Program Analysis and Verification

0368-4479

Noam Rinetzky

Lecture 8: Abstract Interpretation

Slides credit: Roman Manevich, Mooly Sagiv, Eran Yahav

Abstract Interpretation [Cousot'77]

- Mathematical framework for approximating semantics (aka abstraction)
  - Allows designing sound static analysis algorithms
    - Usually compute by iterating to a fixed-point
  - Computes (loop) invariants
    - Can be interpreted as axiomatic verification assertions
    - Generalizes Hoare Logic & WP / SP calculus

Required knowledge

- ✓ Domain theory
- ✓ Collecting semantics
- ✓ Abstract semantics (over lattices)
- ✓ Algorithm to compute abstract semantics (chaotic iteration)
- Connection between collecting semantics and abstract semantics
- Abstract transformers

Recap

Posets

- Poset: A set (D, $\sqsubseteq$) equipped with a partial order
  – Poset = Partially-ordered set
  – E.g., D =2^S, $\sqsubseteq$ = $\subseteq$
- Join: Least upper bound ($\sqcup$)
  – d is an upper bound on $X \subseteq D$ if $\forall d' \in X. d' \sqsubseteq d$
  – d is the LUB on $X \subseteq D$ if
    - d is a UB on X
    - If $d''$is an UB on X then $d \sqsubseteq d''$
- Meet: Greatest lower bound ($\sqcap$)

Chains

- A chain is a countable increasing sequence $\langle x_i \rangle = x_0 \sqsubseteq x_1 \sqsubseteq \ldots$ such that $x_i \in X$

- The least upper bound on $\langle x_i \rangle$ in X is the LUB in X of its elements

Complete Partial Orders

- Complete partial order (cpo): A partial order L= (D, $\sqsubseteq$) is complete if every **chain** in D has a least upper bound also in D
  - (Naturals, $\leq$) is not a CPO
  - (Naturals, ∪ $\{\infty\}$, $\leq$) is a CPO
- A cpo with a least ("bottom") element $\perp$ is a pointed cpo (pcpo)
- *L* satisfies the ascending chain condition (ACC) if every ascending chain eventually stabilizes:

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

  - Hence, L is a CPO

Constructing (P)CPOs

- If D and E are (pointed) cpos, then so is their
- Cartesian product:
  - $D \times E$: $(x, y) \sqsubseteq_{D \times E} (x', y')$ iff $x \sqsubseteq_D x'$ and $y \sqsubseteq_E y'$
- Finite maps:
  - $D \to E$: $f \sqsubseteq f'$ iff $\forall d \in D$: $f(d) \sqsubseteq_E f'(d)$

Complete Lattices

- Let (D, $\sqsubseteq$) be a partial order
- (D, $\sqsubseteq$) is a complete lattice if every **subset** has
  - greatest lower bound
  - least upper bound
  - Recall: A CPO has a LUB for every **chain**
- L = (D, $\sqsubseteq$, $\sqcup$, $\sqcap$, $\perp$, T)

Constructing Complete Lattices

- For two complete lattices
  $L_1 = (D_1, \sqsubseteq_1, \sqcup_1, \sqcap_1, \bot_1, \top_1)$ and $L_2 = (D_2, \sqsubseteq_2, \sqcup_2, \sqcap_2, \bot_2, \top_2)$

- Cartesian product: L_{cart} =Cart(L_1,L_2)= $(D_1 \times D_2, \sqsubseteq_{cart}, \sqcup_{cart}, \sqcap_{cart}, \bot_{cart}, \top_{cart})$
  - $(x_1, x_2) \sqsubseteq_{cart} (y_1, y_2)$ *iff* $x_1 \sqsubseteq_1 y_1\ x_2 \sqsubseteq_2 y_2$

- Finite maps: $L_{V \to L}$ = Map(V, L) $= (V \to D, \sqsubseteq_{V\to L}, \sqcup_{V\to L}, \sqcap_{V\to L}, \bot_{V\to L}, \top_{V\to L})$
  - $f_1 \sqsubseteq_{V\to L} f_2 \Leftrightarrow \forall v \in V\ .\ f_1(v) \sqsubseteq f_2(v)$

- Disjunctive completion (Powerset): $L_\vee$ = Disj(L$_1$)=$(2^{D_1}, \sqsubseteq_\vee, \sqcup_\vee, \sqcap_\vee, \bot_\vee, \top_\vee)$
  - $X \sqsubseteq_\vee Y$ *iff* $\forall x \in X\ .\ \exists y \in Y\ .\ x \sqsubseteq_1 y$

- Relational product: $L_{rel} = (2^{D1 \times D2}, \sqsubseteq_{rel}, \sqcup_{rel}, \sqcap_{rel}, \bot_{rel}, \top_{rel})$
  - L_{rel} = Disj(Cart(L_1,L_2))

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, $\sqsubseteq$)

  $f : D \rightarrow D$

Knaster-Tarski Theorem

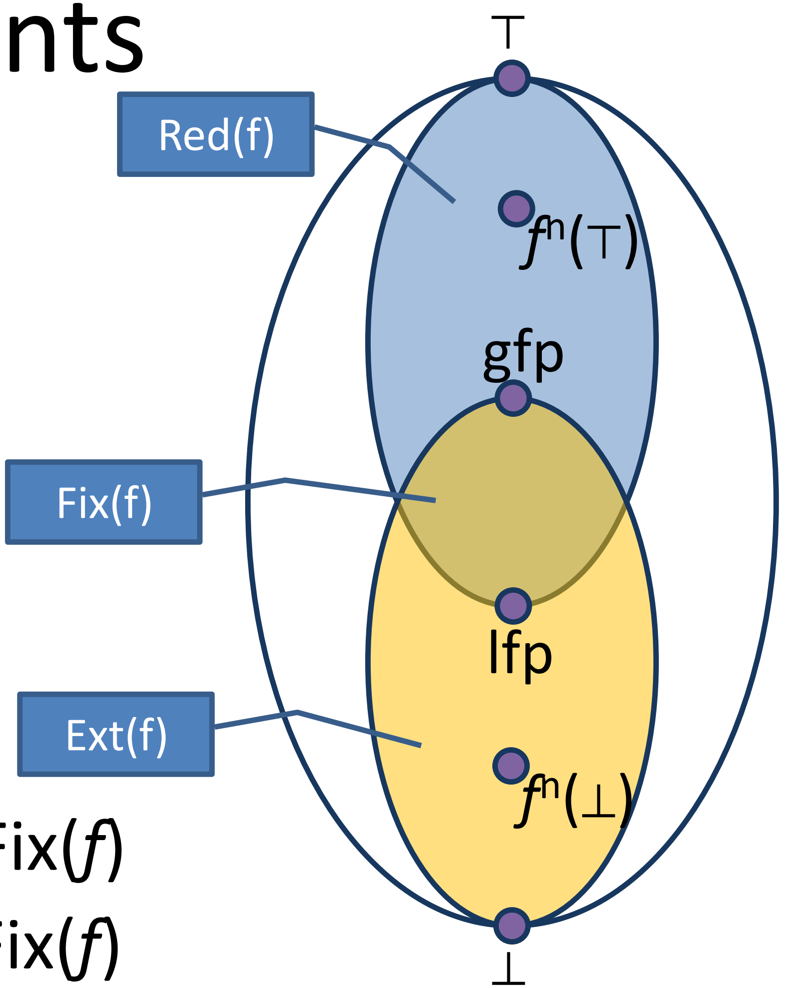
- Let f: L $\rightarrow$L be a monotonic function on a complete lattice L
- The least fixed point lfp(f) exists

Extensive/reductive functions

- Let $L=(D, \sqsubseteq)$ be a poset
- A function $f : D \rightarrow D$ is extensive if for every $x \in D$, we have that $x \sqsubseteq f(x)$
- A function $f : D \rightarrow D$ is reductive if for every $x \in D$, we have that $x \sqsubseteq f(x)$

Fixed-points

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



1. Does a solution always exist? Yes
2. If so, is it unique? No, but it has least/greatest solutions
3. If so, is it computable? Under some conditions…

Continuous Functions

- 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 $Y \subseteq D^*$,

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

- **Lemma:** A continuous function is monotonic

Continuity vs. Monotonicity

- A monotonic function maps a chain of inputs into a chain of outputs:
  $x_0 \sqsubseteq x_1 \sqsubseteq \ldots \Rightarrow f(x_0) \sqsubseteq f(x_1) \sqsubseteq \ldots$

- It is always true that: $\sqcup_i \langle f(x_i) \rangle \sqsubseteq f(\sqcup_i \langle x_i \rangle)$

- But $f(\sqcup_i \langle x_i \rangle) \sqsubseteq \sqcup_i \langle f(x_i) \rangle$ is not always true

Fixed-point theorem [Kleene]

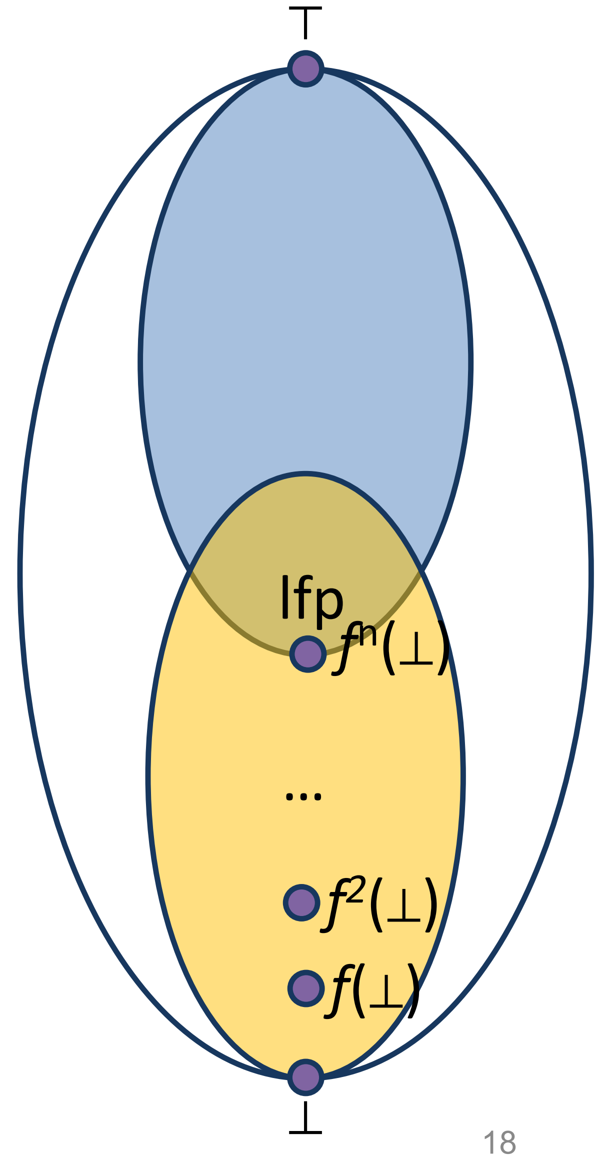
- Let $L = (D, \sqsubseteq, \sqcup, \bot)$ be a complete partial order and a **continuous** function $f: D \to D$ then

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

- **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

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

Algorithm

Terminates if D satisfies ACC

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

Collecting Semantics

- Fixpoint-based definition of the program semantics

  - Think of a program as a CFG

The collecting lattice

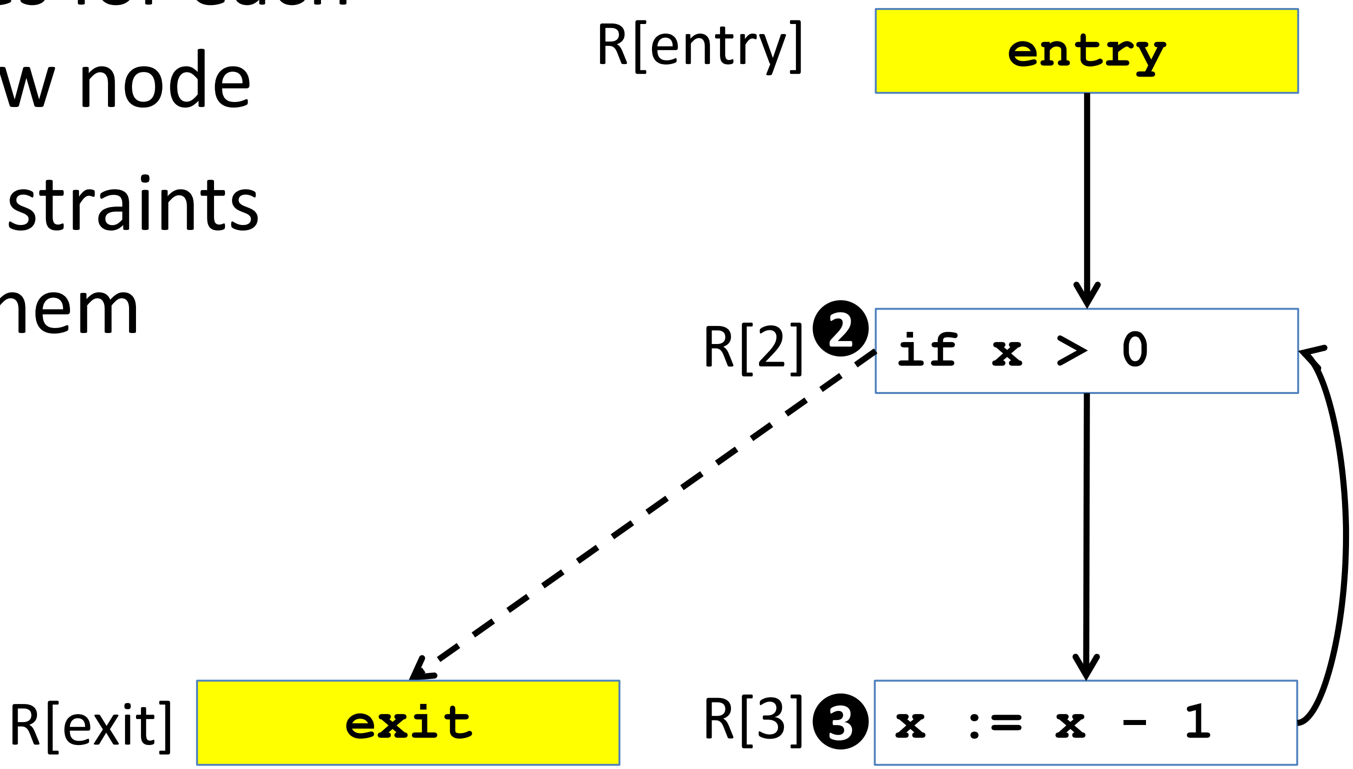
- Lattice for a given control-flow node *v*: $L_v = (2^{\mathbf{State}}, \subseteq, \cup, \cap, \emptyset, \mathbf{State})$
- Lattice for entire control-flow graph with nodes *V*:

$$L_{CFG} = \mathrm{Map}(V, L_v)$$

- We will use this lattice as a baseline for static analysis and define abstractions of its elements

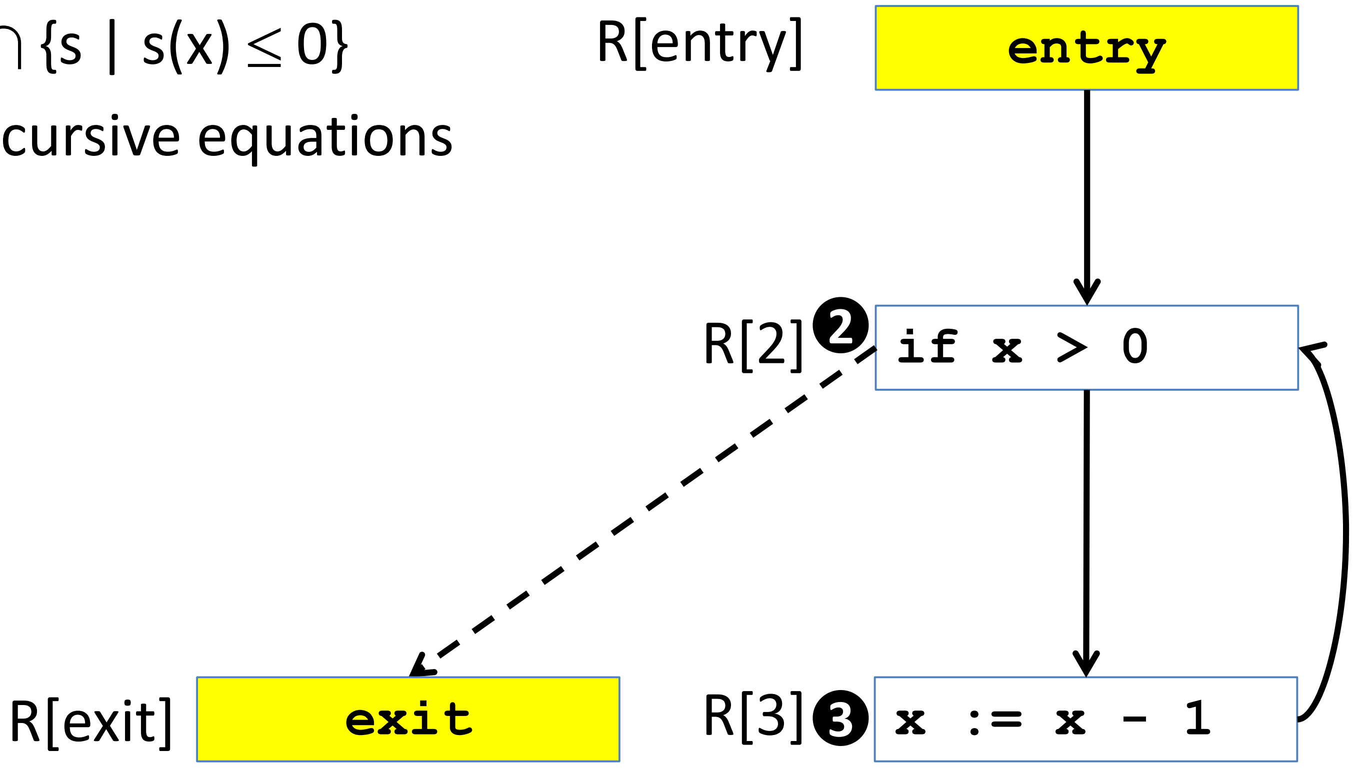
Equational definition of the semantics

- Define variables of type set of states for each control-flow node
- Define constraints between them



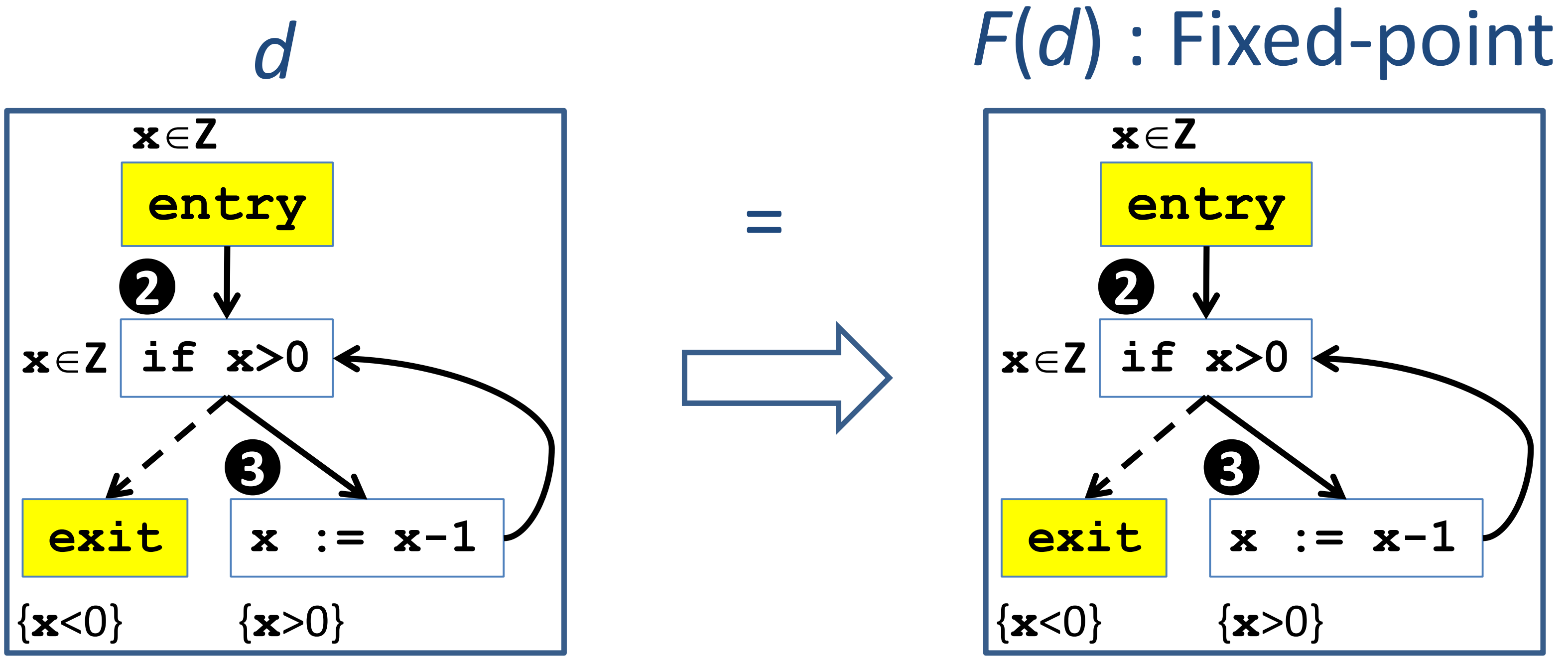
Equational definition of the semantics

- R[2] = R[entry] $\cup$ $[\![$**`x:=x-1`**$]\!]$ R[3]
- R[3] = R[2] $\cap$ {s | s(x) > 0}
- R[exit] = R[2] $\cap$ {s | s(x) $\leq$ 0}
- A system of recursive equations



Fixed point example for program

- $R[0] = \{\mathbf{x} \in Z\}$
  $R[1] = R[0] \cup R[4]$
  $R[2] = R[1] \cap \{s \mid s(x) > 0\}$
  $R[3] = R[1] \cap \{s \mid s(x) \leq 0\}$
  $R[4] = [\![\mathbf{x:=x-1}]\!]\ R[2]$



Equation systems in general

- Let *L* be a complete lattice (D, $\sqsubseteq$, $\sqcup$, $\sqcap$, $\perp$, $\top$)
- Let R be a vector of variables R[0, …, *n*] $\in D\times\ldots\times D$
- Let F be a vector of functions of the type
  f: $D \times \cdots \times D \rightarrow D$
- A system of equations
  R[0] = f[0](R[0], …, R[*n*])
  …
  R[n] = f[n](R[0], …, R[*n*])
- In vector notation R = F(R)

F(R) is monotonic

- Special cases of monotonic functions:
  - Join: $f(X, Y) = X \sqcup Y$
  - For a set X and any function g: $F(X) = \{ g(x) \mid x \in X \}$

- The collecting semantics function is defined using
  - Join (set union)
  - Meet (set intersection)
  - Semantic function for atomic statements lifted to sets of states

- L_{CFG} is a Lattice, hence has a fixpoint

Abstract Semantics

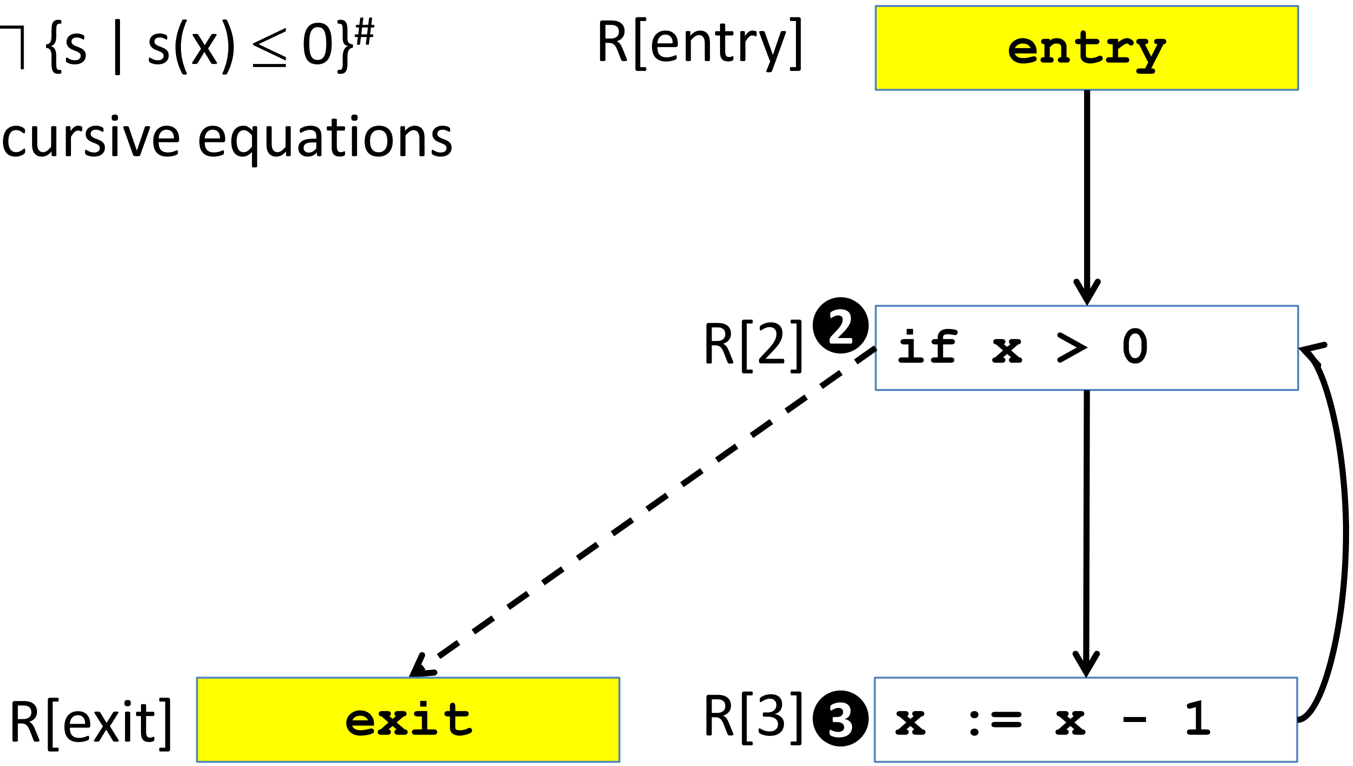
- Over-approximating the collecting semantics

An abstract semantics

- $R[2] = R[entry] \sqcup [\![\mathtt{x:=x-1}]\!]^{\#}\, R[3]$
- $R[3] = R[2] \sqcap \{s \mid s(x) > 0\}^{\#}$
- $R[exit] = R[2] \sqcap \{s \mid s(x) \leq 0\}^{\#}$
- A system of recursive equations

Abstract transformer for **`x:=x-1`**

Abstract representation of {s | s(x) < 0}



Chaotic Iterations

- An algorithm to compute the abstract fixpoint

Resulting algorithm

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

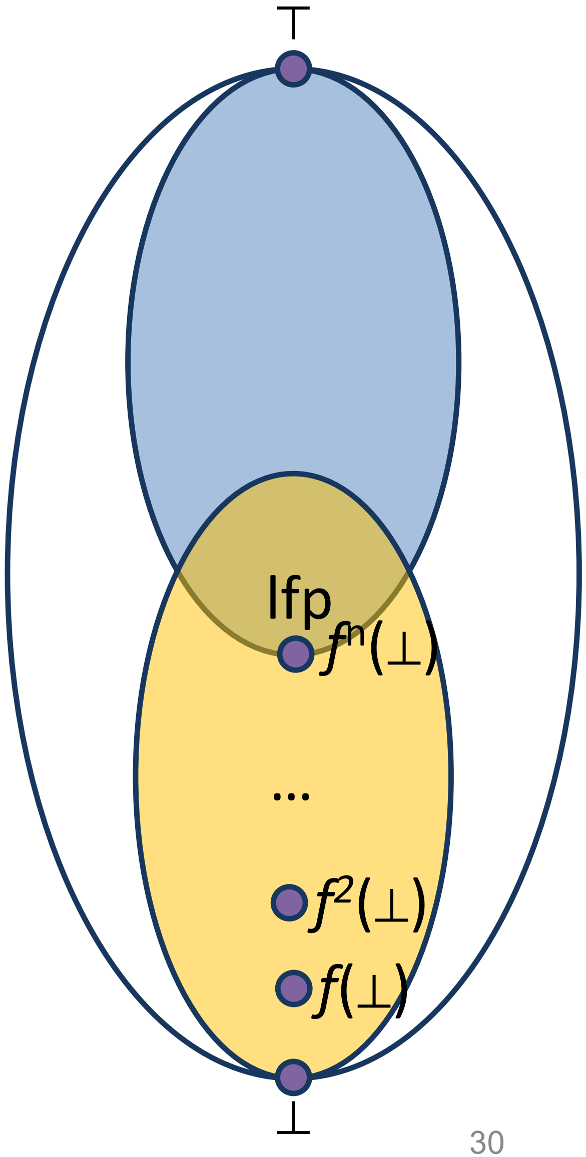
Mathematical definition

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

Algorithm

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

Terminates if D satisfies ACC



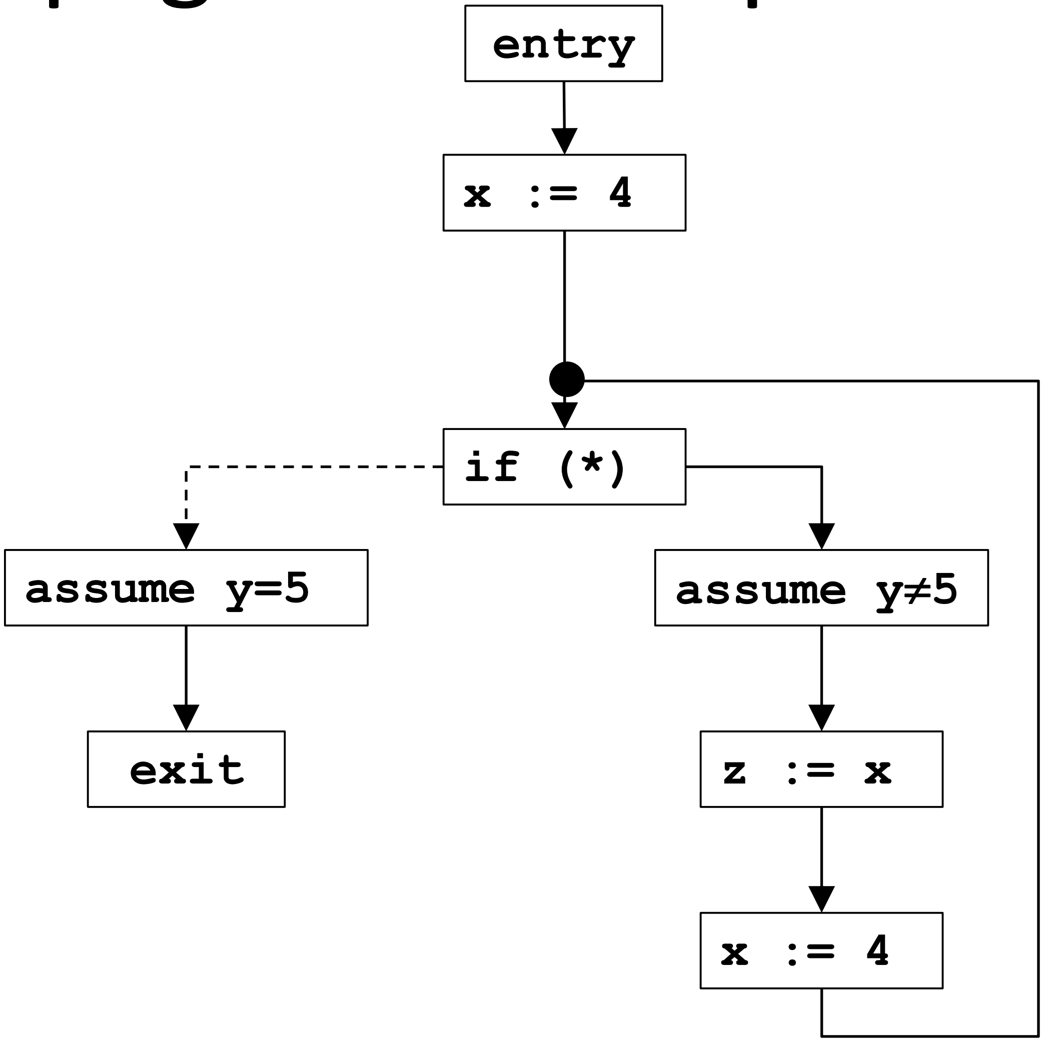
Chaotic iteration

- 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 $\{ X[\mathrm{i}] \mid f(X) \mid 1 \leq i \leq n \}$
- 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
```

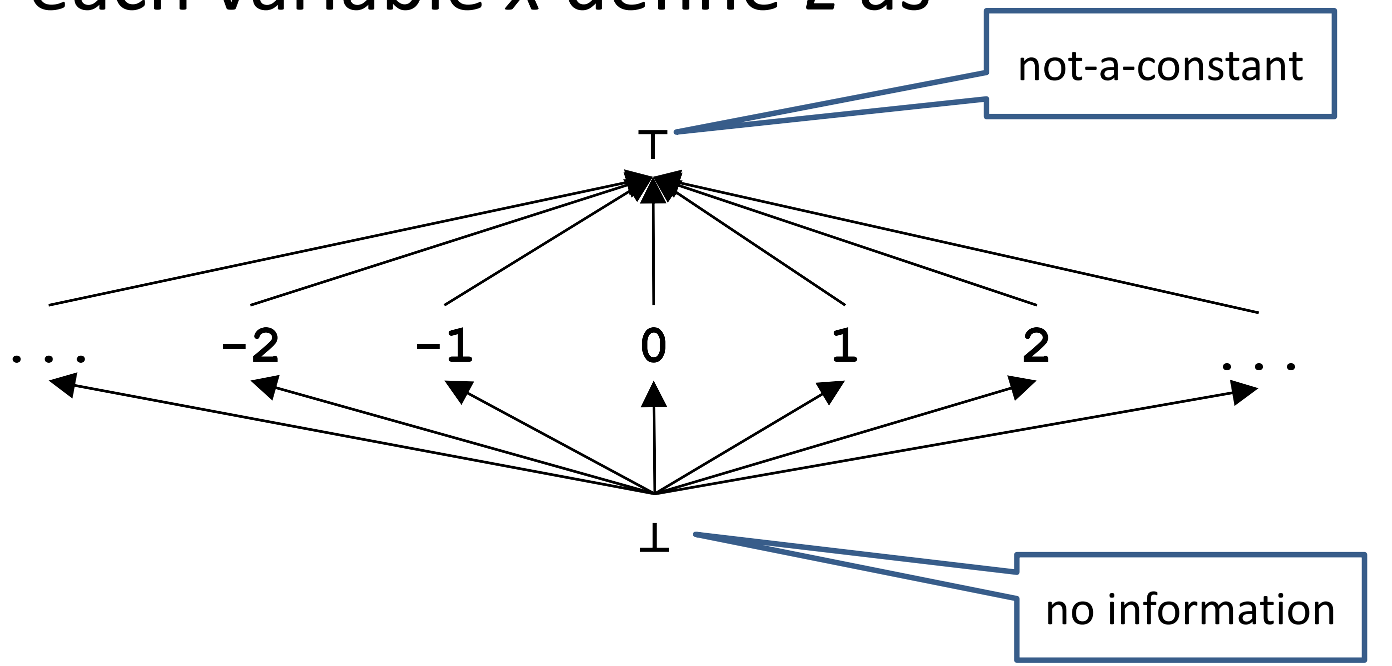
Constant propagation example

```
x := 4;
while (y≠5) do
    z := x;
    x := 4
```



Constant propagation lattice

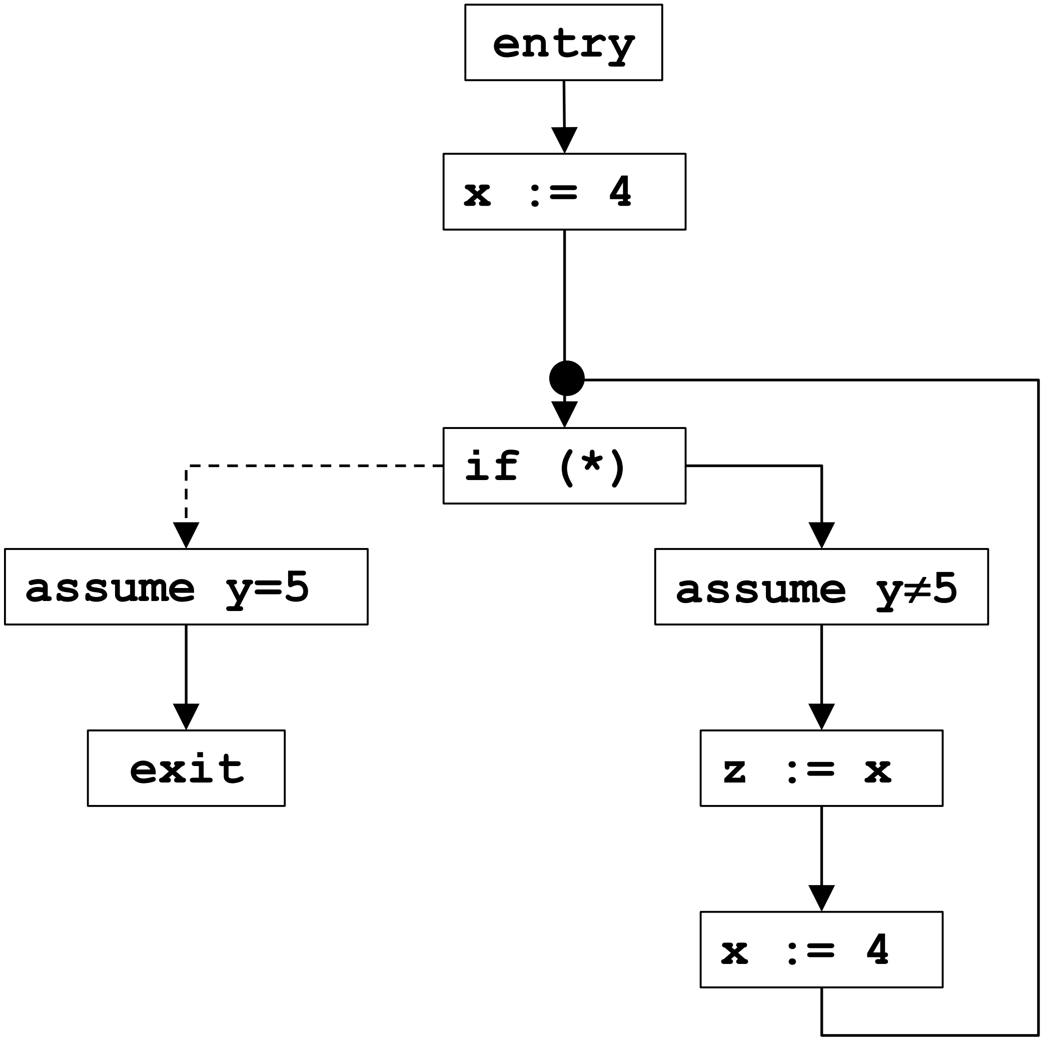
- For each variable *x* define *L* as



- For a set of program variables Var=$x_1,\ldots,x_n$
  $L^n = L \times L \times \ldots \times L$

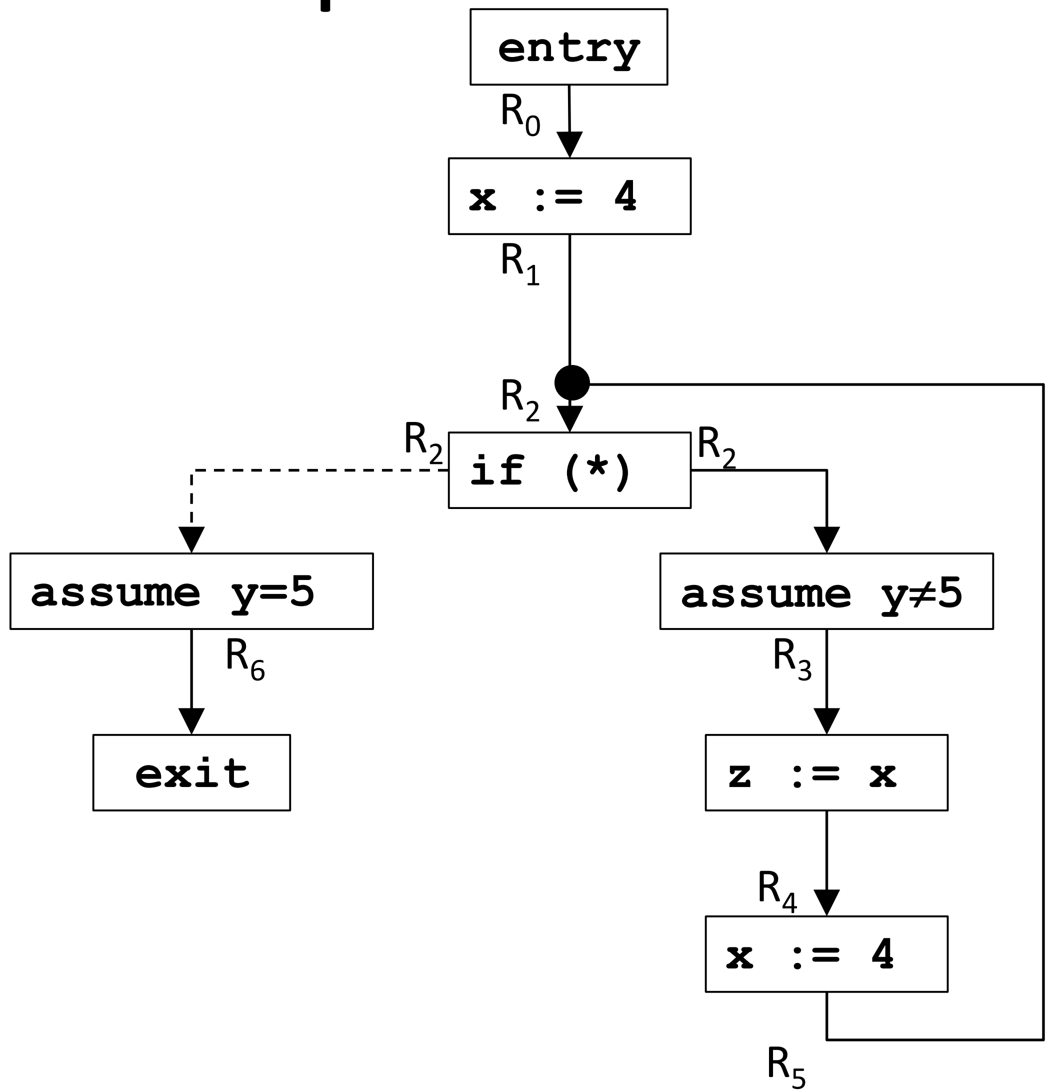
Write down variables

```
x := 4;
while (y≠5) do
     z := x;
     x := 4
```



Write down equations

```
x := 4;
while (y≠5) do
    z := x;
    x := 4
```



Collecting semantics equations

R_0 = **State**

$R_1 = [\![$ **`x:=4`** $]\!]\ R_0$

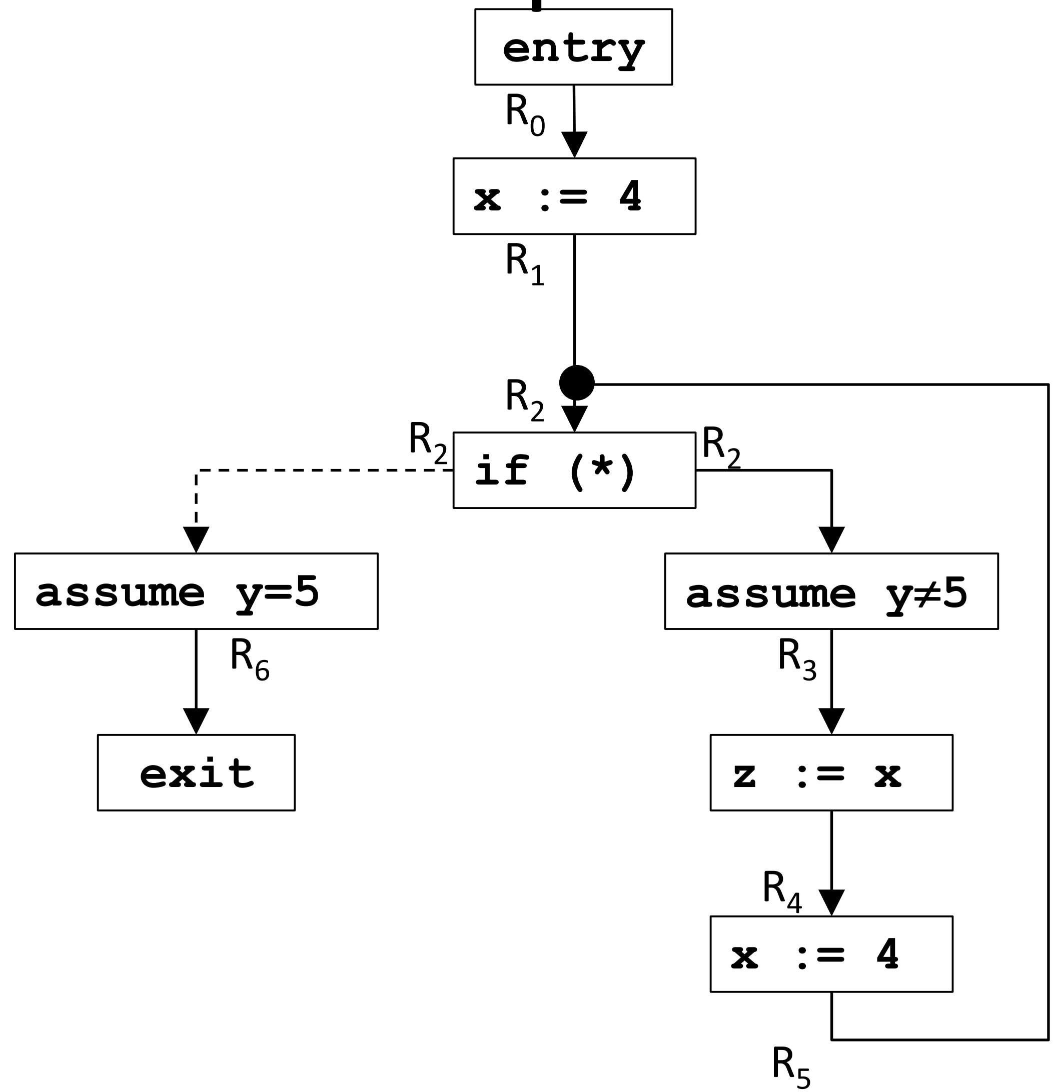
$R_2 = R_1 \cup R_5$

$R_3 = [\![$ **`assume y≠5`** $]\!]\ R_2$

$R_4 = [\![$ **`z:=x`** $]\!]\ R_3$

$R_5 = [\![$ **`x:=4`** $]\!]\ R_4$

$R_6 = [\![$ **`assume y=5`** $]\!]\ R_2$



Constant propagation equations

$R_0 = \top$

$R_1 = [\![\mathbf{x:=4}]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

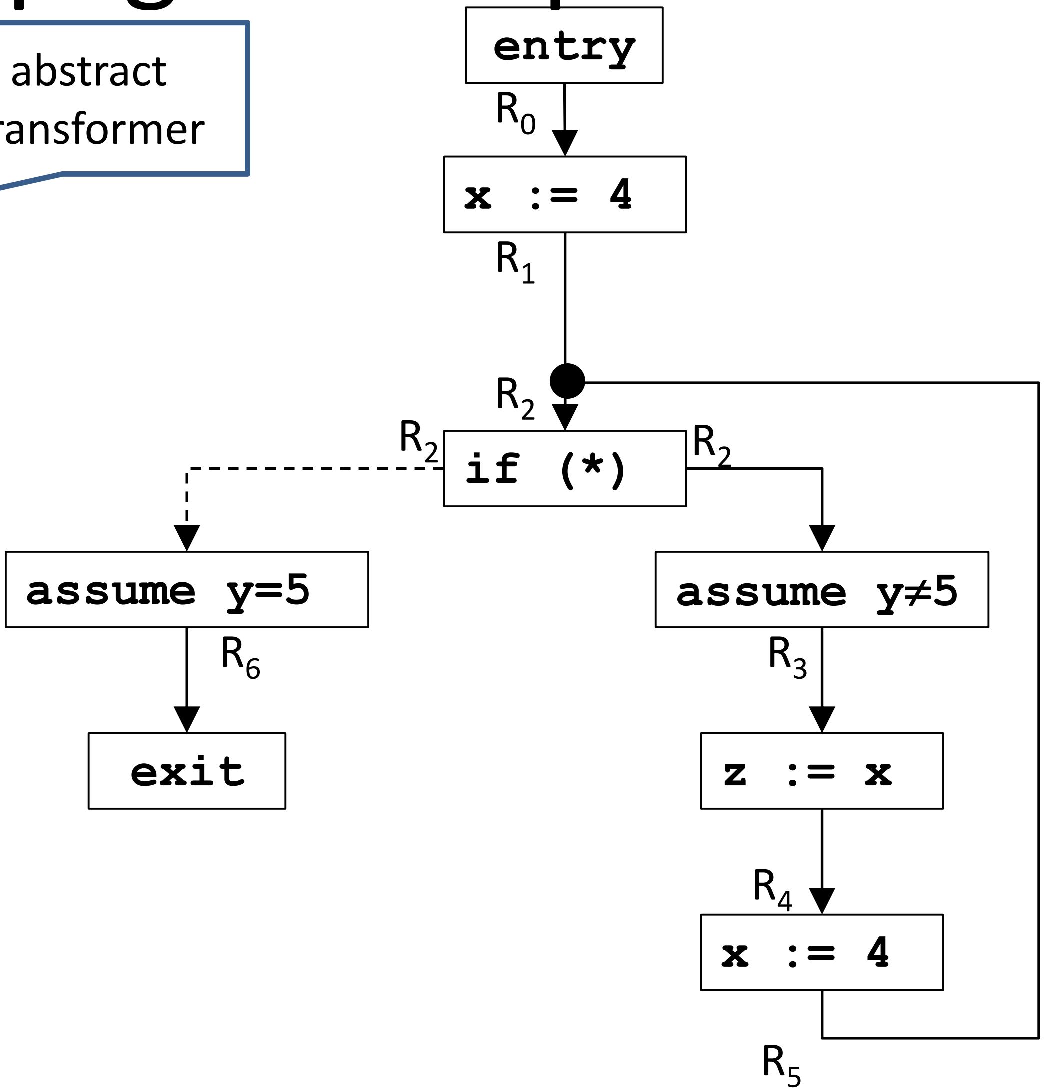
$R_3 = [\![\mathbf{assume\ y \neq 5}]\!]^\# R_2$

$R_4 = [\![\mathbf{z:=x}]\!]^\# R_3$

$R_5 = [\![\mathbf{x:=4}]\!]^\# R_4$

$R_6 = [\![\mathbf{assume\ y=5}]\!]^\# R_2$

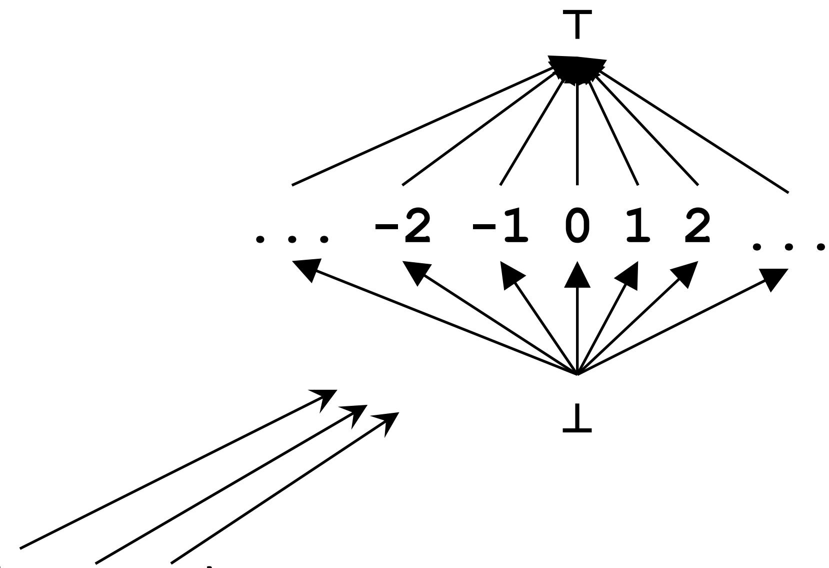
abstract transformer



Abstract operations for CP

$R_0 = \top$
$R_1 = \llbracket \mathbf{x:=4} \rrbracket^\# R_0$
$R_2 = R_1 \sqcup R_5$
$R_3 = \llbracket \mathbf{assume\ y{\neq}5} \rrbracket^\# R_2$
$R_4 = \llbracket \mathbf{z:=x} \rrbracket^\# R_3$
$R_5 = \llbracket \mathbf{x:=4} \rrbracket^\# R_4$
$R_6 = \llbracket \mathbf{assume\ y{=}5} \rrbracket^\# R_2$

CP lattice for a single variable

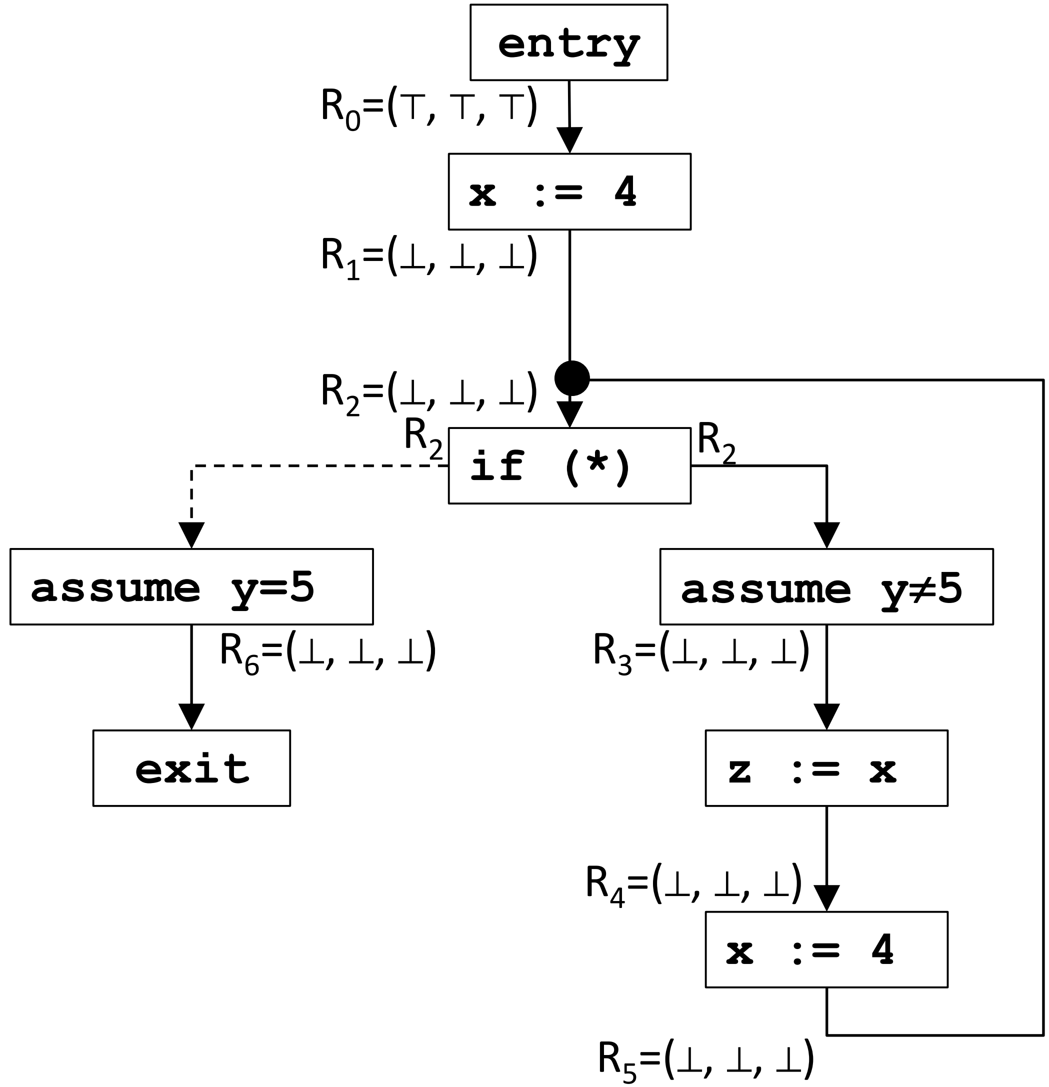


Lattice elements have the form: (v_x, v_y, v_z)
$\llbracket \mathbf{x:=4} \rrbracket^\# (v_x,v_y,v_z) = (4, v_y, v_z)$
$\llbracket \mathbf{z:=x} \rrbracket^\# (v_x,v_y,v_z) = (v_x, v_y, v_x)$
$\llbracket \mathbf{assume\ y{\neq}5} \rrbracket^\# (v_x,v_y,v_z)$ = if $v_y = \mathbf{5}$ then $(\bot, \bot, \bot)$ else (v_x, v_y, v_z)
$\llbracket \mathbf{assume\ y{=}5} \rrbracket^\# (v_x,v_y,v_z)$ = if $v_y = k \neq \mathbf{5}$ then $(\bot, \bot, \bot)$ else $(v_x, 5, v_z)$
$k \neq \top$
$R_1 \sqcup R_5 = (a_1, b_1, c_1) \sqcup (a_5, b_5, c_5) = (a_1 \sqcup a_5, b_1 \sqcup b_5, c_1 \sqcup c_5)$

Chaotic iteration for CP: initialization

$R_0 = \top$
$R_1 = [\![\mathbf{x:=4}]\!]^\# R_0$
$R_2 = R_1 \sqcup R_5$
$R_3 = [\![\mathbf{assume\ y\neq5}]\!]^\# R_2$
$R_4 = [\![\mathbf{z:=x}]\!]^\# R_3$
$R_5 = [\![\mathbf{x:=4}]\!]^\# R_4$
$R_6 = [\![\mathbf{assume\ y=5}]\!]^\# R_2$

WL = $\{R_0, R_1, R_2, R_3, R_4, R_5, R_6\}$



Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![\mathbf{x:=4}]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

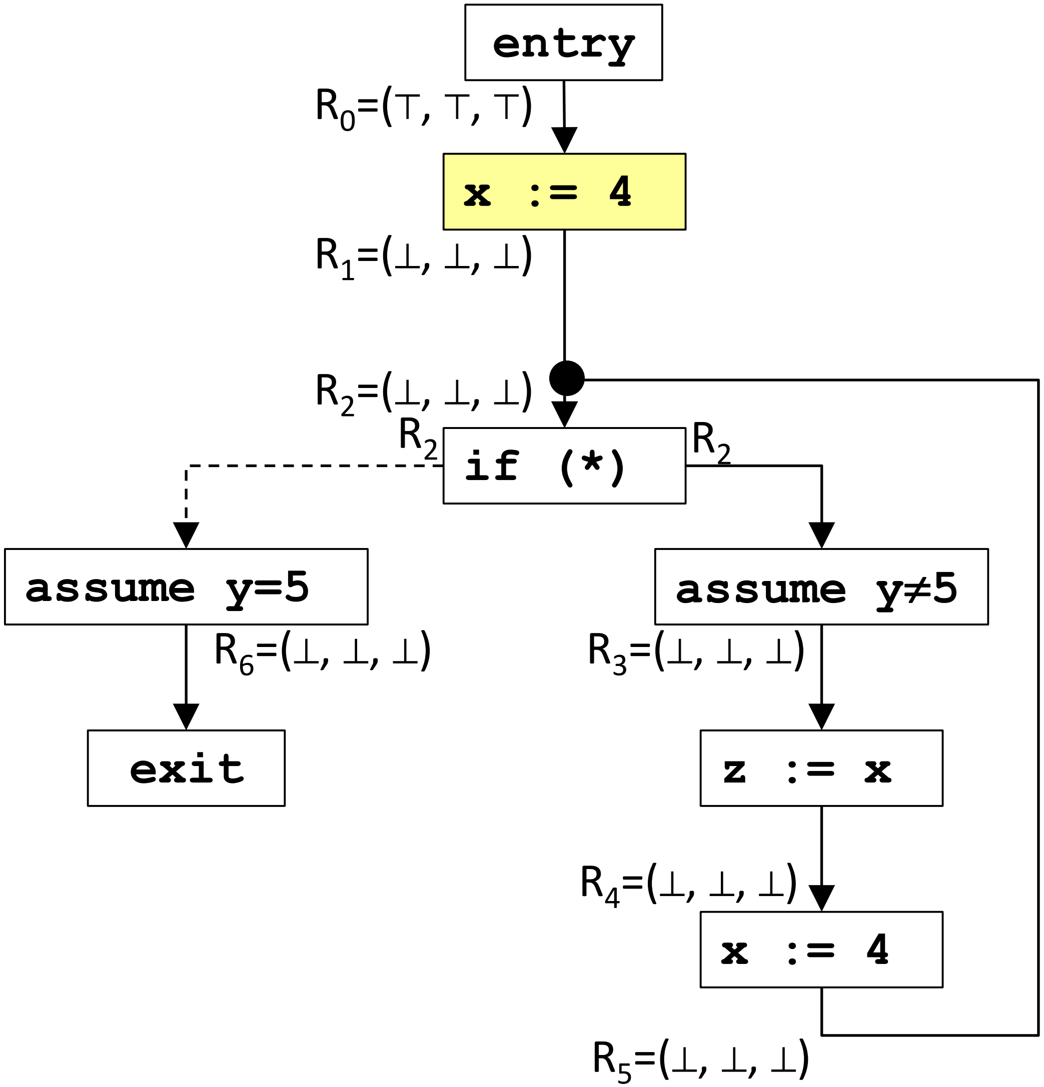
$R_3 = [\![\mathbf{assume\ y{\neq}5}]\!]^\# R_2$

$R_4 = [\![\mathbf{z:=x}]\!]^\# R_3$

$R_5 = [\![\mathbf{x:=4}]\!]^\# R_4$

$R_6 = [\![\mathbf{assume\ y{=}5}]\!]^\# R_2$

WL = $\{R_1, R_2, R_3, R_4, R_5, R_6\}$



Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![\mathbf{x:=4}]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

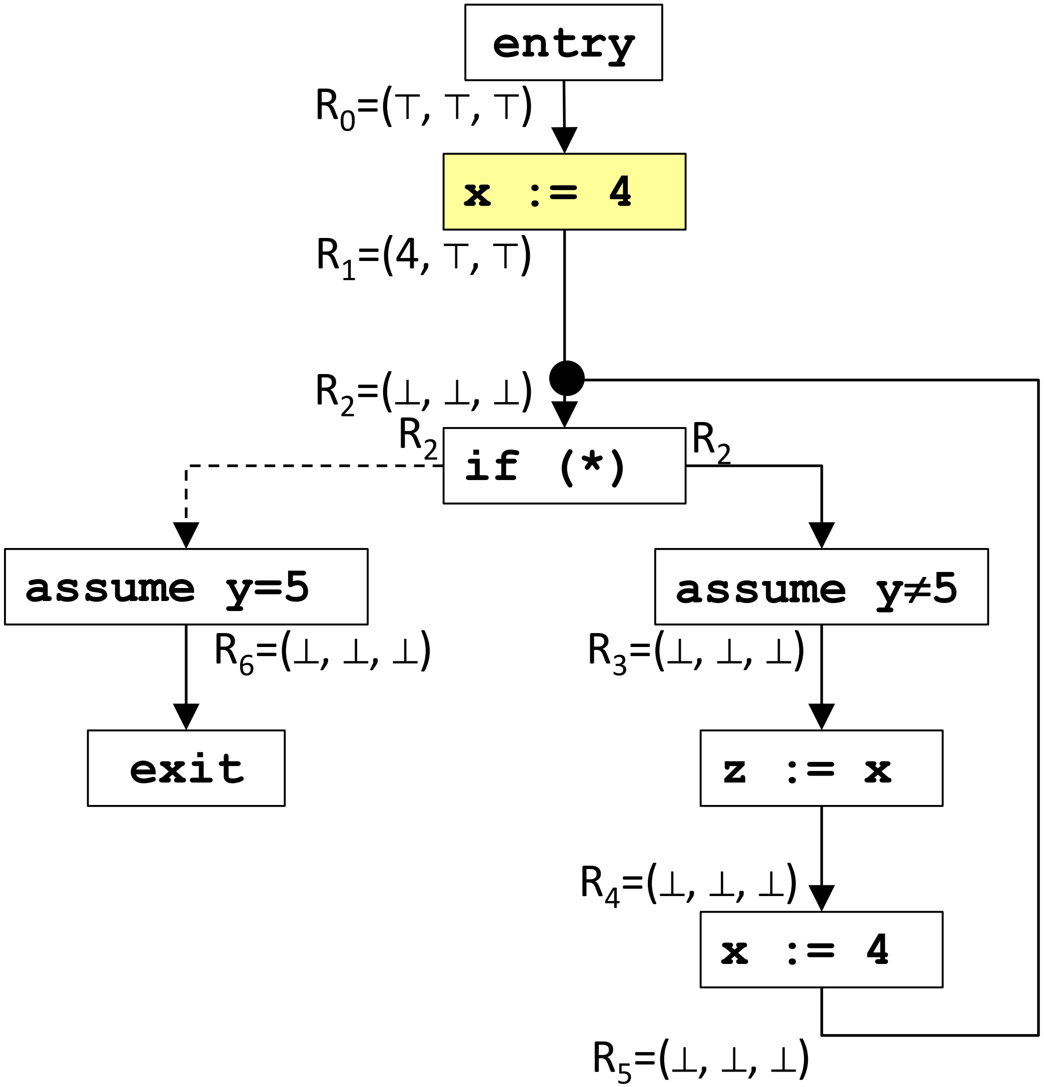
$R_3 = [\![\mathbf{assume\ y \neq 5}]\!]^\# R_2$

$R_4 = [\![\mathbf{z:=x}]\!]^\# R_3$

$R_5 = [\![\mathbf{x:=4}]\!]^\# R_4$

$R_6 = [\![\mathbf{assume\ y=5}]\!]^\# R_2$

WL = $\{R_2, R_3, R_4, R_5, R_6\}$



Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![\texttt{x:=4}]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

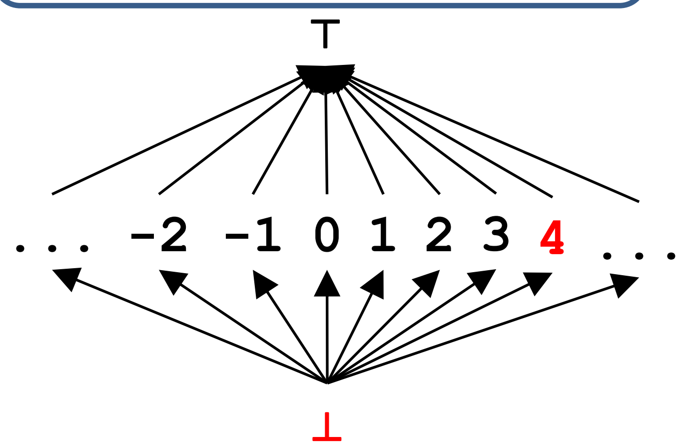
$R_3 = [\![\texttt{assume y≠5}]\!]^\# R_2$

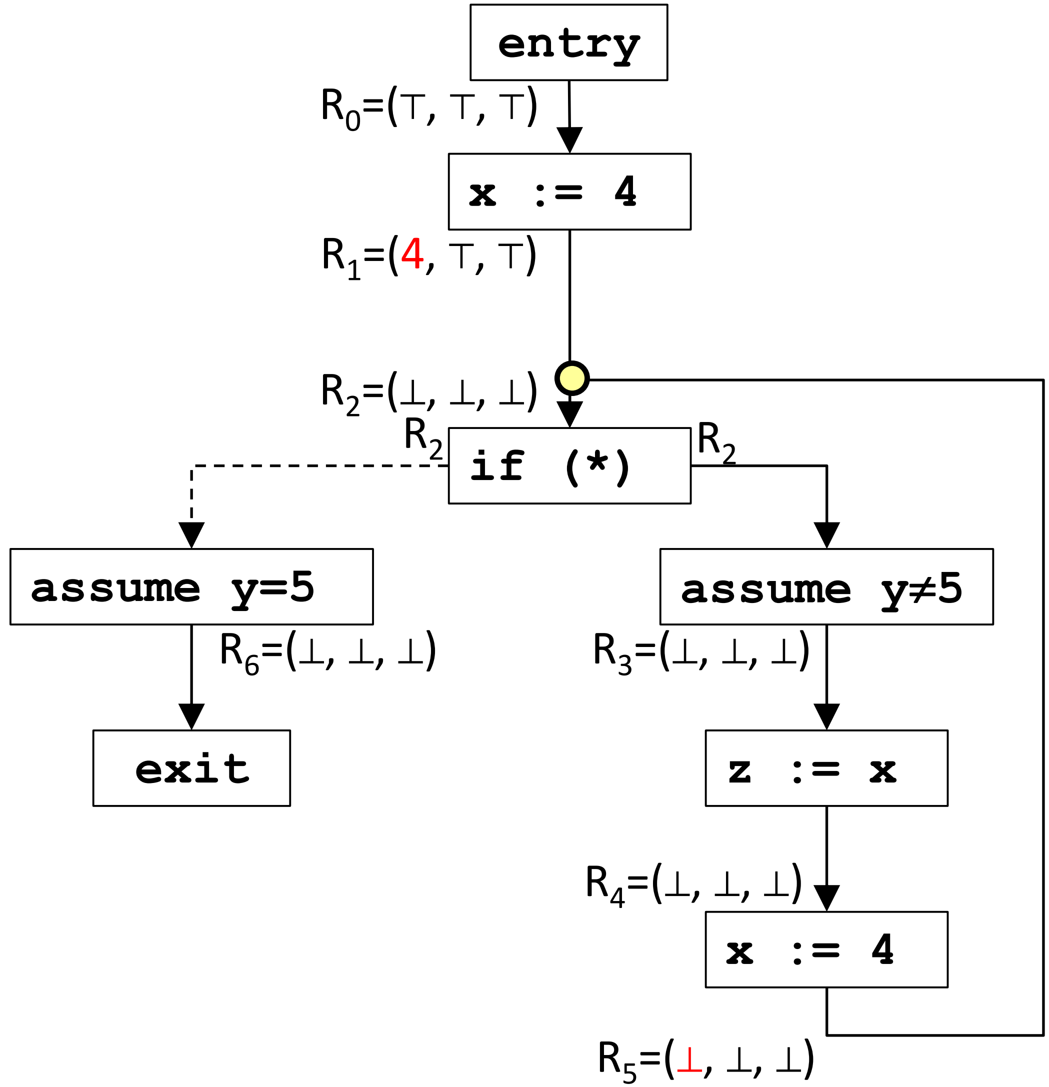
$R_4 = [\![\texttt{z:=x}]\!]^\# R_3$

$R_5 = [\![\texttt{x:=4}]\!]^\# R_4$

$R_6 = [\![\texttt{assume y=5}]\!]^\# R_2$

WL = $\{R_2, R_3, R_4, R_5, R_6\}$





Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![\mathbf{x:=4}]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

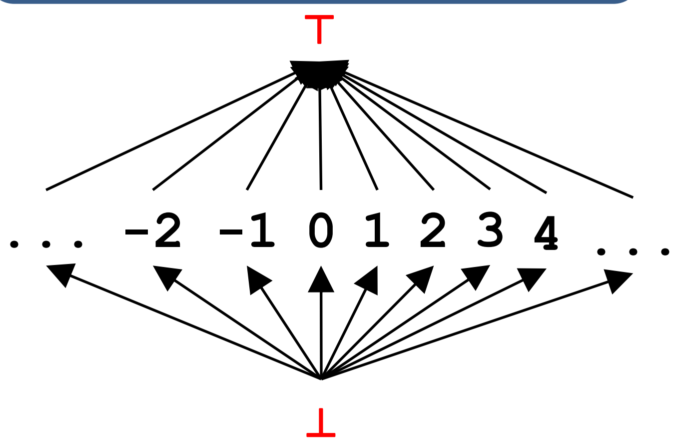
$R_3 = [\![\mathbf{assume\ y{\neq}5}]\!]^\# R_2$

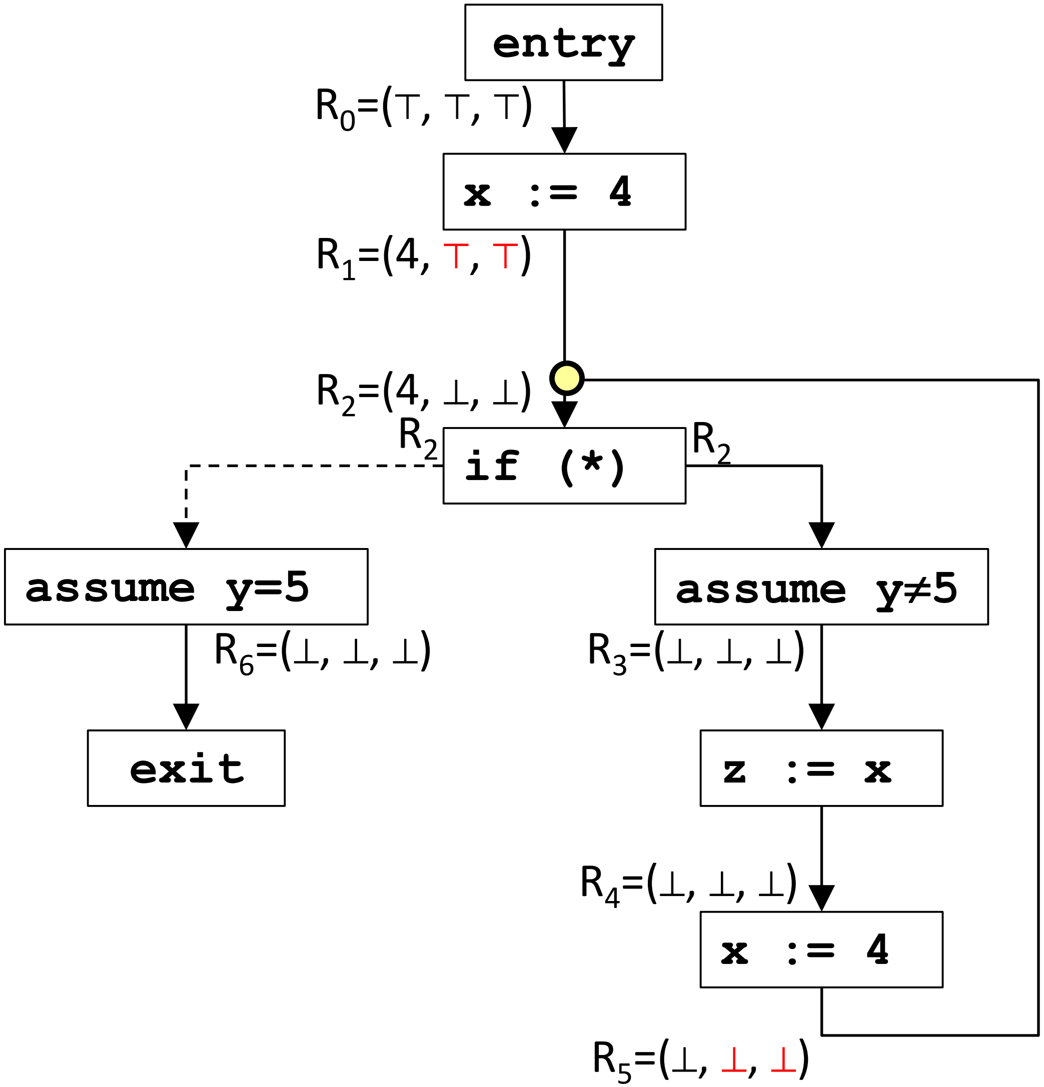
$R_4 = [\![\mathbf{z:=x}]\!]^\# R_3$

$R_5 = [\![\mathbf{x:=4}]\!]^\# R_4$

$R_6 = [\![\mathbf{assume\ y{=}5}]\!]^\# R_2$

WL = $\{R_2, R_3, R_4, R_5, R_6\}$





Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![\mathbf{x:=4}]\!]^{\#}\ R_0$

$R_2 = R_1 \sqcup R_5$

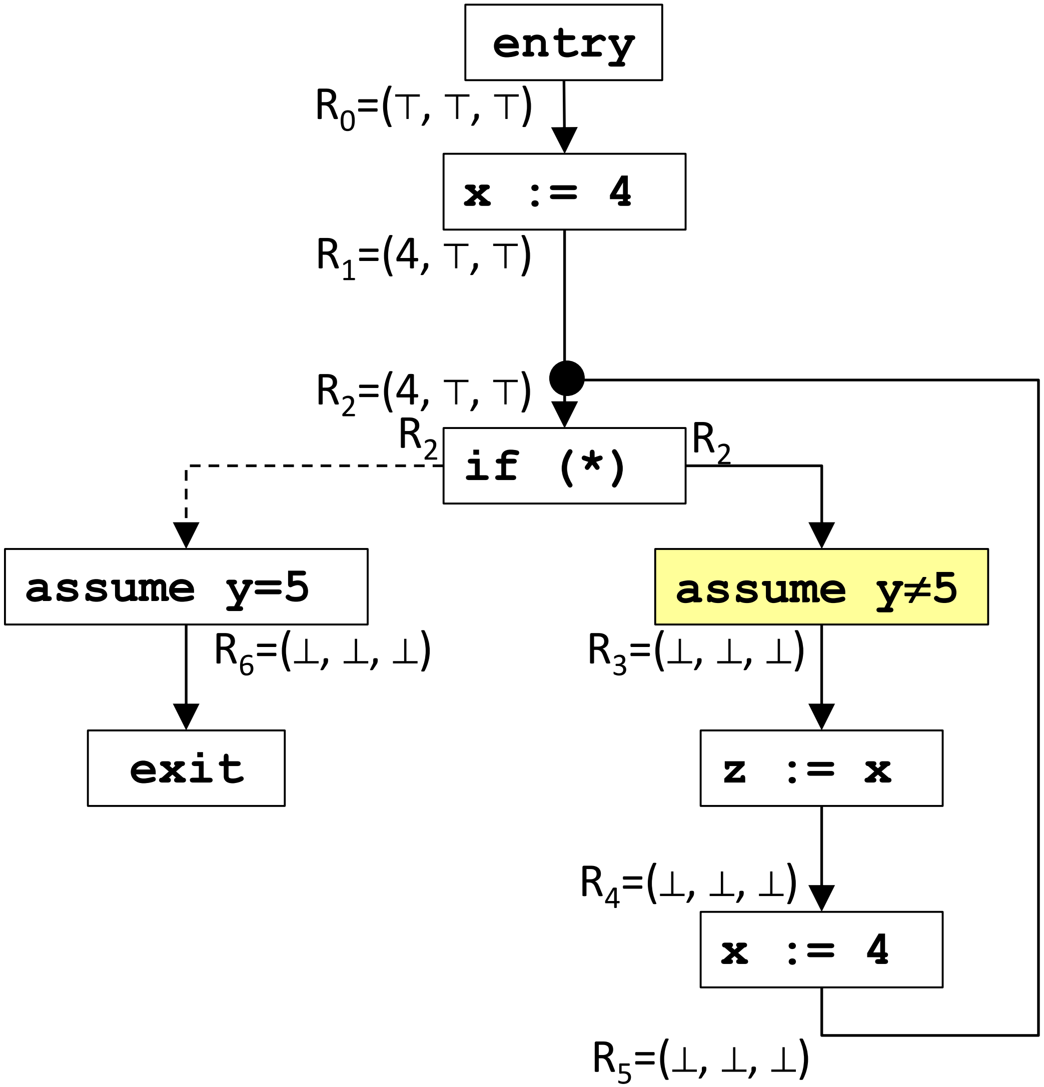
$R_3 = [\![\mathbf{assume\ y{\neq}5}]\!]^{\#}\ R_2$

$R_4 = [\![\mathbf{z:=x}]\!]^{\#}\ R_3$

$R_5 = [\![\mathbf{x:=4}]\!]^{\#}\ R_4$

$R_6 = [\![\mathbf{assume\ y{=}5}]\!]^{\#}\ R_2$

WL = $\{R_3, R_4, R_5, R_6\}$



Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![\mathbf{x:=4}]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

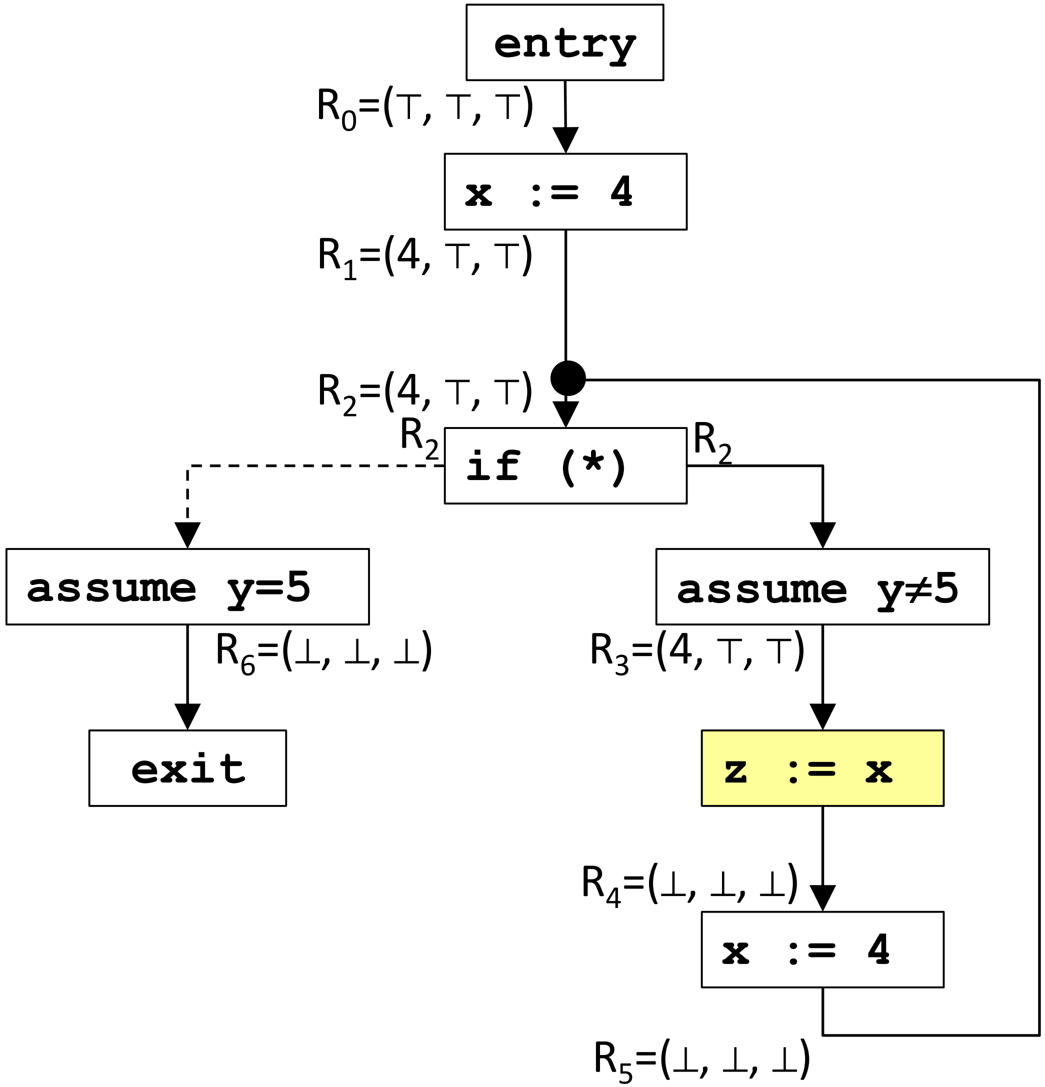
$R_3 = [\![\mathbf{assume\ y{\neq}5}]\!]^\# R_2$

$R_4 = [\![\mathbf{z:=x}]\!]^\# R_3$

$R_5 = [\![\mathbf{x:=4}]\!]^\# R_4$

$R_6 = [\![\mathbf{assume\ y{=}5}]\!]^\# R_2$

WL = $\{R_4, R_5, R_6\}$



Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![$ **x:=4** $]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

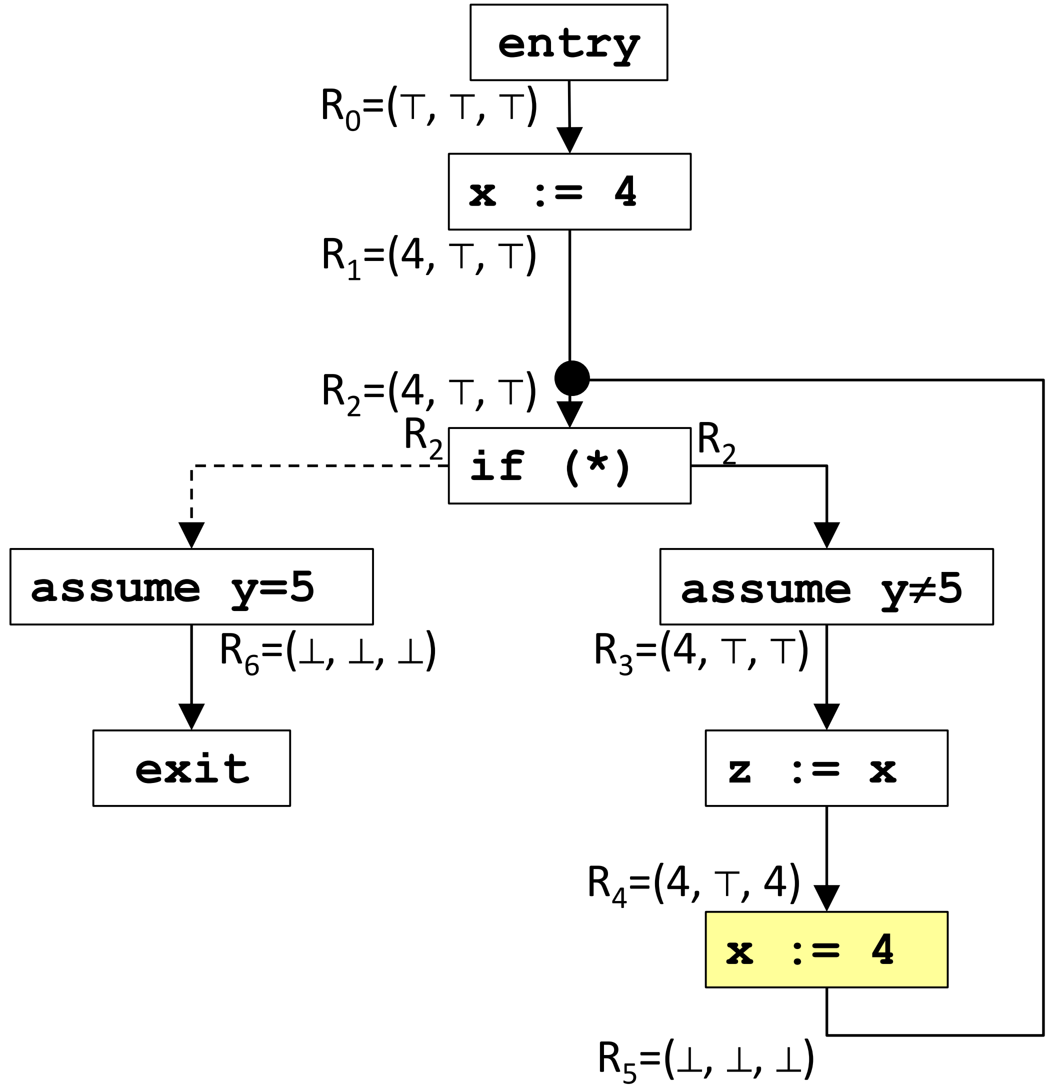
$R_3 = [\![$ **assume y≠5** $]\!]^\# R_2$

$R_4 = [\![$ **z:=x** $]\!]^\# R_3$

$R_5 = [\![$ **x:=4** $]\!]^\# R_4$

$R_6 = [\![$ **assume y=5** $]\!]^\# R_2$

WL = $\{R_5, R_6\}$



Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![\mathbf{x:=4}]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

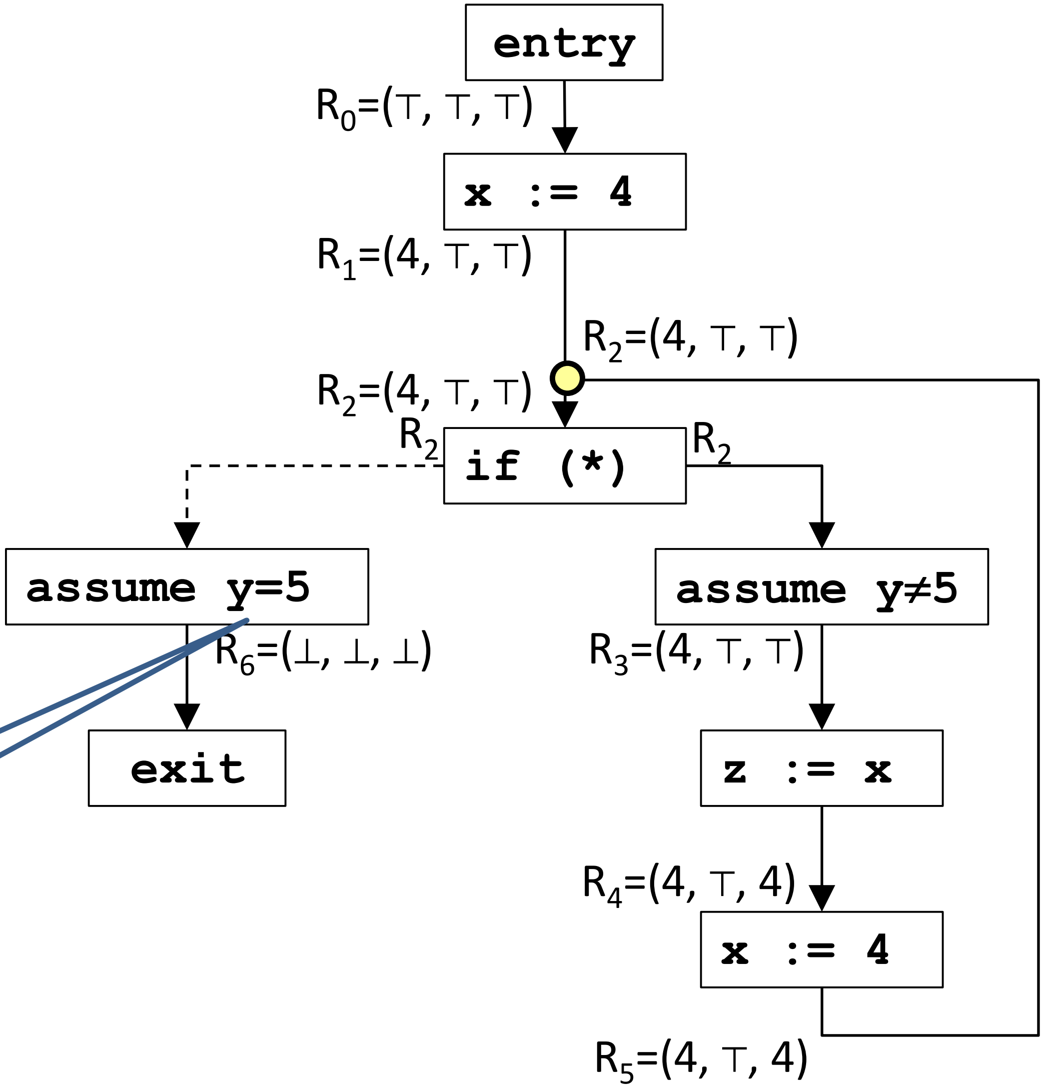
$R_3 = [\![\mathbf{assume\ y\neq5}]\!]^\# R_2$

$R_4 = [\![\mathbf{z:=x}]\!]^\# R_3$

$R_5 = [\![\mathbf{x:=4}]\!]^\# R_4$

$R_6 = [\![\mathbf{assume\ y=5}]\!]^\# R_2$

WL = $\{R_2, R_6\}$



added R_2 back to worklist since it depends on R_5

Chaotic iteration for CP

$R_0 = \top$

$R_1 = [\![$**x:=4**$]\!]^\# R_0$

$R_2 = R_1 \sqcup R_5$

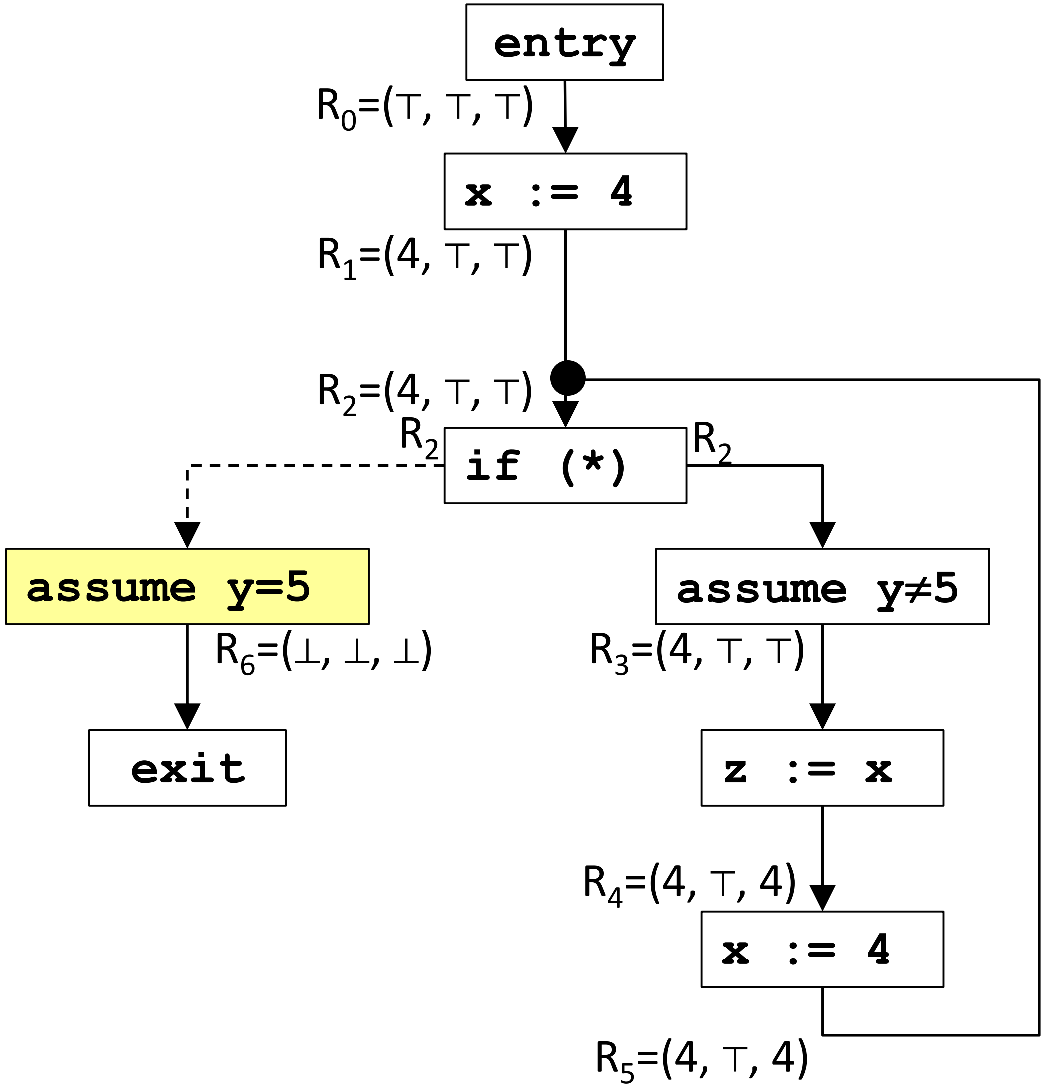
$R_3 = [\![$**assume y≠5**$]\!]^\# R_2$

$R_4 = [\![$**z:=x**$]\!]^\# R_3$

$R_5 = [\![$**x:=4**$]\!]^\# R_4$

$R_6 = [\![$**assume y=5**$]\!]^\# R_2$

WL = $\{R_6\}$



Chaotic iteration for CP

$R_0 = \top$

$R_1 = \llbracket \texttt{x:=4} \rrbracket^\# R_0$

$R_2 = R_1 \sqcup R_5$

$R_3 = \llbracket \texttt{assume y} \neq \texttt{5} \rrbracket^\# R_2$

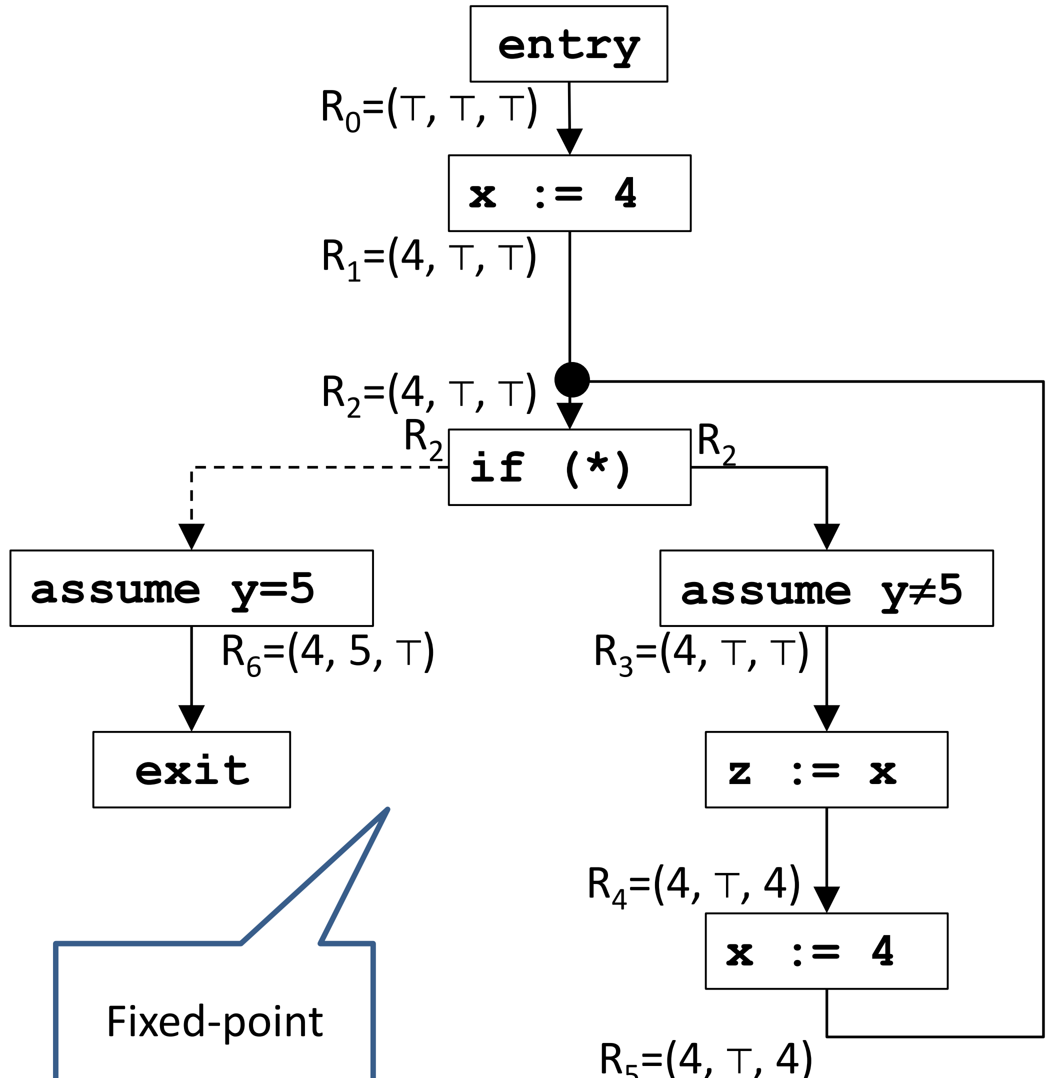
$R_4 = \llbracket \texttt{z:=x} \rrbracket^\# R_3$

$R_5 = \llbracket \texttt{x:=4} \rrbracket^\# R_4$

$R_6 = \llbracket \texttt{assume y=5} \rrbracket^\# R_2$

WL = {}

In practice maintain a worklist of nodes



Fixed-point

Chaotic iteration for static analysis

- Specialize chaotic iteration for programs
- Create a CFG for program
- Choose a cpo of properties for the static analysis to infer: $L = (D, \sqsubseteq, \sqcup, \bot)$
- Define variables R[0,…,n] for input/output of each CFG node such that $R[i] \in D$
- For each node v let v_{out} be the variable at the output of that node:
  $v_{out} = F[v](\sqcup\ u \mid (u,v) \text{ is a CFG edge})$
  - Make sure each F[v] is monotone
- Variable dependence determined by outgoing edges in CFG

Required knowledge

- ✓ Collecting semantics
- ✓ Abstract semantics (over lattices)
- ✓ Algorithm to compute abstract semantics (chaotic iteration)
- Connection between collecting semantics and abstract semantics
- Abstract transformers

Are we sound?

- We defined a reference semantics – the collecting semantics
- We defined an abstract semantics for a given lattice and abstract transformers
- We defined an algorithm to compute abstract least fixed-point when transformers are monotone and lattice obeys ACC
- Questions:

1. What is the connection between the two least fixed-points?
2. Transformer monotonicity is required for termination – what should we require for correctness?

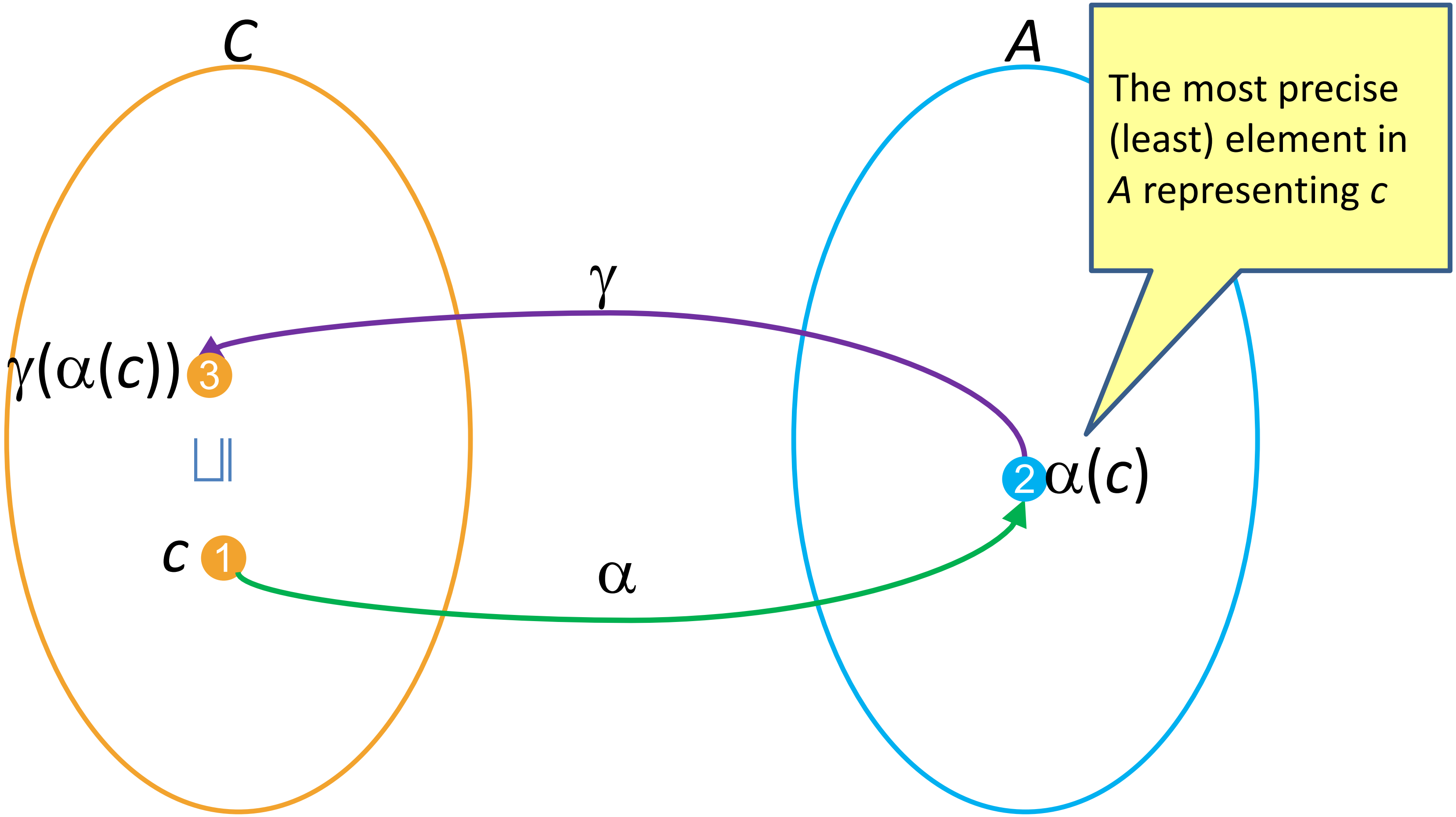
Galois Connection

- 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 $a \in D^A$

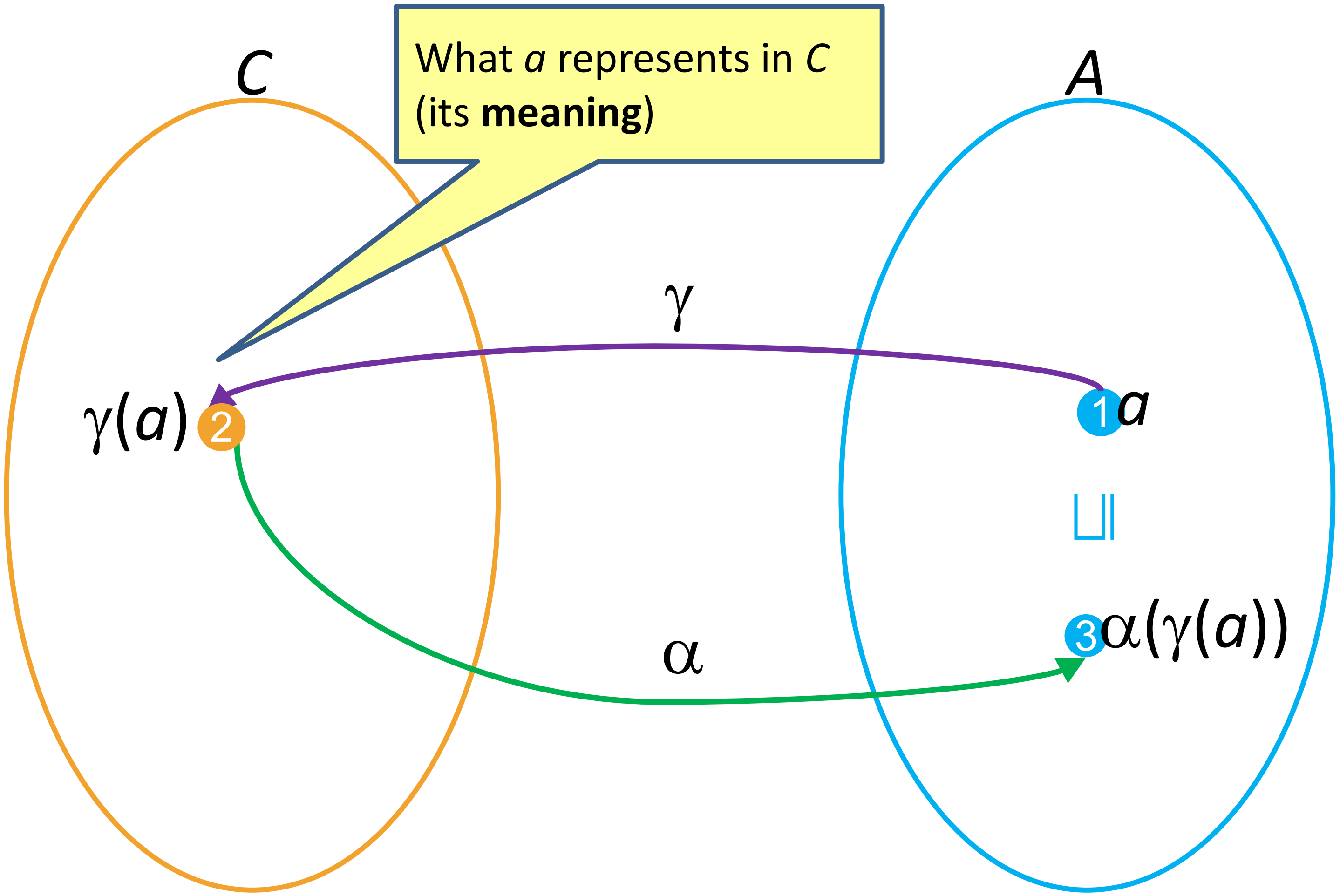
  $$\alpha(\gamma(a)) \sqsubseteq a \text{ and } c \sqsubseteq \gamma(\alpha(c))$$

- Alternatively $\alpha(c) \sqsubseteq a$ iff $c \sqsubseteq \gamma(a)$

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



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



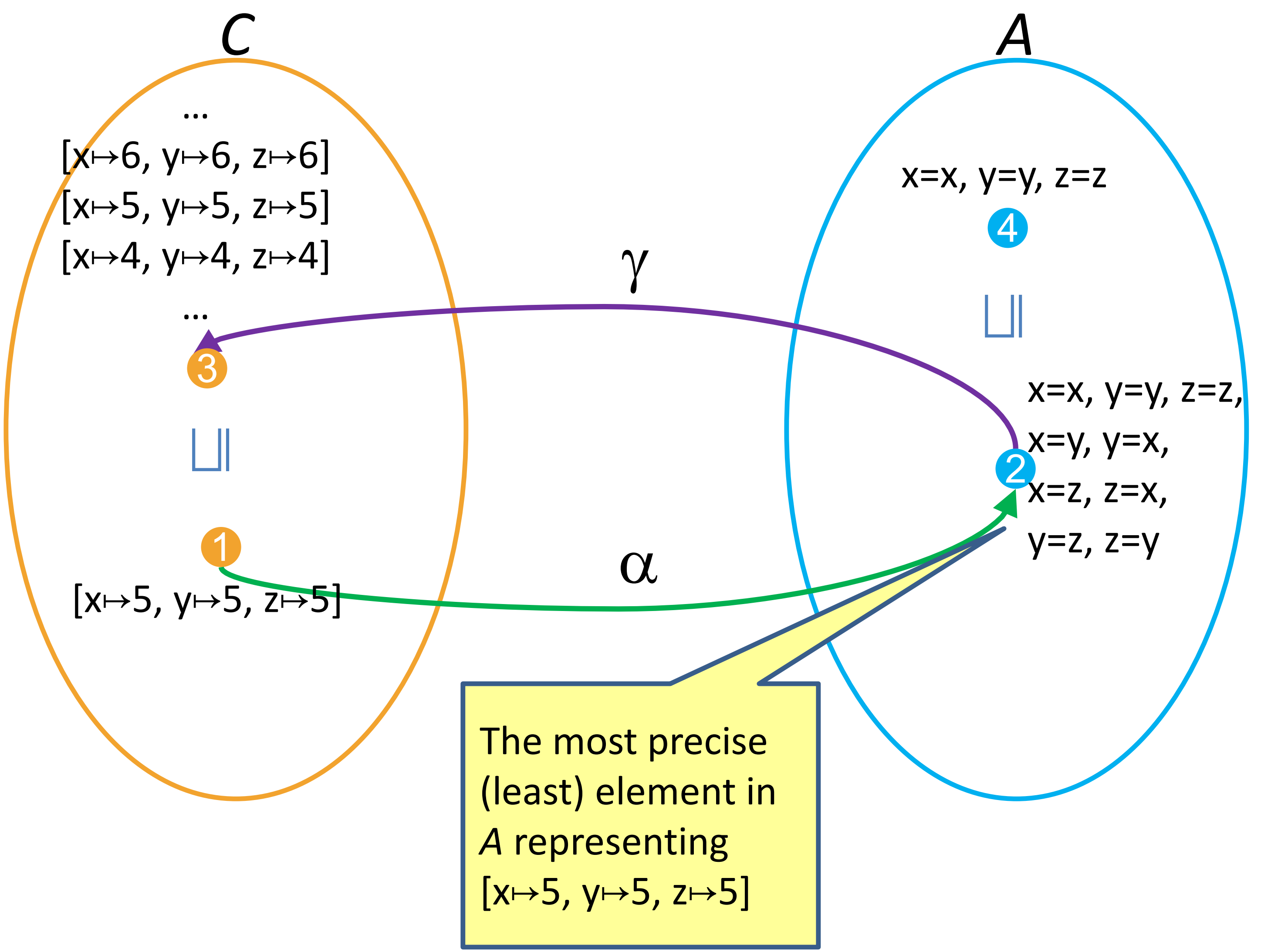
Example: lattice of equalities

- Concrete lattice:
  $C = (2^{\mathbf{State}}, \subseteq, \cup, \cap, \emptyset, \mathbf{State})$
- Abstract lattice:
  $EQ = \{ x\text{=}y \mid x, y \in \mathrm{Var}\}$
  $A = (2^{EQ}, \supseteq, \cap, \cup, EQ, \emptyset)$
  - Treat elements of *A* as both formulas and sets of constraints
- Useful for copy propagation – a compiler optimization
  - $\alpha(X) = ?$
  - $\gamma(\mathrm{Y}) = ?$

Example: lattice of equalities

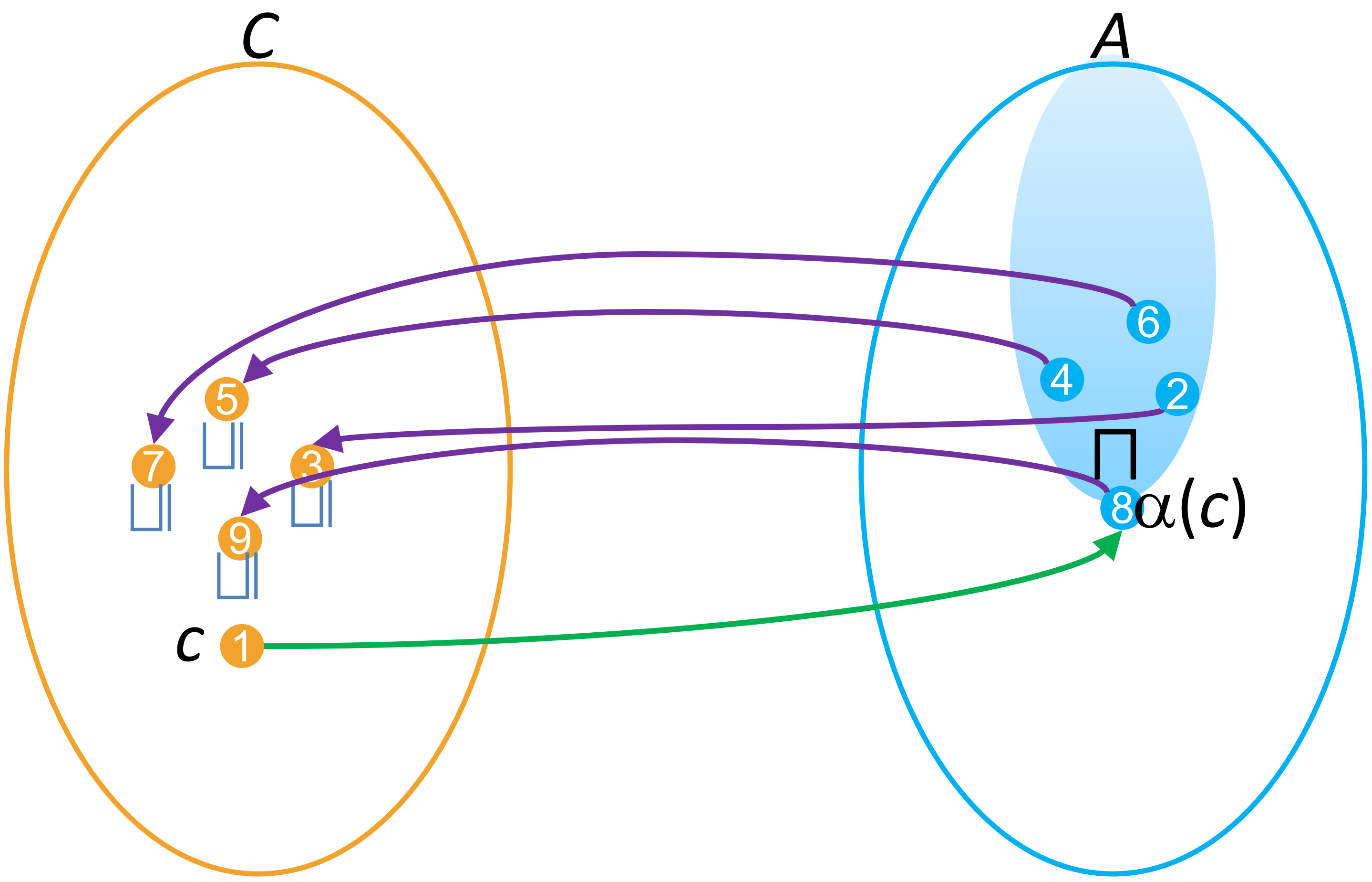
- Concrete lattice:
  $C = (2^{\mathbf{State}}, \subseteq, \cup, \cap, \emptyset, \mathbf{State})$
- Abstract lattice:
  $EQ = \{ x{=}y \mid x, y \in \mathrm{Var}\}$
  $A = (2^{EQ}, \supseteq, \cap, \cup, EQ, \emptyset)$
  – Treat elements of *A* as both formulas and sets of constraints
- Useful for copy propagation – a compiler optimization
- $\beta(s) = \alpha(\{s\}) = \{ x{=}y \mid s\ x = s\ y\}$ that is $s \models x{=}y$
  $\alpha(X) = \cap\{\beta(s) \mid s \in X\} = \sqcup^{A} \{\beta(s) \mid s \in X\}$
  $\gamma(\mathrm{Y}) = \{ s \mid s \models \wedge \mathrm{Y} \} = \mathrm{models}(\wedge \mathrm{Y})$

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



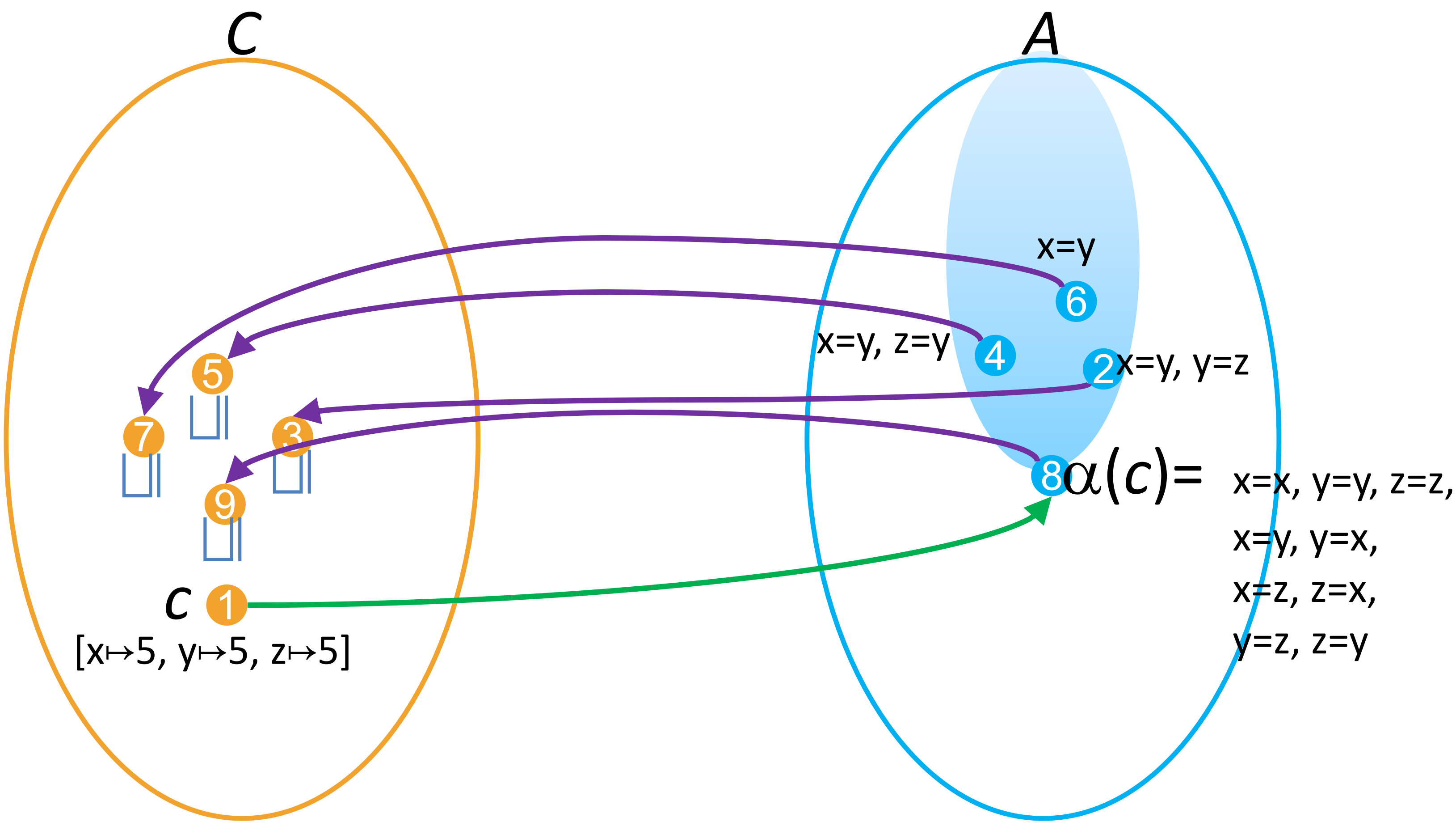
Most precise abstract representation

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

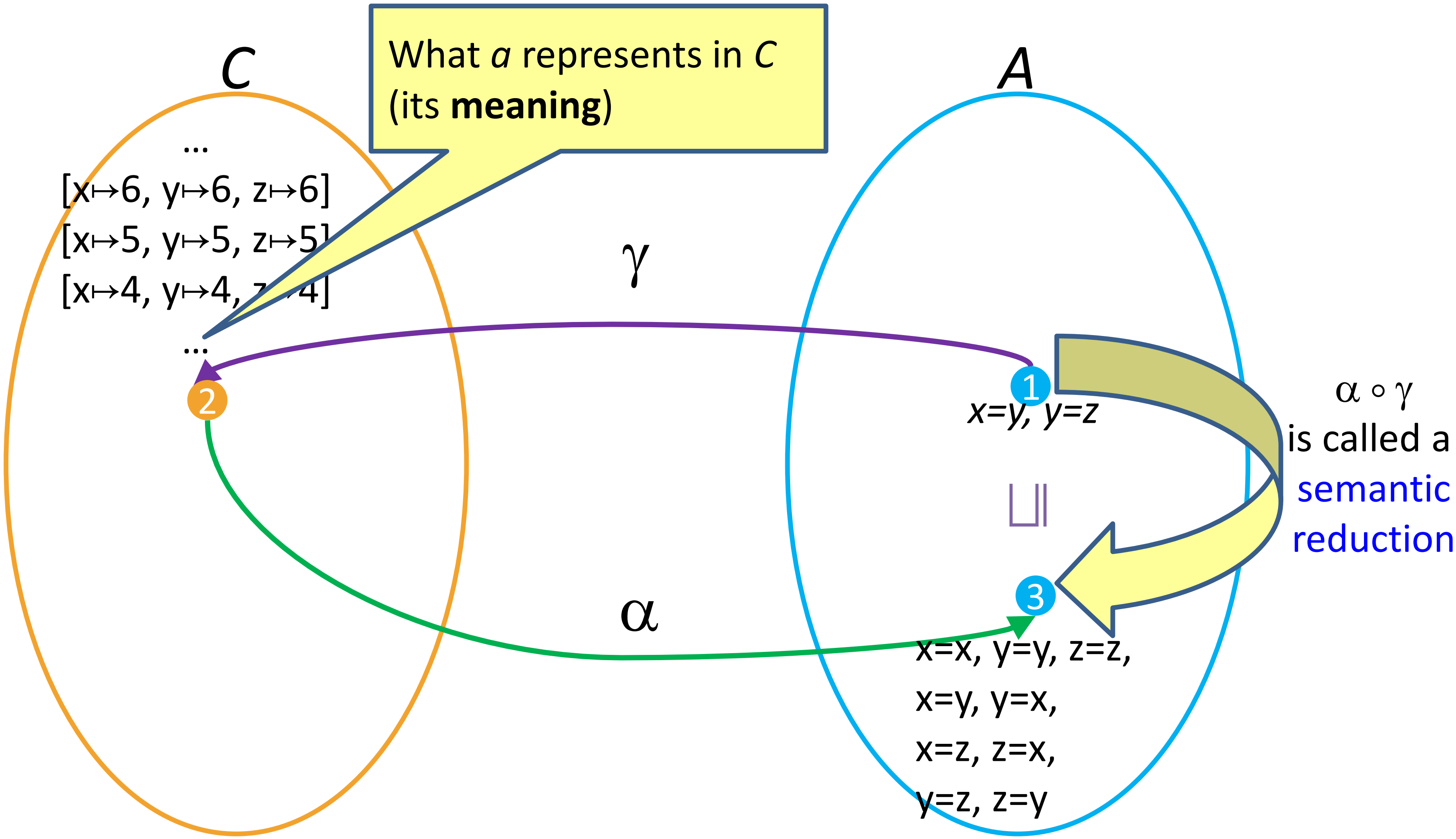


Most precise abstract representation

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

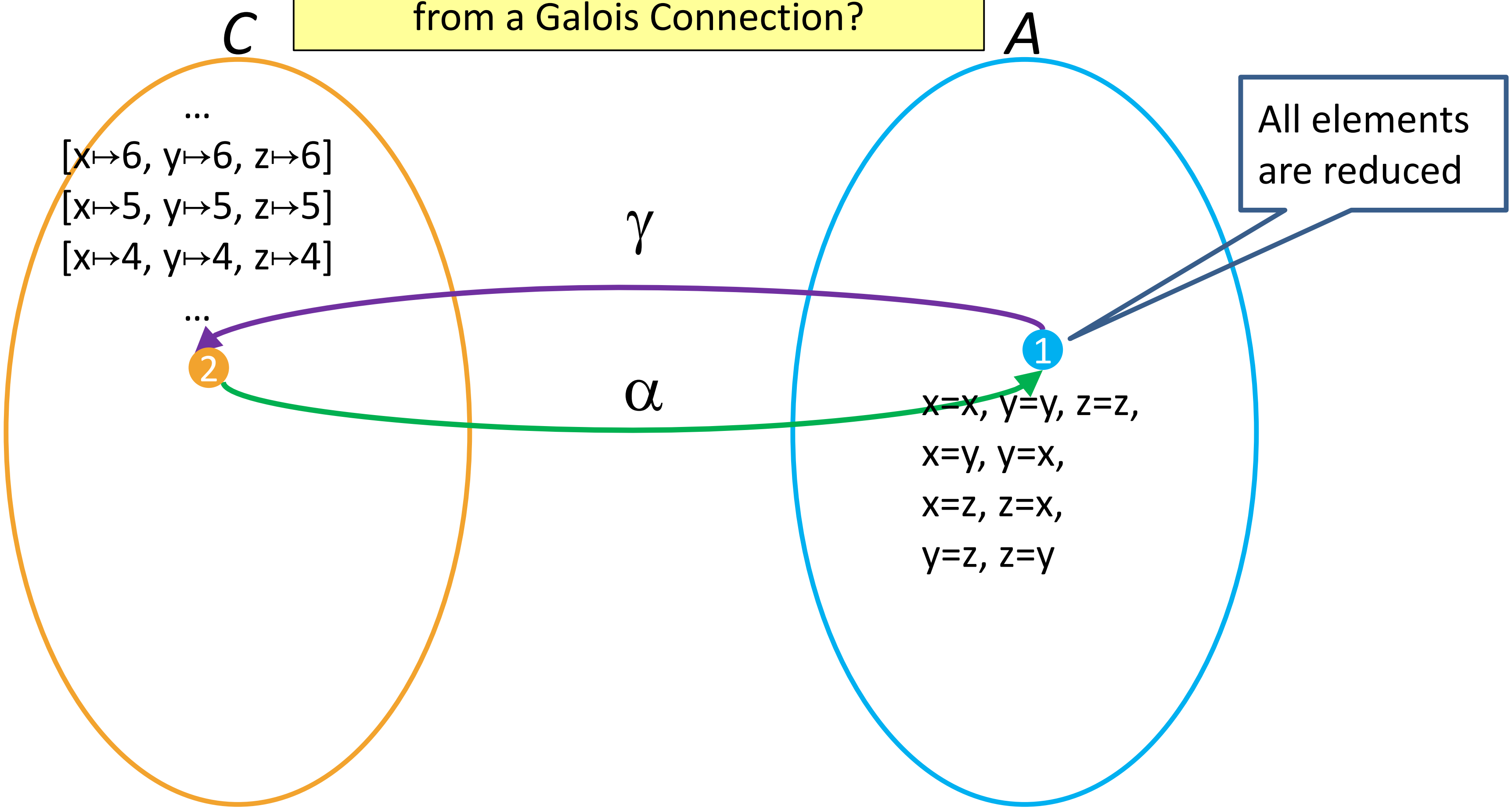


Galois Connection: $\alpha(\gamma(a)) \sqsubseteq a$



Galois Insertion ∀a: $\alpha(\gamma(a))=a$

How can we obtain a Galois Insertion from a Galois Connection?



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 (disjunctive) 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

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