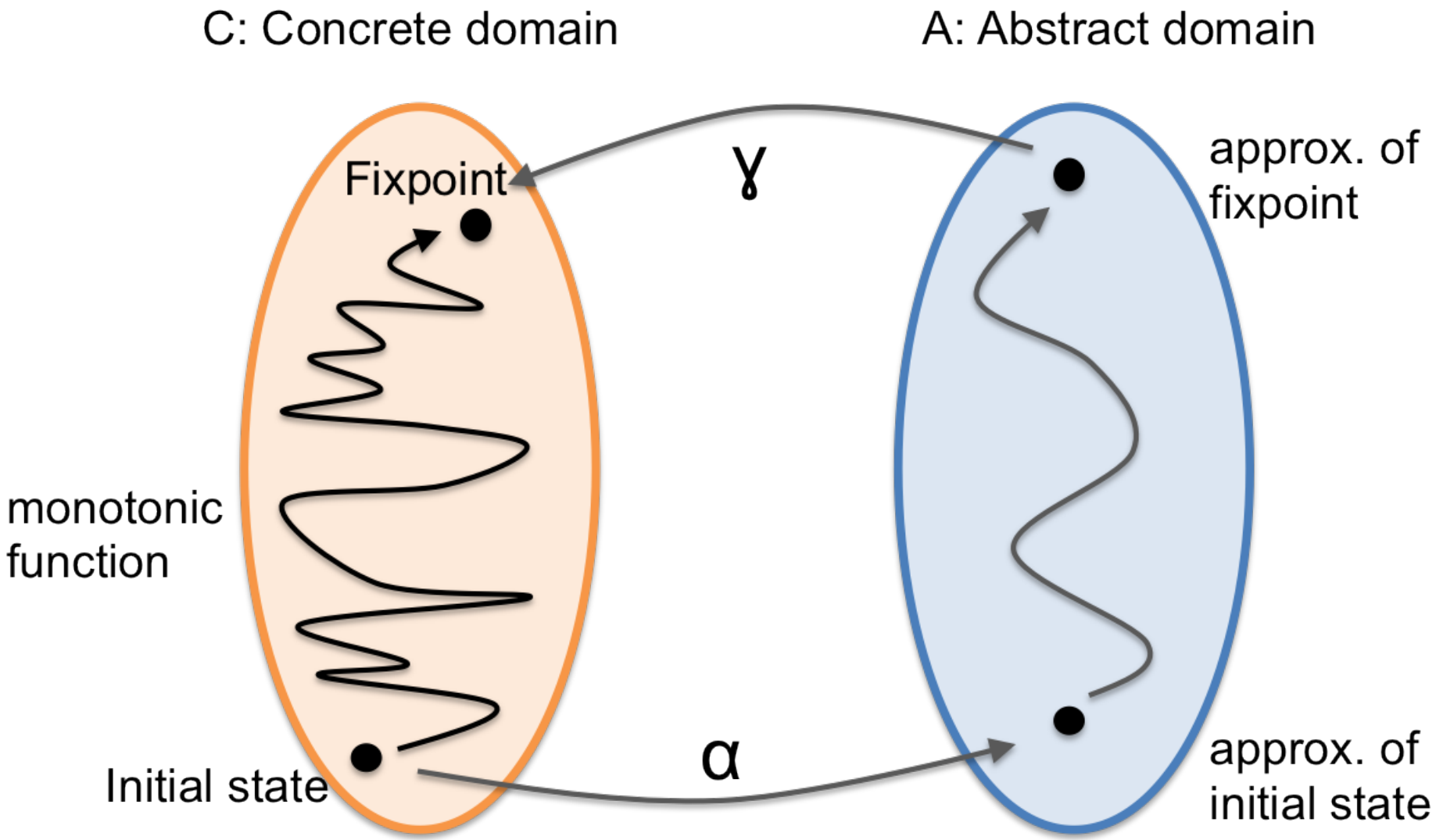
Lecturecise 10
Galois Connection and Abstract Interpretation

Viktor Kuncak

Abstract Interpretation Big Picture



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 $g^{\#} : V \to I$ where I denotes the set of such intervals.

Height of 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 H_i with function

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

We require the condition:

$$x \sqsubseteq W(x)$$

for all x.
The condition holds for the example above.

Approaches

- always apply widening (we will assume this)
- iterate a few times with H_i only (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!

Standard iteration: $\bot, \ldots, (F^\#)^n(\bot), \ldots$
Widening: $\bot, \ldots, ((W \circ F^\#)^n(\bot), \ldots$

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

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

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:

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

Narrowing

Observation

Consider a monotonic function, such as $f(x) = 0.5x + 1$ on the set of real numbers.

If we consider a sequence $x_0, f(x_0), \ldots$, 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 $x_0 = 3$)

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 $W : A \to A$ 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:

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

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

- x_* is the least fixpoint of $F^{\#}$ and z_*, is the least fixpoint of $W \circ F^{\#}$ (by Tarski's Fixpoint Theorem), and
- $x_* \sqsubseteq z_* \sqsubseteq y_*$.

Proof

By induction, for each n we have

$$(F^{\#})^n(\bot) \sqsubseteq (W \circ F^{\#})^n(\bot)$$

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

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

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 $x_* \sqsubseteq z_*$.

Note

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 are:

```
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)
```

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.