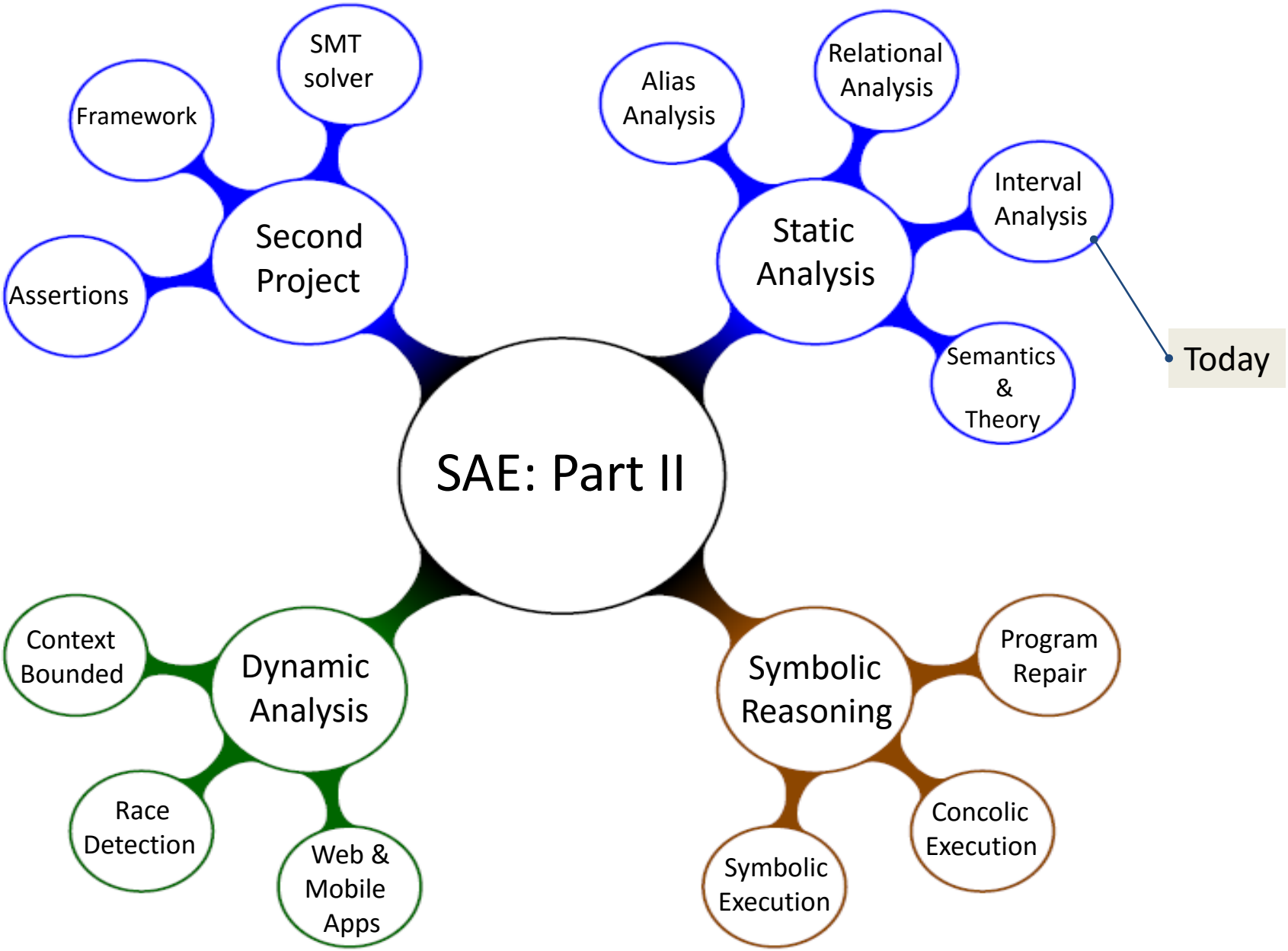
Software Architecture and Engineering: Part II

ETH Zurich, Spring 2016

Prof. Martin Vechev
http://www.srl.inf.ethz.ch/


SAE: Part II
Second Project
SMT solver
Framework
Assertions
Static Analysis
Alias Analysis
Relational Analysis
Interval Analysis
Semantics & Theory
Today
Dynamic Analysis
Context Bounded
Race Detection
Web & Mobile Apps
Symbolic Reasoning
Program Repair
Concolic Execution
Symbolic Execution

Cheat Sheet: Connecting Math and Analysis

Mathematical Concept	Use in Static Analysis
Complete Lattice	Defines Abstract Domain and ensure joins exist.
Joins ($\sqcup$)	Combines facts arriving at a program point
Bottom ($\bot$)	Used for initialization of all but initial elements
Top (T)	Used for initialization of initial elements
Widening (∇)	Used to guarantee analysis termination
Function Approximation	Critical to make sure abstract semantics approximate the concrete semantics
Fixed Points	This is what is computed by the analysis
Tarski's Theorem	Ensures fixed points exist.

So Far

So far, we saw a bunch of mathematical concepts such as lattices, functions, fixed points, function approximation, etc.

Next, we will see how to put these together in order to build static analyzers.

Starting point

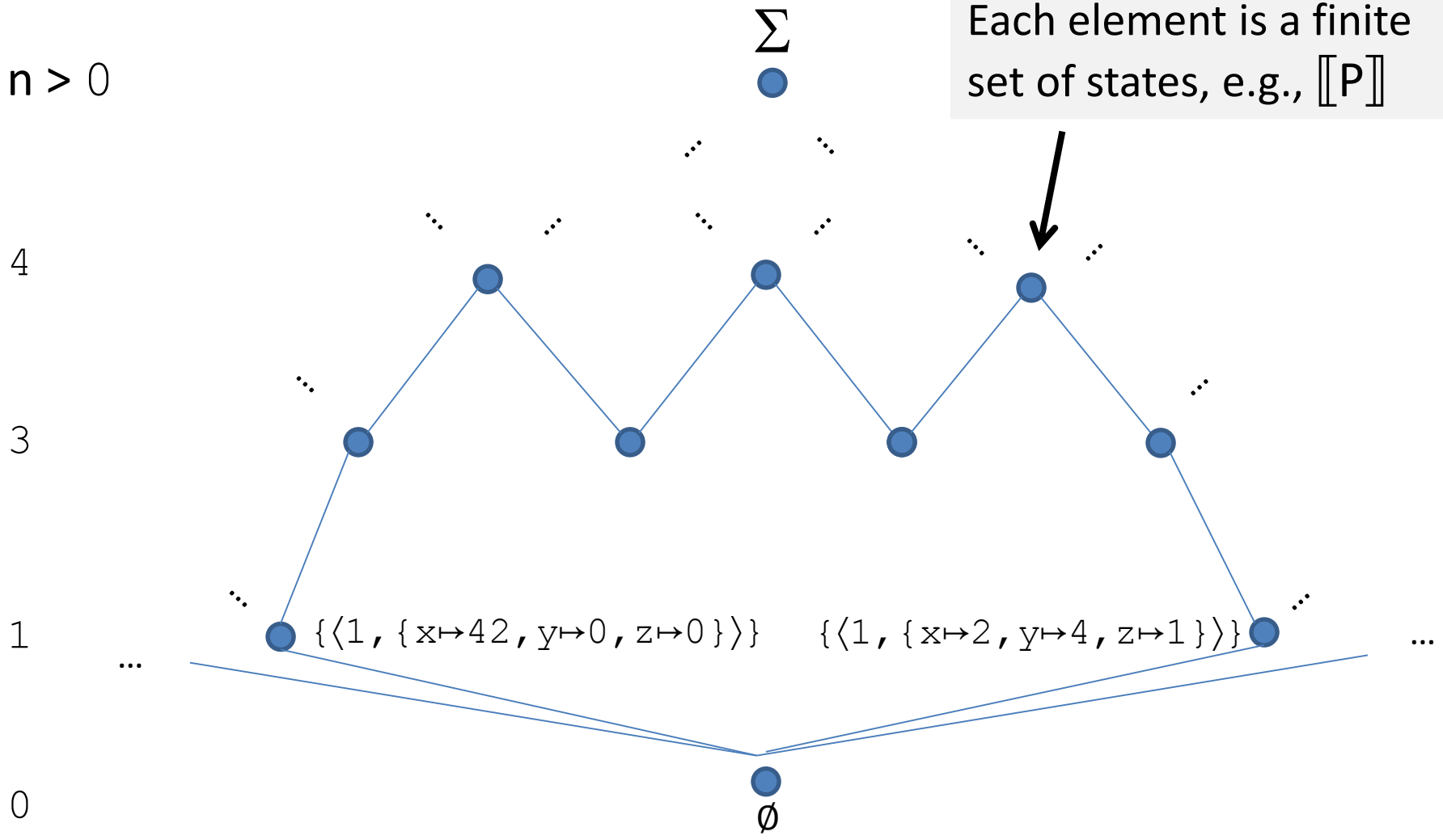
Our starting point is a domain where each element of the domain is a set of states. The domain of states is a complete lattice:

$$(\wp(\Sigma), \subseteq, \cup, \cap, \emptyset, \Sigma)$$

$$\Sigma = \text{Lab} \times \text{Store}$$

Starting Point: Domain of States

Size of Set:



Recall: Representing $[\![P]\!]$

Let $[\![P]\!]$ be the set of reachable state of a program P.

Let function F be (where I is an initial set of states):

$$F(S) = I \cup \{ c' \mid c \in S \wedge c \rightarrow c' \}$$

Then, $[\![P]\!]$ is a fixed point of F: $F([\![P]\!]) = [\![P]\!]$

(in fact, $[\![P]\!]$ is the least fixed point of F)

Abstract Interpretation: step-by-step

1. select/define an abstract domain
   - selected based on the type of **properties** you want to prove

2. define abstract semantics **for the language** w.r.t. to the domain
   - prove **sound** w.r.t **concrete semantics**
   - involves defining abstract transformers
     - that is, effect of statement / expression on the abstract domain

3. iterate abstract transformers over the abstract domain
   - until we reach a **fixed point**

The **fixed point** is the **over-approximation** of the program

Abstract Interpretation: Step 1

1. select/define an abstract domain
   - selected based on the type of **properties** you want to prove

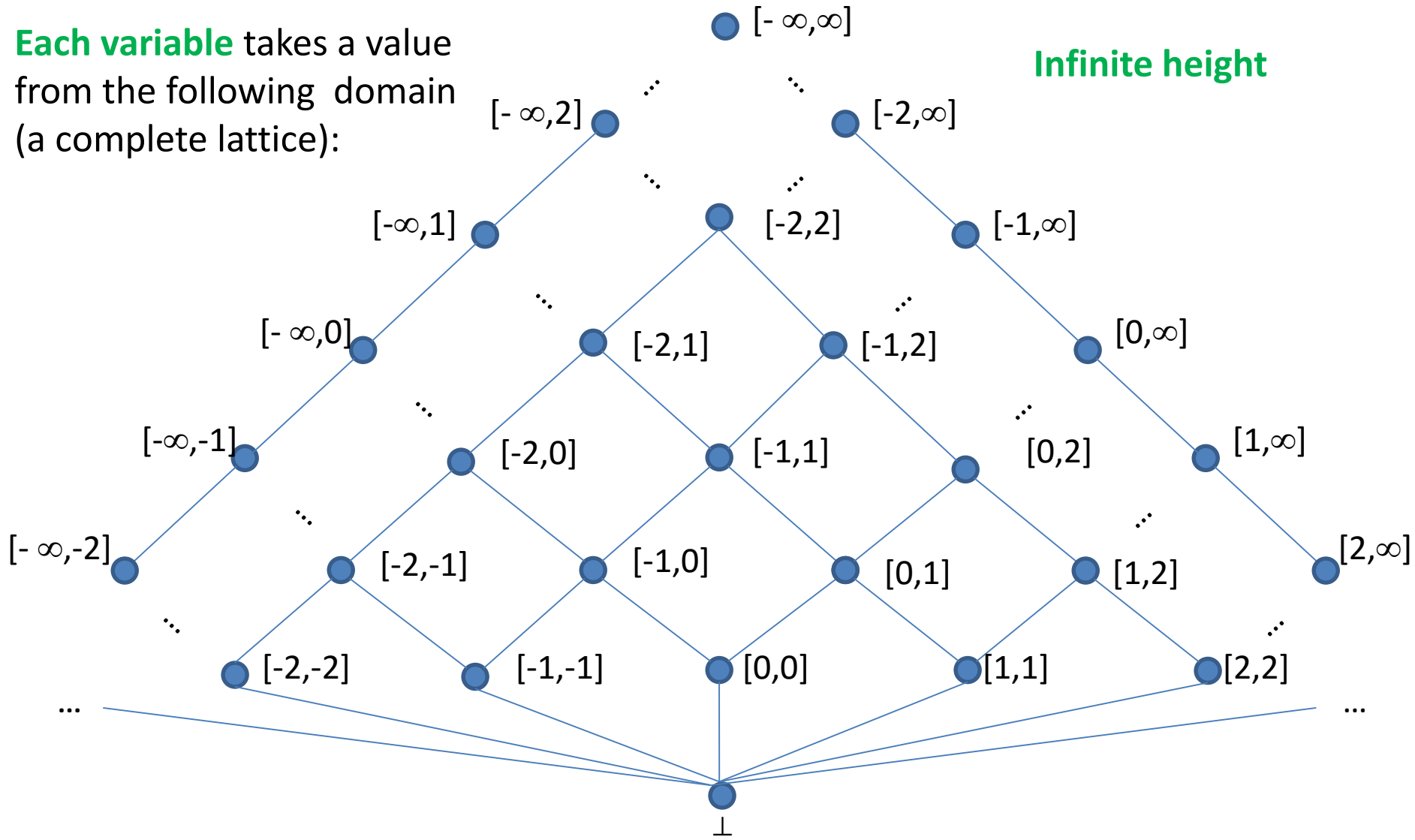
Interval Domain

If we are interested in properties that involve the range of values that a variable can take, we can abstract the set of states into a map which captures the range of values that a variable can take.

Interval Domain

Each variable takes a value from the following domain (a complete lattice):

Infinite height



Interval Domain: Lets Define it

Let the interval domain be: $(L^i, \sqsubseteq_i, \sqcup_i, \sqcap_i, \bot_i, [-\infty,\infty])$

We denote $Z^\infty = Z \cup \{-\infty,\infty\}$
The set $L^i = \{[x,y] \mid x,y \in Z^\infty,\ x \leq y\} \cup \{\bot_i\}$

For a set $S \subseteq Z^\infty$, $\min(S)$ returns the minimum number in S, $\max(S)$ returns the maximum number in S.

- $[a,b] \sqsubseteq_i [c,d]$ if $c \leq a$ and $b \leq d$
- $[a,b] \sqcup_i [c,d] = [\min(\{a,c\}),\max(\{b,d\})]$
- $[a,b] \sqcap_i [c,d] = [meet(\max(\{a,c\}),\min(\{b,d\}))]$
  where $meet(a,b)$ returns $[a,b]$ if $a \leq b$ and $\bot_i$ otherwise

Intervals: Applied to Programs

The $\mathtt{L}^{\mathtt{i}}$ domain simply defines intervals, but to apply it to programs we need to take into account program labels (program counters) and program variables.

Therefore, for programs, we use the domain $\mathtt{Lab} \rightarrow (\mathtt{Var} \rightarrow \mathtt{L}^{\mathtt{i}})$

That is, at each label and for each variable, we will keep the range for that variable. This domain is also a complete lattice.

The operators of $\mathtt{L}^{\mathtt{i}}$ $\sqsubseteq_{\mathtt{i}}, \sqcup_{\mathtt{i}}, \sqcap_{\mathtt{i}}$ are lifted directly to both domains:

- $\mathtt{Var} \rightarrow \mathtt{L}^{\mathtt{i}}$
- $\mathtt{Lab} \rightarrow (\mathtt{Var} \rightarrow \mathtt{L}^{\mathtt{i}})$

Intervals: Applied to Programs

$$\alpha^{\mathtt{i}}: \quad \wp(\Sigma) \rightarrow (\mathtt{Lab} \rightarrow (\mathtt{Var} \rightarrow \mathtt{L}^{\mathtt{i}}))$$
$$\gamma^{\mathtt{i}}: \quad (\mathtt{Lab} \rightarrow (\mathtt{Var} \rightarrow \mathtt{L}^{\mathtt{i}})) \rightarrow \wp(\Sigma)$$

Using $\alpha^{\mathtt{i}}$, we abstract a set of states into a map from program labels to interval ranges for each variable.

Using $\gamma^{\mathtt{i}}$, we concretize the intervals maps to a set of states

Formal definition of $\alpha^{\mathtt{i}}$ and $\gamma^{\mathtt{i}}$ is left as an exercise.

Let us consider an example to give an intuition.

Example of Abstraction and Concretization

α^i (
{ ⟨1,{x↦1,y↦9,q↦-2}⟩, ⟨1,{x↦5,y↦9,q↦-2}⟩, ⟨1,{x↦8,y↦9,q↦-2}⟩,
⟨1,{x↦1,y↦9,q↦4}⟩, ⟨1,{x↦5,y↦9,q↦4}⟩, ⟨1,{x↦8,y↦9,q↦4}⟩}
)
= 1 → (x ↦ [1,8], y ↦ [9,9], q ↦ [-2,4])

γ^i (1 → (x ↦ [1,8], y ↦ [9,9], q ↦ [-2,4])

= { ⟨1,{x↦1,y↦9,q↦-2}⟩, ⟨1,{x↦5,y↦9,q↦-2}⟩, ⟨1,{x↦8,y↦9,q↦-2}⟩,
⟨1,{x↦1,y↦9,q↦4}⟩, ⟨1,{x↦5,y↦9,q↦4}⟩, ⟨1,{x↦8,y↦9,q↦4}⟩,
⟨1,{x↦7,y↦9,q↦3}⟩, ⟨1,{x↦3,y↦9,q↦4}⟩, ⟨1,{x↦1,y↦9,q↦-1}⟩,
…, …, … }

Concretization includes many more states (in red) than what was abstracted…

Abstract Interpretation: Step 2

1. select/define an abstract domain
   - selected based on the type of **properties** you want to prove

2. define abstract semantics **for the language** w.r.t. to the domain
   - prove **sound** w.r.t **concrete semantics**
   - involves defining abstract transformers
     - that is, effect of statement / expression on the abstract domain

we still need to actually compute α^i ($[\![P]\!]$)
(or an over-approximation of it)

We need to approximate F

We want a function F^i where:

$$F^i : (\mathtt{Lab} \rightarrow (\mathtt{Var} \rightarrow \mathtt{L}^i)) \rightarrow (\mathtt{Lab} \rightarrow (\mathtt{Var} \rightarrow \mathtt{L}^i))$$

such that: $\alpha^i (\mathrm{lfp}\, F) \sqsubseteq \mathrm{lfp}\ F^i$

The best transformer is: $\alpha^i \circ F \circ \gamma^i$

Lets define F^i

$$F^i : (\texttt{Lab} \rightarrow (\texttt{Var} \rightarrow \texttt{L}^i)) \rightarrow (\texttt{Lab} \rightarrow (\texttt{Var} \rightarrow \texttt{L}^i))$$

Here is a definition of F^i which approximates the best transformer but works only on the abstract domain:

$$F^i(m)\ell = \begin{cases} \lambda\, v.\, [-\infty,\infty] & \text{if } \ell \text{ is initial label} \\ \bigsqcup_{(\ell',\text{action},\ell)} [\![\text{action}]\!]_i(m(\ell')) & \text{otherwise} \end{cases}$$

$$[\![\text{action}]\!]_i : (\texttt{Var} \rightarrow \texttt{L}^i) \rightarrow (\texttt{Var} \rightarrow \texttt{L}^i)$$

what is $(\ell', \text{action}, \ell)$?

```
foo (int i) {

1: int x := 5;
2: int y := 7;

3: if (0 ≤ i) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:}
```

Actions:

```
(1, x := 5, 2)
(2, y := 7, 3)
(3, 0 ≤ i, 4)
(3, 0 > i, 7)
(4, y = y + 1, 5)
(5, i := i − 1, 6)
(6, goto 3, 3)
```

Multiple (two) actions reach label 3

what is $(\ell', \text{action}, \ell)$?

- $(\ell', \text{action}, \ell)$ is an edge in the control-flow graph

- More formally, if there exists a transition $t = \langle \ell', \sigma' \rangle \rightarrow \langle \ell, \sigma \rangle$ in a program trace in P, where t was performed by statement called action, then $(\ell', \text{action}, \ell)$ must exist. This says that we are sound: we never miss a flow.

- However, $(\ell', \text{action}, \ell)$ may exist even if no such transition t above occurs. In this case, the analysis would be imprecise as we would unnecessarily create more flows.

what is $(\ell', \text{action}, \ell)$?

An action can be:

- b $\in$ BExp — boolean expression in a conditional
- x:= a — here, a $\in$ AExp
- skip

Next, we will define the effect of some of these things formally, while with others we will proceed by example.

The key point is to make sure that $[\![\text{action}]\!]_i$ produces sound and precise results.

$[\![x := a]\!]_i$

$$[\![x := a]\!]_i\,(m) = m\,[x \mapsto v] \quad , \quad \text{where } \langle a, m\rangle \Downarrow_i v$$

$\langle a, m\rangle \Downarrow_i v$ says that given a map m, the expression a evaluates to a value $v \in \mathbb{L}^i$

The operational semantics rules for expression evaluation are as before, except:

- any constant Z is abstracted to an element in $\mathbb{L}^i$
- operators $+$, $-$ and $*$ are re-defined for the Interval domain

Arithmetic Expressions

If we add $\perp_i$ to any other element, we get $\perp_i$.

If both operands are not $\perp_i$, we get:

$$[x,y] + [z,q] = [x + z,\ y + q]$$

what about $*$?

is $[x,y] * [z,q] = [x * z,\ y * q]$ sound ?

$[\![b]\!]_i$

Let us first look at the expression: $a_1 \; c \; a_2$

For a map m, we need to define : $[\![a_1 \; c \; a_2]\!]_i(m)$

which produces another map as a result.

Here, c is a condition such as: $\leq, \; =, \; <$

Recall our example: what is $[\![x \leq y]\!]$?

Suppose we have the program:

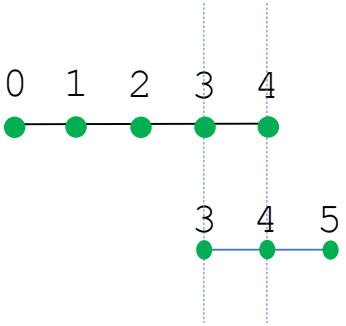
```
// Here, x is [0,4] and y is [3,5]
if (x ≤ y){
   1: …
}
```

What does $[\![x \leq y]\!]$ produce at label 1 ?

That is, what are the intervals for x and y at label 1 ?

Definition of $[l_1, u_1] \leq [l_2, u_2]$

what should $[0,4] \leq [3,5]$ produce ?



one answer is: $([0,3],[3,5])$. Is it sound ?

another non-comparable answer is: $([0,4],[4,5])$. Is it sound ?

can you find a more precise answer ?

Definition of $[l_1,u_1] \leq [l_2,u_2]$

$$[l_1,u_1] \leq [l_2,u_2] \quad = ([l_1,u_1] \sqcap_i [-\infty,u_2],\ [l_1,\infty] \sqcap_i [l_2,u_2])$$

$$[0,4] \leq [3,5] = ([0,4] \sqcap_i [-\infty,5],[0,\ \infty] \sqcap_i [3,\ 5]\ )$$
$$= (\ [0,4],\ [3,5]\ )$$

Exercise: define < and =

Evaluating $[\![b]\!]_i$

$$[\![b_1 \vee b_2]\!]_i\ (m) = [\![b_1]\!]_i\ (m) \sqcup [\![b_2]\!]_i\ (m)$$

$$[\![b_1 \wedge b_2]\!]_i\ (m) = [\![b_1]\!]_i\ (m) \sqcap [\![b_2]\!]_i\ (m)$$

Example

F^i on an example

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

What is F^i?

Abstract Interpretation: Step 3

1. select/define an abstract domain
   - selected based on the type of **properties** you want to prove

2. define abstract semantics **for the language** w.r.t. to the domain
   - prove **sound** w.r.t **concrete semantics**
   - involves defining abstract transformers
     - that is, effect of statement / expression on the abstract domain

3. iterate abstract transformers over the abstract domain
   - until we reach a **fixed point**

F^i on an example

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

$F^i(m)1 = \quad \lambda\, v.\ [-\infty,\infty]$

$F^i(m)2 = \quad [\![x := 5]\!]_i(m(1))$

$F^i(m)3 = \quad [\![y := 7]\!]_i(m(2)) \sqcup [\![\text{goto } 3]\!]_i(m(6))$

$F^i(m)4 = \quad [\![i \geq 0]\!]_i(m(3))$

$F^i(m)5 = \quad [\![y := y + 1]\!]_i(m(4))$

$F^i(m)6 = \quad [\![i := i - 1]\!]_i(m(5))$

$F^i(m)7 = \quad [\![i < 0]\!]_i(m(3))$

Fixed point of F^i

Let us compute the least fixed point of F^i

Iterate 0

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

The collection of these lines denote the current iterate. The iterate is a map

1: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

2: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

3: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

4: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

5: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

6: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

7: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

Iterate 1

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

3: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

4: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

Iterate 2

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

4: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

Iterate 3

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

Iterate 4

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

Notice how we propagated to both labels 4 and 7 at the same time

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 5

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [0,\infty]$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 6

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 7

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,8],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,7],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,8],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,8],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,7],\ i \to [-\infty,-1]$

Iterate 8

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,8],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,8],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,8],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,8],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,8],\ i \to [-\infty,-1]$

Iterate 9

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,8],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,8],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,9],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,8],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,8],\ i \to [-\infty,-1]$

Iterate 10

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,8],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,8],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,9],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,9],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,8],\ i \to [-\infty,-1]$

Iterate 11

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,9],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,8],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,9],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,9],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,8],\ i \rightarrow [-\infty,-1]$

Iterate 12

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,9],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,9],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,9],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,9],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,9],\ i \rightarrow [-\infty,-1]$

Iterate 13

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,9],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,9],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,10],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,9],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,9],\ i \to [-\infty,-1]$

Iterate 14

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,9],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,9],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,10],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,10],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,9],\ i \to [-\infty,-1]$

Iterate 15

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,10],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,9],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,10],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,10],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,9],\ i \to [-\infty,-1]$

Iterate 16

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \to [-\infty,\infty],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

2: $x \to [5,5],\ y \to [-\infty,\infty],\ i \to [-\infty,\infty]$

3: $x \to [5,5],\ y \to [7,10],\ i \to [-\infty,\infty]$

4: $x \to [5,5],\ y \to [7,10],\ i \to [0,\infty]$

5: $x \to [5,5],\ y \to [8,10],\ i \to [0,\infty]$

6: $x \to [5,5],\ y \to [8,10],\ i \to [-1,\infty]$

7: $x \to [5,5],\ y \to [7,10],\ i \to [-\infty,-1]$

The issue is that the iterates:

$$F^{i\,(0)}\,(\lambda\,\ell\,.\,\lambda\,v.\,\bot_i\,),\ F^{i\,(1)}\,(\lambda\,\ell\,.\,\lambda\,v.\,\bot_i\,),\ F^{i\,(2)}\,(\lambda\,\ell\,.\,\lambda\,v.\,\bot_i\,),\ \ldots$$

will keep going on forever as the value of variable y will keep increasing. Hence, we will not be able to compute all of the iterates that we need in order to apply the fixed point theorem.

what should we do in this case ?

Generally, if we have a complete lattice $(L, \sqsubseteq, \sqcup, \sqcap)$ and a monotone function F, if the height is infinite or the computation of the iterates of F takes too long, we need to find a way to approximate the least fixed point of F.

The interval domain and its function F^i is an example of this case.

We need to find a way to compute an element A such that:

$$\mathrm{lfp}^{\sqsubseteq} F \sqsubseteq A$$

Widening Operator

The operator $\nabla: L \times L \rightarrow L$ is called a widening operator if:

- $\forall\, a,b \in L: a \sqcup b \sqsubseteq a \nabla b$ (widening approximates the join)
- if $x^0 \sqsubseteq x^1 \sqsubseteq x^2 \sqsubseteq \ldots \sqsubseteq x^n \sqsubseteq \ldots$ is an increasing sequence then $y^0 \sqsubseteq y^1 \sqsubseteq y^2 \sqsubseteq \ldots \sqsubseteq y^n$ stabilizes after a finite number of steps

  where $y^0 = x^0$ and $\forall i \geq 0: y^{i+1} = y^i \nabla x^{i+1}$

Note that widening is completely **independent of the function F.**

Much like the join, it is an operator defined for the particular domain.

Useful Theorem

If L is a complete lattice, $\nabla: L \times L \rightarrow L$, $F: L \rightarrow L$ is monotone
Then the sequence:

$$y^0 = \bot$$
$$y^1 = y^0 \nabla F(y^0)$$
$$y^2 = y^1 \nabla F(y^1)$$
$$\ldots$$
$$y^n = y^{n-1} \nabla F(y^{n-1})$$

will stabilizes after a finite number of steps with y^n being a **post-fixedpoint** of F.

By Tarski's theorem, we know that a post-fixedpoint is above the least fixed point. Hence, it follows that: $\text{lfp}^{\sqsubseteq} F \sqsubseteq y^n$

Widening for Interval Domain

Let us define a widening operator for the intervals

$$\nabla_i : \mathbb{L}^i \times \mathbb{L}^i \longrightarrow \mathbb{L}^i$$

$[a, b] \ \nabla_i \ [c, d] = [e, f]$ where:

if $c < a$, then $e = -\infty$, else $e = a$

if $d > b$, then $f = \infty$, else $f = b$

if one of the operands is $\perp$ the result is the other operand.
The basic intuition is that if we see that an end point is unstable, we move its value to the extreme case.

Exercise: show this operator satisfies the conditions for widening.

Examples: widening for Interval

$[1, 2] \nabla_i [0, 2] =$

$[0, 2] \nabla_i [1, 2] =$

$[1, 5] \nabla_i [1, 5] =$

$[2, 3] \nabla_i [2, 4] =$

Examples: widening for Interval

$$[1, 2] \nabla_i [0, 2] = [-\infty, 2]$$

$$[0, 2] \nabla_i [1, 2] = [0, 2]$$

$$[1, 5] \nabla_i [1, 5] = [1, 5]$$

$$[2, 3] \nabla_i [2, 4] = [2, \infty]$$

Let us again consider our program with the Interval domain but this time we will apply the widening operator

Iterate 0

$it^0 = \bot$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

2: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

3: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

4: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

Iterate 1

$$it^1 = it^0 \ \nabla\ F(it^0)$$
$$= \bot \ \nabla\ F(\bot)$$
$$= F(\bot)$$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

3: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

4: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

Iterate 2

$it^2 = it^1 \triangledown F(it^1)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

4: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

5: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

6: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

7: $x \rightarrow \perp_i,\ y \rightarrow \perp_i,\ i \rightarrow \perp_i$

Iterate 3

$it^3 = it^2 \triangledown F(it^2)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

Iterate 4

Notice how we propagated to both labels 4 and 7 at the same time

$it^4 = it^3 \triangledown F(it^3)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 5

$it^5 = it^4 \triangledown F(it^4)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [0,\infty]$

6: $x \rightarrow \bot_i,\ y \rightarrow \bot_i,\ i \rightarrow \bot_i$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 6

$it^6 = it^5 \triangledown F(it^5)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 7: first compute $F(it^6)$

$$it^7 = it^6 \triangledown F(it^6)$$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,8],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 7: then $it^6 \triangledown F(it^6)$

we have:

3: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,\infty]$

$$\triangledown$$

3: $x \rightarrow [5,5],\ y \rightarrow [7,8],\ i \rightarrow [-\infty,\infty]$

$$=$$

3: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,\infty]$

Iterate 7: final result

$it^7 = it^6 \triangledown F(it^6)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,7],\ i \rightarrow [-\infty,-1]$

Iterate 8

$it^8 = it^7 \triangledown F(it^7)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,-1]$

Iterate 9

$it^9 = it^8 \triangledown F(it^8)$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,\infty],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,8],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,-1]$

Iterate 10

$it^{10} = it^{9} \triangledown F(it^{9})$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,\infty],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,\infty],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,-1]$

Iterate 11: a post fixed point is reached

$$it^{11} = it^{10} \triangledown F(it^{10})$$

```
foo (int i) {

1: int x :=5;
2: int y :=7;

3: if (i ≥ 0) {
4:    y := y + 1;
5:    i := i - 1;
6:    goto 3;
   }
7:
}
```

1: $x \rightarrow [-\infty,\infty],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

2: $x \rightarrow [5,5],\ y \rightarrow [-\infty,\infty],\ i \rightarrow [-\infty,\infty]$

3: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,\infty]$

4: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [0,\infty]$

5: $x \rightarrow [5,5],\ y \rightarrow [8,\infty],\ i \rightarrow [0,\infty]$

6: $x \rightarrow [5,5],\ y \rightarrow [8,\infty],\ i \rightarrow [-1,\infty]$

7: $x \rightarrow [5,5],\ y \rightarrow [7,\infty],\ i \rightarrow [-\infty,-1]$

Chaotic (Asynchronous) Iteration

$x_1 := \bot;\ x_2 = \bot;\ \ldots;\ x_n = \bot;$
$W := \{1,\ldots,n\};$

while $(W \neq \{\})$ **do** {
  $\ell := \text{removeLabel}(W);$
  $\text{prev}_\ell := x_\ell;$
  $x_\ell := \text{prev}_\ell \ \nabla\ f_\ell(x_1,\ldots,x_n)\ ;$
  if $(x_\ell \neq \text{prev}_\ell)$
    $W := W \cup \text{influence}(\ell);$
}

- W is the worklist , a set of labels left to be processed
- influence(ℓ) returns the set of labels where the value at those labels is influenced by the result at ℓ
- Re-compute only when necessary, thanks to influence (ℓ)
- Asynchronous computation can be parallelized