Lecturecise 12
Abstract Interpretation

How to automatically find
program invariants and postconditions

ASTREE

2013

Basic idea of abstract interpretation

Abstract interpretation is a way to infer properties of e.g. computations. Consider the assignment: $z = x + y$.

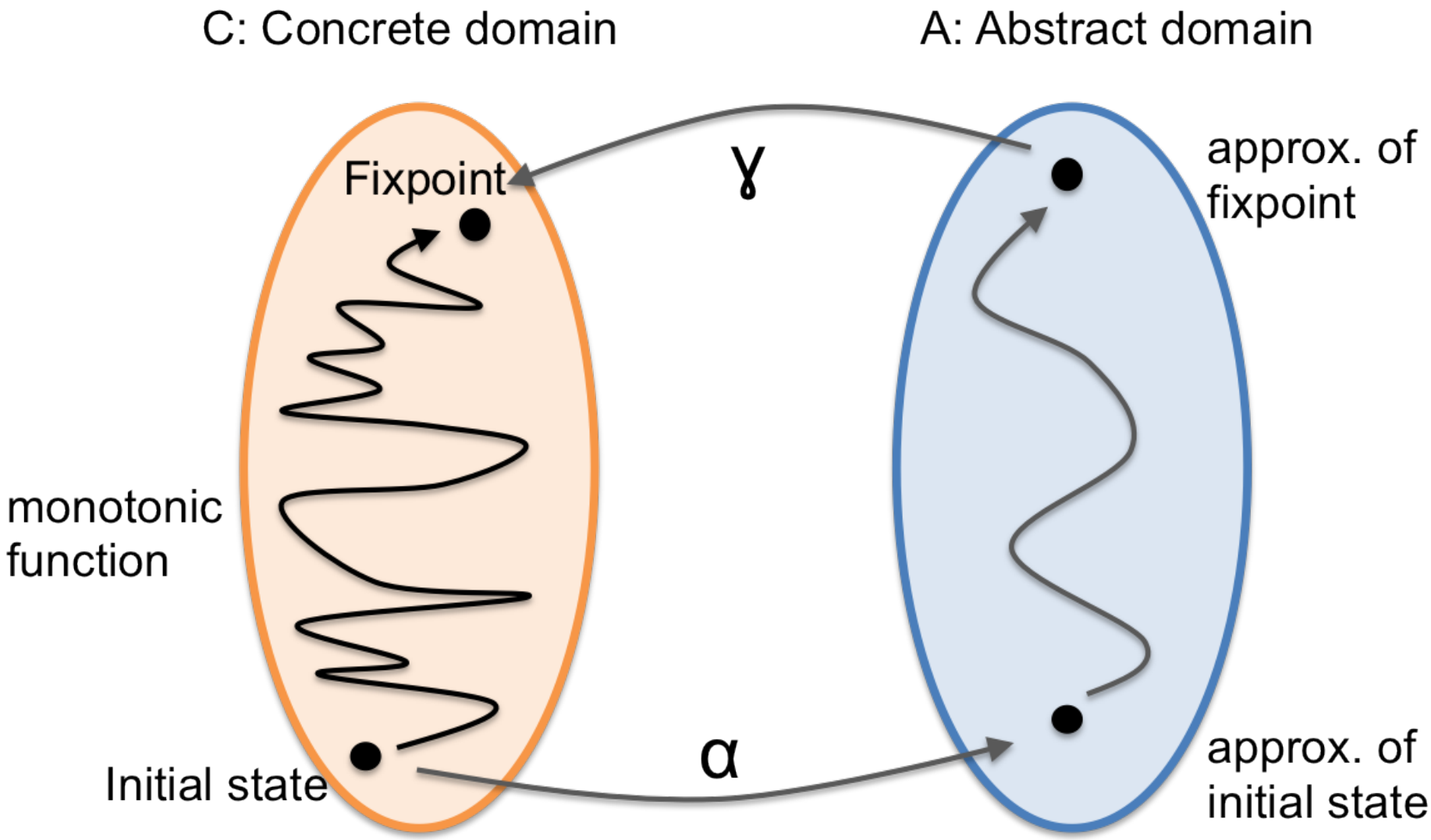
Interpreter:

$$\begin{pmatrix} x : 10 \\ y : -2 \\ z : 3 \end{pmatrix} \xrightarrow{z=x+y} \begin{pmatrix} x : 10 \\ y : -2 \\ z : 8 \end{pmatrix}$$

Abstract interpreter:

$$\begin{pmatrix} x \in & [0, 10] \\ y \in & [-5, 5] \\ z \in & [0, 10] \end{pmatrix} \xrightarrow{z=x+y} \begin{pmatrix} x \in & [0, 10] \\ y \in & [-5, 5] \\ z \in & [-5, 15] \end{pmatrix}$$

Program Meaning is a Fixpoint. We Approximate It.



Proving through Fixpoints of Approximate Functions

Meaning of a program (e.g. a relation) is a least fixpoint of F.
Given specification s, the goal is to prove $\mathbf{lfp(F)} \subseteq \mathbf{s}$
If $F(s) \subseteq s$ then $lfp(F) \subseteq s$ and we are done.

s' $F(s') \subseteq s'$
$lfp(F) \subseteq s'$

Otherwise, we need to search for s' (inductive invariant) such that:

- $F(s') \subseteq s'$ (s' is inductive). If so, theorem says $lfp(F) \subseteq s'$
- $s' \subseteq s$ (s' implies the desired specification). Then $lfp(F) \subseteq s' \subseteq s$

How to find s'? One solution is $lfp(F) = \bigcup_{k \geq 0} F^k(\emptyset)$
Infinite union, unless $F^{n+1}(\emptyset) = F^n$ for some n. This rarely happens.
Instead, we try our luck with some simpler function $F_\#$

- suppose $F_\#$ is *approximation*: $F(r) \subseteq F_\#(r)$ for all r
- we can find s' such that: $F_\#(s') \subseteq s'$ (e.g. $s' = F_\#^n(\emptyset)$ for some n)

Then: $F(s') \subseteq F_\#(s') \subseteq s'$. So, $lfp(F) \subseteq s'$.
If $s' \subseteq s$, we have shown that $lfp(F) \subseteq s$
Static analysis: automatically construct $F_\#$ from F (and sometimes s)

Programs as control-flow graphs

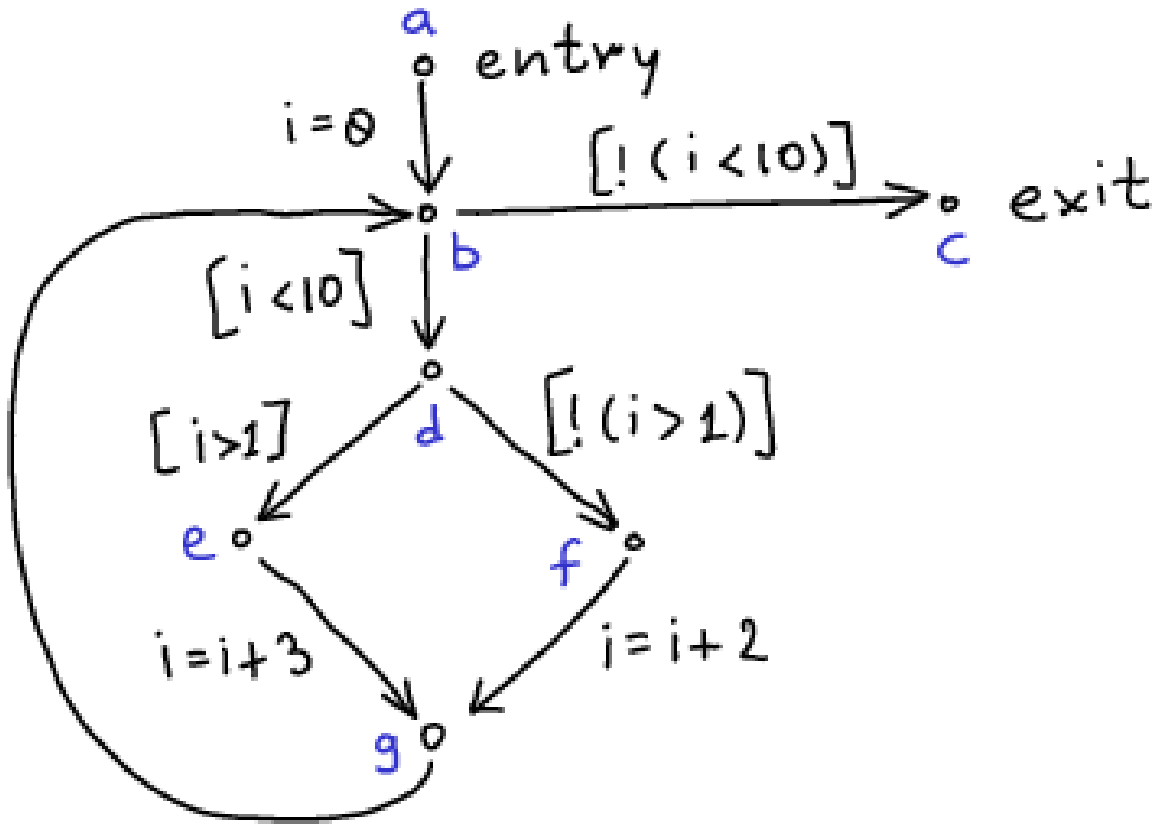
```
//a
i = 0;
      //b
while (i < 10) {
  //d
  if (i > 1)
    //e
    i = i + 3;
  else
    //f
    i = i + 2;
  //g
}
//c
```

A possible corresponding control-flow graph is:

Programs as control-flow graphs

```
//a
i = 0;
      //b
while (i < 10) {
  //d
  if (i > 1)
    //e
    i = i + 3;
  else
    //f
    i = i + 2;
  //g
}
//c
```

A possible corresponding control-flow graph is:



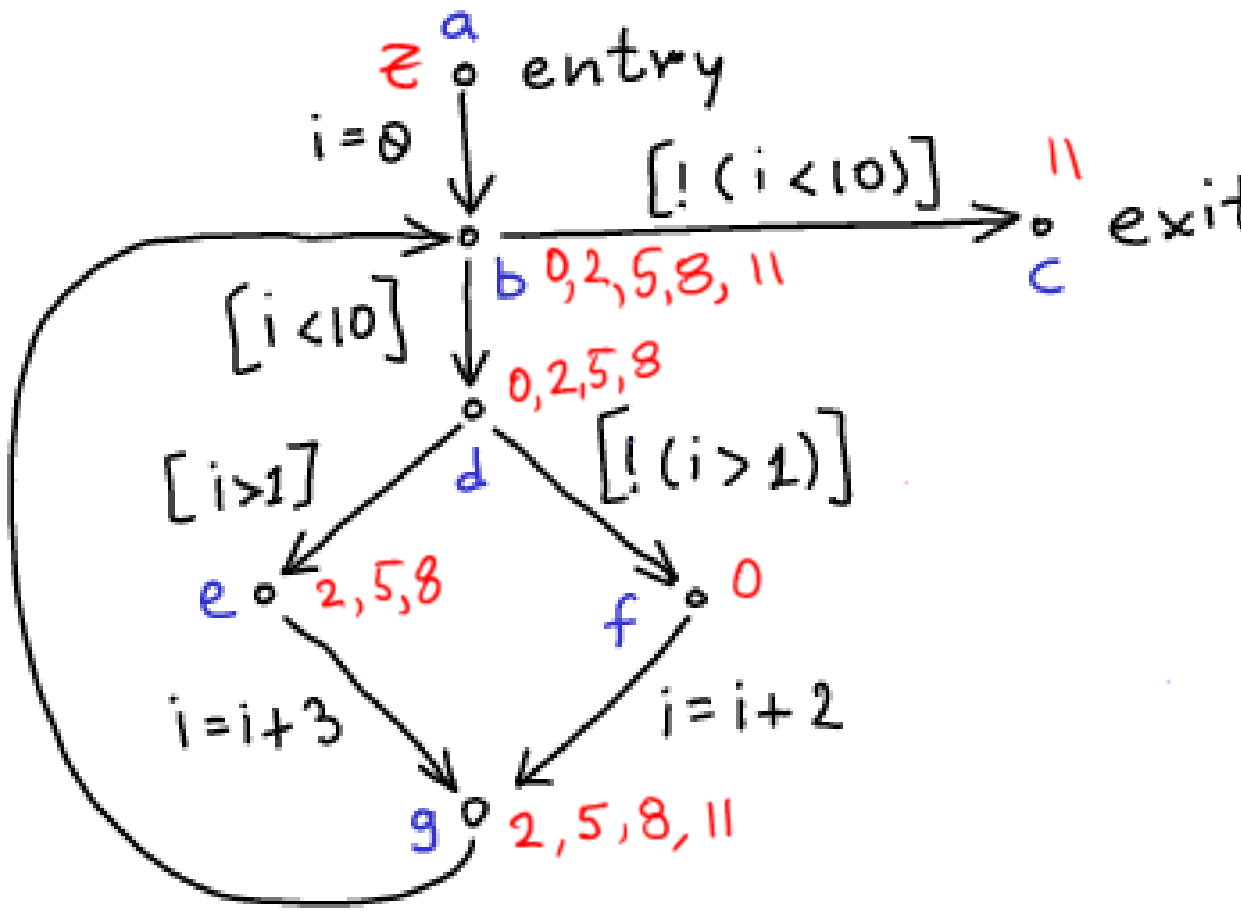
Exercise: Sets of states at each program point

Suppose that

- program state is given by the value of the integer variable *i*
- initially, it is possible that i has any value

Compute the set of states at each vertex in the CFG.

```
//a
i = 0;
      //b
while (i < 10) {
  //d
  if (i > 1)
    //e
    i = i + 3;
  else
    //f
    i = i + 2;
  //g
}
//c
```



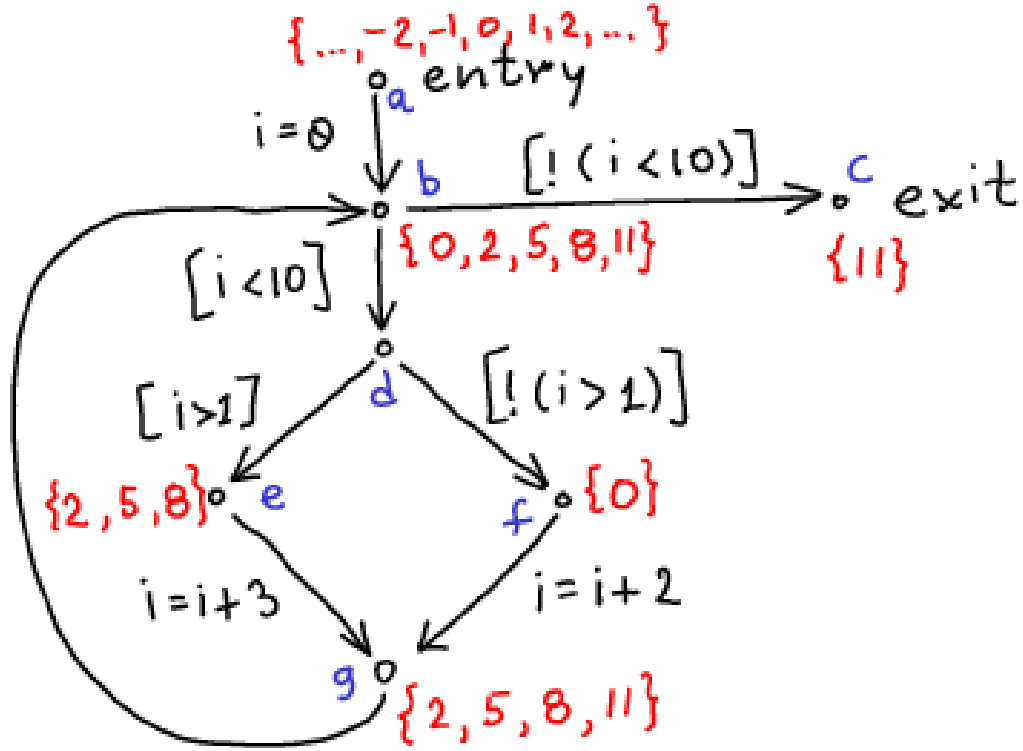
Exercise: Sets of states at each program point

Suppose that

- program state is given by the value of the integer variable *i*
- initially, it is possible that i has any value

Compute the set of states at each vertex in the CFG.

```
//a
i = 0;
      //b
while (i < 10) {
  //d
  if (i > 1)
    //e
    i = i + 3;
  else
    //f
    i = i + 2;
  //g
}
//c
```



Sets of states at each program point

Running the Program

One way to describe the set of state: for each initial state, run the CFG with this state and insert the modified states at appropriate points.

Reachable States as A Set of Recursive Equations

If c is the label on the edge of the graph, let $\rho(c)$ denotes the relation between initial and final state that describes the meaning of statement. For example,

$$\rho(i = 0) = \{(i, i') \mid i' = 0\}$$
$$\rho(i = i + 2) = \{(i, i') \mid i' = i + 2\}$$
$$\rho(i = i + 3) = \{(i, i') \mid i' = i + 3\}$$
$$\rho([i < 10]) = \{(i, i') \mid i' = i \wedge i < 10\}$$

Sets of states at each program point

We will write $T(S, c)$ (transfer function) for the image of set S under relation $\rho(c)$. For example,

$$T(\{10, 15, 20\}, i = i + 2) = \{12, 17, 22\}$$

General definition can be given using the notion of strongest postcondition

$$T(S, c) = sp(S, \rho(c))$$

If [p] is a condition (assume(p), coming from 'if' or 'while') then

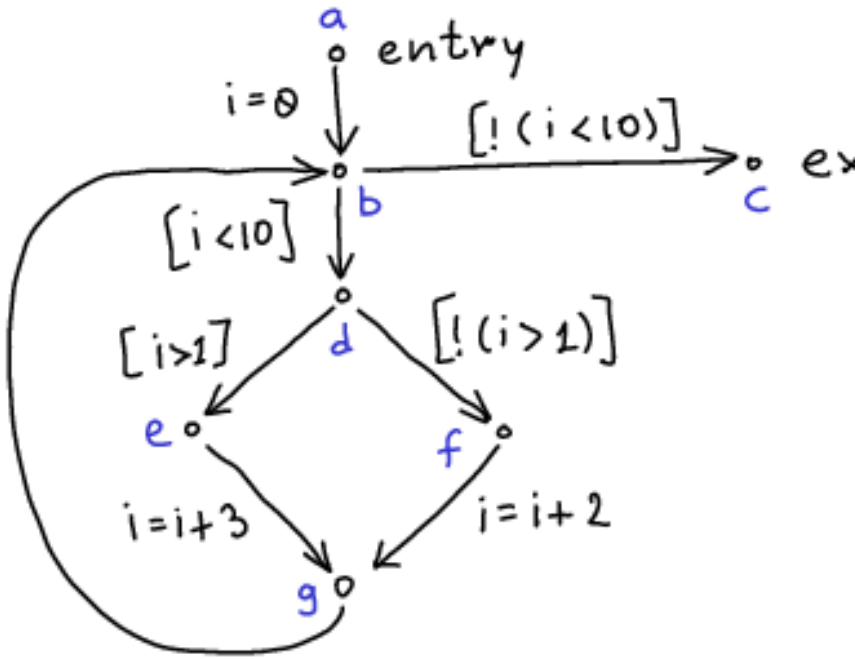
$$T(S, [p]) = \{x \in S \mid p\}$$

If an edge has no label, we denote it skip. So, $T(S, skip) = S$.

Reachable States as A Set of Recursive Equations

Now we can describe the meaning of our program using recursive equations:

$$
\begin{aligned}
S(a) &= \{\ldots, -2, -1, 0, 1, 2, \ldots\} \\
S(b) &= T(S(a), i = 0) \cup T(S(g), \mathit{skip}) \\
S(c) &= T(S(b), [\neg(i < 10)]) \\
S(d) &= T(S(b), [i < 10]) \\
S(e) &= T(S(d), [i > 1]) \\
S(f) &= T(S(d), [\neg(i > 1)]) \\
S(g) &= T(S(e), i = i + 3) \\
&\quad \cup T(S(f), i = i + 2)
\end{aligned}
$$



Note:

- solution we computed satisfies the recursive equations
- our solution is the unique **least** solution of these equations

The problem:

These exact equations are as difficult to compute as running the program on all possible input states. Instead, when we do data-flow analysis; we will consider approximate descriptions of these sets of states.

New Analysis Domain

We continue with the same example but instead of allowing to denote all possible sets, we will allow sets represented by expressions

$$[L, U]$$

which denote the set $\{x \mid L \leq x \wedge x \leq U\}$.
Example: $[0, 127]$ denotes integers between 0 and 127.

- L is the lower bound and U is the upper bound, with $L \leq U$.
- to ensure that we have only a few elements, we let

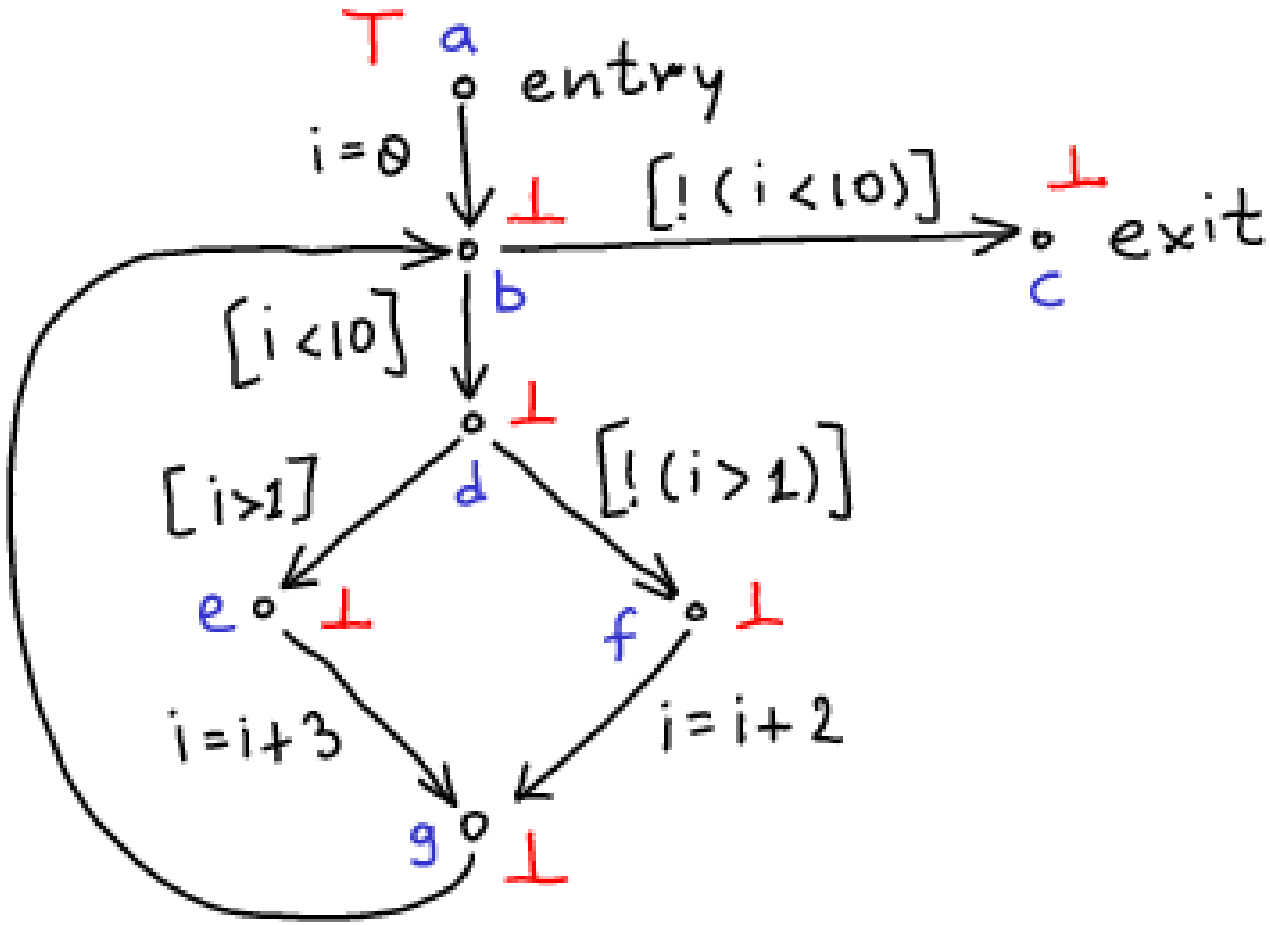
$$L, U \in \{\text{MININT}, -128, 1, 0, 1, 127, \text{MAXINT}\}$$

- $[\text{MININT}, \text{MAXINT}]$ denotes all possible integers, denote it $\top$
- instead of writing $[1, 0]$ and other empty sets, we will always write $\bot$

So, we only work with a finite number of sets $1 + \binom{7}{2} = 22$.
Denote the family of these sets by D (domain).

Initial Sets

In the 'entry' point, we put $\top$, in all others we put $\bot$.



New Set of Recursive Equations

We want to write the same set of equations as before, but because we have only a finite number of sets, we must approximate. We approximate sets with possibly larger sets.

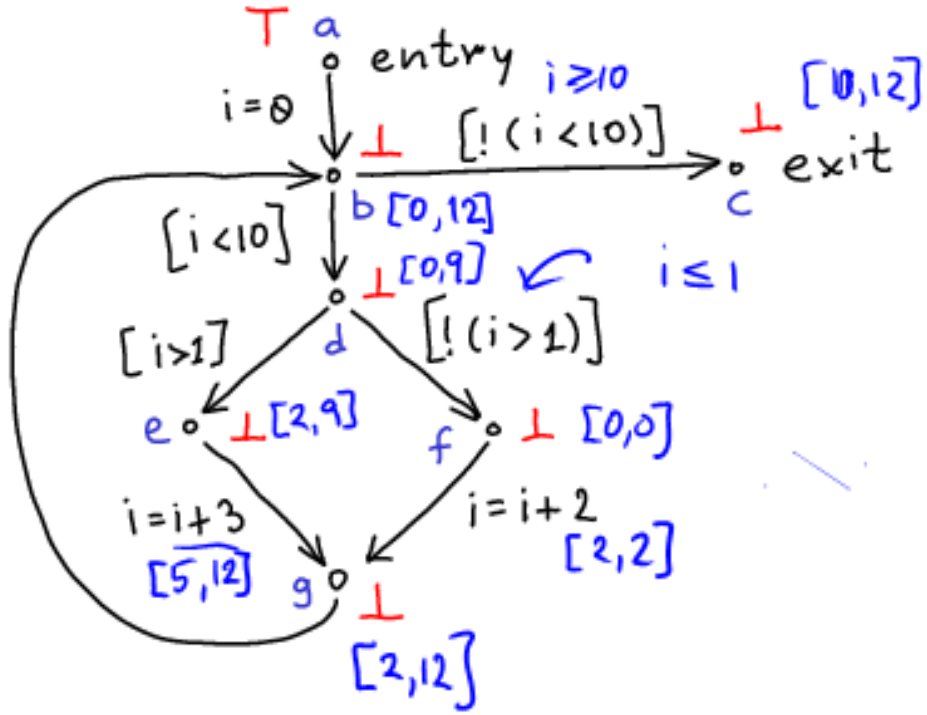
$$
\begin{aligned}
S^{\#}(a) &= \top \\
S^{\#}(b) &= T^{\#}(S^{\#}(a), i = 0) \sqcup T^{\#}(S^{\#}(g), \mathit{skip}) \\
S^{\#}(c) &= T^{\#}(S^{\#}(b), [\neg(i < 10)]) \\
S^{\#}(d) &= T^{\#}(S^{\#}(b), [i < 10]) \\
S^{\#}(e) &= T^{\#}(S^{\#}(d), [i > 1]) \\
S^{\#}(f) &= T^{\#}(S^{\#}(d), [\neg(i > 1)]) \\
S^{\#}(g) &= T^{\#}(S^{\#}(e), i = i + 3) \sqcup T^{\#}(S^{\#}(f), i = i + 2)
\end{aligned}
$$

- $S_1 \sqcup S_2$ denotes the approximation of $S_1 \cup S_2$: it is the set that contains both S_1 and S_2, that belongs to D, and is otherwise as small as possible.
- We use approximate functions $T^{\#}(S, c)$ that give a result in D.

Updating Sets

We solve the equations by starting in the initial state and repeatedly applying them.

$$
\begin{aligned}
S^{\#}(a) &= \top \\
S^{\#}(b) &= T^{\#}(S^{\#}(a), i = 0) \sqcup \\
&\quad T^{\#}(S^{\#}(g), \mathit{skip}) \\
S^{\#}(c) &= T^{\#}(S^{\#}(b), [\neg(i < 10)) \\
S^{\#}(d) &= T^{\#}(S^{\#}(b), [i < 10]) \\
S^{\#}(e) &= T^{\#}(S^{\#}(d), [i > 1]) \\
S^{\#}(f) &= T^{\#}(S^{\#}(d), [\neg(i > 1)] \\
S^{\#}(g) &= T^{\#}(S^{\#}(e), i = i + 3) \sqcup \\
&\quad T^{\#}(S^{\#}(f), i = i + 2)
\end{aligned}
$$

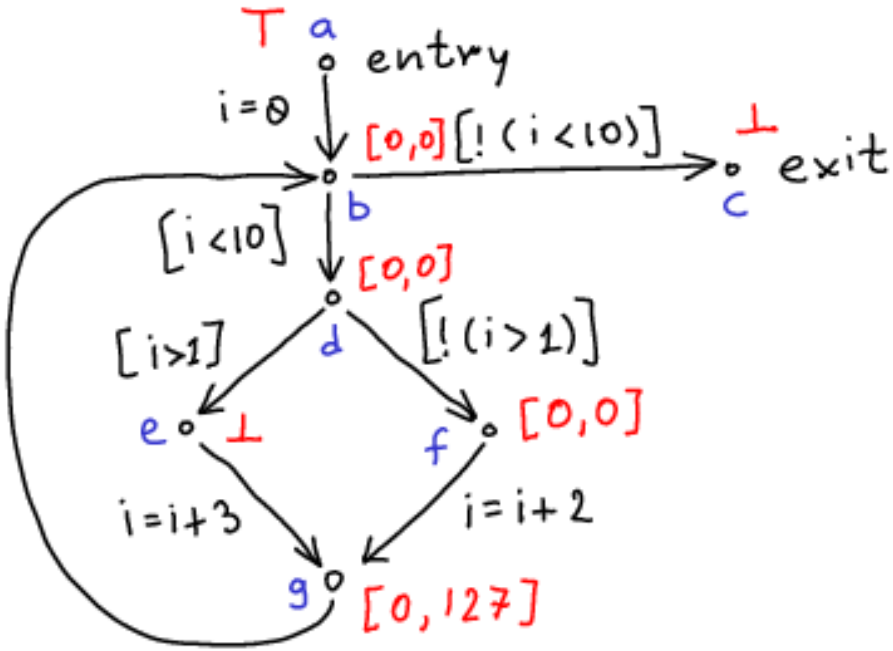


$[0,0] \sqcup [2,2] = [0,2]$

Updating Sets

Sets after first iteration:

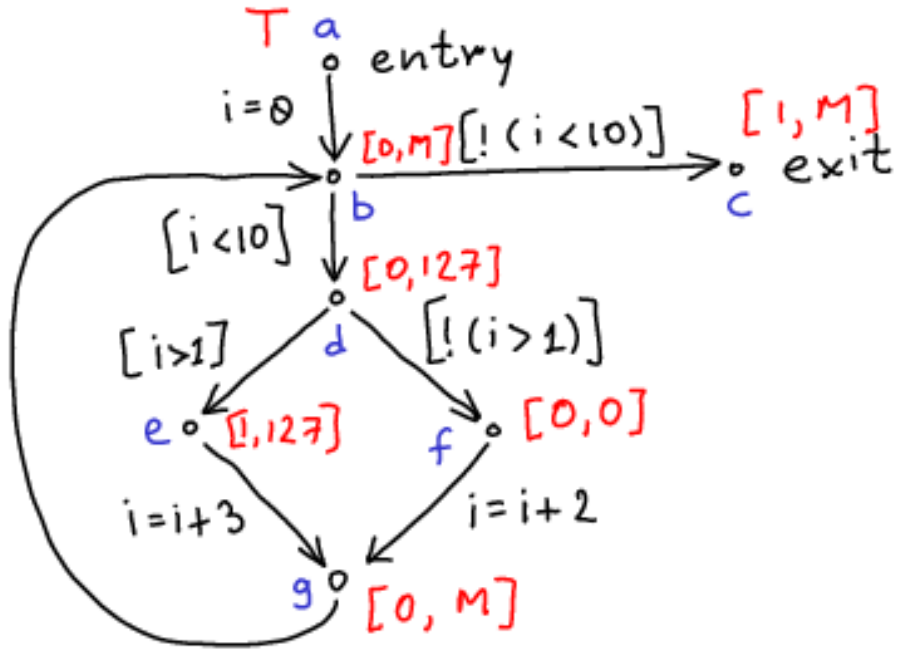
$$
\begin{aligned}
S^{\#}(a) &= \top \\
S^{\#}(b) &= T^{\#}(S^{\#}(a), i = 0) \sqcup \\
&\quad\ T(S(g), skip) \\
S^{\#}(c) &= T^{\#}(S^{\#}(b), [\neg(i < 10)) \\
S^{\#}(d) &= T^{\#}(S^{\#}(b), [i < 10]) \\
S^{\#}(e) &= T^{\#}(S^{\#}(d), [i > 1]) \\
S^{\#}(f) &= T^{\#}(S^{\#}(d), [\neg(i > 1)] \\
S^{\#}(g) &= T^{\#}(S^{\#}(e), i = i + 3) \sqcup \\
&\quad\ T(S(f), i = i + 2)
\end{aligned}
$$



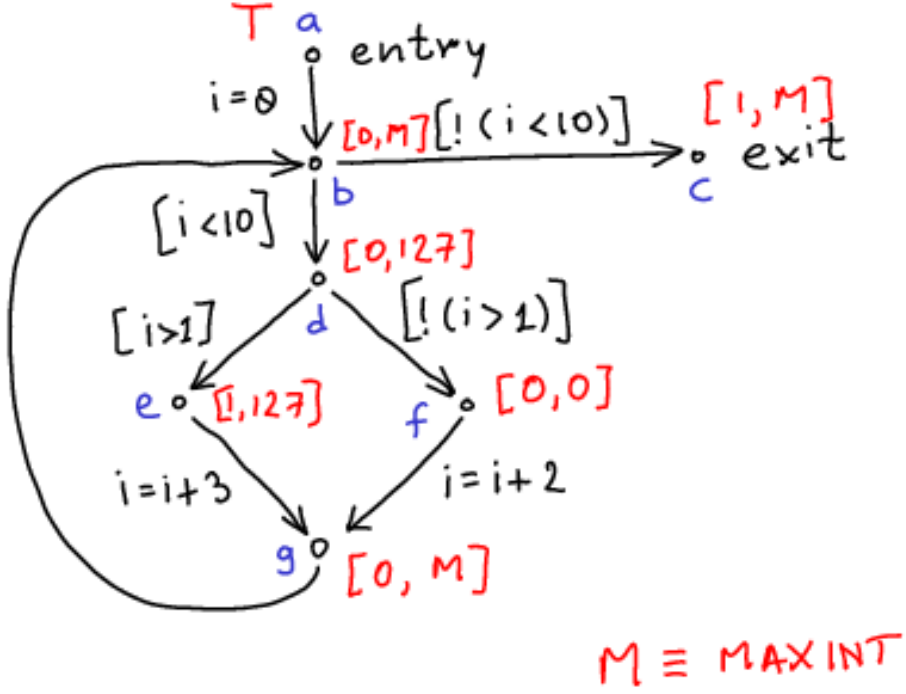
Updating Sets

Final values of sets:

$$
\begin{array}{lll}
S^{\#}(a) = & \top \\
S^{\#}(b) = & T^{\#}(S^{\#}(a), i = 0) \sqcup \\
 & T(S(g), skip) \\
S^{\#}(c) = & T^{\#}(S^{\#}(b), [\neg(i < 10)) \\
S^{\#}(d) = & T^{\#}(S^{\#}(b), [i < 10]) \\
S^{\#}(e) = & T^{\#}(S^{\#}(d), [i > 1]) \\
S^{\#}(f) = & T^{\#}(S^{\#}(d), [\neg(i > 1)] \\
S^{\#}(g) = & T^{\#}(S^{\#}(e), i = i + 3) \sqcup \\
 & T(S(f), i = i + 2)
\end{array}
$$



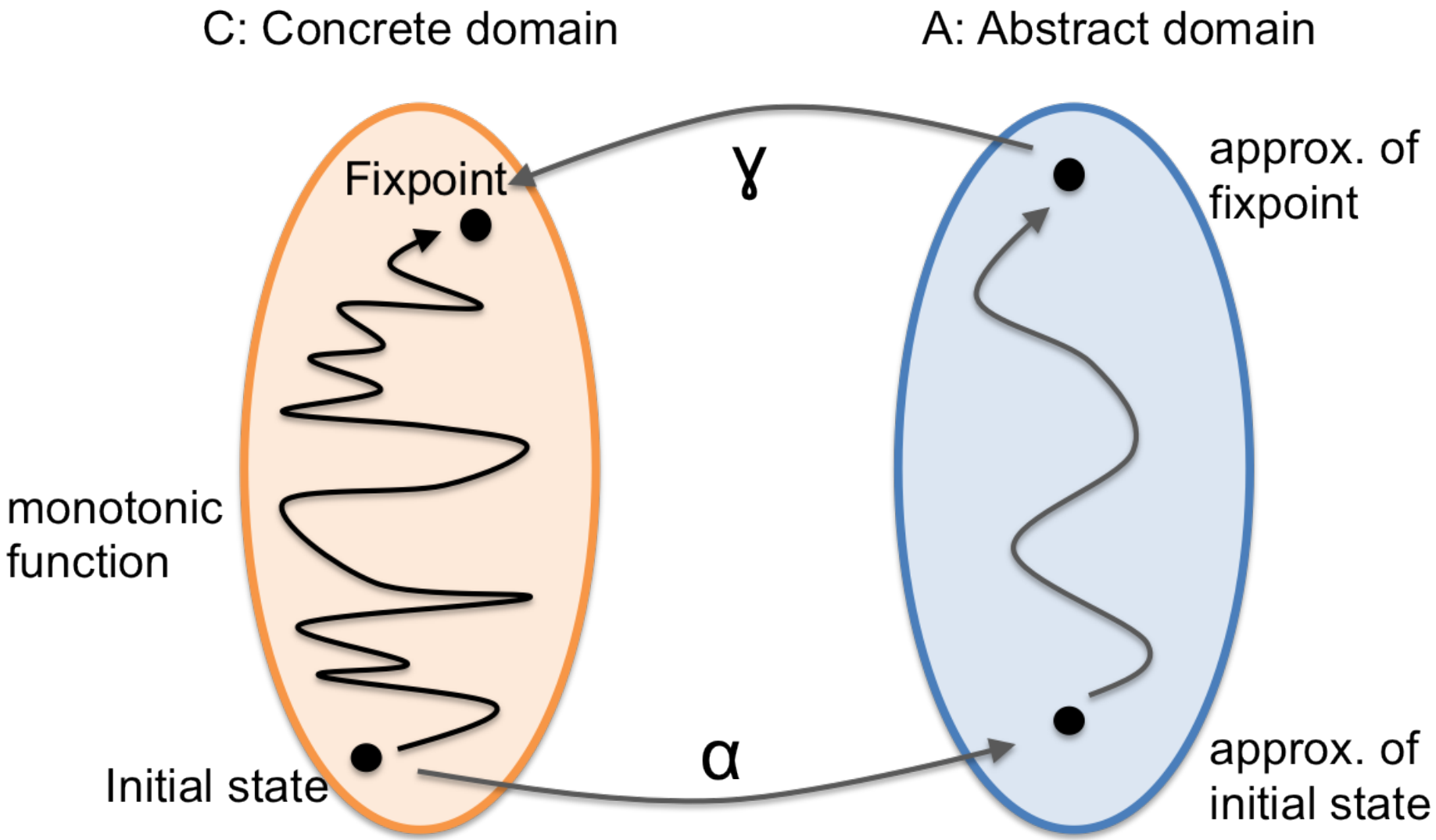
Final values of sets:



- analysis terminates (because this computation is monotonic) even if the original program does not terminate or takes very long (important for a compiler!)
- it proves that:
  - in all executions i=0 at point *f*
  - *i* cannot be zero at point *e*
  - value of *i* is always non-negative

With a larger domain *D* we can get better results, but analysis can take longer.

Abstract Interpretation Big Picture



Abstract Domains are Partial Orders

Program semantics is given by certain sets (e.g. sets of reachable states).

- subset relation $\subseteq$: used to compare sets
- union of states: used to combine sets coming from different executions (e.g. if statement)

Our goal is to approximate such sets. We introduce a domain of elements $d \in D$ where each d represents a set.

- $\gamma(d)$ is a set of states. γ is called **concretization function**
- given d_1 and d_2, it could happen that there is **no element** d representing union

$$\gamma(d_1) \cup \gamma(d_2) = \gamma(d)$$

  Instead, we use a set d that approximates union, and denote it $d_1 \sqcup d_2$

This leads us to review the theory of **partial orders** and **(semi)lattices**.

Partial Orders

Partial ordering relation is a binary relation $\leq$ that is reflexive, antisymmetric, and transitive, that is, the following properties hold for all x, y, z:

- $x \leq x$ (R)
- $x \leq y \wedge y \leq x \rightarrow x = y$ (A. S.)
- $x \leq y \wedge y \leq z \rightarrow x \leq z$ (T)

If A is a set and $\leq$ a binary relation on A, we call the pair $(A, \leq)$ a **partial order**.

Given a partial ordering relation $\leq$, the corresponding **strict ordering relation** $x < y$ is defined by $x \leq y \wedge x \neq y$ and can be viewed as a shorthand for this conjunction.

We can view the partial order (A, r) as a first-order interpretation $I = (A, \alpha)$ of language $\mathcal{L} = \{\leq\}$ where $\alpha(\leq) = r$.

Example Partial Orders

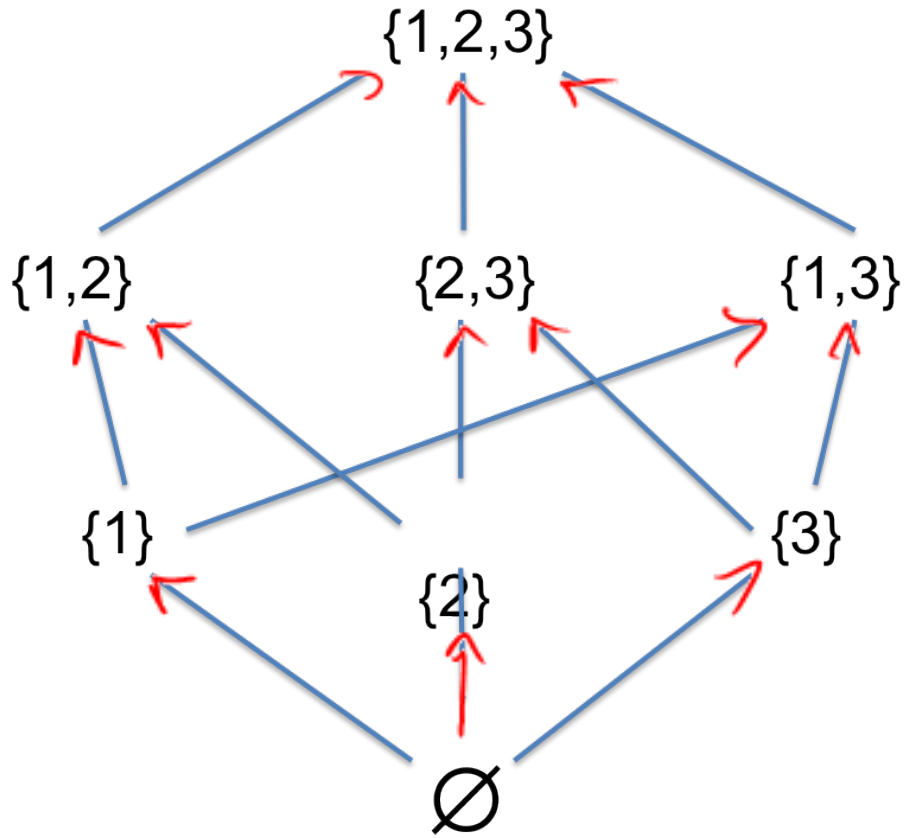
- Orders on integers, rationals, reals are all special cases of partial orders called *linear orders*.
- Given a set U, let A be any set of subsets of U, that is $A \subseteq 2^U$. Then $(A, \subseteq)$ is a partial order.

Example: Let $U = \{1, 2, 3\}$ and let $A = \{\emptyset, \{1\}, \{2\}, \{3\}, \{2, 3\}\}$. Then $(A, \subseteq)$ is a partial order. We can draw it as a *Hasse diagram*.

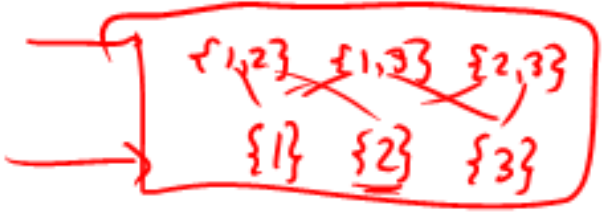
Hasse diagram

presents the relation as a directed graph in a plane, such that

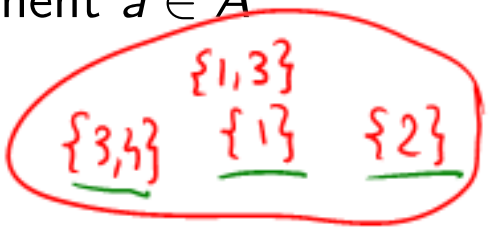
- the direction of edge is given by which nodes is drawn above
- transitive and reflexive edges are not represented (they can be derived)



Extreme Elements in Partial Orders



Given a partial order $(A, \leq)$ and a set $S \subseteq A$, we call an element $a \in A$



- **upper bound** of S if for all $a' \in S$ we have $a' \leq a$
- **lower bound** of S if for all $a' \in S$ we have $a \leq a'$
- **minimal element** of S if $a \in S$ and there is no element $a' \in S$ such that $a' < a$
- **maximal element** of S if $a \in S$ and there is no element $a' \in S$ such that $a < a'$
- **greatest element** of S if $a \in S$ and for all $a' \in S$ we have $a' \leq a$
- **least element** of S if $a \in S$ and for all $a' \in S$ we have $a \leq a'$

- **least upper bound** (lub, supremum, join, $\sqcup$) of S if a is the least element in the set of all upper bounds of S
- **greatest lower bound** (glb, infimum, meet, $\sqcap$) of S if a is the greatest element in the set of all lower bounds of S

Taking $S = A$ we obtain minimal, maximal, greatest, least elements for the entire partial order.

Extreme Elements in Partial Orders

$\bigsqcap S = a$ $\quad a \in S$ $\quad \forall x \in S.\ a \leq x$

Notes

- minimal element need not exist: $(0, 1)$ interval of rationals
- there may be multiple minimal elements: $\{\{a\}, \{b\}, \{a, b\}\}$
- if minimal element exists, it need not be least: above example
- there are no two distinct least elements for the same set
- least element is always glb and minimal
- if glb belongs to the set, then it is always least and minimal
- for relation $\subseteq$ on sets, *glb* is intersection, *lub* is union (not all families of sets are closed under $\cap$, $\cup$)

S $\quad a$ is least elem of S.

$\forall x \in S \quad a \leq x$

$\forall x' \in S. \quad b \leq x'$

$\uparrow a \qquad b \leq a$

Least upper bound

Denoted $lub(S)$, least upper bound of S is an element M, if it exists, such that M is the least element of the set

$$U = \{x \mid x \text{ is upper bound on } S\}$$

In other words:

- M is an upper bound on S
- for every other upper bound M' on S, we have that $M \leq M'$

Note: this is the same definition as infinum in real analysis.

Real Analysis

Take as S the open interval of reals $(0, 1) = \{x \mid 0 < x < 1\}$
Then

- S has no maximal element
- S thus has no greatest element
- 2, 2.5, 3,... are all upper bounds on S
- $lub(S) = 1$

~~If we had rational numbers, there would be no $lub(S')$ in general.~~

Least upper bound (shorthand: $\sqcup$)

$a_1 \sqcup a_2$ denotes $lub(\{a_1, a_2\})$

$$(\ldots(a_1 \sqcup a_2)\ldots) \sqcup a_n \quad \text{is in fact } lub(\{a_1, \ldots, a_n\})$$

So the operation is

- associative
- commutative
- idempotent

$\sqcup$, lub, sup

Execise: subsets of U

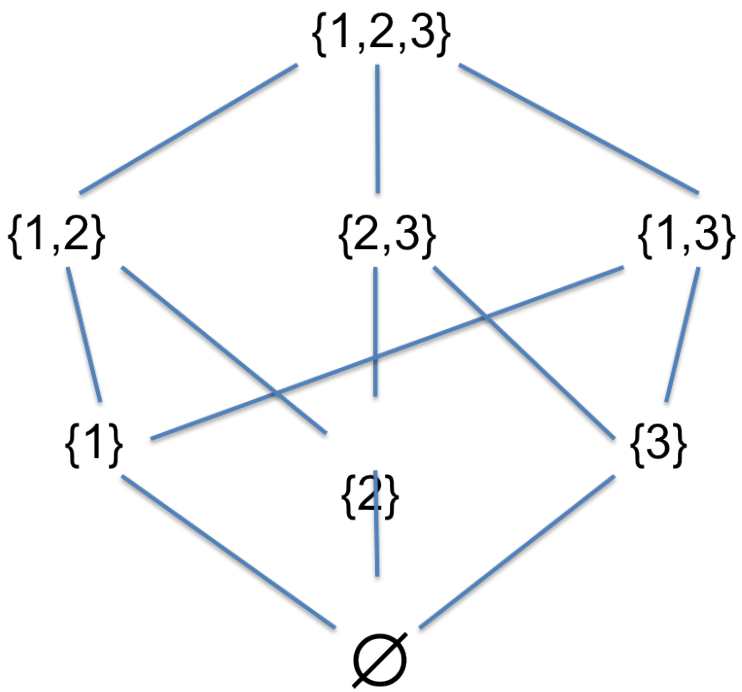
Consider

$$A = 2^U = \{S \mid S \subseteq U\} \qquad \text{and} \qquad (A, \subseteq)$$

Do these exist, and if so, what are they?

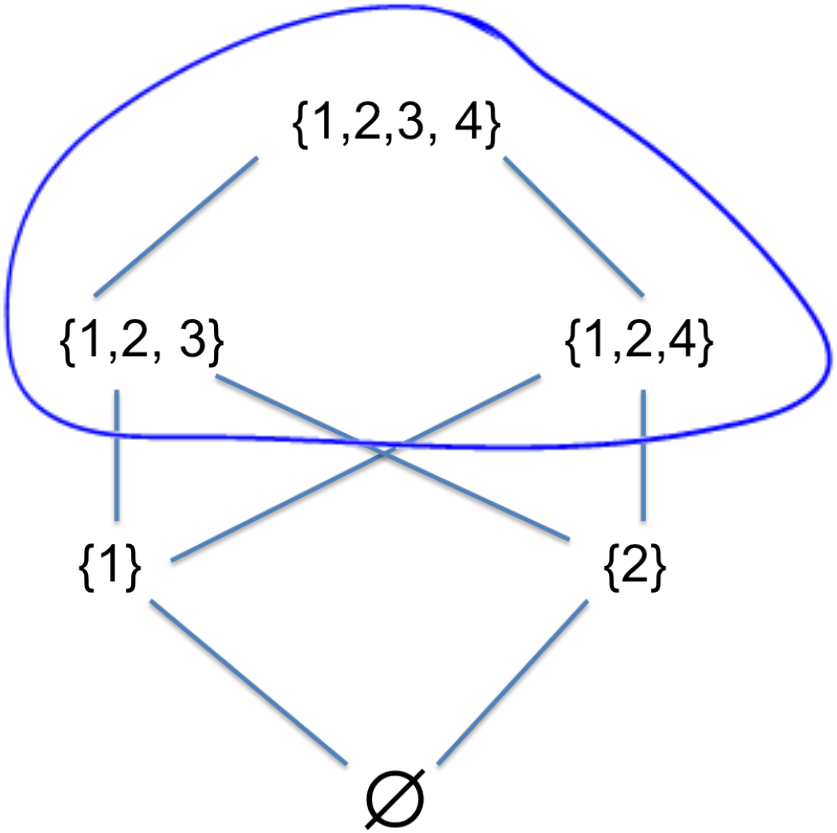
- $s_1 \subseteq S, s_2 \subseteq S, lub(\{s_1, s_2\}) = ?$
- $lub(S) = ?$

Exercise: find the lub



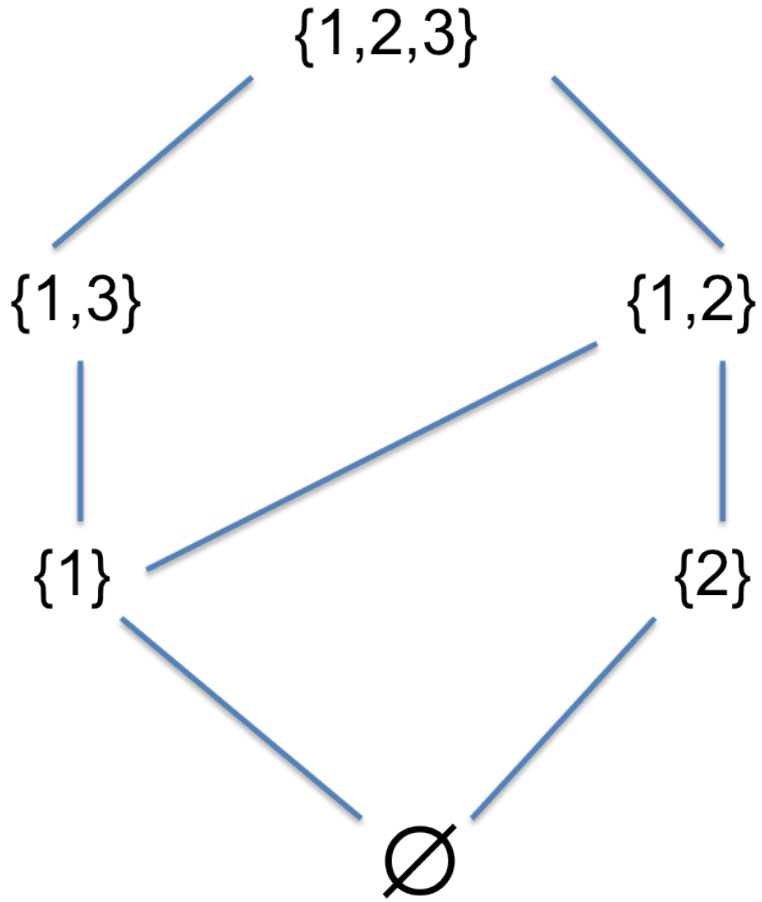
$\{1\} \sqcup \{2\} = \{1, 2\}$

$\sqcup, \sqcap$



$\{1\} \sqcup \{2\} =$

Does every pair of elements in this order have a least upper bound?



Dually, does it have a **greatest lower bound**?

Partial order for the domain of intervals

Domain: $D = \{\bot\} \cup \{(L, U) \mid L \in \{-\infty\} \cup \mathbb{Z}, U \in \{+\infty\} \cup \mathbb{Z}$ such that $L \leq U$.

The associated set of elements is given by the function γ:

$$\gamma : D \to 2^{\mathbb{Z}}, \qquad \gamma((L, U)) = \{x \mid L \leq x \wedge x \leq U\}$$

Lub: for $d_1, d_2 \in D$, $d_1 \sqsubseteq d_2 \quad \leftrightarrow \quad \gamma(d_1) \subseteq \gamma(d_2)$ hence

$$(L_1, U_1) \sqsubseteq (L_2, U_2) \quad \leftrightarrow \quad L_2 \leq L_1 \wedge U_1 \leq U_2$$

$$\bot \sqsubseteq d \quad \forall d \in D$$

$$(L_1, U_1) \sqcup (L_2, U_2) = (min(L_1, L_2), max(U_1, U_2))$$

Exercises

Prove the following:

1. $(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$
2. $\sqcup A \sqsubseteq \sqcap B \Leftrightarrow \forall x \in A. \forall y \in B. x \sqsubseteq y$
3. Let $(A, \sqsubseteq)$ be a partial order such that every set $S \subseteq A$ has the greatest lower bound.
   Prove that then every set $S \subseteq A$ has the least upper bound.

We do this in class.

Lattices

Definition: A lattice is a partial order in which every two-element set has a least upper bound and a greatest lower bound.

Lemma: In a lattice every non-empty finite set has a lub ($\sqcup$) and glb ($\sqcap$).

Lattices

Definition: A lattice is a partial order in which every two-element set has a least upper bound and a greatest lower bound.

Lemma: In a lattice every non-empty finite set has a lub ($\sqcup$) and glb ($\sqcap$).

Proof: is by induction!
Case where the set S has three elements x,y and z:
Let $a = (x \sqcup y) \sqcup z$.
By definition of $\sqcup$ we have $z \sqsubseteq a$ and $x \sqcup y \sqsubseteq a$.
Then we have again by definition of $\sqcup$, $x \sqsubseteq x \sqcup y$ and $y \sqsubseteq x \sqcup y$. Thus by transitivity we have $x \sqsubseteq a$ and $y \sqsubseteq a$.
Thus we have $S \sqsubseteq a$ and a is an upper bound.
Now suppose that there exists a' such that $S \sqsubseteq a'$. We want $a \sqsubseteq a'$ (a least upper bound):
We have $x \sqsubseteq a'$ and $y \sqsubseteq a'$, thus $x \sqcup y \sqsubseteq a'$. But $z \sqsubseteq a'$, thus $((x \sqcup y) \sqcup z) \sqsubseteq a'$.
Thus a is the lub of our 3 elements set.

Lattices

Lemma: Every linear order is a lattice.

Example: Every subset of the set of real numbers has a lub. This is an axiom of real numbers, the way they are defined (or constructed from rationals).

- If a lattice has least and greatest element, then every finite set (including empty set) has a lub and glb.
- This does not imply there are lub and glb for infinite sets.
  Example: In the oder $([0,1),\leq)$ with standard ordering on reals is a lattice, the entire set has no lub. The set of all rationals of interval $[0,10]$ is a lattice, but the set $\{x \mid 0 \leq x \wedge x^2 < 2\}$ has no lub.

Complete Lattices

Definition: A **complete** lattice is a lattice where every set S of elements has lub, denoted $\sqcup S$, and glb, denoted $\sqcap S$

(this implies that there is top and bottom as $\sqcup\emptyset = \bot$ and $\sqcap\emptyset = \top$. This is because every element is an upper bound and a lower bound of $\emptyset$: $\forall x.\forall y \in \emptyset.y \sqsubseteq x$ is valid, as well as $\forall x.\forall y \in \emptyset.y \sqsupseteq x$).

Lattices

Lemma: In every lattice, $x \sqcup (x \sqcap y) = x$.

Proof:
We trivially have $x \sqsubseteq x \sqcup (x \sqcap y)$.
Let's prove that $x \sqcup (x \sqcap y) \sqsubseteq x$:
x is an upper bound of x and $x \sqcap y$, $x \sqcup (x \sqcap y)$ is the least upper bound of x and $x \sqcap y$, thus $x \sqcup (x \sqcap y) \sqsubseteq x$.

Monotonic functions

Given two partial orders $(C, \leq)$ and $(A, \sqsubseteq)$, we call a function $\alpha : C \to A$ *monotonic* iff for all $x, y \in C$,

$$x \leq y \;\rightarrow\; \alpha(x) \sqsubseteq \alpha(y)$$

Reminder: Fixpoints

Definition: Given a set A and a function $f : A \to A$ we say that $x \in A$ is a fixed point (fixpoint) of f if $f(x) = x$.

Definition: Let $(A, \leq)$ be a partial order, let $f : A \to A$ be a monotonic function on $(A, \leq)$, and let the set of its fixpoints be $S = \{x \mid f(x) = x\}$. If the least element of S exists, it is called the **least fixpoint**, if the greatest element of S exists, it is called the **greatest fixpoint**.

Fixpoints

Let $(A, \sqsubseteq)$ be a complete lattice and $G : A \to A$ a monotonic function.

Definition:
Post $= \{x \mid G(x) \sqsubseteq x\}$ - the set of *postfix points* of G
(e.g. $\top$ is a postfix point)
Pre $= \{x \mid x \sqsubseteq G(x)\}$ - the set of *prefix points* of G
Fix $= \{x \mid G(x) = x\}$ - the set of *fixed points* of G.

Note that Fix $\subseteq$ Post.

Tarski's fixed point theorem

Theorem: Let $a = \sqcap \text{Post}$. Then a is the least element of Fix (dually, $\sqcup \text{Pre}$ is the largest element of Fix).

Proof:

Let x range over elements of Post.

- applying monotonic G from $a \sqsubseteq x$ we get $G(a) \sqsubseteq G(x) \sqsubseteq x$
- so $G(a)$ is a lower bound on Post, but a is the greatest lower bound, so $G(a) \sqsubseteq a$
- therefore $a \in \text{Post}$
- Post is closed under G, by monotonicity, so $G(a) \in \text{Post}$
- a is a lower bound on Post, so $a \sqsubseteq G(a)$
- from $a \sqsubseteq G(a)$ and $G(a) \sqsubseteq a$ we have $a = G(a)$, so $a \in \text{Fix}$
- a is a lower bound on Post so it is also a lower bound on a smaller set Fix

In fact, the set of all fixpoints Fix is a lattice itself.

Tarski's fixed point theorem

Tarski's Fixed Point theorem shows that in a complete lattice with a monotonic function G on this lattice, there is at least one fixed point of G, namely the least fixed point $\sqcap$Post.

- Tarski's theorem guarantees fixpoints in complete lattices, but the above proof does not say how to find them.
- How difficult it is to find fixpoints depends on the structure of the lattice.

Let G be a monotonic function on a lattice. Let $a_0 = \bot$ and $a_{n+1} = G(a_n)$. We obtain a sequence $\bot \sqsubseteq G(\bot) \sqsubseteq G^2(\bot) \sqsubseteq \cdots$. Let $a_* = \bigsqcup_{n \geq 0} a_n$.

Lemma: The value a_* is a prefix point.
Observation: a_* need not be a fixpoint (e.g. on lattice [0,1] of real numbers).

Omega continuity

Definition: A function G is ω-continuous if for every chain $x_0 \sqsubseteq x_1 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq \ldots$ we have

$$G(\bigsqcup_{i \geq 0} x_i) = \bigsqcup_{i \geq 0} G(x_i)$$

Lemma: For an ω-continuous function G, the value $a_* = \bigsqcup_{n \geq 0} G^n(\bot)$ is the least fixpoint of G.

Iterating sequences and omega continuity

Lemma: For an ω-continuous function G, the value $a_* = \bigsqcup_{n\geq 0} G^n(\bot)$ is the least fixpoint of G.

Proof:

- By definition of ω-continuous we have $G(\bigsqcup_{n\geq 0} G^n(\bot)) = \bigsqcup_{n\geq 0} G^{n+1}(\bot) = \bigsqcup_{n\geq 1} G^n(\bot)$.
- But $\bigsqcup_{n\geq 0} G^n(\bot) = \bigsqcup_{n\geq 1} G^n(\bot) \sqcup \bot = \bigsqcup_{n\geq 1} G^n(\bot)$ because $\bot$ is the least element of the lattice.
- Thus $G(\bigsqcup_{n\geq 0} G^n(\bot)) = \bigsqcup_{n\geq 0} G^n(\bot)$ and a_* is a fixpoint.

Now let's prove it is the least. Let c be such that $G(c) = c$. We want $\bigsqcup_{n\geq 0} G^n(\bot) \sqsubseteq c$. This is equivalent to $\forall n \in \mathbb{N}. G^n(\bot) \sqsubseteq c$.
We can prove this by induction : $\bot \sqsubseteq c$ and if $G^n(\bot) \sqsubseteq c$, then by monotonicity of G and by definition of c we have $G^{n+1}(\bot) \sqsubseteq G(c) \sqsubseteq c$.

Iterating sequences and omega continuity

Lemma: For an ω-continuous function G, the value $a_* = \bigsqcup_{n \geq 0} G^n(\bot)$ is the least fixpoint of G.

When the function is not ω-continuous, then we obtain a_* as above (we jump over a discontinuity) and then continue iterating. We then take the limit of such sequence, and the limit of limits etc., ultimately we obtain the fixpoint.

Exercise

Let $C[0,1]$ be the set of continuous functions from $[0,1]$ to the reals. Define $\leq$ on $C[0,1]$ by $f \leq g$ if and only if $f(a) \leq g(a)$ for all $a \in [0,1]$.

i) Show that $\leq$ is a partial order and that $C[0,1]$ with this order forms a lattice.

ii) Does an analogous statement hold if we consider the set of differentiable functions from $[0,1]$ to the reals? That is, instead of requiring the functions to be continuous, we require them to have a derivative on the entire interval. (The order is defined in the same way.)

Exercise

Let $A = [0,1] = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$ be the interval of real numbers. Recall that, by definition of real numbers and complete lattice, $(A, \leq)$ is a complete lattice with least lattice element 0 and greatest lattice element 1. Here $\sqcup$ is the least upper bound operator on sets of real numbers, also called *supremum* and denoted *sup* in real analysis.

Let function $f : A \to A$ be given by

$$f(x) = \begin{cases} \frac{1}{2} + \frac{1}{4}x, & \text{if } x \in [0, \frac{2}{3}) \\ \\ \frac{3}{5} + \frac{1}{5}x, & \text{if } x \in [\frac{2}{3}, 1] \end{cases}$$

(It may help you to try to draw f.)

a) Prove that f is monotonic and injective (so it is strictly monotonic).

b) Compute the set of fixpoints of f.

c) Define $iter(x) = \sqcup\{f^n(x) \mid n \in \{0, 1, 2, \ldots\}\}$. (This is in fact equal to $\lim_{n\to\infty} f^n(x)$ when f is a monotonic bounded function.)

Compute $iter(0)$ (prove that the computed value is correct by definition of *iter*, that is, that the value is indeed $\sqcup$ of the set of values). Is $iter(0)$ a fixpoint of f? Is $iter(iter(0))$ a fixpoint of f?

Galois Connection

Galois connection (named after Évariste Galois) is defined by two monotonic functions $\alpha : C \to A$ and $\gamma : A \to C$ between partial orders $\leq$ on C and $\sqsubseteq$ on A, such that

$$\forall c, a. \quad \alpha(c) \sqsubseteq a \iff c \leq \gamma(a) \qquad (*)$$

(intuitively the condition means that c is approximated by a).

Lemma: The condition $(*)$ holds iff the conjunction of these two conditions:

$$\begin{aligned} c &\leq \gamma(\alpha(c)) \\ \alpha(\gamma(a)) &\sqsubseteq a \end{aligned}$$

holds for all c and a.

Abstract Interpretation Recipe

Key steps (details to be filled in):

- design abstract domain A that represents sets of program states
- define $\gamma : A \to C$ giving meaning to elements of A
- define lattice ordering $\sqsubseteq$ on A such that $a_1 \sqsubseteq a_2 \to \gamma(a_1) \subseteq \gamma(a_2)$
- define $sp^{\#} : A \times 2^{S \times S} \to A$ that maps an abstract element and a CFG statement to new abstract element, such that $sp(\gamma(a), r) \subseteq \gamma(sp^{\#}(a, r))$
  (for example, by defining function α so that (α, γ) becomes a *Galois Connection*)
- extend $sp^{\#}$ to work on control-flow graphs, by defining $F\#$ (handling multiple incoming edges)
- compute least fixpoint of $F\#$