Program Analysis and Verification

0368-4479

http://www.cs.tau.ac.il/~maon/teaching/2013-2014/paav/paav1314b.html

Noam Rinetzky

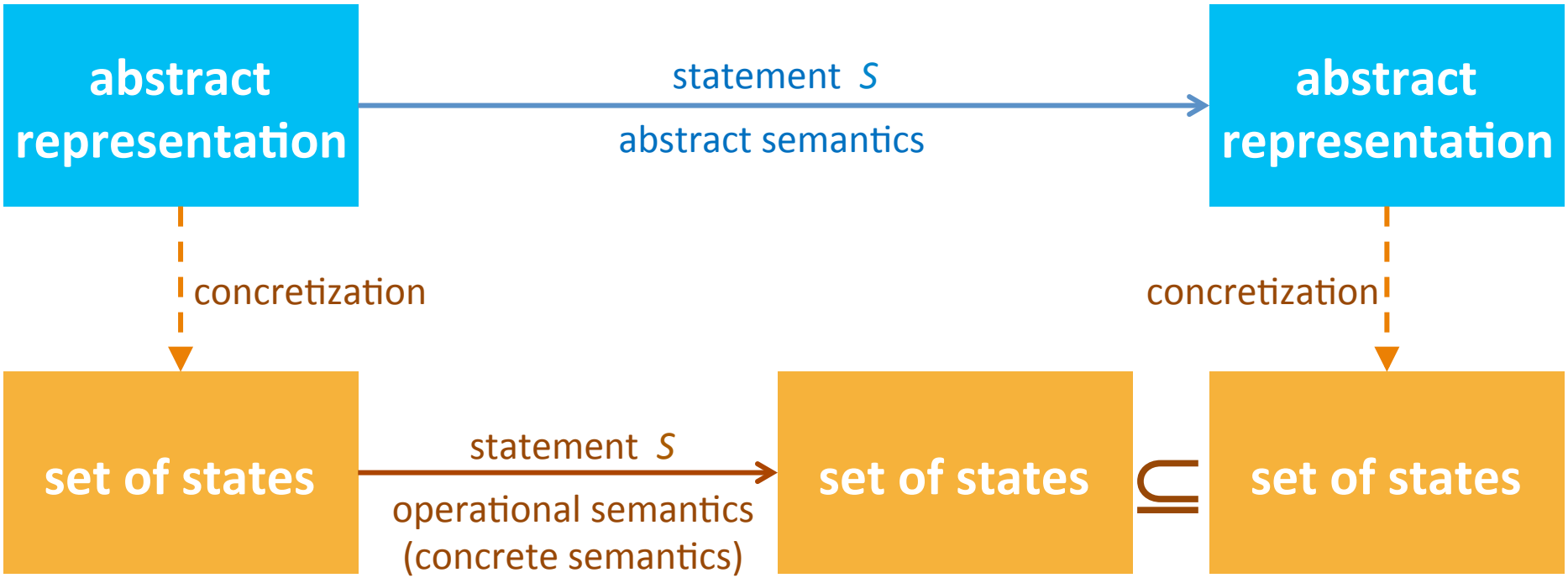
Lecture 12: Abstract Interpretation IV

Slides credit: Roman Manevich, Mooly Sagiv, Eran Yahav, Ganesan Ramalingam

Abstract Interpretation [Cousot'77]

- Mathematical foundation of static analysis
  - Abstract domains
    - Abstract states
    - Join ($\sqcup$)
  - Transformer functions
    - Abstract steps
  - Chaotic iteration
    - Structured Programs
    - Abstract computation

Abstract (conservative) interpretation



Abstract Interpretation [Cousot'77]

- Mathematical foundation of static analysis
  - Abstract domains
    - Abstract states
    - Join ($\sqcup$)
  - Transformer functions
    - Abstract steps
  - Chaotic iteration
    - Abstract computation
    - Structured Programs

Lattices
$(D, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$

Monotonic functions

Fixpoints

Chains

- $d \sqsubset d'$ means $d \sqsubseteq d'$ and $d \neq d'$
- An ascending chain is a sequence
  $x_1 \sqsubset x_2 \sqsubset \ldots \sqsubset x_k \ldots$
- A descending chain is a sequence
  $x_1 \sqsupset x_2 \sqsupset \ldots \sqsupset x_k \ldots$
- The height of a poset $(D, \sqsubseteq)$ is the length of the maximal ascending chain in D

Complete partial order (CPO)

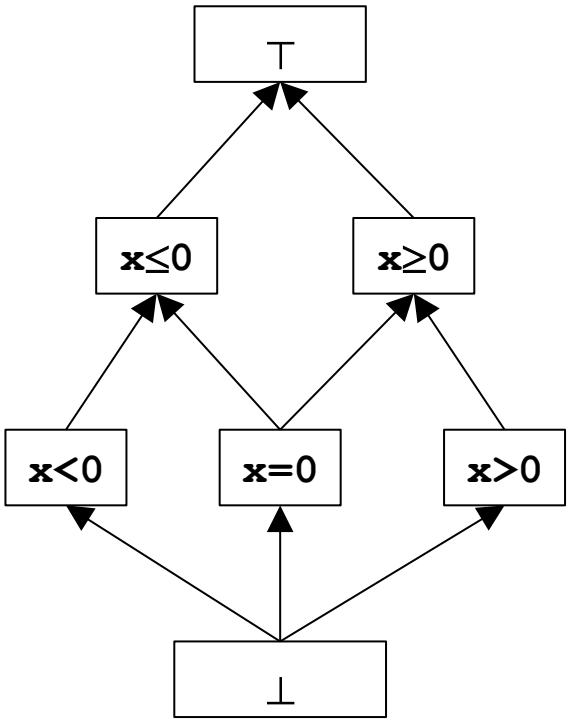
- A poset (D , $\sqsubseteq$) is a complete partial if every ascending chain $x_1 \sqsubset x_2 \sqsubset \ldots \sqsubset x_k \ldots$ has a LUB

Lattices

- $(D, \sqsubseteq, \bigsqcup, \bigsqcap, \bot, \top)$ is a lattice if
  - $(D, \sqsubseteq)$ is a partial order
  - $\forall X \subseteq_{FIN} D$. $\bigsqcup X$ is defined
  - A top element $\top$
  - $\forall X \subseteq_{FIN} D$. $\bigsqcap X$ is defined
  - A bottom element $\bot$

- A lattice $(D, \sqsubseteq, \bigsqcup, \bigsqcap, \bot, \top)$ is a complete lattice if
  - $\bigsqcup X$ and $\bigsqcap Y$ are defined for arbitrary sets

Example I: Sign lattice



Example II: Powerset lattices

- $(2^X, \subseteq, \cup, \cap, \varnothing, X)$ is the powerset lattice of X
  - A complete lattice

Domain Constructors

- Cartesian products: $0 < x \wedge 0 < y$:
- Disjunctive completion: $x < 0 \vee 0 < x$
- Relational product: $(0 < x \wedge 0 < y) \vee (0 < x \wedge y < 0)$
- Finite maps: $[L1 \mapsto x = \top, L2 \mapsto x = 0]$
  - $|\{L1, L2\}|$ finite

Concrete Semantics

Collecting semantics

- For a set of program states **State**, we define the collecting lattice

  $(2^{\mathbf{State}}, \subseteq, \cup, \cap, \varnothing, \mathbf{State})$

- The collecting semantics accumulates the (possibly infinite) sets of states generated during the execution
  - Not computable in general

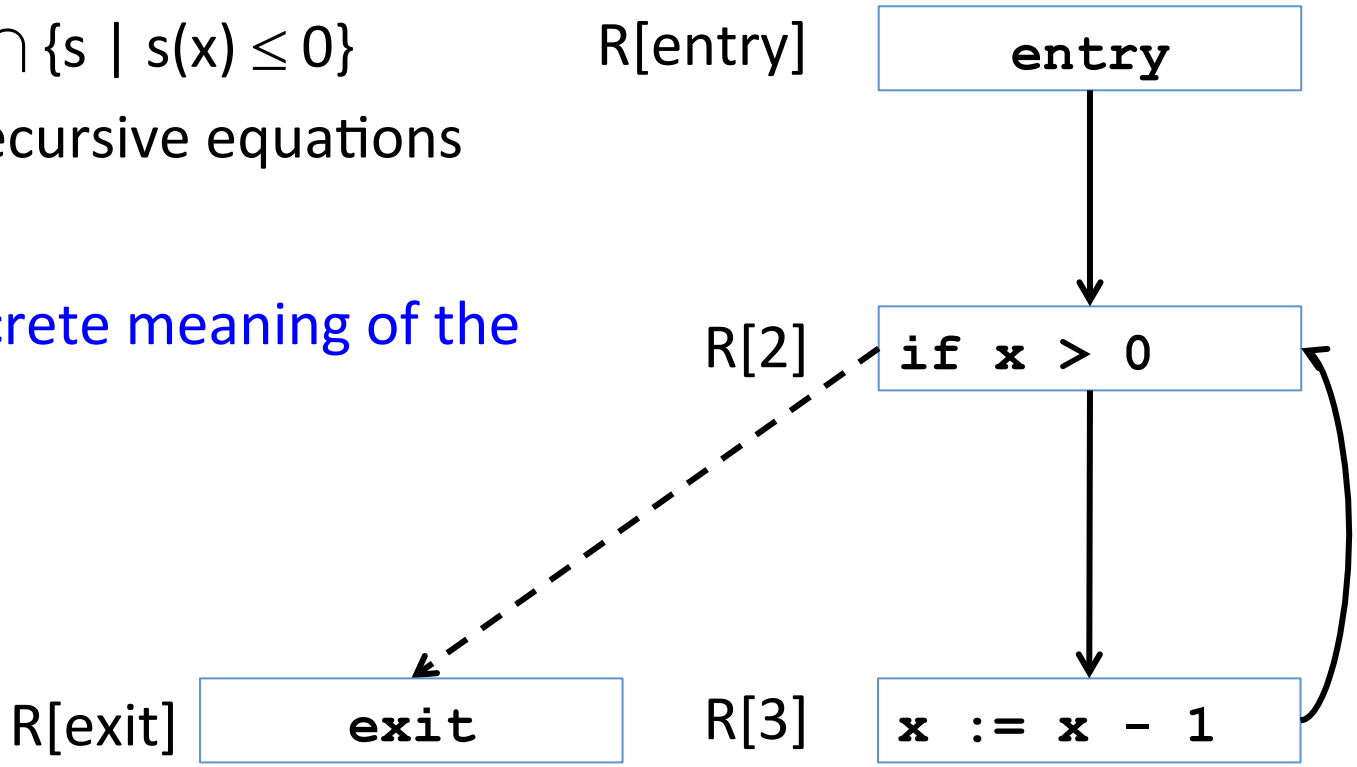
The collecting lattice

- Lattice for a given control-flow node *v*:
  $L_v = (2^{\mathbf{State}}, \subseteq, \cup, \cap, \varnothing, \mathbf{State})$
- Lattice for entire control-flow graph with nodes *V*:
  $$L_{CFG} = \text{Map}(V, L_v)$$
- We will use this lattice as a baseline for static analysis and define abstractions of its elements

Equational definition of the semantics

- R[2] = R[entry] ∪ ⟦**x:=x-1**⟧ R[3]
- R[3] = R[2] ∩ {s | s(x) > 0}
- R[exit] = R[2] ∩ {s | s(x) ≤ 0}
- A system of recursive equations

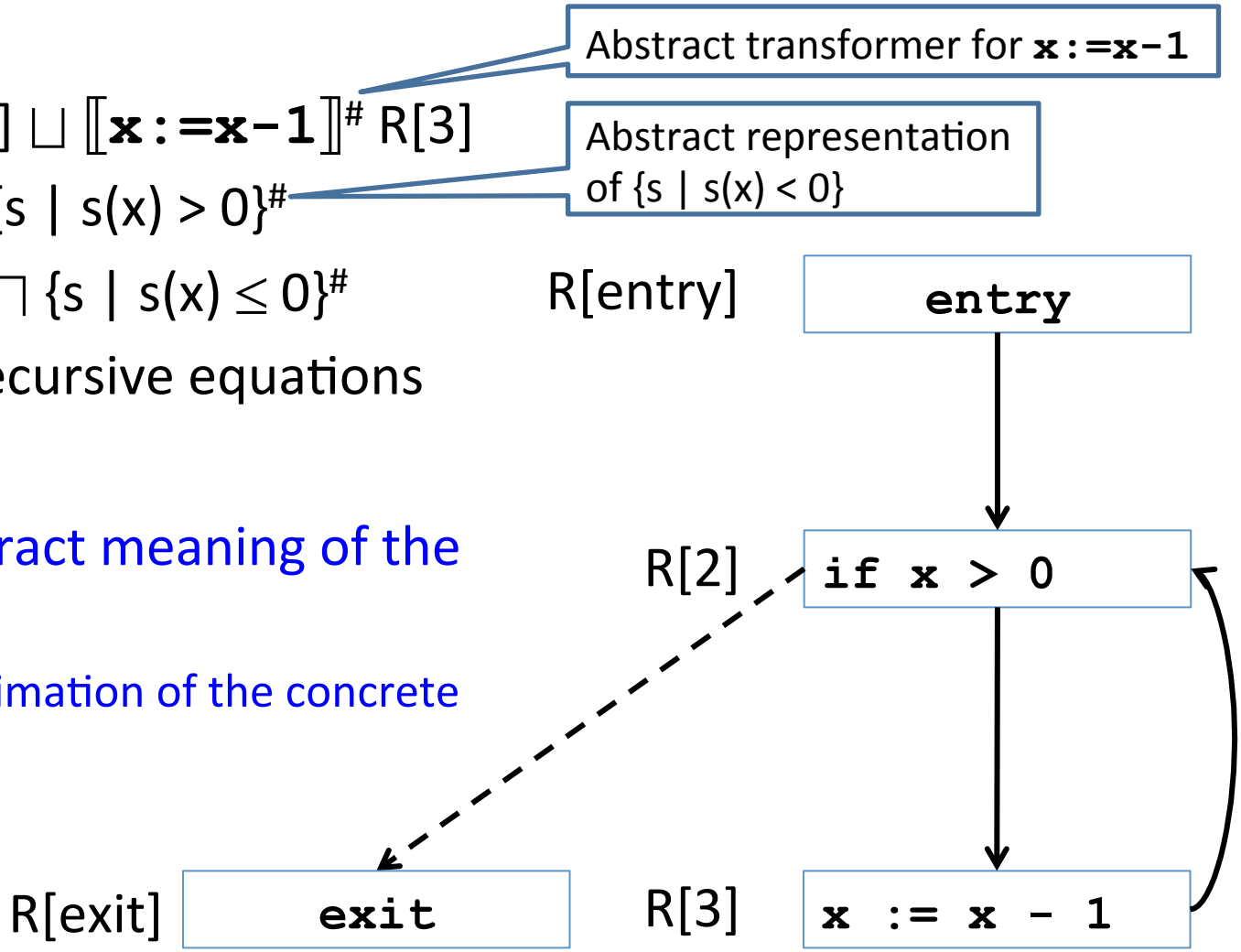
- Solution: concrete meaning of the program



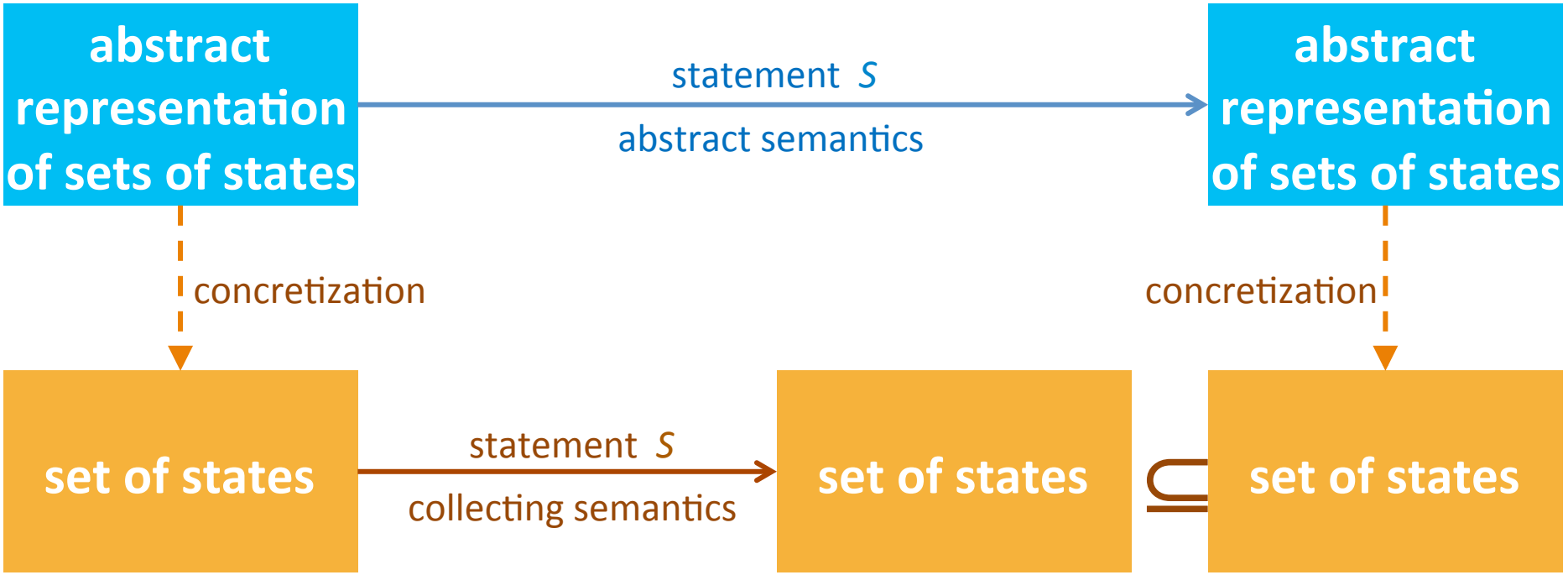
An abstract semantics

- R[2] = R[entry] ⊔ ⟦`x:=x-1`⟧$^{\#}$ R[3]
- R[3] = R[2] ⊓ {s | s(x) > 0}$^{\#}$
- R[exit] = R[2] ⊓ {s | s(x) ≤ 0}$^{\#}$
- A system of recursive equations

- Solution: abstract meaning of the program
  - Over-approximation of the concrete semantics



Abstract interpretation via concretization



Can we always find a solution?

Monotone functions

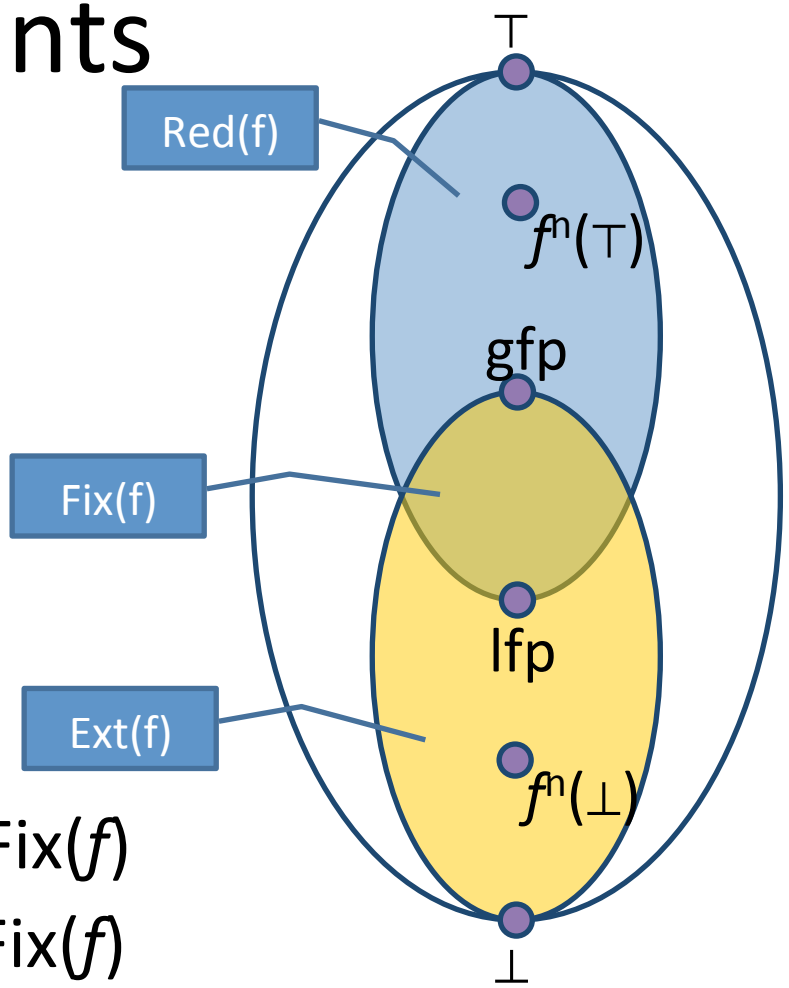
- Let $L_1=(D_1, \sqsubseteq)$ and $L_2=(D_2, \sqsubseteq)$ be two posets
- A function $f : D_1 \rightarrow D_2$ is monotone if for every pair $x, y \in D_1$

  $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$
- A special case: $L_1=L_2=(D, \sqsubseteq)$

  $f : D \rightarrow D$

Fixed-points

- $L = (D, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$
- $f : D \rightarrow D$ **monotone**
- $\mathrm{Fix}(f) = \{ d \mid f(d) = d \}$
- $\mathrm{Red}(f) = \{ d \mid f(d) \sqsubseteq d \}$
- $\mathrm{Ext}(f) = \{ d \mid d \sqsubseteq f(d) \}$
- **Theorem** [Tarski 1955]
  - $\mathrm{lfp}(f) = \sqcap \mathrm{Fix}(f) = \sqcap \mathrm{Red}(f) \in \mathrm{Fix}(f)$
  - $\mathrm{gfp}(f) = \sqcup \mathrm{Fix}(f) = \sqcup \mathrm{Ext}(f) \in \mathrm{Fix}(f)$



1. A solution always exist
2. It unique
3. Not always computable

Continuity and ACC condition

- Let $L = (D, \sqsubseteq, \sqcup, \bot)$ be a complete partial order
  - Every ascending chain has an upper bound
- A function f is continuous if for every increasing chain $Y \subseteq D^*$,

$$f(\sqcup Y) = \sqcup \{ f(y) \mid y \in Y \}$$

- L satisfies the ascending chain condition (ACC) if every ascending chain eventually stabilizes:

$$d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_n = d_{n+1} = \ldots$$

Fixed-point theorem [Kleene]

- Let $L = (D, \sqsubseteq, \sqcup, \bot)$ be a complete partial order and a **continuous** function $f: D \to D$ then

$$\mathrm{lfp}(f) = \bigsqcup_{n \in \mathbb{N}} f^n(\bot)$$

- **Lemma:** Monotone functions on posets satisfying ACC are continuous

Resulting algorithm

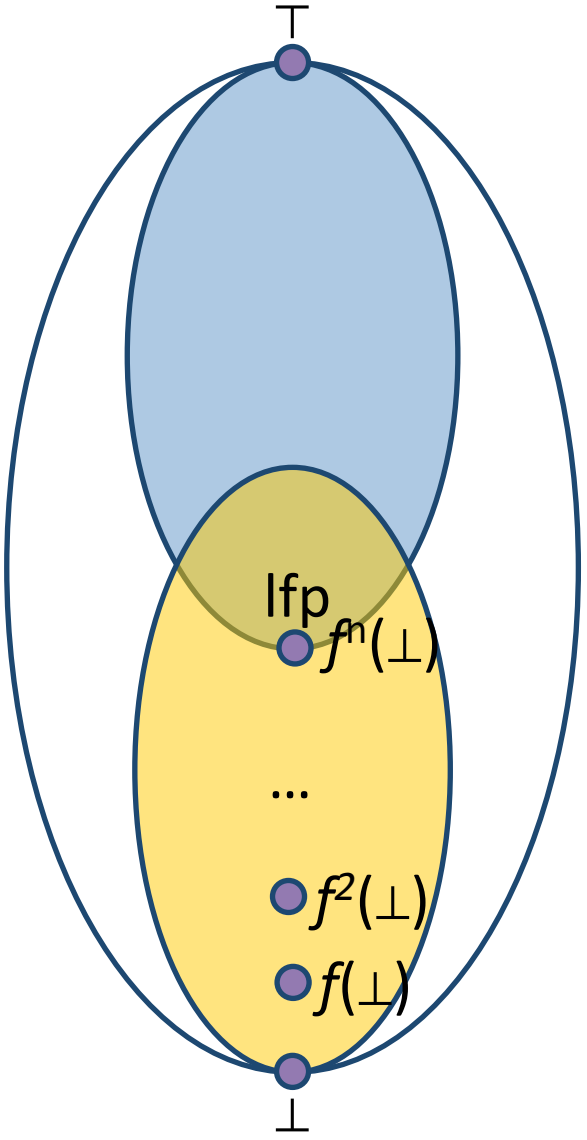
- Kleene's fixed point theorem gives a constructive method for computing the lfp

Mathematical definition

$$\text{lfp}(f) = \bigsqcup_{n \in \mathbb{N}} f^n(\bot)$$

Algorithm

$d := \bot$
while $f(d) \neq d$ **do**
  $d := d \sqcup f(d)$
return d



Correctness

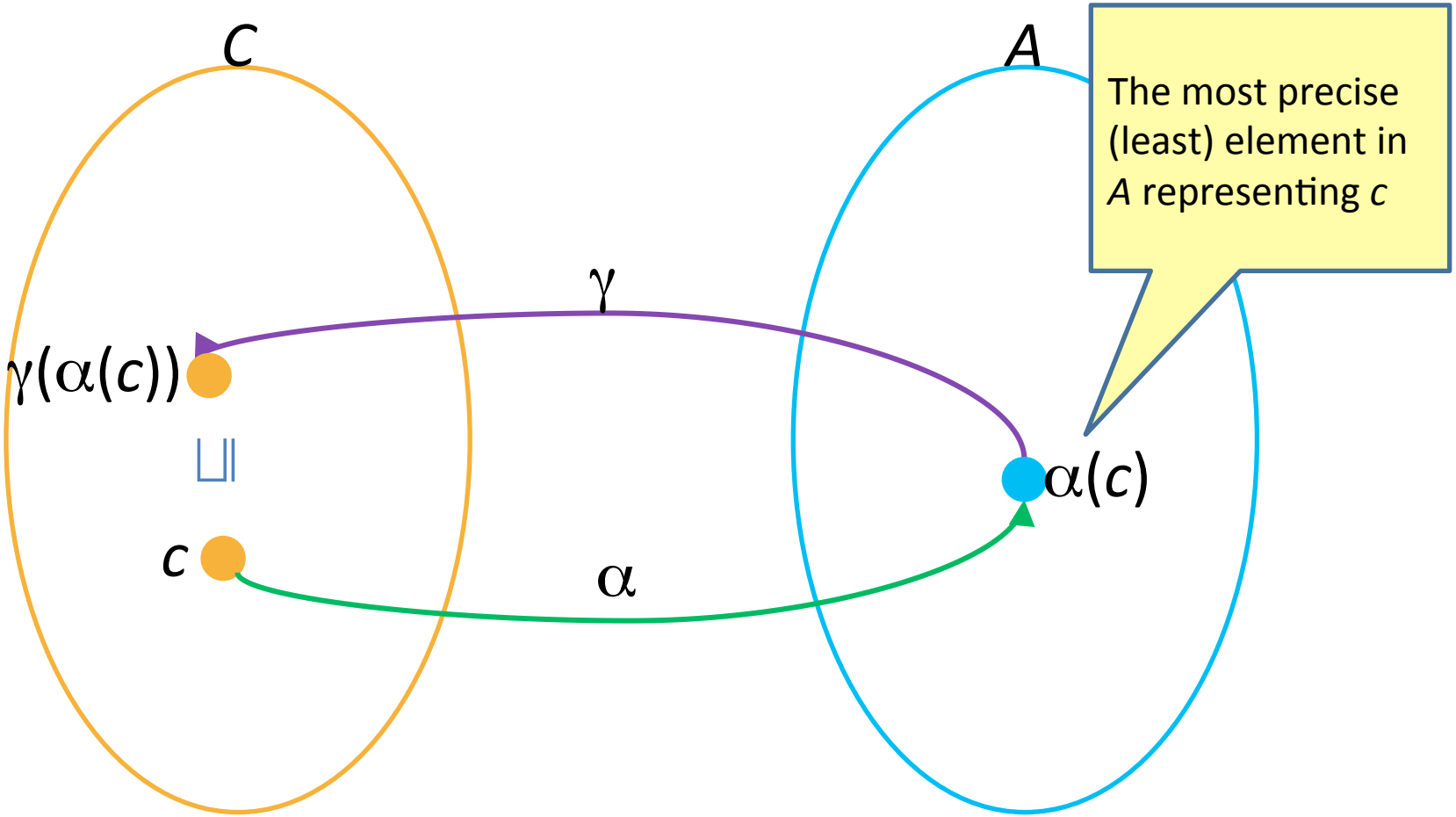
- Transformer monotonicity is required for termination – what should we require for correctness?

- What is the connection between the two least fixed-points of the concrete and abstract semantics?

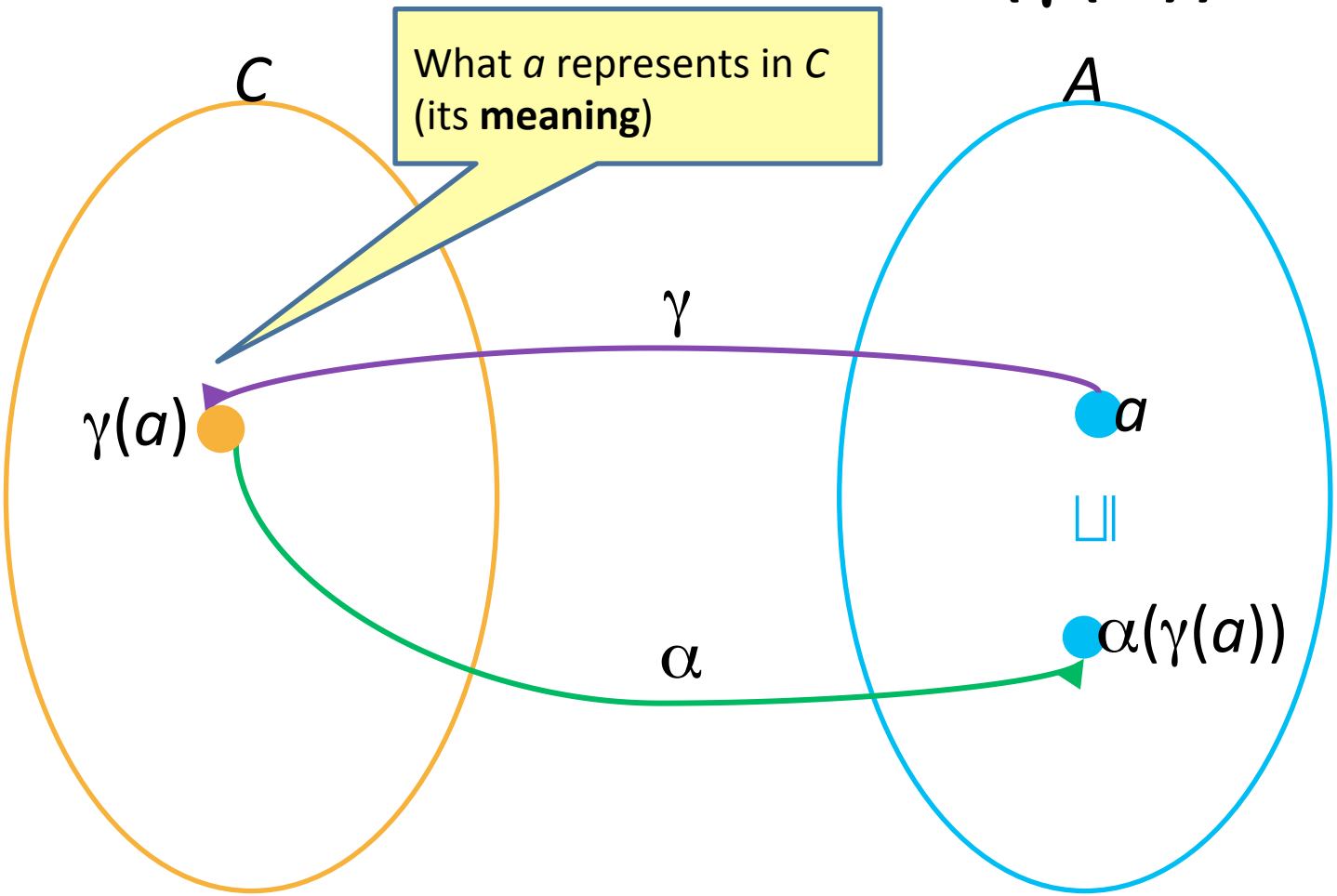
Abstraction/Concretization

- Given two complete lattices
  $C = (D^C, \sqsubseteq^C, \sqcup^C, \sqcap^C, \bot^C, \top^C)$ – concrete domain
  $A = (D^A, \sqsubseteq^A, \sqcup^A, \sqcap^A, \bot^A, \top^A)$ – abstract domain
- A Galois Connection (GC) is quadruple (C, α, γ, A) that relates C and A via the monotone functions
  - The abstraction function $\alpha : D^C \rightarrow D^A$
  - The concretization function $\gamma : D^A \rightarrow D^C$
- for every concrete element $c \in D^C$ and abstract element $a \in D^A$
  $\alpha(\gamma(a)) \sqsubseteq a$ and $c \sqsubseteq \gamma(\alpha(c))$
- Alternatively $\alpha(c) \sqsubseteq a$ iff $c \sqsubseteq \gamma(a)$
  - Homework

Galois Connection: $c \sqsubseteq \gamma(\alpha(c))$



Galois Connection: $\alpha(\gamma(a)) \sqsubseteq a$



Properties of a Galois Connection

- The abstraction and concretization functions uniquely determine each other:

$$\gamma(a) = \bigsqcup\{c \mid \alpha(c) \sqsubseteq a\}$$

$$\alpha(c) = \bigsqcap\{a \mid c \sqsubseteq \gamma(a)\}$$

Abstracting sets

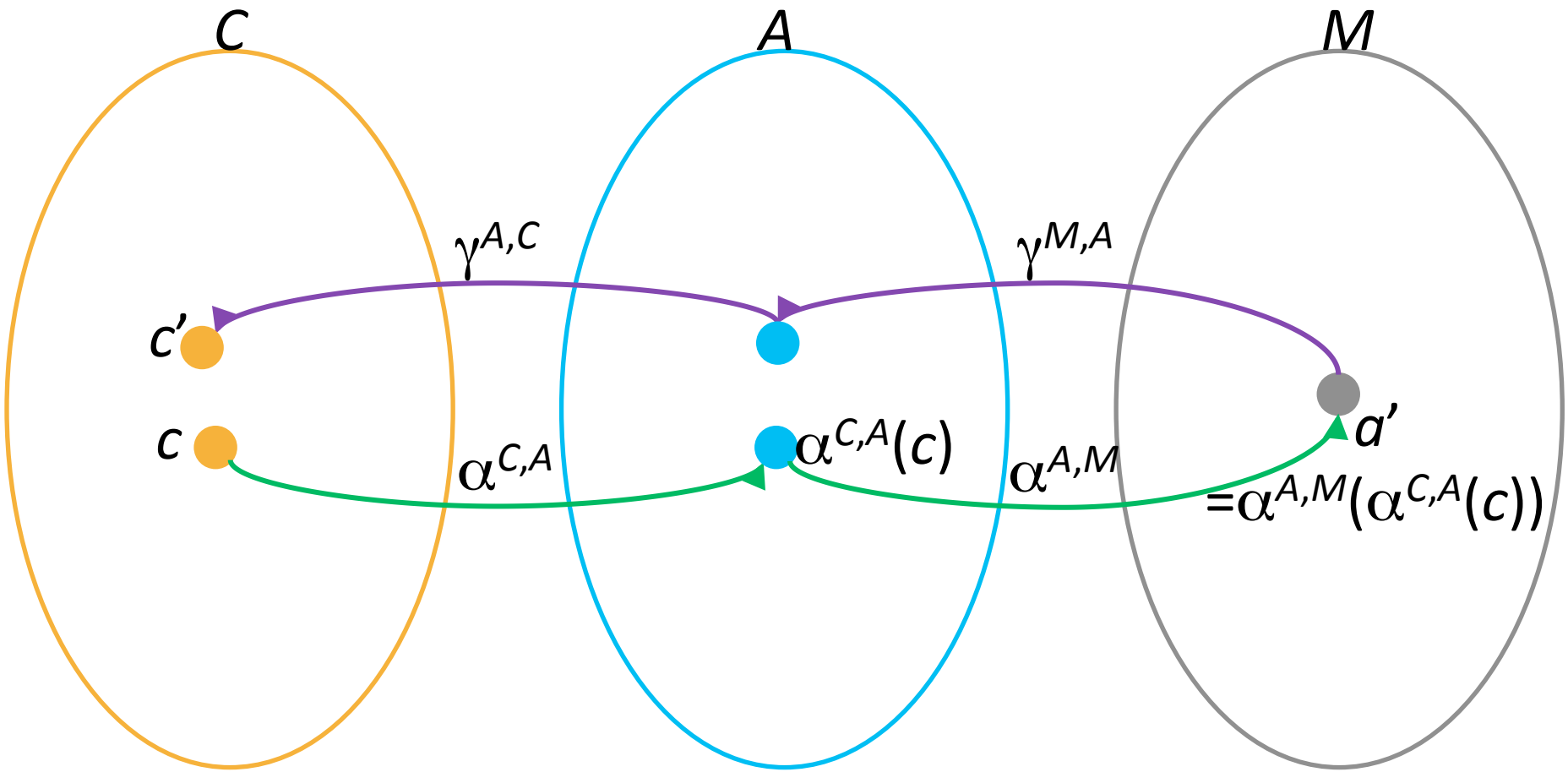
- It is usually convenient to first define the abstraction of single elements

$$\beta(s) = \alpha(\{s\})$$

- Then lift the abstraction to sets of elements

$$\alpha(X) = \bigsqcup^{A} \{\beta(s) \mid s \in X\}$$

Inducing along the connections

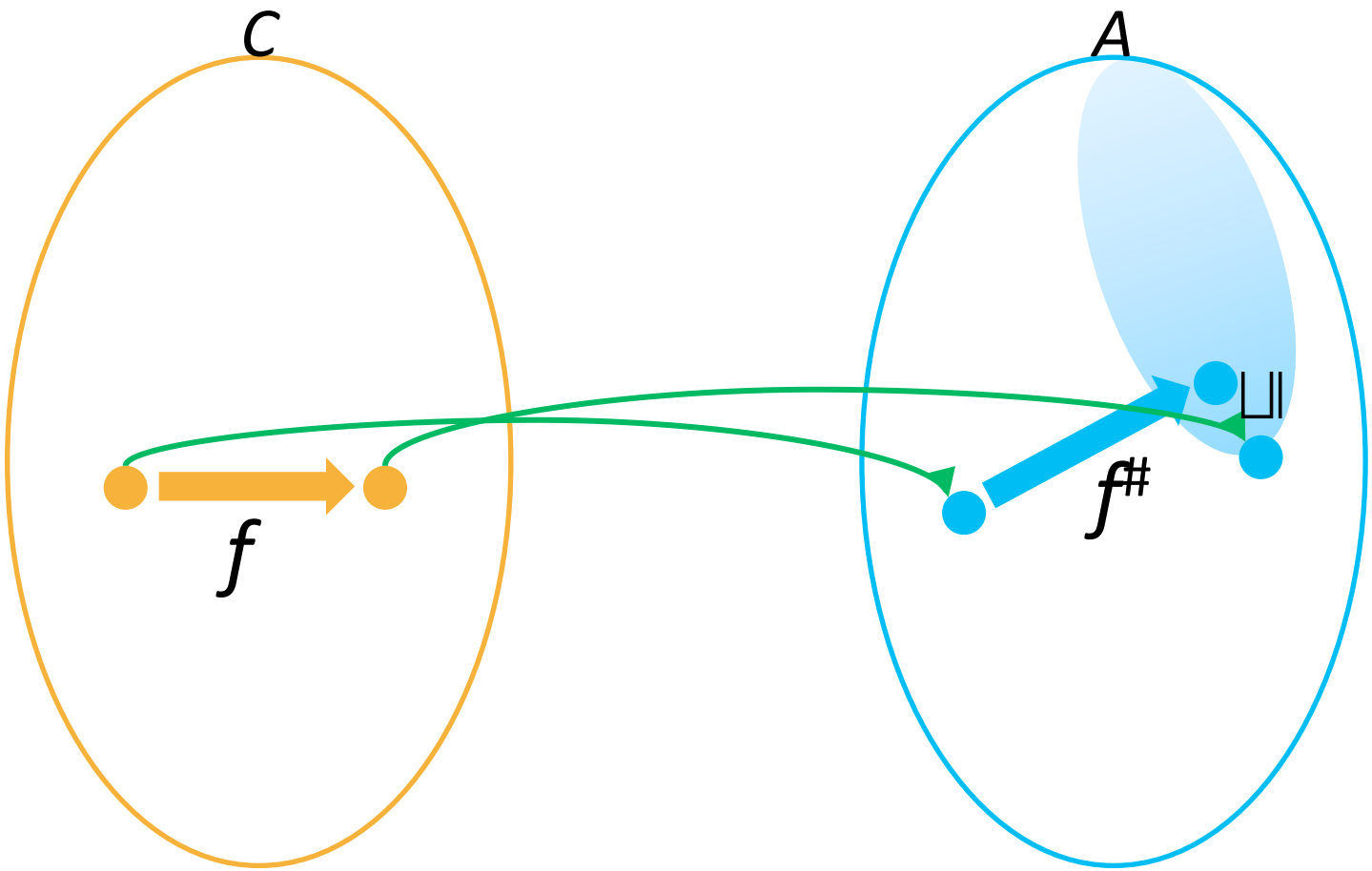


Sound abstract transformer

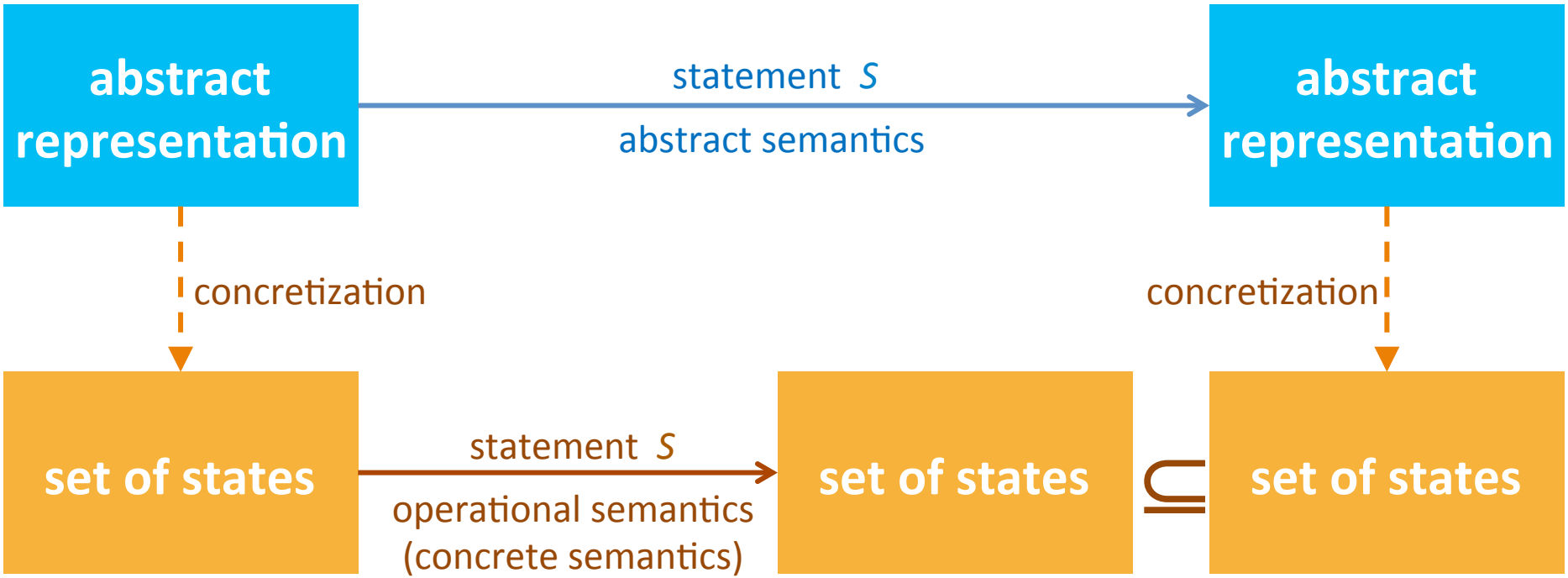
- Given two lattices
  $C = (D^C, \sqsubseteq^C, \sqcup^C, \sqcap^C, \bot^C, \top^C)$
  $A = (D^A, \sqsubseteq^A, \sqcup^A, \sqcap^A, \bot^A, \top^A)$
  and $GC^{C,A}=(C, \alpha, \gamma, A)$ with
- A concrete transformer $f : D^C \rightarrow D^C$
  an abstract transformer $f^\# : D^A \rightarrow D^A$
- We say that $f^\#$ is a sound transformer (w.r.t. f) if
  - $\forall c: \alpha(f(c)) \sqsubseteq f^\#(\alpha(c))$
  - $\forall a: \alpha(f(\gamma(a))) \sqsubseteq^A f^\#(a)$

Transformer soundness condition 1

$$\forall c: f(c)=c' \Rightarrow f^{\#}(\alpha(c)) \sqsupseteq \alpha(c')$$

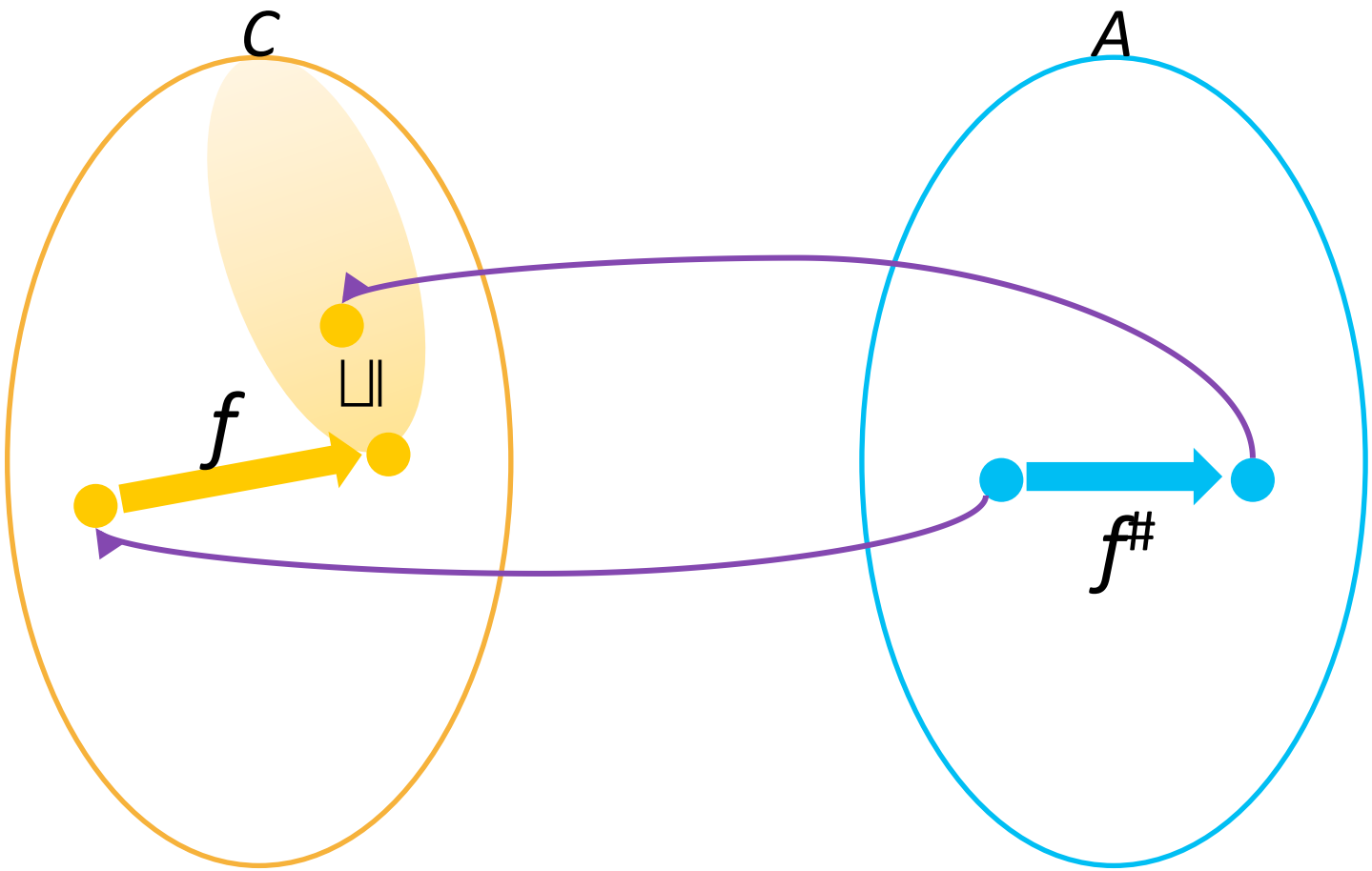


Abstract (conservative) interpretation

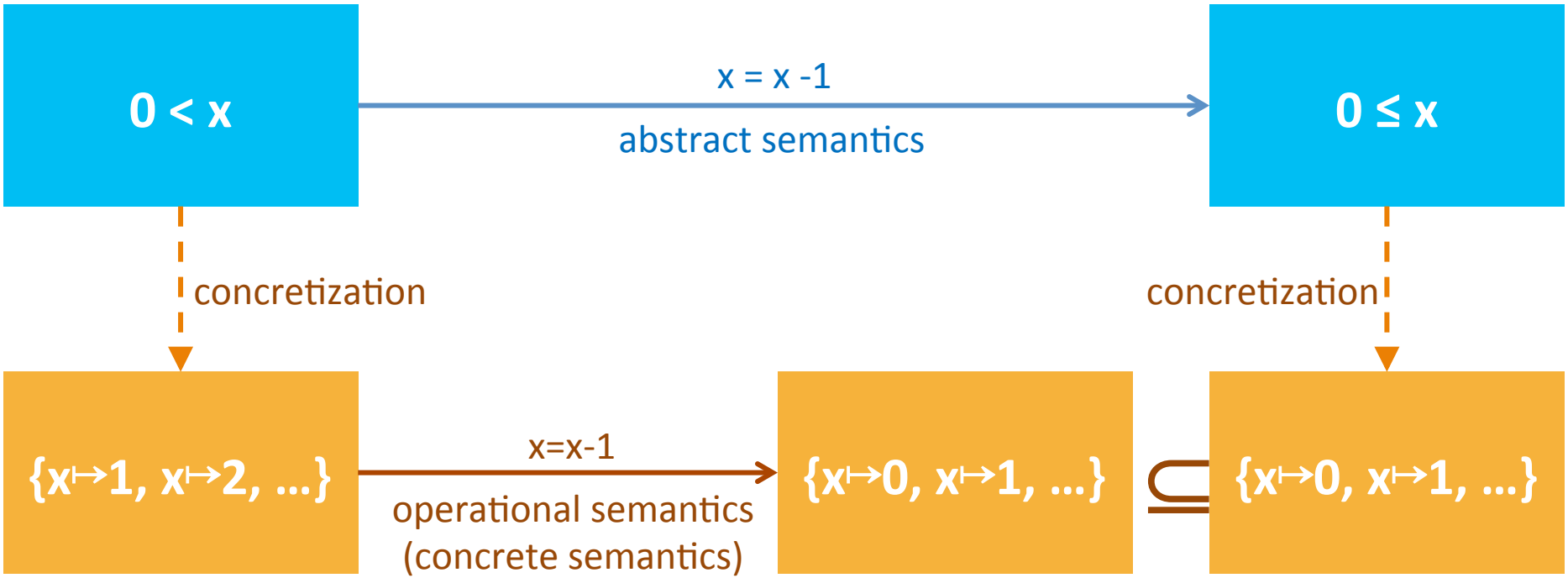


Transformer soundness condition 2

$$\forall a: f^{\#}(a)=a' \Rightarrow f(\gamma(a)) \sqsubseteq \gamma(a')$$

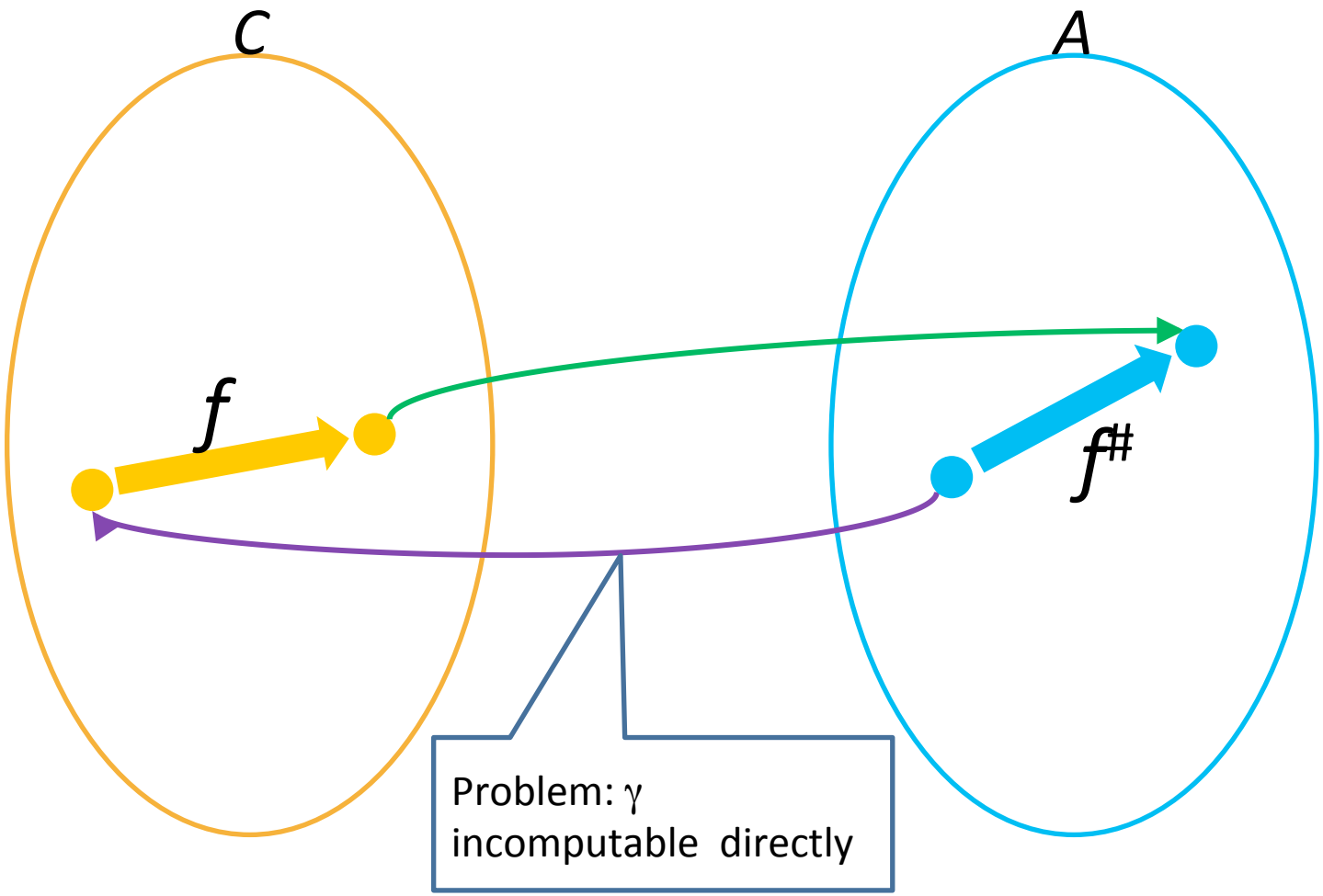


Abstract (conservative) interpretation



Best (induced) transformer

$$f^{\#}(a) = \alpha(f(\gamma(a)))$$



Best abstract transformer [CC'77]

- Best in terms of **precision**
  - Most precise abstract transformer
  - May be too expensive to compute
- Constructively defined as

$$f^{\#} = \alpha \circ f \circ \gamma$$

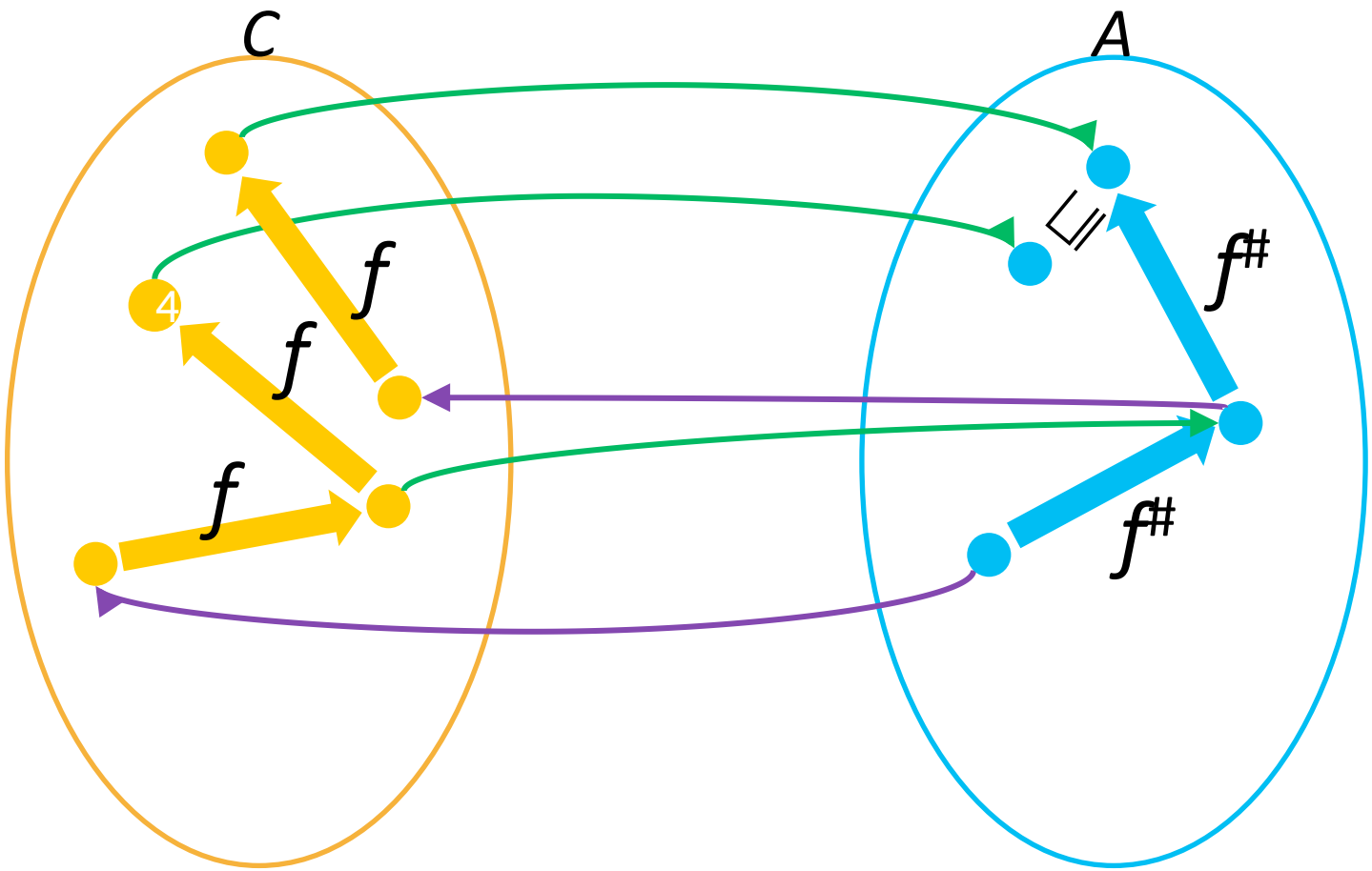
  - Induced by the GC
- Not directly computable because first step is concretization
- We often compromise for a "good enough" transformer
  - Useful tool: partial concretization

Negative property of best transformers

- Let $f^{\#} = \alpha \circ f \circ \gamma$
- Best transformer does not compose

$$\alpha(f(f(\gamma(a)))) \sqsubseteq f^{\#}(f^{\#}(a))$$

$$\alpha(f(f(\gamma(a)))) \sqsubseteq f^{\#}(f^{\#}(a))$$



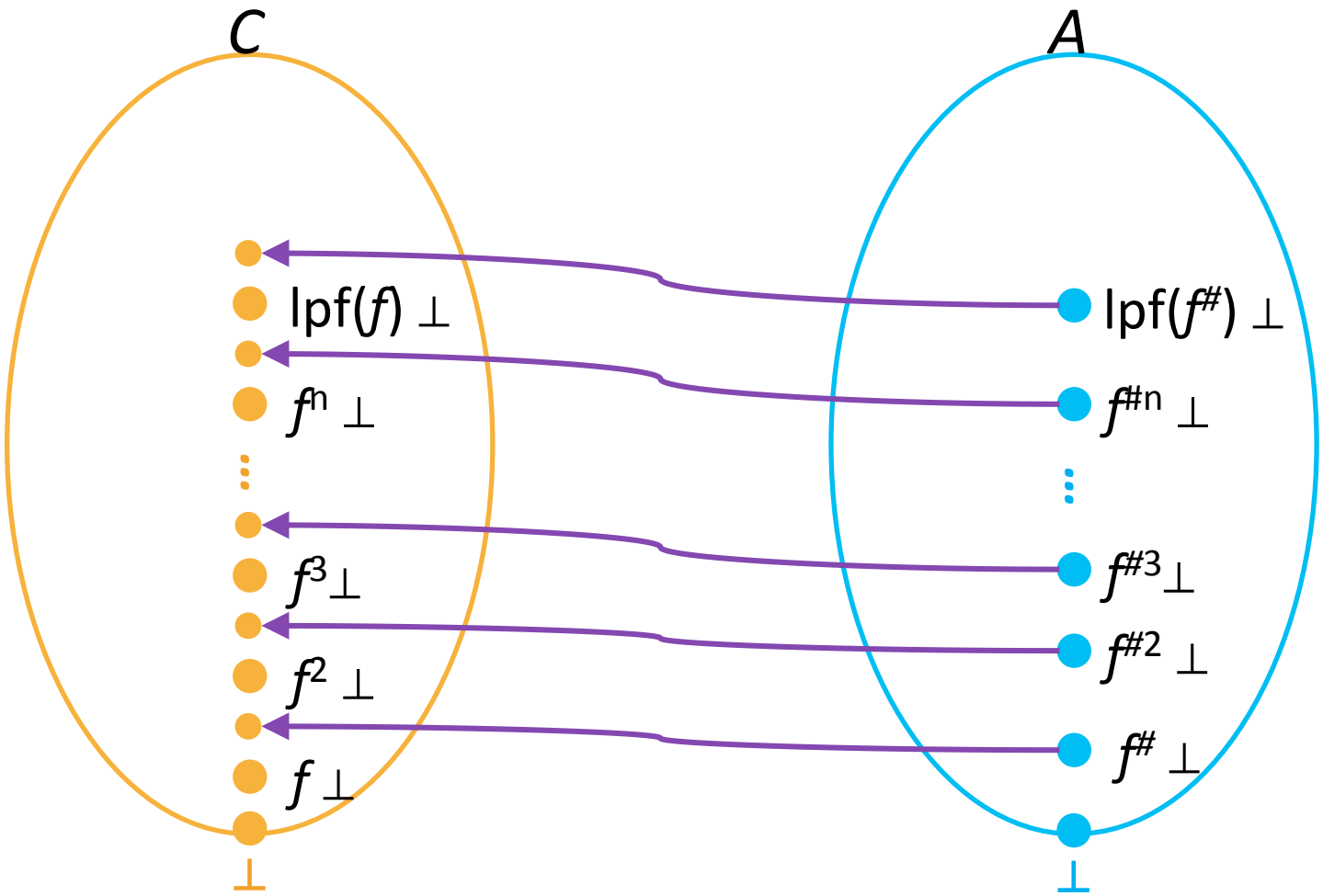
Soundness theorem 1

1. Given two complete lattices
   $C = (D^C, \sqsubseteq^C, \sqcup^C, \sqcap^C, \bot^C, \top^C)$
   $A = (D^A, \sqsubseteq^A, \sqcup^A, \sqcap^A, \bot^A, \top^A)$
   and $GC^{C,A}=(C, \alpha, \gamma, A)$ with
2. Monotone concrete transformer $f : D^C \rightarrow D^C$
3. Monotone abstract transformer $f^\# : D^A \rightarrow D^A$
4. $\forall a \in D^A : f(\gamma(a)) \sqsubseteq \gamma(f^\#(a))$

Then

$$\mathrm{lfp}(f) \sqsubseteq \gamma(\mathrm{lfp}(f^\#))$$
$$\alpha(\mathrm{lfp}(f)) \sqsubseteq \mathrm{lfp}(f^\#)$$

Soundness theorem 1



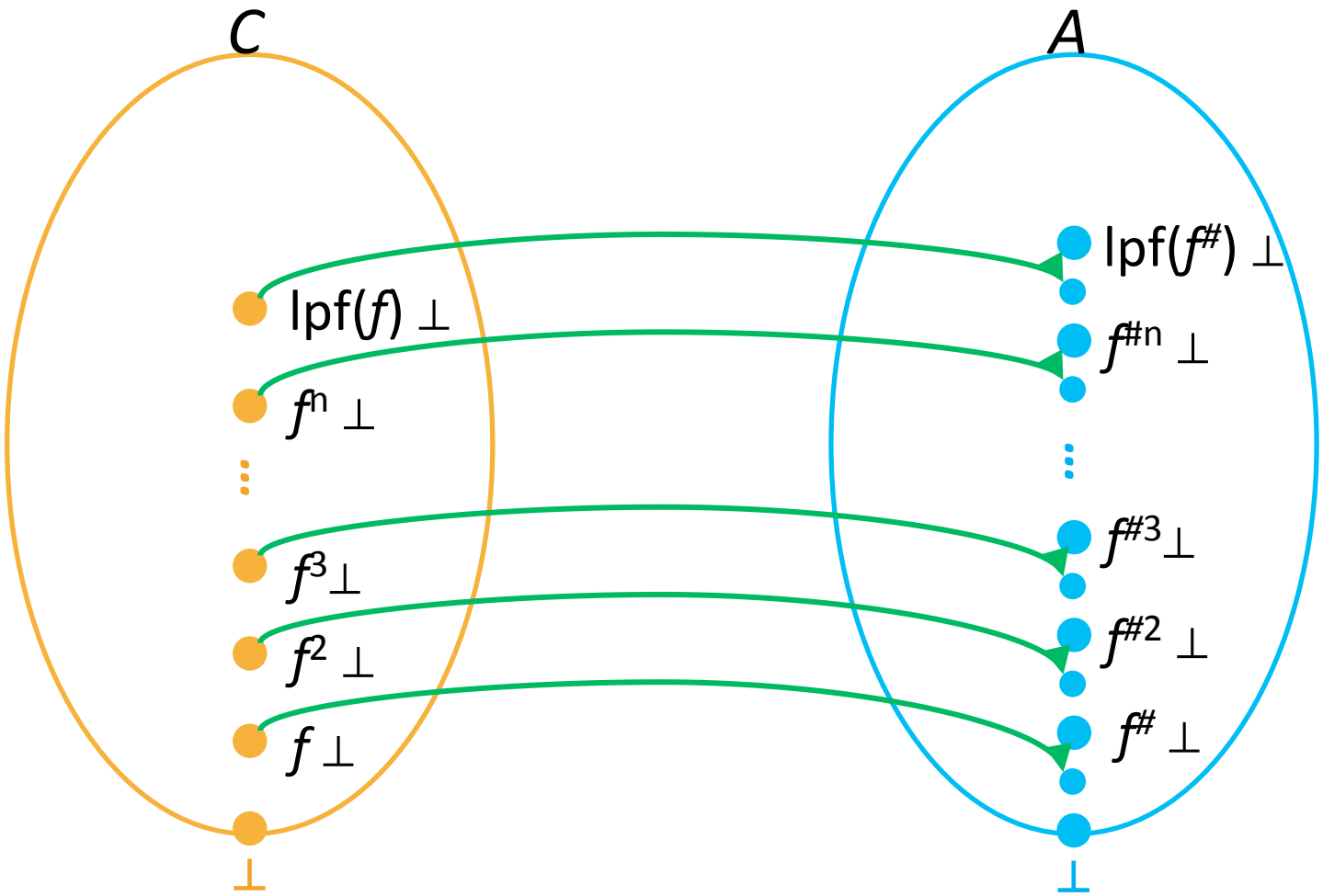
Soundness theorem 2

1. Given two complete lattices
   $C = (D^C, \sqsubseteq^C, \sqcup^C, \sqcap^C, \bot^C, \top^C)$
   $A = (D^A, \sqsubseteq^A, \sqcup^A, \sqcap^A, \bot^A, \top^A)$
   and $GC^{C,A}=(C, \alpha, \gamma, A)$ with
2. Monotone concrete transformer $f : D^C \rightarrow D^C$
3. Monotone abstract transformer $f^\# : D^A \rightarrow D^A$
4. $\forall c \in D^C : \alpha(f(c)) \sqsubseteq f^\#(\alpha(c))$

Then

$$\alpha(\mathrm{lfp}(f)) \sqsubseteq \mathrm{lfp}(f^\#)$$
$$\mathrm{lfp}(f) \sqsubseteq \gamma(\mathrm{lfp}(f^\#))$$

Soundness theorem 2



A recipe for a sound static analysis

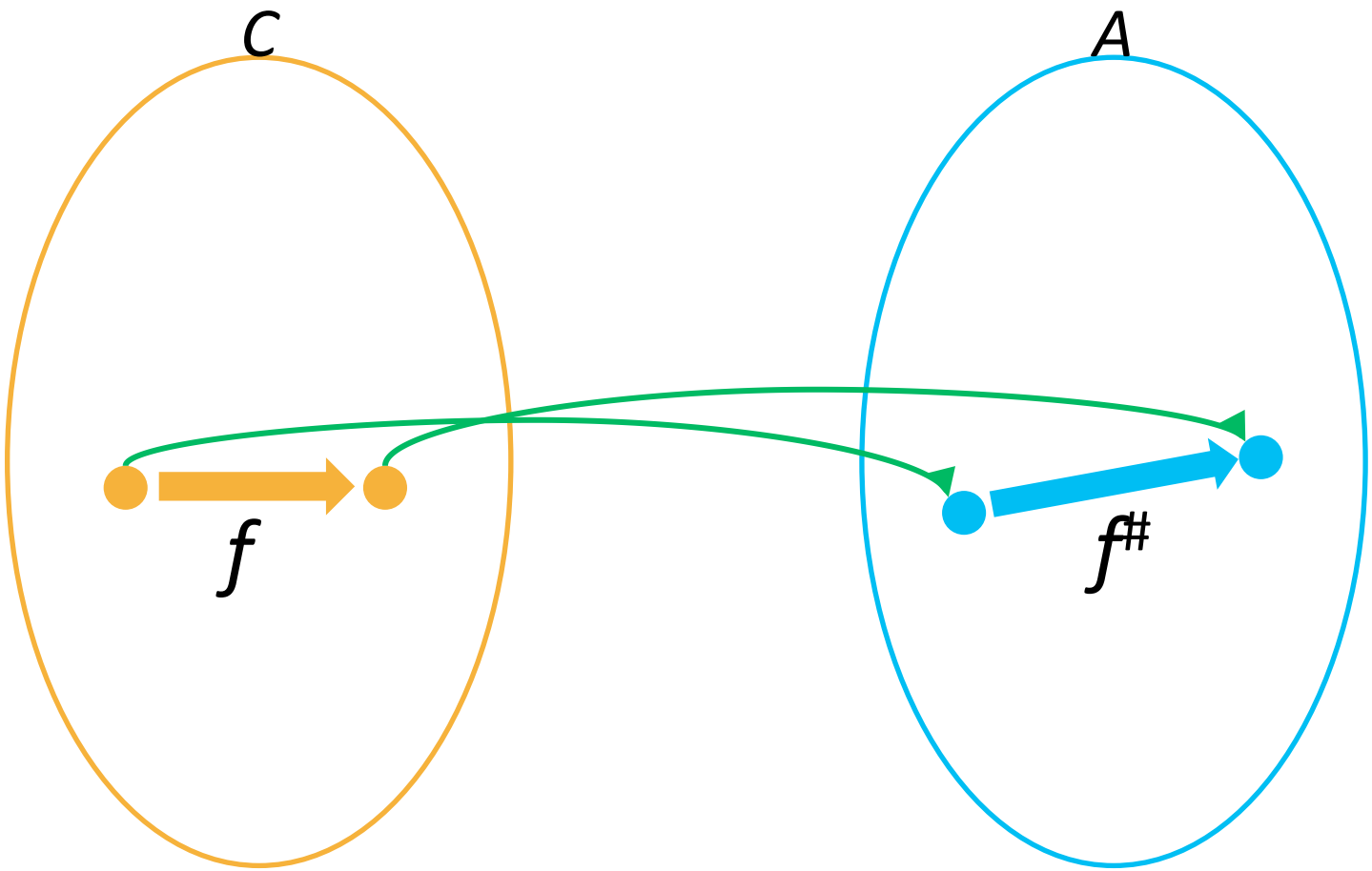
- Define an “appropriate” operational semantics
- Define “collecting” structural operational semantics
- Establish a Galois connection between collecting states and abstract states
- **Local correctness:** show that the abstract interpretation of every atomic statement is sound
  w.r.t. the collecting semantics
- **Global correctness:** conclude that the analysis is sound

Completeness

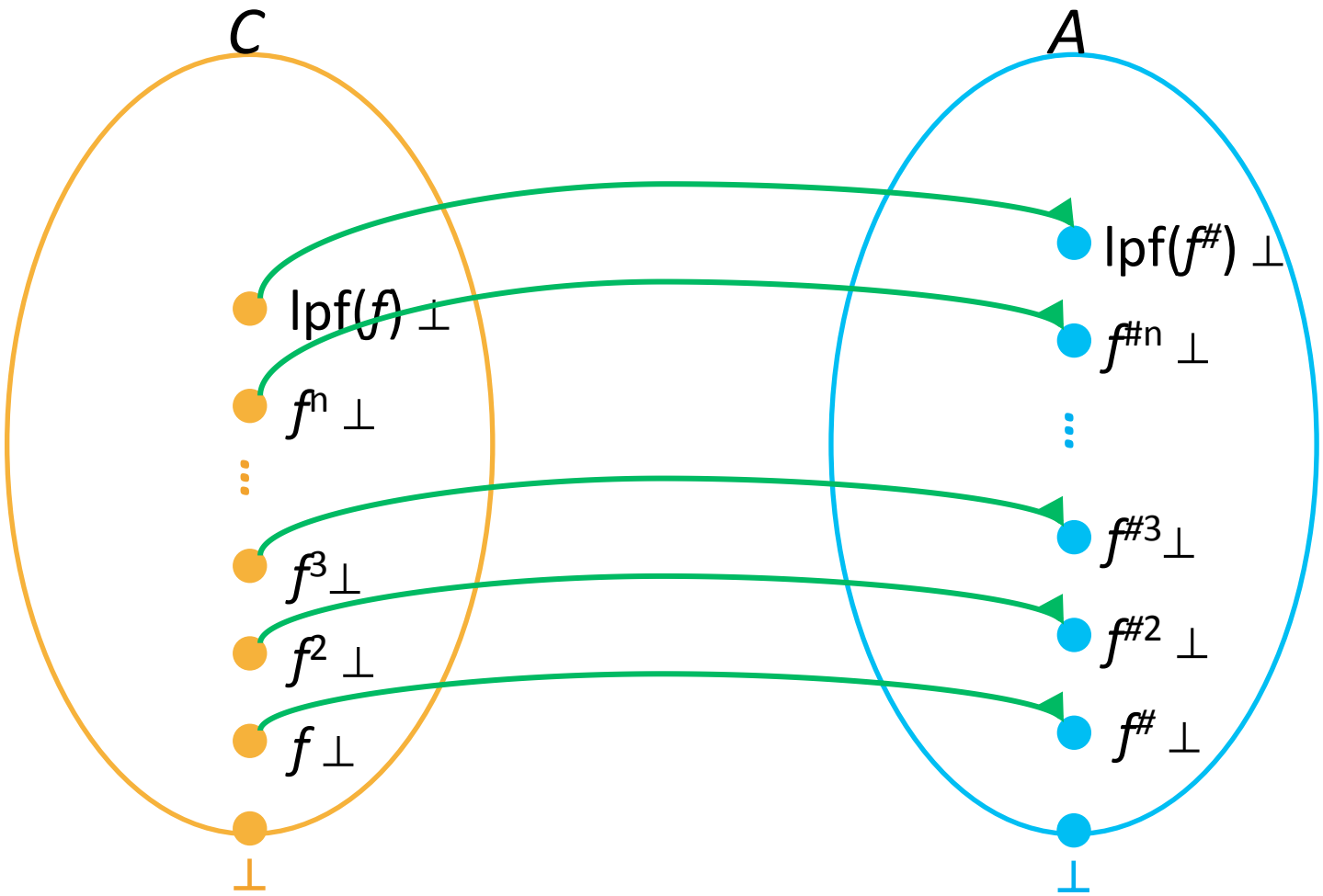
- Local property:
  - forward complete: $\forall c: \alpha(f^{\#}(c)) = \alpha(f(c))$
  - backward complete: $\forall a: f(\gamma(a)) = \gamma(f^{\#}(a))$
- A property of domain and the (best) transformer
- Global property:
  - $\alpha(\text{lfp}(f)) = \text{lfp}(f^{\#})$
  - $\text{lfp}(f) = \gamma(\text{lfp}(f^{\#}))$
- Very ideal but usually not possible unless we change the program model (apply strong abstraction) and/or aim for very simple properties

Forward complete transformer

$$\forall c: \alpha(f^{\#}(c)) = \alpha(f(c))$$

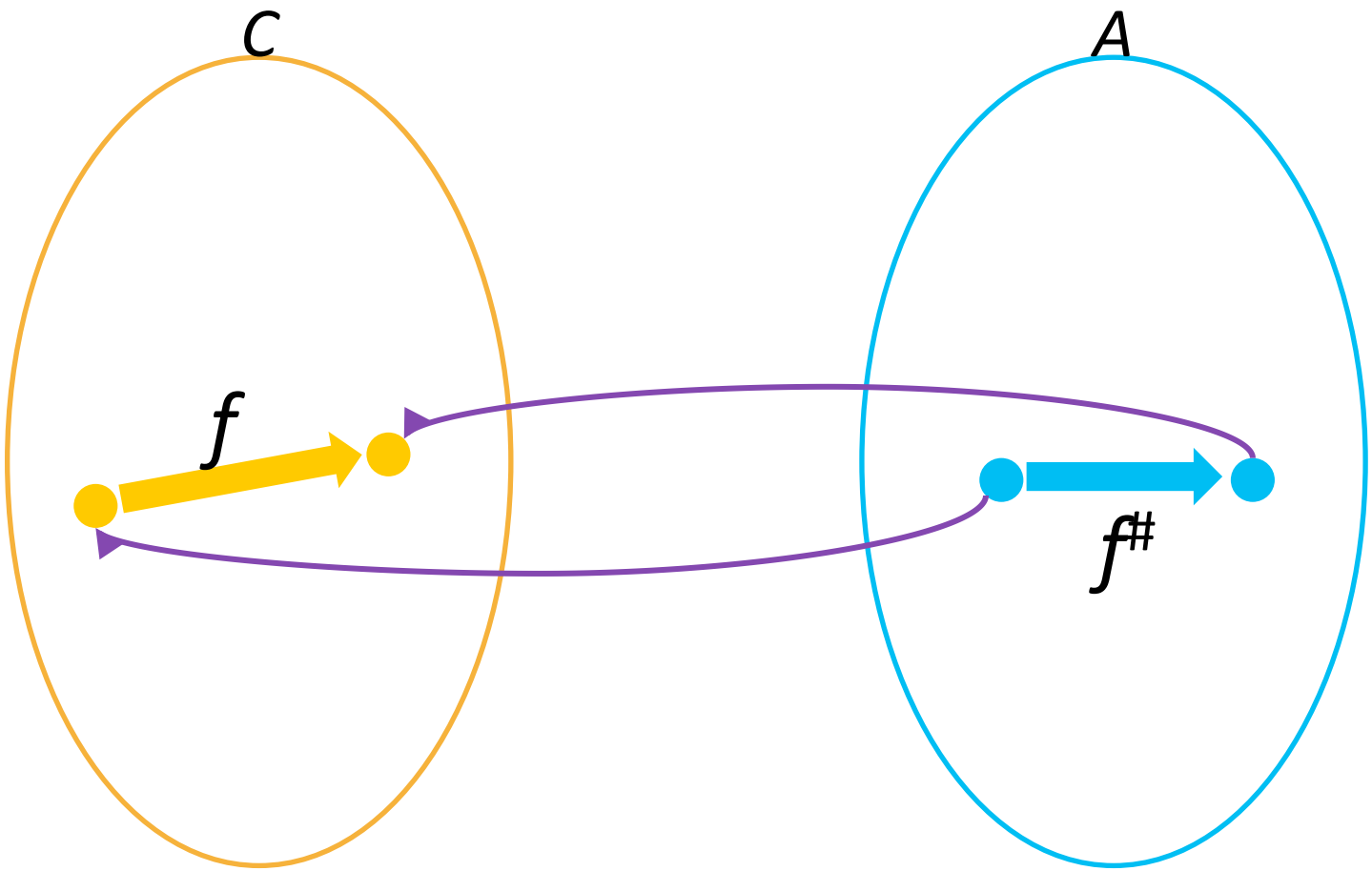


Global (forward) completeness

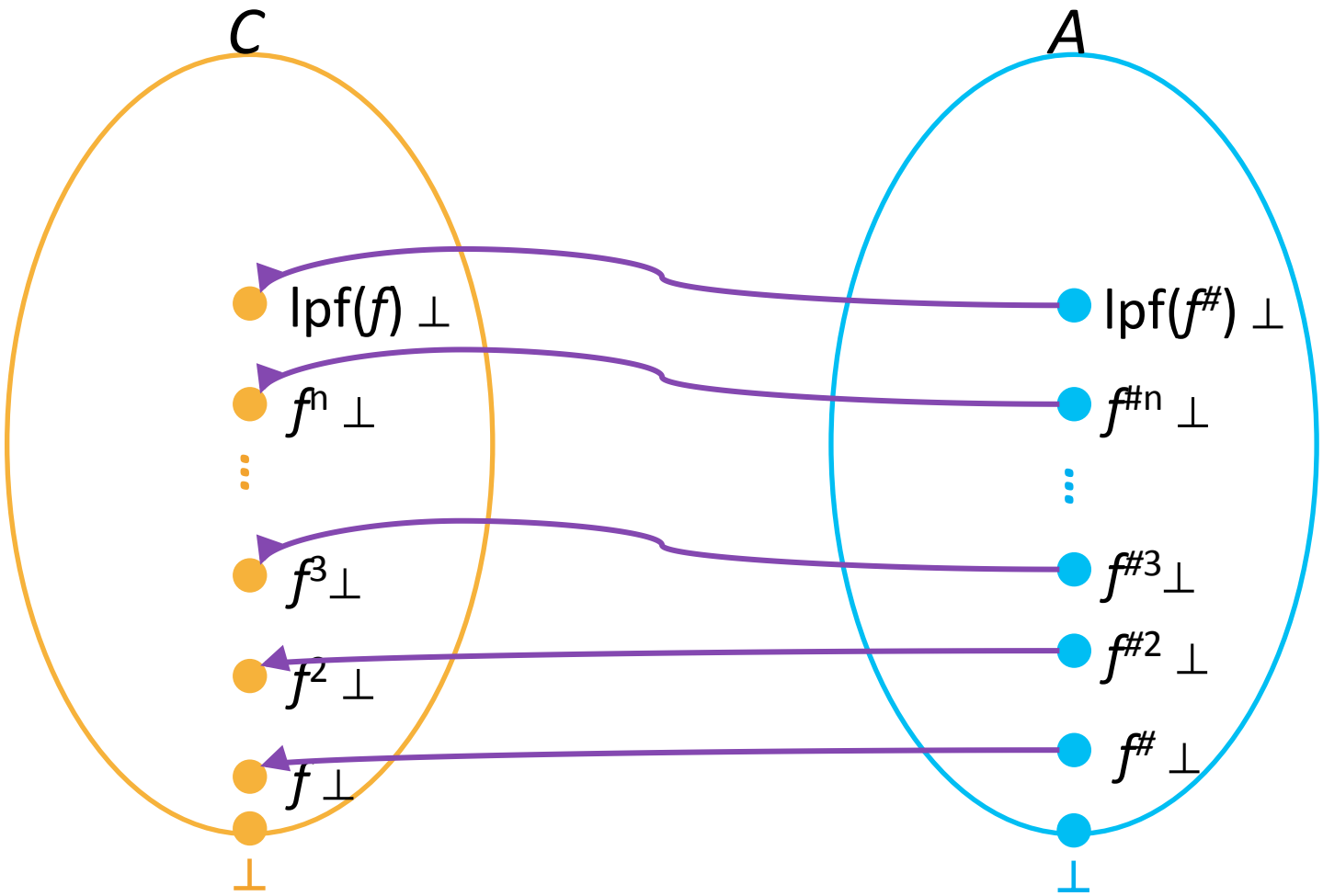


Backward complete transformer

$$\forall a: f(\gamma(a)) = \gamma(f^{\#}(a))$$



Global (backward) completeness



Three example analyses

- **V**ariable **E**qualities
- **C**onstant **P**ropagation
- **A**vailable **E**xpressions

Combining GC and Transformers

- Cartesian Product
  - $L_1 = (D_1, \sqsubseteq_1, \sqcup_1, \sqcap_1, \bot_1, \top_1)$
    $L_2 = (D_2, \sqsubseteq_2, \sqcup_2, \sqcap_2, \bot_2, \top_2)$
  - $\mathrm{Cart}(L_1, L_2) = (D_1 \times D_2, \sqsubseteq_{cart}, \sqcup_{cart}, \sqcap_{cart}, \bot_{cart}, \top_{cart})$
- Disjunctive completion
  - $L = (D, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$
  - $\mathrm{Disj}(L) = (2^D, \sqsubseteq_\vee, \sqcup_\vee, \sqcap_\vee, \bot_\vee, \top_\vee)$
- Relational Product
  - $\mathrm{Rel}(L_1, L_2) = \mathrm{Disj}(\mathrm{Cart}(L_1, L_2))$

Cartesian product of GCs

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$
- Cartesian Product

  $$GC^{C,A\times B} = (C, \alpha^{C,A\times B}, \gamma^{A\times B,C}, A\times B)$$

  – $\alpha^{C,A\times B}(X) = (\alpha^{C,A}(X), \alpha^{C,B}(X))$
  – $\gamma^{A\times B,C}(Y) = \gamma^{A,C}(X) \cap \gamma^{B,C}(X)$

Cartesian product transformers

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$ $F^A[st] : A \rightarrow A$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$ $F^B[st] : B \rightarrow B$
- Cartesian Product

  $GC^{C,A\times B} = (C, \alpha^{C,A\times B}, \gamma^{A\times B,C}, A\times B)$

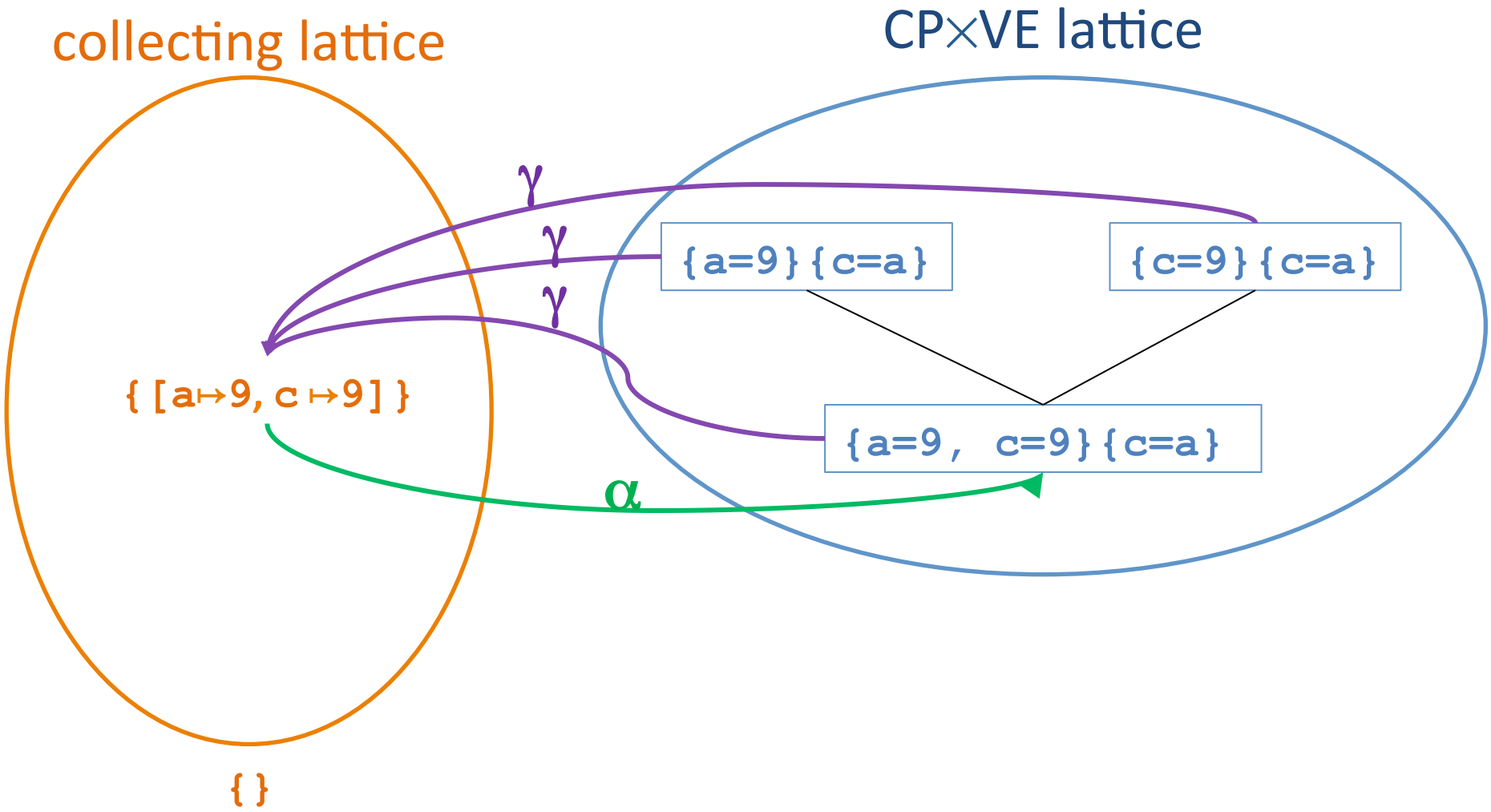
  – $\alpha^{C,A\times B}(X) = (\alpha^{C,A}(X), \alpha^{C,B}(X))$

  – $\gamma^{A\times B,C}(Y) = \gamma^{A,C}(X) \cap \gamma^{B,C}(X)$

- $F^{A\times B}[st](a, b) = (F^A[st]\ a, F^B[st]\ b)$

- Is this the best we can do?

Product vs. reduced product



Reduced product

- For two complete lattices

  $L_1 = (D_1, \sqsubseteq_1, \sqcup_1, \sqcap_1, \bot_1, \top_1)$

  $L_2 = (D_2, \sqsubseteq_2, \sqcup_2, \sqcap_2, \bot_2, \top_2)$

- Define the reduced poset

  $D_1 \sqcap D_2 = \{(d_1,d_2) \in D_1 \times D_2 \mid (d_1,d_2) = \alpha \circ \gamma\ (d_1,d_2)\ \}$

  $L_1 \sqcap L_2 = (D_1 \sqcap D_2, \sqsubseteq_{cart}, \sqcup_{cart}, \sqcap_{cart}, \bot_{cart}, \top_{cart})$

Transformers for Cartesian product

- Do we get the best transformer by applying component-wise transformer followed by reduction?
  - Unfortunately, no (what's the intuition?)
  - Can we do better?
  - Logical Product [Gulwani and Tiwari, PLDI 2006]

Combining Abstract Interpreters

Sumit Gulwani
Microsoft Research
sumitg@microsoft.com

Ashish Tiwari
SRI International
tiwari@csl.sri.com

Abstract

We present a methodology for automatically combining abstract interpreters over given lattices to construct an abstract interpreter for the combination of those lattices. This lends modularity to the process of design and implementation of abstract interpreters.

We define the notion of logical product of lattices. This kind of combination is more precise than the reduced product combination. We give algorithms to obtain the join operator and the existential quantification operator for the combined lattice from the corresponding operators of the individual lattices. We also give a bound on the number of steps required to reach a fixed point across loops during analysis over the combined lattice in terms of the corresponding bounds for the individual lattices. We prove that our combination methodology yields the most precise abstract interpretation operators over the logical product of lattices when the individual lattices are over theories that are convex, stably infinite, and disjoint.

We also present an interesting application of logical product wherein some lattices can be reduced to combination of other (unrelated) lattices with known abstract interpreters.

Categories and Subject Descriptors D.2.4 [*Software Engineering*]: Software/Program Verification; F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs; F.3.2 [*Logics and Meanings of Programs*]: Semantics of Programming Languages—Program analysis

General Terms Algorithms, Theory, Verification

Keywords Abstract Interpreter, Logical Product, Reduced Product, Nelson-Oppen Combination

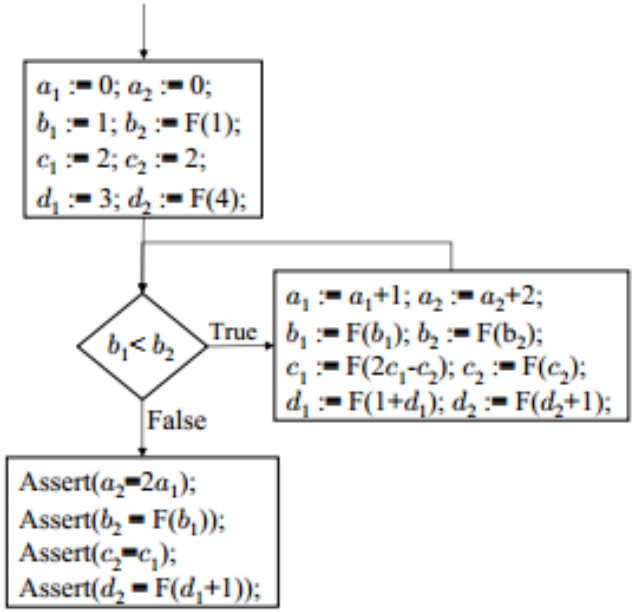


Figure 1. This program illustrates the difference between precision of performing analysis over *direct product*, *reduced product*, and *logical product* of the linear arithmetic lattice and uninterpreted functions lattice. Analysis over direct product can verify the first two assertions, while analysis over reduced product can verify the first three assertions. The analysis over logical product can verify all assertions. F denotes some function without any side-effects and can be modeled as an uninterpreted function for purpose of proving the assertions.

Logical product--

- Assume $A=(D,\ldots)$ is an abstract domain that supports two operations: for $x \in D$
  - inferEqualities(x) = { $a=b$ | $\gamma(x) \models a=b$ }
    returns a set of equalities between variables that are satisfied in all states given by x
  - refineFromEqualities(x, {$a=b$}) = y
    such that
    - $\gamma(x)=\gamma(y)$
    - $y \sqsubseteq x$

Example

Information loss example

```
if (…)          {}
  b := 5        {b=5}
else
  b := -5       {b=-5}
                {b=⊤}
if (b>0)
  b := b-5      {b=⊤}
else
  b := b+5      {b=⊤}
assert b==0     can't prove
```

Disjunctive completion of a lattice

- For a complete lattice
  $L = (D, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$
- Define the powerset lattice
  $L_\vee = (2^D, \sqsubseteq_\vee, \sqcup_\vee, \sqcap_\vee, \bot_\vee, \top_\vee)$
  $\sqsubseteq_\vee$ = ? $\sqcup_\vee$ = ? $\sqcap_\vee$ = ? $\bot_\vee$ = ? $\top_\vee$ = ?
- **Lemma:** $L_\vee$ is a complete lattice
- $L_\vee$ contains all subsets of D, which can be thought of as disjunctions of the corresponding predicates
- Define the disjunctive completion constructor
  $L_\vee$ = Disj(L)

Disjunctive completion for GCs

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$
- Disjunctive completion
  $GC^{C,P(A)} = (C, \alpha^{P(A)}, \gamma^{P(A)}, P(A))$
  - $\alpha^{C,P(A)}(X) = ?$
  - $\gamma^{P(A),C}(Y) = ?$

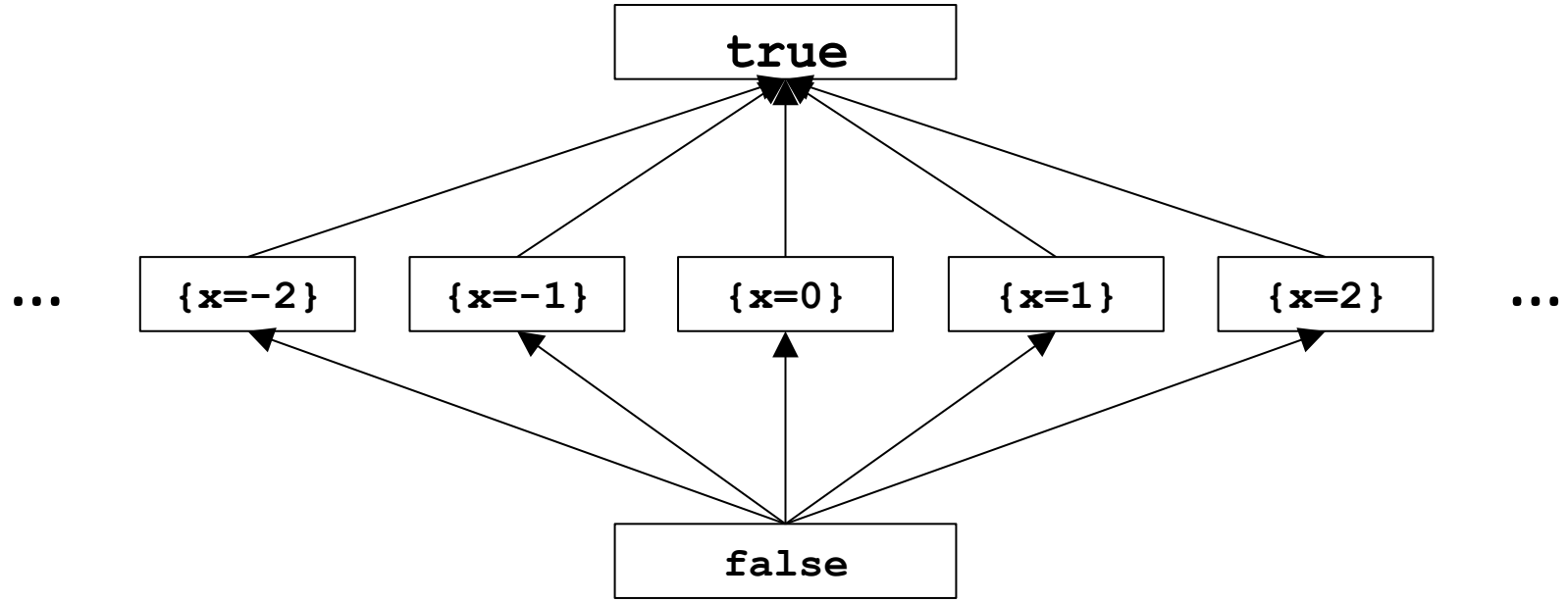
Disjunctive completion for GCs

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$
- Disjunctive completion
  $GC^{C,P(A)} = (C, \alpha^{P(A)}, \gamma^{P(A)}, P(A))$
  - $\alpha^{C,P(A)}(X) = \{\alpha^{C,A}(\{x\}) \mid x \in X\}$
  - $\gamma^{P(A),C}(Y) = \cup\{\gamma^{P(A)}(y) \mid y \in Y\}$
- What about transformers?

Information loss example

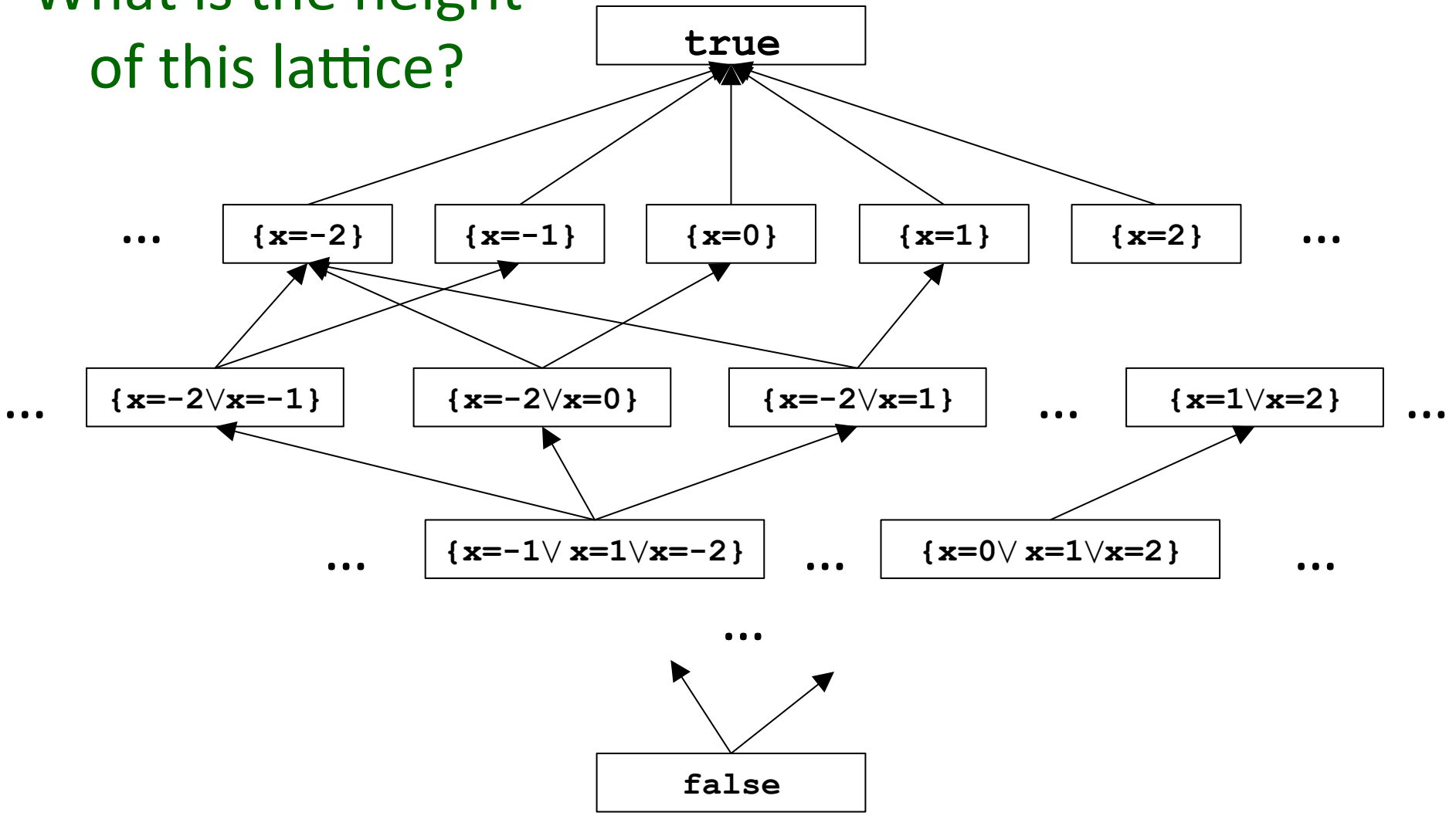
```
if (…)          {}
  b := 5        {b=5}
else
  b := -5       {b=-5}
                {b=5 ∨ b=-5}
if (b>0)
  b := b-5      {b=0}
else
  b := b+5      {b=0}
assert b==0     proved
```

The base lattice CP



The disjunctive completion of CP

What is the height of this lattice?



Taming disjunctive completion

- Disjunctive completion is very precise
  - Maintains correlations between states of different analyses
  - Helps handle conditions precisely
  - But very expensive – number of abstract states grows exponentially
  - May lead to non-termination
- Base analysis (usually product) is less precise
  - Analysis terminates if the analyses of each component terminates
- How can we combine them to get more precision yet ensure termination and state explosion?

Taming disjunctive completion

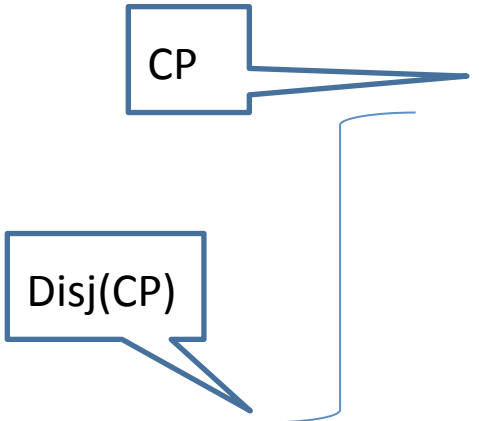
- Use different abstractions for different program locations
  - At loop heads use coarse abstraction (base)
  - At other points use disjunctive completion
- Termination is guaranteed (by base domain)
- Precision increased **inside** loop body

With Disj(CP)

```
while (…) {
  if (…)
    b := 5
  else
    b := -5

  if (b>0)
    b := b-5
  else
    b := b+5
  assert b==0
}
```

With tamed Disj(CP)




```
while (…) {
  if (…)
    b := 5
  else
    b := -5

  if (b>0)
    b := b-5
  else
    b := b+5
  assert b==0
}
```

Reducing disjunctive elements

- A disjunctive set X may contain within it an ascending chain $Y=a \sqsubseteq b \sqsubseteq c\ldots$
- We only need max(Y) – remove all elements below

Relational product of lattices

- $L_1 = (D_1, \sqsubseteq_1, \sqcup_1, \sqcap_1, \bot_1, \top_1)$
  $L_2 = (D_2, \sqsubseteq_2, \sqcup_2, \sqcap_2, \bot_2, \top_2)$
- $L_{rel} = (2^{D_1 \times D_2}, \sqsubseteq_{rel}, \sqcup_{rel}, \sqcap_{rel}, \bot_{rel}, \top_{rel})$
  as follows:
  - L_{rel} = Disj(Cart(L_1, L_2))
- **Lemma:** *L* is a complete lattice
- What does it buy us?
  - How is it relative to Cart(Disj(L_1), Disj(L_2))?
- What about transformers?

Relational product of GCs

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$
- Relational Product
  $GC^{C,P(A\times B)} = (C, \alpha^{C,P(A\times B)}, \gamma^{P(A\times B),C}, P(A\times B))$
  - $\alpha^{C,P(A\times B)}(X) = ?$
  - $\gamma^{P(A\times B),C}(Y) = ?$

Relational product of GCs

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$
- Relational Product
  $GC^{C,P(A\times B)} = (C, \alpha^{C,P(A\times B)}, \gamma^{P(A\times B),C}, P(A\times B))$
  - $\alpha^{C,P(A\times B)}(X) = \{(\alpha^{C,A}(\{x\}), \alpha^{C,B}(\{x\})) \mid x\in X\}$
  - $\gamma^{P(A\times B),C}(Y) = \cup\{\gamma^{A,C}(y_A) \cap \gamma^{B,C}(y_B) \mid (y_A,y_B)\in Y\}$

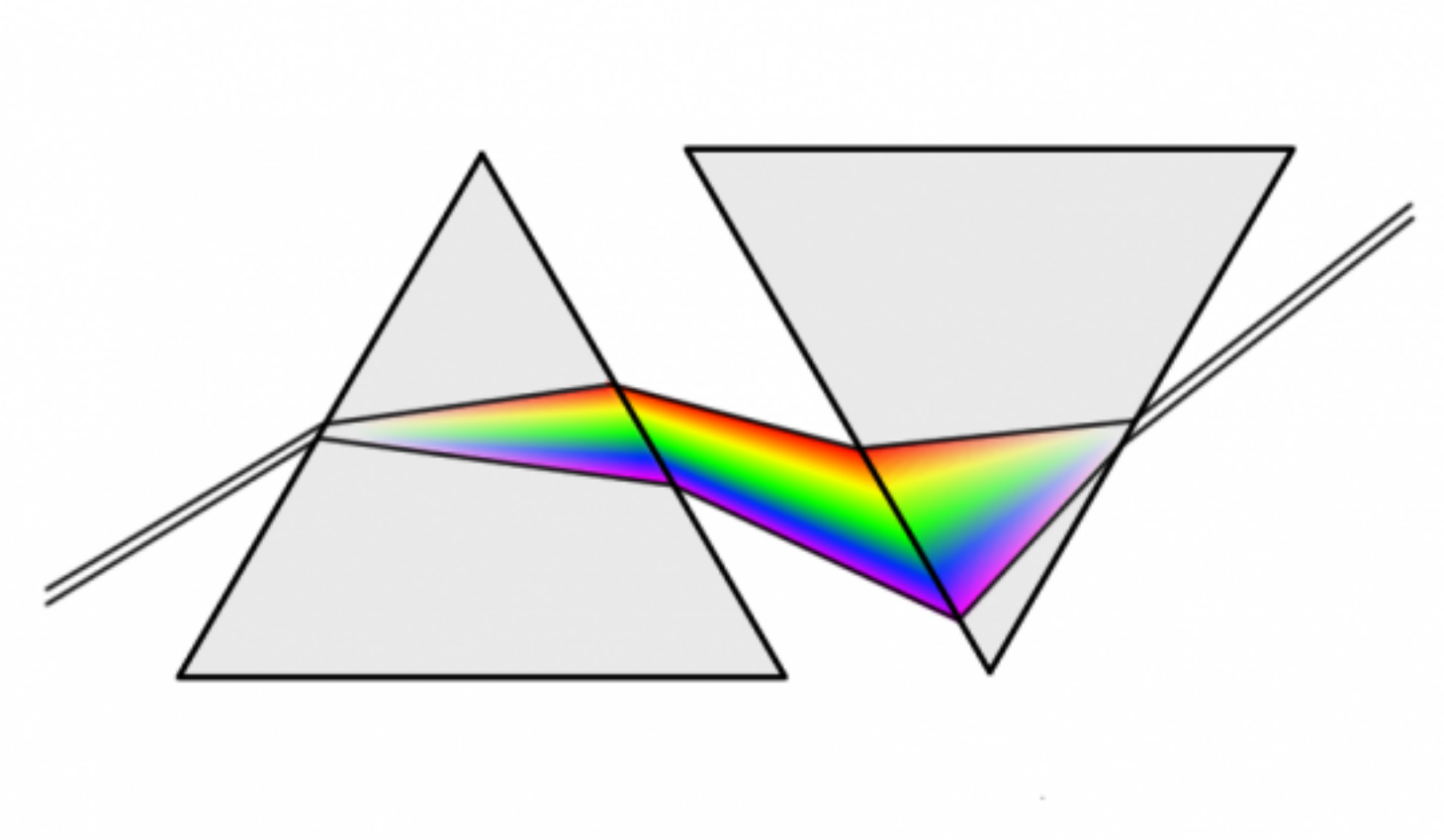
Function space

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$
- Denote the set of monotone functions from A to B by $A \longrightarrow B$
- Define $\sqcup$ for elements of $A \longrightarrow B$ as follows
  $(a_1, b_1) \sqcup (a_2, b_2)$ = if $a_1=a_2$ then $\{(a_1, b_1 \sqcup_B b_1)\}$
  else $\{(a_1, b_1), (a_2, b_2)\}$
- Reduced cardinal power
  $GC^{C,A\rightarrow B} = (C, \alpha^{C,A\rightarrow B}, \gamma^{A\rightarrow B,C}, A \longrightarrow B)$
  - $\alpha^{C,A\rightarrow B}(X) = \sqcup\{(\alpha^{C,A}(\{x\}), \alpha^{C,B}(\{x\})) \mid x \in X\}$
  - $\gamma^{A\rightarrow B,C}(Y) = \cup\{\gamma^{A,C}(y_A) \cap \gamma^{B,C}(y_B) \mid (y_A, y_B) \in Y\}$
- Useful when A is small and B is much larger
  - E.g., typestate verification

Function space

- $GC^{C,A}=(C, \alpha^{C,A}, \gamma^{A,C}, A)$
  $GC^{C,B}=(C, \alpha^{C,B}, \gamma^{B,C}, B)$
- Denote the set of monotone functions from A to B by $A \longrightarrow B$
- Define $\sqcup$ for elements of $A \longrightarrow B$ as follows
  $(a_1, b_1) \sqcup (a_2, b_2)$ = if $a_1=a_2$ then $\{(a_1, b_1 \sqcup_B b_1)\}$
  else $\{(a_1, b_1), (a_2, b_2)\}$
- Reduced cardinal power
  $GC^{C,A\rightarrow B} = (C, \alpha^{C,A\rightarrow B}, \gamma^{A\rightarrow B,C}, A \longrightarrow B)$
  - $\alpha^{C,A\rightarrow B}(X) = \sqcup\{(\alpha^{C,A}(\{x\}), \alpha^{C,B}(\{x\})) \mid x \in X\}$
  - $\gamma^{A\rightarrow B,C}(Y) = \cup\{\gamma^{A,C}(y_A) \cap \gamma^{B,C}(y_B) \mid (y_A, y_B) \in Y\}$
- Useful when A is small and B is much larger
  - E.g., typestate verification

Widening/Narrowing



How can we prove this automatically?

```
public void loopExample() {
    int x = 7;
    while (x < 1000) {
        ++x;
    }
    if (!(x == 1000))
        error("Unable to prove x == 1000!");
}
```

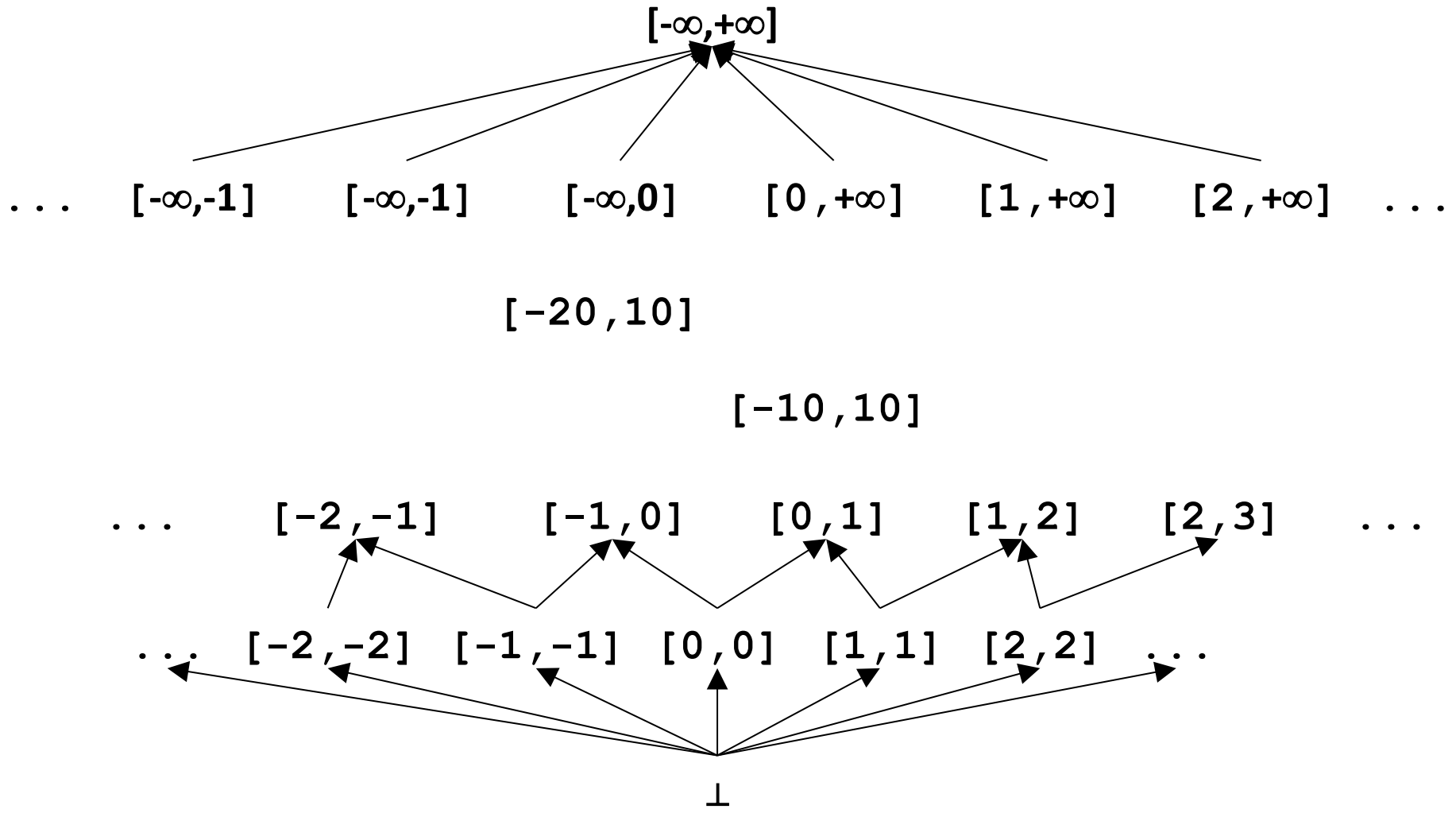
RelProd(CP, VE)

```
Reached fixed-point after 19 iterations.
Solution = {
  V[0] : (true, true)
  V[1] : (true, true)
  V[2] : (x=7, true)
  V[3] : (x=7, true)
  V[4] : (true, true)
  V[7] : (true, true)
  V[5] : (true, true)
  V[6] : (true, true)
  V[8] : (true, true)
  V[9] : (true, true)
  V[10] : (true, true)
  V[12] : (true, true)
  V[11] : (true, true)
}
1 possible errors found.
```

Intervals domain

- One of the simplest numerical domains
- Maintain for each variable *x* an interval [*L*,*H*]
  - *L* is either an integer of -∞
  - *H* is either an integer of +∞
- A (non-relational) numeric domain

Intervals lattice for variable *x*



Intervals lattice for variable *x*

- $D^{int}[x] = \{ (L,H) \mid L \in -\infty,\mathbf{Z} \text{ and } H \in \mathbf{Z},+\infty \text{ and } L \leq H \}$
- $\bot$
- $\top = [-\infty,+\infty]$
- $\sqsubseteq$ = ?
  - $[1,2] \sqsubseteq [3,4]$?
  - $[1,4] \sqsubseteq [1,3]$?
  - $[1,3] \sqsubseteq [1,4]$?
  - $[1,3] \sqsubseteq [-\infty,+\infty]$?
- What is the lattice height?

Intervals lattice for variable *x*

- $D^{int}[x] = \{ (L,H) \mid L \in -\infty,\mathbf{Z} \text{ and } H \in \mathbf{Z},+\infty \text{ and } L \leq H \}$
- $\bot$
- $\top = [-\infty,+\infty]$
- $\sqsubseteq$ = ?
  - $[1,2] \sqsubseteq [3,4]$ no
  - $[1,4] \sqsubseteq [1,3]$ no
  - $[1,3] \sqsubseteq [1,4]$ yes
  - $[1,3] \sqsubseteq [-\infty,+\infty]$ yes
- What is the lattice height? Infinite

Joining/meeting intervals

- $[a,b] \sqcup [c,d] = ?$
  - $[1,1] \sqcup [2,2] = ?$
  - $[1,1] \sqcup [2, +\infty] = ?$
- $[a,b] \sqcap [c,d] = ?$
  - $[1,2] \sqcap [3,4] = ?$
  - $[1,4] \sqcap [3,4] = ?$
  - $[1,1] \sqcap [1,+\infty] = ?$
- Check that indeed $x \sqsubseteq y$ if and only if $x \sqcup y = y$

Joining/meeting intervals

- $[a,b] \sqcup [c,d] = [\min(a,c), \max(b,d)]$
  - $[1,1] \sqcup [2,2] = [1,2]$
  - $[1,1] \sqcup [2,+\infty] = [1,+\infty]$
- $[a,b] \sqcap [c,d] = [\max(a,c), \min(b,d)]$ if a proper interval and otherwise $\bot$
  - $[1,2] \sqcap [3,4] = \bot$
  - $[1,4] \sqcap [3,4] = [3,4]$
  - $[1,1] \sqcap [1,+\infty] = [1,1]$
- Check that indeed $x \sqsubseteq y$ if and only if $x \sqcup y = y$

Interval domain for programs

- $D^{int}[x] = \{ (L,H) \mid L \in -\infty,\mathbf{Z} \text{ and } H \in \mathbf{Z},+\infty \text{ and } L \leq H \}$
- For a program with variables $Var=\{x_1,\ldots,x_k\}$
- $D^{int}[Var] = ?$

Interval domain for programs

- $D^{int}[x] = \{ (L,H) \mid L \in -\infty,\mathbf{Z} \text{ and } H \in \mathbf{Z},+\infty \text{ and } L \leq H \}$
- For a program with variables $Var=\{x_1,\ldots,x_k\}$
- $D^{int}[Var] = D^{int}[x_1] \times \ldots \times D^{int}[x_k]$
- How can we represent it in terms of formulas?

Interval domain for programs

- $D^{int}[x] = \{ (L,H) \mid L \in -\infty, \mathbf{Z} \text{ and } H \in \mathbf{Z}, +\infty \text{ and } L \leq H \}$
- For a program with variables $Var = \{x_1, \ldots, x_k\}$
- $D^{int}[Var] = D^{int}[x_1] \times \ldots \times D^{int}[x_k]$
- How can we represent it in terms of formulas?
  - Two types of factoids $x \geq c$ and $x \leq c$
  - Example: $S = \wedge\{x \geq 9, y \geq 5, y \leq 10\}$
  - Helper operations
    - $c + +\infty = +\infty$
    - remove(S, x) = S without any x-constraints
    - lb(S, x) = k if $k \leq x \leq m$
    - ub(S,x) = m if $k \leq x \leq m$

Assignment transformers

- $[\![x := c]\!]^\# S = ?$
- $[\![x := y]\!]^\# S = ?$
- $[\![x := y+c]\!]^\# S = ?$
- $[\![x := y+z]\!]^\# S = ?$
- $[\![x := y*c]\!]^\# S = ?$
- $[\![x := y*z]\!]^\# S = ?$

Assignment transformers

- $[\![x := c]\!]^\# S = \text{remove}(S,x) \cup \{x \geq c, x \leq c\}$
- $[\![x := y]\!]^\# S = \text{remove}(S,x) \cup \{x \geq \text{lb}(S,y), x \leq \text{ub}(S,y)\}$
- $[\![x := y+c]\!]^\# S = \text{remove}(S,x) \cup \{x \geq \text{lb}(S,y)+c, x \leq \text{ub}(S,y)+c\}$
- $[\![x := y+z]\!]^\# S = \text{remove}(S,x) \cup \{x \geq \text{lb}(S,y)+\text{lb}(S,z),$
  $x \leq \text{ub}(S,y)+\text{ub}(S,z)\}$
- $[\![x := y*c]\!]^\# S = \text{remove}(S,x) \cup \text{if } c>0 \{x \geq \text{lb}(S,y)*c, x \leq \text{ub}(S,y)*c\}$
  else $\{x \geq \text{ub}(S,y)*-c, x \leq \text{lb}(S,y)*-c\}$
- $[\![x := y*z]\!]^\# S = \text{remove}(S,x) \cup ?$

assume transformers

- $[\![\textbf{assume}\ x{=}c]\!]^{\#}\ S$ = ?
- $[\![\textbf{assume}\ x{<}c]\!]^{\#}\ S$ = ?
- $[\![\textbf{assume}\ x{=}y]\!]^{\#}\ S$ = ?
- $[\![\textbf{assume}\ x{\neq}c]\!]^{\#}\ S$ = ?

assume transformers

- $[\![\textbf{assume}\ x{=}c]\!]^{\#}\ S = S \sqcap \{x \geq c, x \leq c\}$
- $[\![\textbf{assume}\ x{<}c]\!]^{\#}\ S = S \sqcap \{x \leq c\text{-}1\}$
- $[\![\textbf{assume}\ x{=}y]\!]^{\#}\ S = S \sqcap \{x \geq \mathrm{lb}(S,y), x \leq \mathrm{ub}(S,y)\}$
- $[\![\textbf{assume}\ x{\neq}c]\!]^{\#}\ S = ?$

assume transformers

- $[\![\textbf{assume}\ x{=}c]\!]^\#\ S = S \sqcap \{x \geq c, x \leq c\}$
- $[\![\textbf{assume}\ x{<}c]\!]^\#\ S = S \sqcap \{x \leq c\text{-}1\}$
- $[\![\textbf{assume}\ x{=}y]\!]^\#\ S = S \sqcap \{x \geq \mathrm{lb}(S,y), x \leq \mathrm{ub}(S,y)\}$
- $[\![\textbf{assume}\ x{\neq}c]\!]^\#\ S = (S \sqcap \{x \leq c\text{-}1\}) \sqcup (S \sqcap \{x \geq c{+}1\})$

Chaotic iteration

- Input:
  - A cpo $L = (D, \sqsubseteq, \sqcup, \bot)$ satisfying ACC
  - $L^n = L \times L \times \ldots \times L$
  - A monotone function $f : D^n \rightarrow D^n$
  - A system of equations $\{ X[i] \mid f(X) \mid 1 \leq i \leq n \}$
- Output: lfp(f)
- A worklist-based algorithm

```
for i:=1 to n do
   X[i] := ⊥
WL = {1,…,n}
while WL ≠ ∅ do
   j := pop WL // choose index non-deterministically
   N := F[i](X)
   if N ≠ X[i] then
      X[i] := N
      add all the indexes that directly depend on i to WL
      (X[j] depends on X[i] if F[j] contains X[i])
return X
```

Concrete semantics equations

```
public void loopExample() {
R[0]  int x = 7; R[1]
R[2]  while (x < 1000) {
R[3]      ++x; R[4]
      }
R[5]  if (!(x == 1000))
R[6]      error("Unable to prove x == 1000!");
}
```

- R[0] = {**x**∈**Z**}
  R[1] = ⟦**x**:=**7**⟧
  R[2] = R[1] ∪ R[4]
  R[3] = R[2] ∩ {s | s(x) < 1000}
  R[4] = ⟦**x**:=**x+1**⟧ R[3]
  R[5] = R[2] ∩ {s | s(x) ≥ 1000}
  R[6] = R[5] ∩ {s | s(x) ≠ 1001}

Abstract semantics equations

```
public void loopExample() {
R[0]  int x = 7; R[1]
R[2]  while (x < 1000) {
R[3]       ++x; R[4]
      }
R[5]  if (!(x == 1000))
R[6]       error("Unable to prove x == 1000!");
}
```

- $R[0] = \alpha(\{\mathbf{x} \in \mathbf{Z}\})$
  $R[1] = [\![\mathbf{x := 7}]\!]^{\#}$
  $R[2] = R[1] \sqcup R[4]$
  $R[3] = R[2] \sqcap \alpha(\{s \mid s(x) < 1000\})$
  $R[4] = [\![\mathbf{x := x+1}]\!]^{\#}\ R[3]$
  $R[5] = R[2] \sqcap \alpha(\{s \mid s(x) \geq 1000\})$
  $R[6] = R[5] \sqcap \alpha(\{s \mid s(x) \geq 1001\}) \sqcup R[5] \sqcap \alpha(\{s \mid s(x) \leq 999\})$

Abstract semantics equations

```
public void loopExample() {
R[0]  int x = 7; R[1]
R[2]  while (x < 1000) {
R[3]      ++x; R[4]
      }
R[5]  if (!(x == 1000))
R[6]      error("Unable to prove x == 1000!");
}
```

- R[0] = ⊤
  R[1] = [7,7]
  R[2] = R[1] ⊔ R[4]
  R[3] = R[2] ⊓ [-∞,999]
  R[4] = R[3] + [1,1]
  R[5] = R[2] ⊓ [1000,+∞]
  R[6] = R[5] ⊓ [999,+∞] ⊔ R[5] ⊓ [1001,+∞]

Too many iterations to converge

```
Iteration 3981: processing V[8] = Interval[x==1000](V[6]) // if x == 1000 goto return
              V[8] : false
              V[6] : and(x=1000)
              V[8]' : and(x=1000)
              Adding [V[12] = Join_IntervalDomain(V[8], V[10]) // return]
              workSet = {V[12]}
Iteration 3982: processing V[12] = Join_IntervalDomain(V[8], V[10]) // return
              V[12] : false
              V[8] : and(x=1000)
              V[10] : false
              V[12]' : and(x=1000)
              Adding [V[11] = V[12] // return]
              workSet = {V[11]}
Iteration 3983: processing V[11] = V[12] // return
              V[11] : false
              V[12] : and(x=1000)
              V[11]' : and(x=1000)
              Adding []
Reached fixed-point after 3983 iterations.
Solution = {
  V[0] : true
  V[1] : true
  V[2] : and(x=7)
  V[3] : and(x=7)
  V[4] : and(8<=x<=1000)
  V[7] : and(7<=x<=1000)
  V[5] : and(7<=x<=999)
  V[6] : and(x=1000)
  V[8] : and(x=1000)
  V[9] : false
  V[10] : false
  V[12] : and(x=1000)
  V[11] : and(x=1000)
}
0 possible errors found.
Writing to sootOutput\IntervalExample.jimple
Soot finished on Wed Jun 12 06:24:14 IDT 2013
Soot has run for 0 min. 1 sec.
```

How many iterations for this one?

```
public void loopExample2(int y) {
    int x = 7;
    if (x < y) {
        while (x < y) {
            ++x;
        }

        if (x != y)
            error("Unable to prove x = y!");
    }
}
```

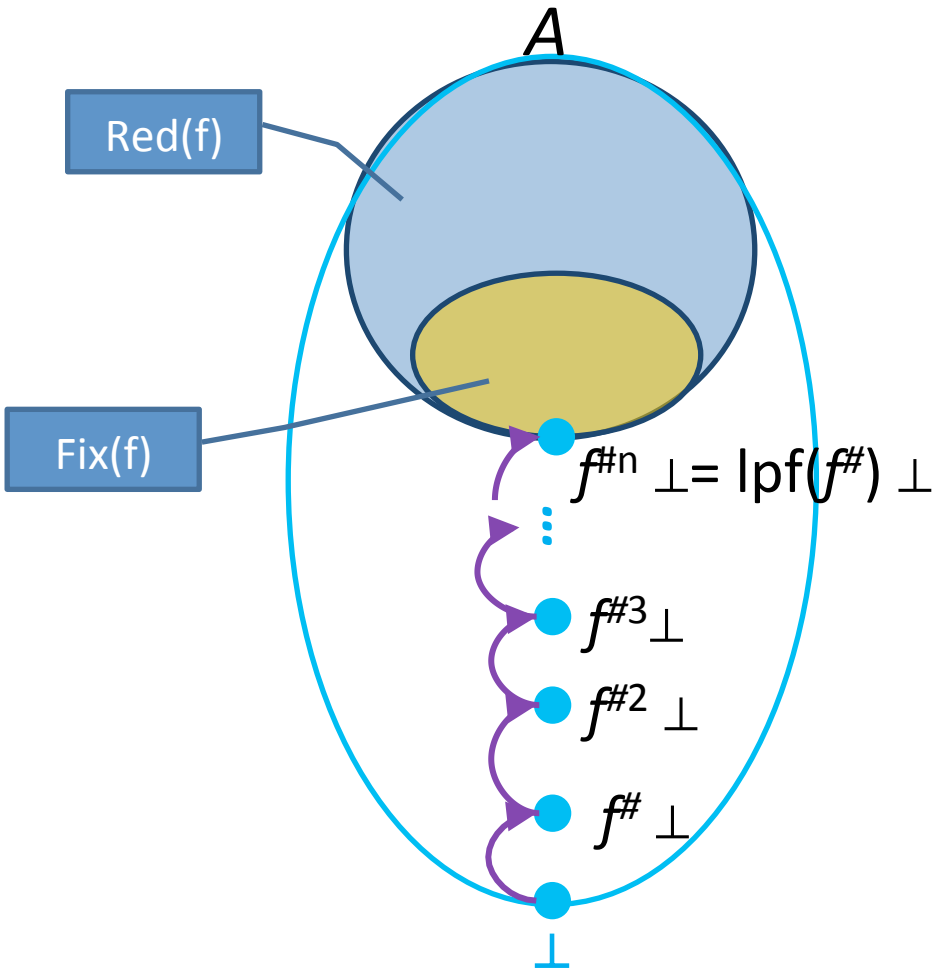
Widening

- Introduce a new binary operator to ensure termination
  - A kind of extrapolation
- Enables static analysis to use infinite height lattices
  - Dynamically adapts to given program
- Tricky to design
- Precision less predictable then with finite-height domains (widening non-monotone)

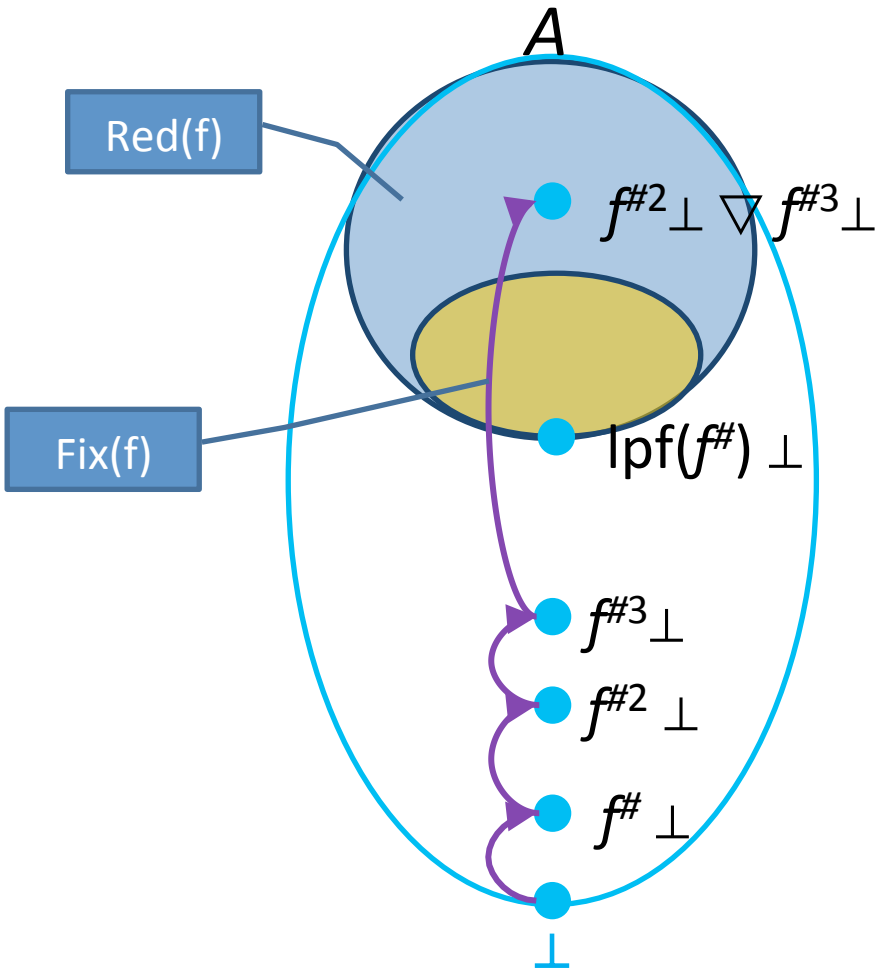
Formal definition

- For all elements $d_1 \sqcup d_2 \sqsubseteq d_1 \bigtriangledown d_2$
- For all ascending chains $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ the following sequence is finite (stabilizes)
  - $y_0 = d_0$
  - $y_{i+1} = y_i \bigtriangledown d_{i+1}$
- For a monotone function $f : D \rightarrow D$ define
  - $x_0 = \bot$
  - $x_{i+1} = x_i \bigtriangledown f(x_i)$
- Theorem:
  - There exits k such that $x_{k+1} = x_k$
  - $x_k \in \text{Red}(f) = \{ d \mid d \in D \text{ and } f(d) \sqsubseteq d \}$

Analysis with finite-height lattice



Analysis with widening



Widening for Intervals Analysis

- $\perp \triangledown [c, d] = [c, d]$
- $[a, b] \triangledown [c, d] = [$
  if $a \leq c$
  then a
  else $-\infty$,
  if $b \geq d$
  then b
  else ∞

Semantic equations with widening

```
public void loopExample() {
R[0]  int x = 7; R[1]
R[2]  while (x < 1000) {
R[3]       ++x; R[4]
      }
R[5]  if (!(x == 1000))
R[6]       error("Unable to prove x == 1000!");
}
```

- $R[0] = \top$
  $R[1] = [7,7]$
  $R[2] = R[1] \sqcup R[4]$
  $R[2.1] = R[2.1] \triangledown R[2]$
  $R[3] = R[2.1] \sqcap [-\infty,999]$
  $R[4] = R[3] + [1,1]$
  $R[5] = R[2] \sqcap [1001,+\infty]$
  $R[6] = R[5] \sqcap [999,+\infty] \sqcup R[5] \sqcap [1001,+\infty]$

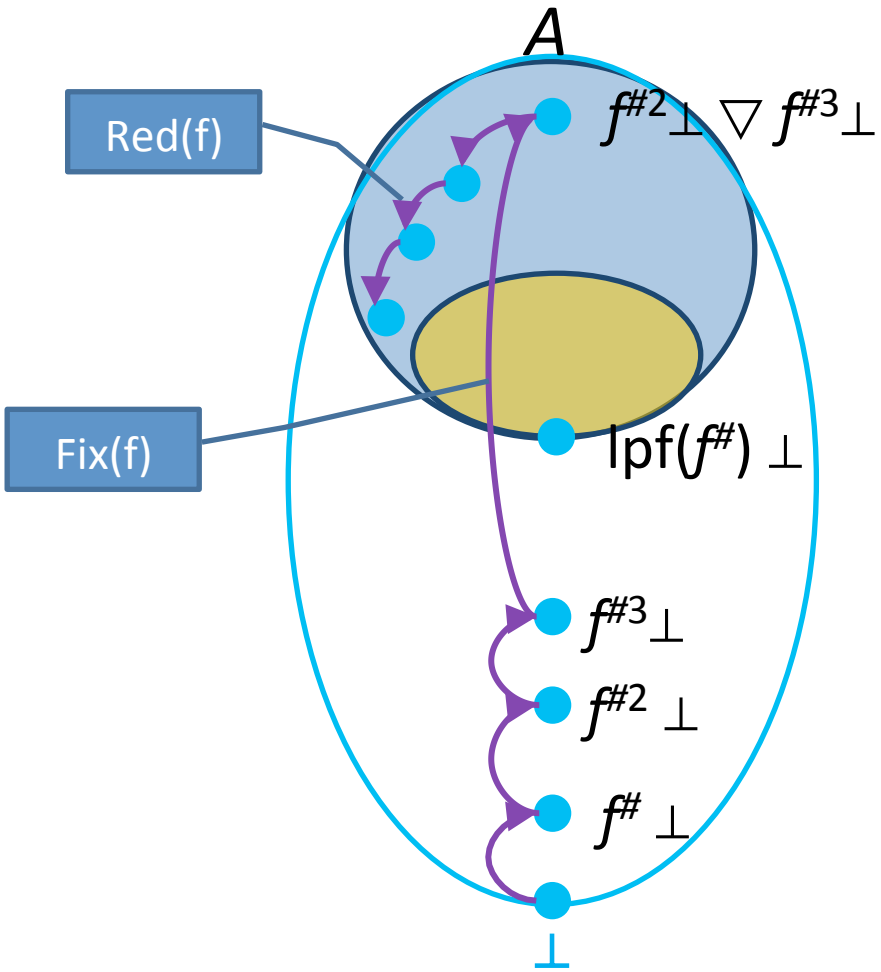
Non monotonicity of widening

- $[0,1] \triangledown [0,2] = ?$
- $[0,2] \triangledown [0,2] = ?$

Non monotonicity of widening

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

Analysis with narrowing



Formal definition of narrowing

- Improves the result of widening
- $y \sqsubseteq x \Rightarrow y \sqsubseteq (x \triangle y) \sqsubseteq x$
- For all decreasing chains $x_0 \sqsupseteq x_1 \sqsupseteq \ldots$ the following sequence is finite (stabilizes)
  - $y_0 = x_0$
  - $y_{i+1} = y_i \triangle x_{i+1}$
- For a monotone function f: $D \rightarrow D$ and $x_k \in \mathrm{Red}(f) = \{ d \mid d \in D \text{ and } f(d) \sqsubseteq d \}$ define
  - $y_0 = x$
  - $y_{i+1} = y_i \triangle f(y_i)$
- Theorem:
  - There exits k such that $y_{k+1} = y_k$

Formal definition of narrowing

- Improves the result of widening
- $y \sqsubseteq x \Rightarrow y \sqsubseteq (x \triangle y) \sqsubseteq x$
- For all decreasing chains $x_0 \sqsupseteq x_1 \sqsupseteq \ldots$ the following sequence is finite
  - $y_0 = x_0$
  - $y_{i+1} = y_i \triangle x_{i+1}$
- For a monotone function f: $D \rightarrow D$ and $x_k \in \mathrm{Red}(f) = \{ d \mid d \in D \text{ and } f(d) \sqsubseteq d \}$ define
  - $y_0 = x$
  - $y_{i+1} = y_i \triangle f(y_i)$
- Theorem:
  - There exits k such that $y_{k+1} = y_k$
  - $y_k \in \mathrm{Red}(f) = \{ d \mid d \in D \text{ and } f(d) \sqsubseteq d \}$

Narrowing for Interval Analysis

- $[a, b] \triangle \perp = [a, b]$
- $[a, b] \triangle [c, d] = [$

  if $a = -\infty$

  then c

  else a,

  if $b = \infty$

  then d

  else b

  $]$

Semantic equations with narrowing

```
public void loopExample() {
    R[0]                R[1]
    int x = 7;
    R[2]
    while (x < 1000) {
    R[3]                R[4]
        ++x;
    }
    R[5]
    if (!(x == 1000))
    R[6]
        error("Unable to prove x == 1000!");
}
```

- R[0] = ⊤
  R[1] = [7,7]
  R[2] = R[1] ⊔ R[4]
  R[2.1] = R[2.1] △ R[2]
  R[3] = R[2.1] ⊓ [-∞,999]
  R[4] = R[3]+[1,1]
  R[5] = $R[2]^{\#}$ ⊓ [1000,+∞]
  R[6] = R[5] ⊓ [999,+∞] ⊔ R[5] ⊓ [1001,+∞]

Analysis with widening/narrowing

- Two phases
  - Phase 1: analyze with widening until converging
  - Phase 2: use values to analyze with narrowing

```
public void loopExample() {
    int x = 7;
    while (x < 1000) {
        ++x;
    }
    if (!(x == 1000))
        error("Unable to prove x == 1000!");
}
```

Phase 1:

$R[0] = \top$

$R[1] = [7,7]$

$R[2] = R[1] \sqcup R[4]$

$R[2.1] = R[2.1] \triangledown R[2]$

$R[3] = R[2.1] \sqcap [-\infty,999]$

$R[4] = R[3] + [1,1]$

$R[5] = R[2] \sqcap [1001,+\infty]$

$R[6] = R[5] \sqcap [999,+\infty] \sqcup R[5] \sqcap [1001,+\infty]$

Phase 2:

$R[0] = \top$

$R[1] = [7,7]$

$R[2] = R[1] \sqcup R[4]$

$R[2.1] = R[2.1] \triangle R[2]$

$R[3] = R[2.1] \sqcap [-\infty,999]$

$R[4] = R[3]+[1,1]$

$R[5] = R[2]^{\#} \sqcap [1000,+\infty]$

$R[6] = R[5] \sqcap [999,+\infty] \sqcup R[5] \sqcap [1001,+\infty]$

Analysis with widening/narrowing

```
Reached fixed-point after 23 iterations.
Solution = {
  V[0] : true
  V[1] : true
  V[2] : and(x=7)
  V[3] : and(x=7)
  V[4] : and(8<=x<=1000)
  V[7] : and(7<=x<=1000)
  V[8] : and(x>=7)
  V[5] : and(7<=x<=999)
  V[6] : and(x>=1000)
  V[9] : and(x=1000)
  V[10] : and(x>=1001)
  V[11] : and(x>=1001)
  V[13] : and(x>=1000)
  V[12] : and(x>=1000)
}

Starting chaotic iteration: narrowing phase...
              workSet = {V[0], V[1], V[2], V[3], V[4], V[7], V[8], V[5], V[6], V[9], V[10], V[11], V[13], V[12]}
Iteration 24: processing V[0] = true // this := @this: IntervalExample
              V[0] : true
              V[0]' : true
              workSet = {V[12], V[1], V[2], V[3], V[4], V[7], V[8], V[5], V[6], V[9], V[10], V[11], V[13]}
```

Analysis results widening/narrowing

```
Iteration 44: processing V[1] = AssignTopTransformer(V[0]) // this := @this: IntervalExample
              V[1] : true
              V[0] : true
              V[1]' : true
Reached fixed-point after 44 iterations.
Solution = {
  V[0] : true
  V[1] : true
  V[2] : and(x=7)
  V[3] : and(x=7)
  V[4] : and(8<=x<=1000)
  V[7] : and(7<=x<=1000)
  V[8] : and(7<=x<=1000)
  V[5] : and(7<=x<=999)
  V[6] : and(x=1000)
  V[9] : and(x=1000)
  V[10] : false
  V[11] : false
  V[13] : and(x=1000)
  V[12] : and(x=1000)
}
0 possible errors found.
Writing to sootOutput\IntervalExample.jimple
Soot finished on Wed Jun 12 06:47:24 IDT 2013
Soot has run for 0 min. 0 sec.
```