2), H_2(x_1, x_2))$$

Consider an imperative program that iterates H . Compare two versions.

First compute both values, then update:

```
while (iter < max) {  
  x1next = H1(x1,x2)  
  x2next = H2(x1,x2)  
  x1 = x1next  
  x2 = x2next  
  iter = iter + 1  
}
```

Immediately update:

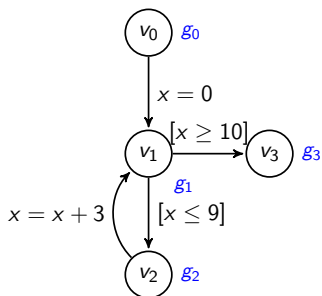
```
while (iter < max) {  
  x1 = H1(x1,x2)  
  x2 = H2(x1,x2)  
  iter = iter + 1  
}
```

Do these two loops behave the same in the limit?

Can one converge and the other not?

Abstract Semantic Function for the Program

In Collecting Semantics for Example Program we had



$$F(g_0, g_1, g_2, g_3) =$$
$$(\mathbb{Z},$$
$$sp(g_0, x := 0) \sqcup sp(g_2, x := x + 3),$$
$$sp(g_1, assume(x \leq 9)),$$
$$sp(g_1, assume(x \geq 10)))$$

Here we have:

$$F^\#(g_0^\#, g_1^\#, g_2^\#, g_3^\#) =$$
$$(\top,$$
$$sp^\#(g_0^\#, x := 0) \sqcup sp^\#(g_2^\#, x := x + 3),$$
$$sp^\#(g_1^\#, assume(x \leq 9)),$$
$$sp^\#(g_1^\#, assume(x \geq 10)))$$

Parallel and Iterative Updates in Abstract Interpretation

Program points v_1, \dots, v_n . Solving the system of n equations in variables g_1, \dots, g_n

$$\begin{aligned}g_1 &= H_1^{\#}(g_1, \dots, g_n) \\ \dots \\ g_n &= H_n^{\#}(g_1, \dots, g_n)\end{aligned}$$

where

$$H_i^{\#}(g_1, \dots, g_n) = \text{Init}_i \sqcup \bigsqcup_{(v_j, v_i) \in E} \text{sp}^{\#}(g_j, r(v_j, v_i))$$

$\text{lfp}(H) = \bigsqcup_{k \geq 0} H^k(\perp, \dots, \perp) = \bigsqcup_{k \geq 0} (g_1^k, \dots, g_n^k)$ (this is parallel iteration)

parallel iteration step

$$g_1^{k+1} = H_1(g_1^k, \dots, g_n^k)$$

\dots

$$g_i^{k+1} = H_i(g_1^k, \dots, g_n^k)$$

\dots

$$g_n^{k+1} = H_n(g_1^k, \dots, g_n^k)$$

chaotic iteration step

$$g_i^{k+1} = H_i(g_1^k, \dots, g_n^k)$$

$$g_j^{k+1} = g_j^k, \text{ for all } j \neq i$$

In **chaotic iteration** we select some equation i to update and keep the rest same.

Questions for Chaotic Iteration

- ▶ What is the cost of doing one chaotic versus one parallel iteration?

Questions for Chaotic Iteration

- ▶ What is the cost of doing one chaotic versus one parallel iteration?
Chaotic is n times cheaper!
- ▶ Does chaotic iteration converge if parallel converges?

Questions for Chaotic Iteration

- ▶ What is the cost of doing one chaotic versus one parallel iteration?
Chaotic is n times cheaper!
- ▶ Does chaotic iteration converge if parallel converges?
- ▶ If chaotic converges, will it converge to same value as parallel?

Questions for Chaotic Iteration

- ▶ What is the cost of doing one chaotic versus one parallel iteration?
Chaotic is n times cheaper!
- ▶ Does chaotic iteration converge if parallel converges?
- ▶ If chaotic converges, will it converge to same value as parallel?
- ▶ If chaotic converges, how many steps will convergence take?

Questions for Chaotic Iteration

- ▶ What is the cost of doing one chaotic versus one parallel iteration?
Chaotic is n times cheaper!
- ▶ Does chaotic iteration converge if parallel converges?
- ▶ If chaotic converges, will it converge to same value as parallel?
- ▶ If chaotic converges, how many steps will convergence take?
- ▶ What is a good way of choosing index i (iteration strategy)?

Questions for Chaotic Iteration

- ▶ What is the cost of doing one chaotic versus one parallel iteration?
Chaotic is n times cheaper!
- ▶ Does chaotic iteration converge if parallel converges?
- ▶ If chaotic converges, will it converge to same value as parallel?
- ▶ If chaotic converges, how many steps will convergence take?
- ▶ What is a good way of choosing index i (iteration strategy)?
Example: take some permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ of equations. At step k select equation $\pi(k \% n)$
More generally: a fair strategy, for each vertex and each position in the sequence, there exists a position afterwards where this vertex is chosen.

Sequences in Chaotic vs Parallel Iteration

$\perp, L_1, L_2, \dots, L_n, \dots$, given by parallel iteration: $(g_1^k, \dots, g_n^k) = F^{\#k}(\bar{\perp})$

$\perp, C_1, C_2, \dots, C_n, \dots$, given by chaotic iteration: $(g_1^k, \dots, g_n^k) = F_C^{\#k}(\bar{\perp})$

These two sequences are given by monotonic functions $F_C^{\#}$ and $F^{\#}$. Clearly

$$F_C^{\#}(g_1, \dots, g_n) \sqsubseteq F^{\#}(g_1, \dots, g_n)$$

Compare values $L_1, C_1, \dots, L_n, C_n$ in the lattice:

Sequences in Chaotic vs Parallel Iteration

$\perp, L_1, L_2, \dots, L_n, \dots$, given by parallel iteration: $(g_1^k, \dots, g_n^k) = F^{\#k}(\bar{\perp})$

$\perp, C_1, C_2, \dots, C_n, \dots$, given by chaotic iteration: $(g_1^k, \dots, g_n^k) = F_L^{\#k}(\bar{\perp})$

These two sequences are given by monotonic functions $F_C^{\#}$ and $F^{\#}$. Clearly

$$F_C^{\#}(g_1, \dots, g_n) \sqsubseteq F_L^{\#}(g_1, \dots, g_n)$$

Compare values $L_1, C_1, \dots, L_n, C_n$ in the lattice:

- ▶ $C_0 \sqsubseteq L_0$, generally $C_i \sqsubseteq L_i$ (proof by induction)

Sequences in Chaotic vs Parallel Iteration

$\perp, L_1, L_2, \dots, L_n, \dots$, given by parallel iteration: $(g_1^k, \dots, g_n^k) = F^{\#k}(\bar{\perp})$

$\perp, C_1, C_2, \dots, C_n, \dots$, given by chaotic iteration: $(g_1^k, \dots, g_n^k) = F_L^{\#k}(\bar{\perp})$

These two sequences are given by monotonic functions $F_C^{\#}$ and $F^{\#}$. Clearly

$$F_C^{\#}(g_1, \dots, g_n) \sqsubseteq F_L^{\#}(g_1, \dots, g_n)$$

Compare values $L_1, C_1, \dots, L_n, C_n$ in the lattice:

- ▶ $C_0 \sqsubseteq L_0$, generally $C_i \sqsubseteq L_i$ (proof by induction)
- ▶ when selecting equations by fixed permutation, $L_1 \sqsubseteq C_n$, generally $L_i \sqsubseteq C_{ni}$

Therefore, using the Lemma from on Comparing Fixpoints of Sequences twice, we have that these two sequences converge to the same value.

Worklist Algorithm and Iteration Strategies

$H_i(g_1, \dots, g_n)$ depends only on small number of g_j , namely predecessors of node v_i .

If we chose i , next time it suffices to look at successors of i
(This saves traversing CFG).

This leads to a worklist algorithm:

- ▶ initialize lattice, put all CFG node indices into worklist
- ▶ choose i , compute new g_i , remove i from worklist
- ▶ if g_i has changed, update it and add to worklist all j where v_j is a successor of v_i

Algorithm terminates when worklist is empty (no more changes).

Useful iteration strategy: reverse postorder and strongly connected components.

Reverse postorder: follow changes through successors in the graph.

Strongly connected component (SCC) of a directed graph: path between each two nodes of component.

- ▶ compute until fixpoint within each SCC

For intervals of integers, what is the height of the lattice?

What is the bound on the number of iterations?

Widening and Narrowing in Variable Range Analysis

Interval analysis domain, for each program point, maps each program variable to an interval.

Analysis domain has elements $g^\# : V \rightarrow I$ where I denotes the set of such intervals.

Height of the entire lattice for unbounded integers:

Widening and Narrowing in Variable Range Analysis

Interval analysis domain, for each program point, maps each program variable to an interval.

Analysis domain has elements $g^\# : V \rightarrow I$ where I denotes the set of such intervals.

Height of the entire lattice for unbounded integers: infinite.

Widening and Narrowing in Variable Range Analysis

Interval analysis domain, for each program point, maps each program variable to an interval.

Analysis domain has elements $g^\# : V \rightarrow I$ where I denotes the set of such intervals.

Height of the entire lattice for unbounded integers: infinite.

Height of lattice of one interval for 64-bit integers:

Widening and Narrowing in Variable Range Analysis

Interval analysis domain, for each program point, maps each program variable to an interval.

Analysis domain has elements $g^\# : V \rightarrow I$ where I denotes the set of such intervals.

Height of the entire lattice for unbounded integers: infinite.

Height of lattice of one interval for 64-bit integers: around 2^{64}

Widening and Narrowing in Variable Range Analysis

Interval analysis domain, for each program point, maps each program variable to an interval.

Analysis domain has elements $g^\# : V \rightarrow I$ where I denotes the set of such intervals.

Height of the entire lattice for unbounded integers: infinite.

Height of lattice of one interval for 64-bit integers: around 2^{64}

Moreover, if we have q variables in program and p program points, height of lattice for the analysis domain is pq times larger.

Widening and Narrowing in Variable Range Analysis

Interval analysis domain, for each program point, maps each program variable to an interval.

Analysis domain has elements $g^\# : V \rightarrow I$ where I denotes the set of such intervals.

Height of the entire lattice for unbounded integers: infinite.

Height of lattice of one interval for 64-bit integers: around 2^{64}

Moreover, if we have q variables in program and p program points, height of lattice for the analysis domain is pq times larger.

How to guarantee (reasonably fast) termination?

Widening technique

If the iteration does not seem to be converging, take a "jump" and make the interval much **wider** (larger).

Finite set of *jump points* J (e.g. set of all integer constants in the program)

In fixpoint computation, compose $F_i^\#$ with function

$$w([a, b]) = [\max\{x \in J \mid x \leq a\}, \min\{x \in J \mid b \leq x\}]$$

(For multiple variables, do this for every interval.)

We require the condition:

$$x \sqsubseteq W(x)$$

for all x .

The condition holds for the example above.

When to Apply Widening

- ▶ Iterate a few times without using w , if we are not at a fixpoint at this program point, then widen.
This is not monotonic: if you start at fixpoint, it converges, if start below, can jump over fixpoint!
- ▶ Always apply widening. We will assume this.

Standard iteration: $\perp, F^\#(\perp), \dots, (F^\#)^n(\perp), \dots$

Widening: $\perp, (W \circ F^\#)(\perp), \dots, ((W \circ F^\#)^n(\perp), \dots$

Example where widening works nicely

Consider program:

```
x = 0;
while (x < 1000) {
  x = x + 1;
}
```

Interval analysis without widening will need around 1000 iterations to converge to interval $[1000, 1000]$ for x at the end of the program.

This may be too slow.

Let us derive the set J by taking all constants that appear in the program, as well as $-\infty$ and $+\infty$:

$$J = \{-\infty, 0, 1, 1000, +\infty\}$$

After a few iterations, widening maps interval $[0, 2]$ into $[0, 1000]$. This gives $[0, 999]$ for x at loop entry and again $[1000, 1000]$ for x at the end of the program, but in many fewer iterations.

Example showing problems with widening

Consider program:

```
x = 0;
y = 1;
while (x < 1000) {
  x = x + 1;
  y = 2*x;
  y = y + 1;
  print(y);
}
```

Interval analysis without widening will need around 1000 iterations to converge to

$$x \mapsto [1000, 1000]; \quad y \mapsto [1, 2001]$$

This may be too slow.

Now apply widening with the same J as before. When within loop we obtain $x \mapsto [0, 1000]$, applying widening function to the interval $[0, 2000]$ for y results in $[0, +\infty)$. We obtain $y \mapsto [1, +\infty)$ at the end of the program:

$$x \mapsto [1000, 1000]; \quad y \mapsto [1, +\infty)$$

Narrowing

Observation

Consider a monotonic function, such as $f(x) = x/2 + 1$ on the set of real numbers.

If we consider a sequence $x_0, f(x_0), \dots$, this sequence is

- ▶ monotonically increasing iff $x_0 < x_1$ (e.g. for $x_0 = 0$)
- ▶ monotonically decreasing iff $x_1 < x_0$ (e.g. for $x_0 = 3$)

$$\begin{array}{l} f(x_0) \\ \parallel \\ x_0 < x_1 \quad / f \\ \hline f(x_0) < f(x_1) \\ \hline x_1 < x_2 \end{array}$$

Informally, the sequence continues of the direction in which it starts in the first step.

This is because $x_0 < x_1$ implies by monotonicity of f that $x_1 < x_2$ etc., whereas $x_1 < x_0$ implies $x_2 < x_1$.

The Idea

Let $W : A \rightarrow A$ such that $x \sqsubseteq W(x)$.

After finding fixpoint of $(W \circ F)^\#$, apply $F^\#$ to improve precision.

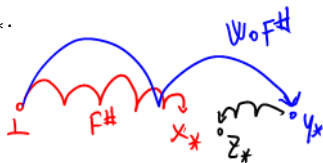
Widen and Narrow

Lemma: Let $F^\#$ and W be monotonic functions on a partial order \sqsubseteq such that $x \sqsubseteq W(x)$ for all x . Define the following:

- ▶ $x_* = \sqcup_{n \geq 0} (F^\#)^n(\perp)$
- ▶ $y_* = \sqcup_{n \geq 0} (W \circ F^\#)^n(\perp)$
- ▶ $z_* = \sqcap_{n \geq 0} (F^\#)^n(y_*)$

where we also assume that the two \sqcup and one \sqcap exist. Then

- ▶ x_* is the least fixpoint of $F^\#$
- ▶ z_* , is the least fixpoint of $W \circ F^\#$
- ▶ $x_* \sqsubseteq z_* \sqsubseteq y_*$.



Proof

By induction, for each n we have

$$(F^\#)^n(\perp) \sqsubseteq (W \circ F^\#)^n(\perp)$$

Thus by Comparing Fixpoints of Sequences, we have $x_* \sqsubseteq y_*$.

Next, we have that

$$x_* = F^\#(x_*) \sqsubseteq F^\#(y_*) \sqsubseteq (W \circ F^\#)(y_*) \sqsubseteq y_*$$

Thus, $F^\#(y_*) \sqsubseteq y_*$. From there by induction and monotonicity of $F^\#$ we obtain

$$(F^\#)^{n+1}(y_*) \sqsubseteq (F^\#)^n(y_*) \sqsubseteq y_*$$

i.e. the sequence $(F^\#)^n(y_*)$ is **decreasing**. Therefore, y_* is its upper bound and therefore $z_* \sqsubseteq y_*$.

On the other hand, we have by monotonicity of $F^\#$, the fact that x_* is fixpoint, and $x_* \sqsubseteq y_*$ that:

$$x_* = (F^\#)^n(x_*) \sqsubseteq (F^\#)^n(y_*)$$

Thus, x_* is the lower bound on $(F^\#)^n(y_*)$, so $x_* \sqsubseteq z_*$.

What is we do not have \sqcap ?

Can we still do something useful if

$$\sqcap_{n \geq 0} (F^\#)^n(y_*)$$

does not exist?

What is we do not have \sqcap ?

Can we still do something useful if

$$\sqcap_{n \geq 0} (F^\#)^n(y_*)$$

does not exist?

Even if z_* does not exist, we can simply compute $(F^\#)^n(y_*)$ for any chosen value of n , it is still a sound over-approximation, because it approximates x_* , which approximates the concrete value:

$$x_* \sqsubseteq z_n$$

so

$$s_* \subseteq \gamma(x_*) \subseteq \gamma(z_n)$$

Being able to stop at any point gives us an **anytime algorithm**.

Example showing how narrowing may improve result after widening

In the above example for the program, the results obtained using widening

```
x = 0;
y = 1;
// x -> [0,0], y -> [1,1]
// (merge point)
// x -> [0,1000], y -> [1,+infty)
while (x < 1000) {
  // x -> [0,999], y -> [1,+infty)
  x = x + 1;
  // x -> [0,1000], y -> [1,+infty)
  y = 2*x;
  // x -> [0,1000], y -> [0,+infty)
  y = y + 1;
  // x -> [0,1000], y -> [1,+infty)
  print(y);
}
// x -> [1000,1000], y -> [1,+infty)
```

are:

Example cont.

Let us now apply one ordinary iteration, without widening. We obtain:

```
x = 0;
y = 1;
// x -> [0,0], y -> [1,1]
// (merge point)
// x -> [0,1000], y -> [1,2001]
while (x < 1000) {
    // x -> [0,999], y -> [1,+infty) .
    x = x + 1;
    // x -> [0,1000], y -> [1,+infty)
    y = 2*x;
    // x -> [0,1000], y -> [0,2000]
    y = y + 1;
    // x -> [0,1000], y -> [1,2001]
    print(y);
}
// x -> [1000,1000], y -> [1,2001]
```

Thus, we obtained a good first approximation by a few iterations with widening and then improved it with a single iteration without widening.