Tutorial on Widening (and Narrowing)

Hongseok Yang
Seoul National University

Goal of Abstract Interpretation

```
int P() {
   int x = 2;
   while (x >= 0) {
      // Question: which value can "x" have here?
      x = x + 1;
   }
   return(x);
}
```

- An abstract interpretation answers this question by "executing" a program "abstractly".
- The answer is an upper approximation.

Usual Abstract Interpretation

```
x = 2; while (x >= 0) {/* i2I */ x = x+1;}
```

- First, define a finite lattice <L,?,t,u,>>.
  - Each element of L denotes a set of integers that “x” can have in the loop.
- Second, define a monotone function F:L ! L.
  - F is the abstraction of the loop body.
- Finally, compute the least fixpoint fix(F).
  - fix(F) is the limit of ?, F(?), F^2(?), ⋯
  - Since L is finite and F^n(?) is increasing, the sequence “ter minates”.
- Example: L = {?, +, 0, -, >} and F(X) = (+ t (X © +))

Infinite Abstract Domain

- Question: Can we use an infinite lattice L?
  - Using a larger L, we can obtain a better estimate of a program invariant.
  - But, fix(F) might not be computable.
- Answer: Yes, we can, if we have widening.
  - Intuitively, widening works by picking a different finite subset of L for each program F.

Goal of this Talk

- My goal is to demystify widening and narrowing:
  1. What are widening and narrowing?
  2. How do they allow us to use an infinite lattice?
  3. How to design widening and narrowing?
  4. Are they really necessary? Can we just live with finite lattices, or lattices with no strictly increasing chain?
- I’ll mostly focus on widening.

Problem Mathematically

- Question: Given an infinite lattice <L,?,t,u,>> and monotone F:L!L, compute fix(F).
  - ?, F(?), F^2(?), … converges to fix(F).
  - But, the sequence might strictly increase.
- Widening approach asks a different question: Find an upper approximation “a” of fix(F),

  fix(F) v a.

  1. First, approximate the sequence {F^n(?)} by a “terminating” sequence {a_n}: F^n(?) v a_n.
  2. Then, compute the limit “a” of {a_n}.

Widening “r: L£ L ! L”

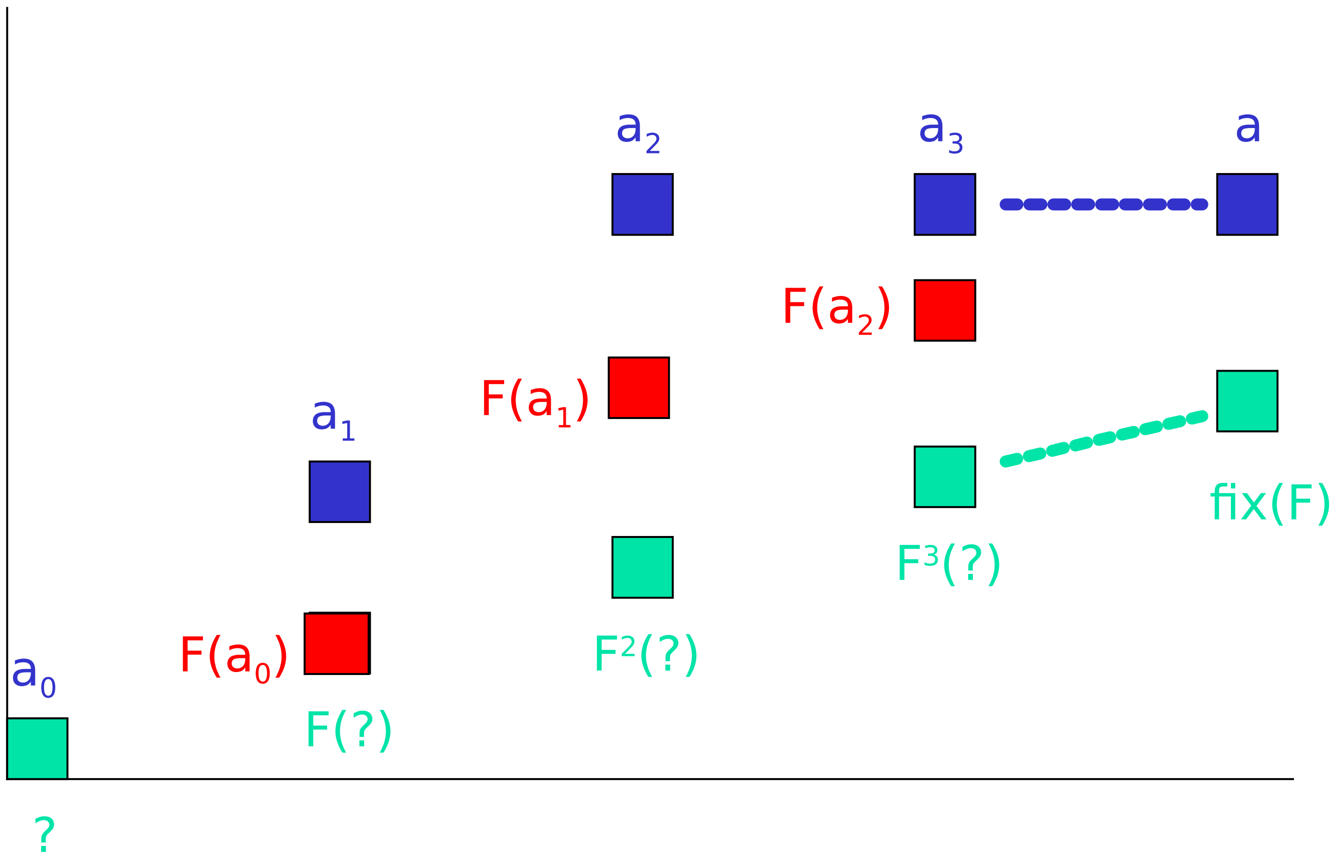
- General Dfn: Widening is what gives us $\{a_n\}$ for every F.
  - For every monotone function F, the below sequence $\{a_n\}$ approximates $\{F^n(?)\}$ and terminates:

    $a_0 = ?$ $a_{n+1} = a_n$ r $F(a_n)$

- Specific Dfn: Widening r : L £ L ! L is a function s.t.
  1. x v x r y and y v x r y; and
  2. for all increasing sequences $\{x_n\}_n$, the “widened” sequence y_n terminates:

     $y_0 = x_0$ $y_{n+1} = y_n$ r x_{n+1}

Widening



Example

- Find fix(F) in the interval domain L:

  L = {?} [{[l,u] | l,u 2 (Z [{-inf,inf}) Æ l · u}

  F(X) = [0,0] t (X + [1,1])

- F^n(?) = [0,n-1]. Thus, {F^n(?)} is strictly increasing.
- Use the following widening r:

  [l,u] r ? = ? r [l,u] = [l,u]

  [l,u] r [l',u'] = [if l' < l then -inf else l,

  if u' > u then inf else u]

- What is the limit of {a_n}?

Technique for Designing Widening

- For each non-bottom x (!= ?),
  - decide a finite lattice L_x (μ L), and
  - define (xr-) as an "abstraction" fn from L to L_x.

    (xr-) : L L_x : id
- Then, define ?rx = x.
- Example: for r in the previous slide,

  $L_{[l,u]}$ = {?, [l,u], [l,inf], [-inf,u], [-inf,inf]}
- This kind of widening lets us pick a different finite subset L_F (?) for each F.
- Exercise: F(X) = ([1,1] t (X + [-1,1])) u [-inf,4]. Design a widening whose "a" for F is [-inf,4].

Finite Domain on the Fly (My Observation)

Suppose that for each non-bottom x (!= ?), there is a subset L_x of L such that

1. L_x with the order of L is a lattice with finite height;
2. “α_x : L L_x : id” is a Galois embedding; and
3. $\alpha_x(x) = x = ?_x$.

Proposition1: If for all x and all y in L_x, L_y µ L_x, then “urw = α_u(w)” and “?rw = w” define a widening operator.

Proposition2: Moreover, if for all x and y in L_x, α_y(z) = α_x(z) t_x y, t hen for every non-strict monotone function F, this widenin g picks $L_{F(?)}$ in the computation: the n+1-th term a_{n+1} of the w idened sequence is the n-th iterate of ($\alpha_{F(?)}$ o F): $L_{F(?)}$! $L_{F(?)}$.

Widening Sequence $\{r_i\}$

- Is there a widening that uses $F^k(?)$ to decide a finite subset f or F?
  - Need to use different "widening" r_i for a_i.
- For every monotone function F, the below sequence $\{a_n\}$ app roximates $\{F^n(?)\}$ and terminates:

$$a_0 = ? \qquad\qquad a_{n+1} = a_n \, r_n \, F(a_n)$$

- A widening sequence $\{r_n\}$ is a sequence of fns s.t.
  1. $x \text{ v } x r_n y$ and $y \text{ v } x r_n y$; and
  2. for all increasing sequences $\{x_n\}_n$, the "widened" sequenc e y_n terminates:

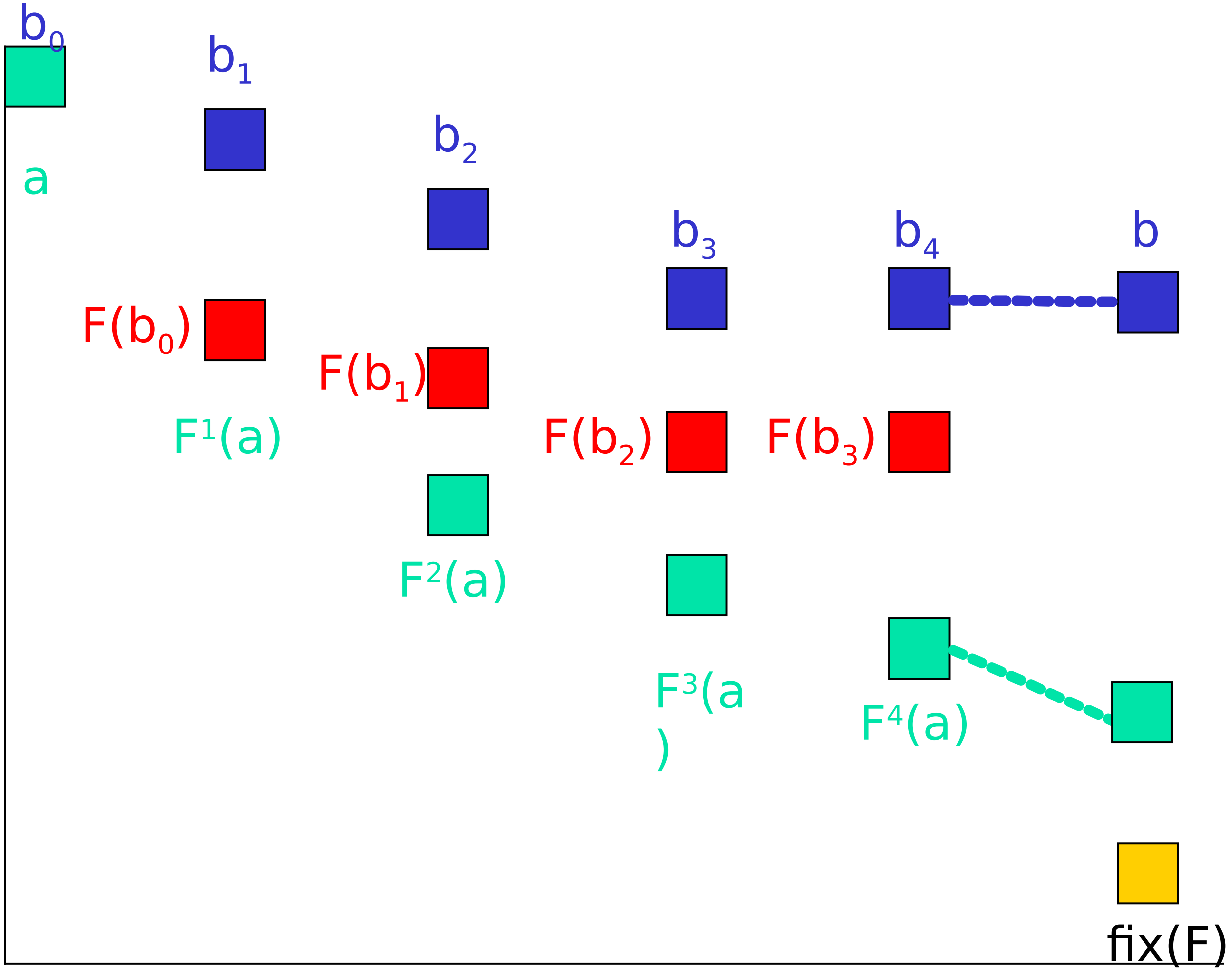
$$y_0 = x_0 \qquad\qquad y_{n+1} = y_n r_n x_{n+1}$$

- Design Technique: Usually, $r_0 = \cdots = r_{k-1} = t$, and $r_k = r_{k+1} = \cdots$
- Exercise: F(X) = ([1,1] t (X + [-10,10])) u [-inf,30].

Increase the Precision

- Problem:
  - Suppose we obtained an approximation "a" of fix(F) via widening.
  - From "a", find a more precise approximation.
- Not-computable "solution":
  - a, F(a), $F^2(a)$, $F^3(a)$, ⋯
  - For each n, fix(F) v $F^n(a)$ and F^{n+1} v $F^n(a)$. (* a w F(a))
- Narrowing approach:
  1. Approximate the sequence $\{F^n(a)\}$ by a "terminating" decreasing sequence $\{b_n\}$: F^n (a) v b_n.
  2. Then, b is the limit of $\{b_n\}$.

Narrowing



Narrowing

- General Dfn: Narrowing is what gives us $\{b_n\}$ for every F.
  - For every monotone function F, the below sequence {b $_n$} approximates $\{F^n(a)\}$, is decreasing and terminates:

$$b_0 = a \qquad\qquad b_{n+1} = b_n \text{ r } F(b_n)$$

- Specific Dfn: Narrowing Δ: L£ L ! L is a function such that
  1. x w x Δ y w y, and
  2. for all decreasing sequences $\{x_n\}_n$, the “narrowed” sequ ence $\{y_n\}$ terminates:

$$y_0 = x_0 \qquad\qquad y_{n+1} = y_n \Delta\, x_{n+1}$$

Widening/Narrowing Example

- F(X) = ([1,1] t (X+[1,1])) u [-inf,100]
- Widening:
  - ?rx = xr? = ?
  - [l,u] r [l’,u’] = [if l > l’ then -inf else l,
    if u < u’ then inf else u]
- Narrowing:
  - ? Δ x = x Δ ? = ?
  - [l,u] Δ [l’,u’] = [if l = -inf then l’ else l,
    if u = inf then u’ else u]
- Exercise: Compute an approximation of fix(F).

Finite Lattices Are Not as Powerful as Widening.

```
int Pnm() {
   int i = n;
   while (i <= m) { // i 2 [n,m]
       i := i + 1; }
   return(i);
}
```

- With a single "finite" abstract domain, we cannot find the invariant of "P_{nm}()" for all n and m.
  - {[n,m] | n · m} has a strictly increasing sequence.
- We can find the invariant of all "P_{nm}()"s, using the interval domain and the widening and narrowing in the previous slide.

Conclusion

- Widening lets us use an infinite lattice in designin g an abstract interpreter.
  - It picks a finite subset depending on an input program.
- A common technique for designing widening is to compute F^k(?) for some fixed k, and to use the res ult to build a finite subset.
- Why don't you design an abstract interpreter bas ed on an infinite lattice and widening?