Abstract Interpretation Part I

Mooly Sagiv

Textbook: Chapter 4

The Abstract Interpretation Technique (Cousot & Cousot)

- The foundation of program analysis
- Defines the meaning of the information computed by static tools
- A mathematical framework
- Allows proving that an analysis is sound in a local way
- Identify design bugs
- Understand where precision is lost
- New analysis from old
- Not limited to certain programming style

Outline

- Monotone Frameworks with Widening
- Galois Connections (Insertions)
- Collecting semantics
- The Soundness Theorem

Specialized Chaotic Iterations

Chaotic(G(V, E): Graph, s: Node, L: lattice, ι: L, f: E $\rightarrow$(L $\rightarrow$L)){

for each v in V to n do $df_{entry}[v] := \bot$

In[v] = ι

WL = {s}

while (WL $\neq$ $\emptyset$) do

select and remove an element u $\in$ WL

for each v, such that. (u, v) $\in$E do

temp = f(e)(df_{entry}[u])

new := $df_{entry}(v) \sqcup$ temp

if (new $\neq$ df_{entry}[v]) then

$df_{entry}[v] :=$ new;

WL := WL $\cup${v}

Widening

- Accelerate the termination of Chaotic iterations by computing a more conservative solution
- Can handle lattices of infinite heights

Specialized Chaotic Iterations+ ∇

Chaotic(G(V, E): Graph, s: Node, L: lattice, ι: L, f: E $\rightarrow$(L $\rightarrow$L)){

for each v in V to n do $df_{entry}[v] := \bot$

In[v] = ι

WL = {s}

while (WL $\neq$ $\emptyset$) do

select and remove an element u $\in$ WL

for each v, such that. (u, v) $\in$E do

temp = f(e)(df_{entry}[u])

new := $df_{entry}(v) \nabla$ temp

if (new $\neq$ df_{entry}[v]) then

$df_{entry}[v] :=$ new;

WL := WL $\cup${v}

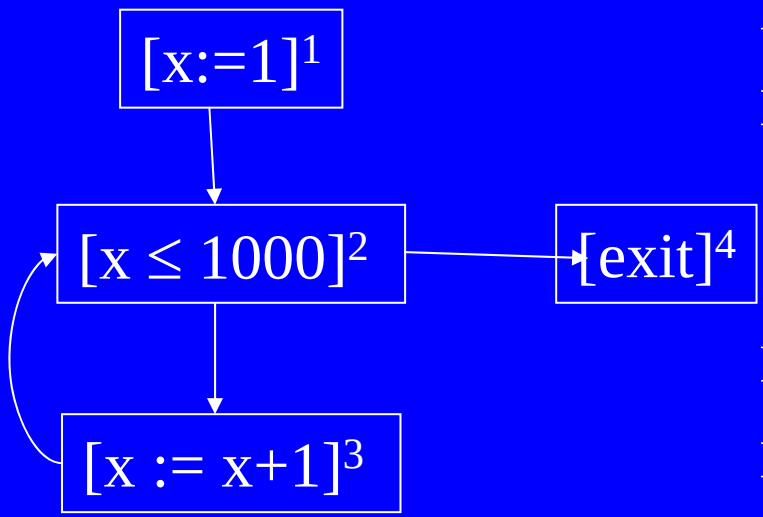
Example Interval Analysis

- Find a lower and an upper bound of the value of a variable
- Usages?
- Lattice

  $L = (Z \cup \{-\infty, \infty\} \times Z \cup \{-\infty, \infty\}, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$
  - $[a, b] \sqsubseteq [c, d]$ if $c \leq a$ and $d \geq b$
  - $[a, b] \sqcup [c, d] = [\min(a, c), \max(b, d)]$
  - $[a, b] \sqcap [c, d] = [\max(a, c), \min(b, d)]$
  - $\top$ =
  - $\bot$ =

Example Program
Interval Analysis

$[x := 1]^1$;
while $[x \leq 1000]^2$ do
    $[x := x + 1;]^3$



IntEntry(1) = [minint,maxint]

IntExit(1) = [1,1]

IntEntry(2) = IntExit(1) ⊔ IntExit(3)

IntExit(2) = IntEntry(2)

IntEntry(3) = IntExit(2) ⊓ [minint,1000]

IntExit(3) = IntEntry(3)+[1,1]

IntEntry(4) = IntExit(2) ⊓ [1001,maxint]

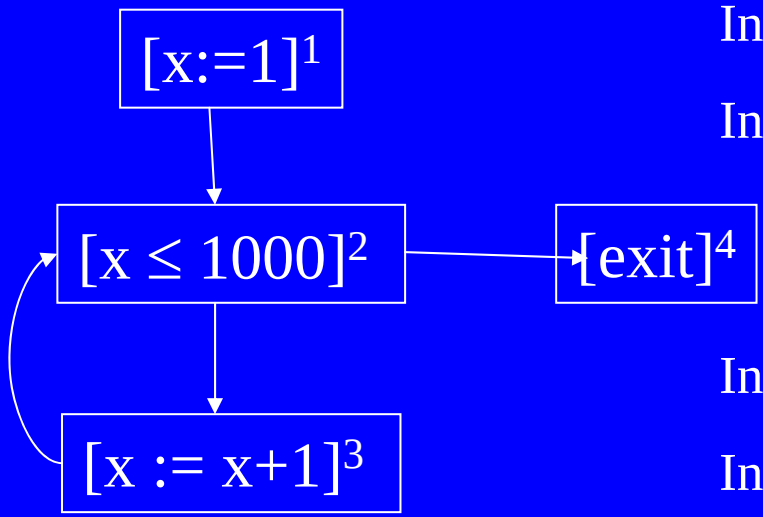
IntExit(4) = IntEntry(4)

Widening for Interval Analysis

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

Example Program
Interval Analysis

$[x := 1]^1$;
while $[x \leq 1000]^2$ do
    $[x := x + 1;]^3$



IntEntry(1) = $[-\infty, \infty]$

IntExit(1) = [1,1]

IntEntry(2) = InExit(2) $\triangledown$ (IntExit(1) $\sqcup$ IntExit(3))

IntExit(2) = IntEntry(2)

IntEntry(3) = IntExit(2) $\sqcap$ $[-\infty,1000]$

IntExit(3) = IntEntry(3)+[1,1]

IntEntry(4) = IntExit(2) $\sqcap$ $[1001, \infty]$

IntExit(4) = IntEntry(4)

Requirements on Widening

- For all elements $l_1 \sqcup l_2 \sqsubseteq l_1 \triangledown l_2$
- For all ascending chains
  $l_0 \sqsubseteq l_1 \sqsubseteq l_2 \sqsubseteq \ldots$
  the following sequence is finite
  - $y_0 = l_0$
  - $y_{i+1} = y_i \triangledown l_{i+1}$
- For a monotonic function
  $f: L \rightarrow L$
  define
  - $x_0 = \perp$
  - $x_{i+1} = x_i \triangledown f(x_i)$
- Theorem:
  - There exits k such that $x_{k+1} = x_k$
  - $x_k \in \text{Red}(f) = \{l: l \in L, f(l) \sqsubseteq l\}$

Narrowing

- Improve 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 monotonic function $f: L \rightarrow L$ and $x \in \mathrm{Red}(f) = \{l: l \in L, f(l) \sqsubseteq l\}$ define
  - $y_0 = x$
  - $y_{i+1} = y_i \triangle f(y_i)$
- Theorem:
  - There exits k such that $y_{k+1} = y_k$
  - $y_k \in \mathrm{Red}(f) = \{l: l \in L, f(l) \sqsubseteq l\}$

Narrowing for Interval Analysis

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

  if $a = -\infty$

  then c

  else a,

  if $b = \infty$

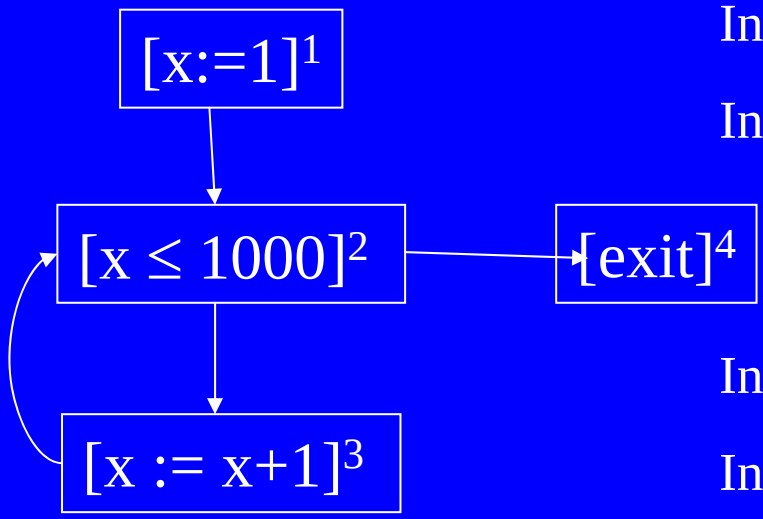
  then d

  else b

  $]$

Example Program
Interval Analysis

$[x := 1]^1$;
while $[x \leq 1000]^2$ do
    $[x := x + 1;]^3$



IntEntry(1) = [-∞ , ∞]

IntExit(1) = [1,1]

IntEntry(2) = InExit(2) △(IntExit(1) ⊔ IntExit(3))

IntExit(2) = IntEntry(2)

IntEntry(3) = IntExit(2) ⊓ [-∞,1000]

IntExit(3) = IntEntry(3)+[1,1]

IntEntry(4) = IntExit(2) ⊓ [1001, ∞]

IntExit(4) = IntEntry(4)

Non Montonicity of Widening

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

Widening and Narrowing Summary

- Very simple but produces impressive precision
- Sometimes non-monotonic
- The McCarthy 91 function

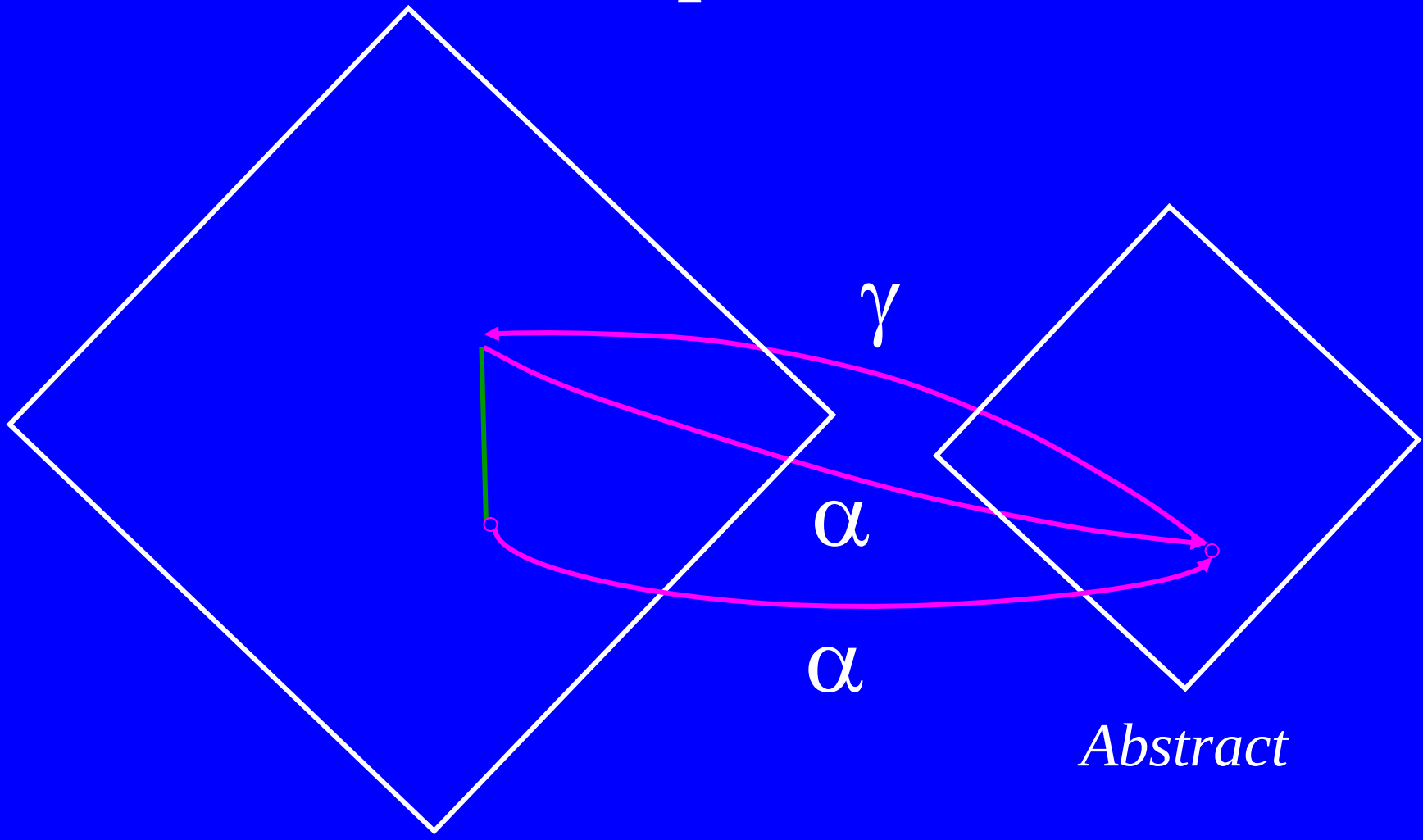
```
int f(x) [-∞ , ∞]
  if x > 100
     then [101, ∞] return x -10 [91, ∞-10];
     else [-∞, 100] return f(f(x+11)) [91, 91] ;
```

- Also useful in the finite case
- Can be used as a methodological tool

Foundation of Static Analysis

- Static analysis can be viewed as interpreting the program over an “abstract domain”
- Execute the program over larger set of execution paths
- Guarantee sound results
  - Every identified constant is indeed a constant
  - But not every constant is identified as such

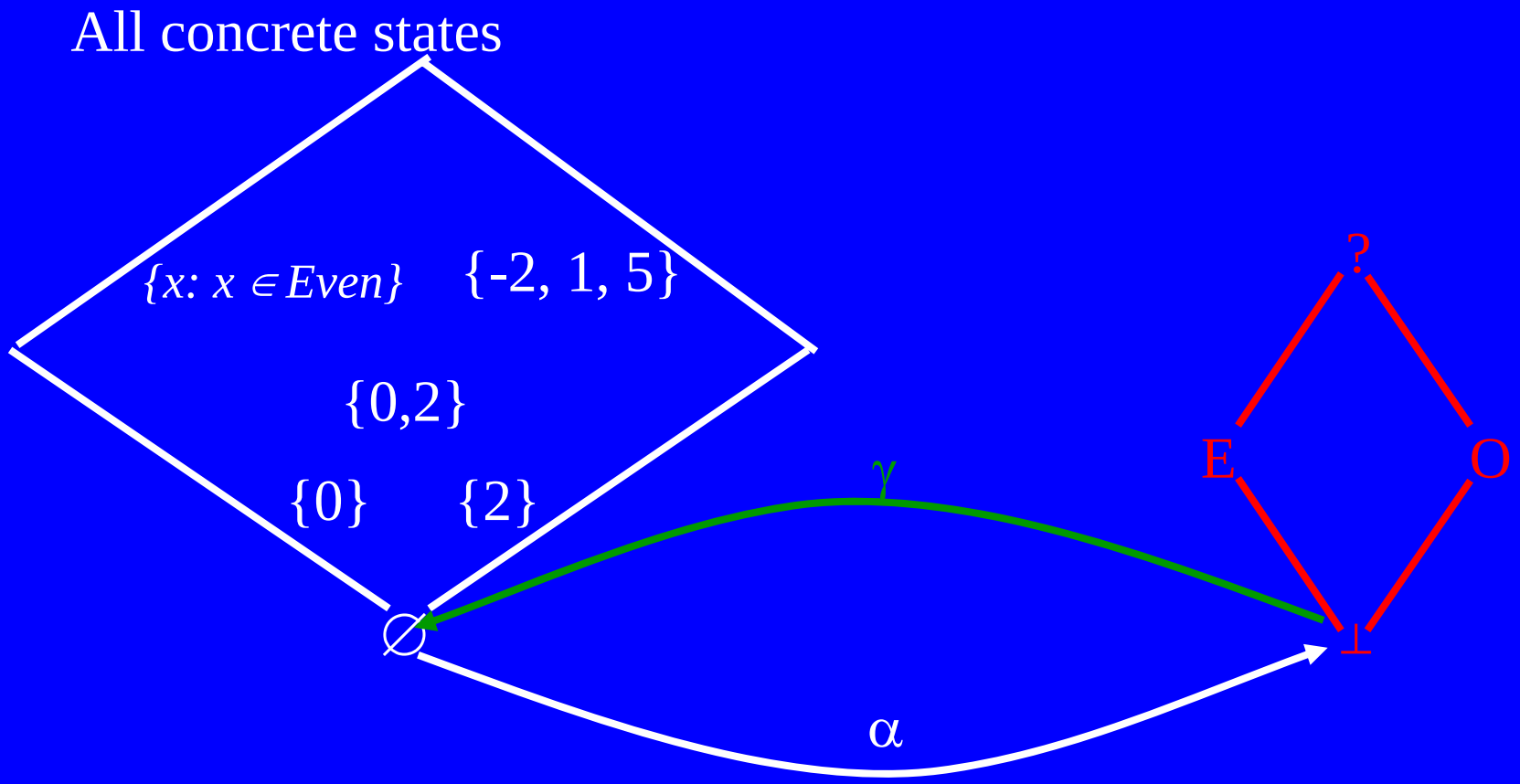
Abstract Interpretation



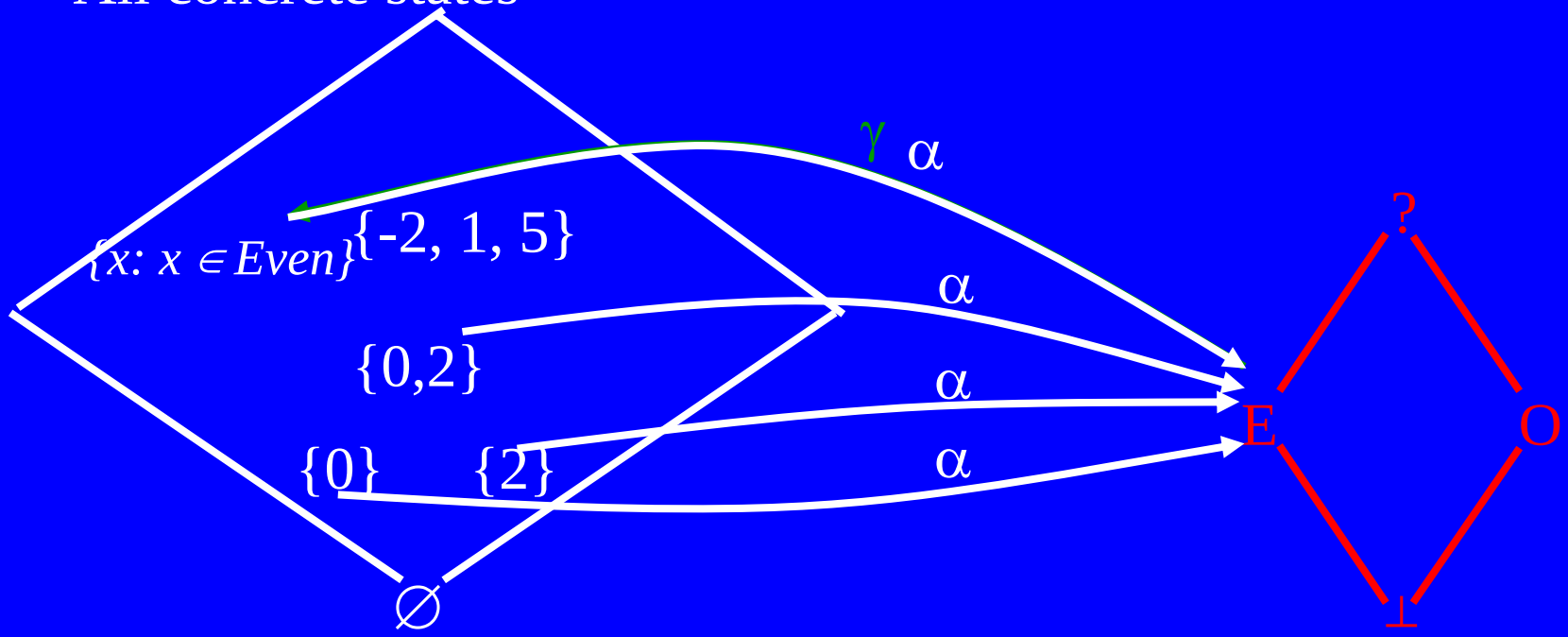
α

Sets of stores → *Descriptors of sets of stores*

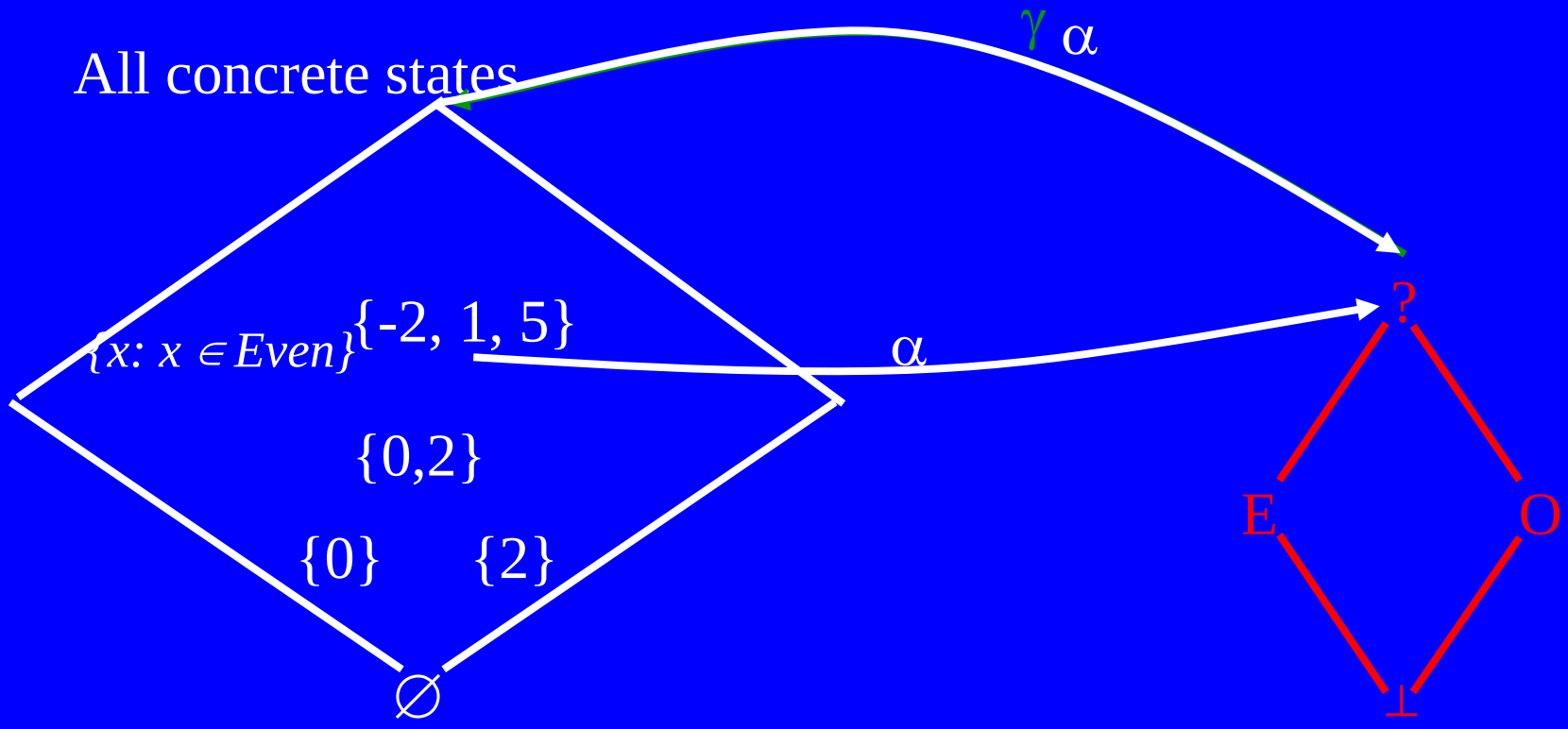
Odd/Even Abstract Interpretation



Odd/Even Abstract Interpretation



Odd/Even Abstract Interpretation



Galois Connections

- A concrete domain $(C, \sqsubseteq)$
- An abstract domain $(A, \sqsubseteq)$
- An abstraction function $\alpha: C \rightarrow A$
- A concretization function $\gamma: A \rightarrow C$
- α is monotone (order-preserving)
- γ is monotone (order-preserving)
- $c \sqsubseteq \gamma(\alpha(c))$
- $\alpha(\gamma(a)) \sqsubseteq a$
- $\alpha(c) \sqsubseteq a \Leftrightarrow c \sqsubseteq \gamma(a)$

More on Galois Connections

- α and γ determine each other
- Defines an upward closure operator up:C $\rightarrow$ C such that c $\sqsubseteq$up(c) and up(up(c)) = up(c) by up(c) = $\gamma(\alpha(c))$
- For C = P($\sum$) let β: C $\rightarrow$ A then the Galois connection is defined by:
  - $\alpha(c) = \sqcup \{\beta(\sigma) \mid \sigma \in c\}$
  - $\gamma(a) = \{\sigma \mid \beta(\sigma) \sqsubseteq a\}$

The Abstraction Function (CP)

- Map collecting states into constants
- The abstraction of an individual state

  $\beta_{CP}:[Var_* \rightarrow Z] \rightarrow [Var_* \rightarrow Z \cup \{\bot, \top\}]$

  $\beta_{CP}(\sigma) = \sigma$
- The abstraction of set of states

  $\alpha_{CP}:P([Var_* \rightarrow Z]) \rightarrow [Var_* \rightarrow Z \cup \{\bot, \top\}]$

  $\alpha_{CP}(CS) = \sqcup \{ \beta_{CP}(\sigma) \mid \sigma \in CS\} = \sqcup\{\sigma \mid \sigma \in CS\}$
- Soundness

  $\alpha_{CP}(\text{Reach}(v)) \sqsubseteq df(v)$
- Completeness

The Concretization Function

- Map constants into collecting states
- The formal meaning of constants
- The concretization

  γ_{CP}: $[Var_* \rightarrow Z \cup \{\bot, \top\}] \rightarrow P([Var_* \rightarrow Z])$

  $\gamma_{CP}(df) = \{\sigma \mid \beta_{CP}(\sigma) \sqsubseteq df\} = \{\sigma \mid \sigma \sqsubseteq df\}$
- Soundness

  $Reach(v) \subseteq \gamma_{CP}(df(v))$
- Optimality

Galois Connection Constant Propagation

- α_{CP} is monotone
- γ_{CP} is monotone
- $\forall$ df $\in [Var_* \rightarrow Z \cup \{\bot, \top\}]$
  - $\alpha_{CP}(\gamma_{CP}(df)) \sqsubseteq df$
- $\forall$ c $\in$ P($[Var_* \rightarrow Z]$)
  - $c_{CP} \sqsubseteq \gamma_{CP}(\alpha_{CP}(C))$

Upper Closure (CP)

More Examples

- Interval Analysis
- Points-to analysis
- Reaching definitions
- Live variable analysis

Collecting Semantics

- The input state is not known at compile-time
- “Collect” all the states for all possible inputs to the program
- The set of reachable states
- No lost of precision
- Need not be computable

A Simple Example Program

$\{[x \mapsto 0, y \mapsto 0, z \mapsto 0]\}$

z = 3

$\{[x \mapsto 0, y \mapsto 0, z \mapsto 3]\}$

x = 1

$\{[x \mapsto 1, y \mapsto 0, z \mapsto 3]\}$

while (x > 0) (

$\{[x \mapsto 1, y \mapsto 0, z \mapsto 3], [x \mapsto 3, y \mapsto 0, z \mapsto 3],\}$

if (x = 1) then y = 7

$\{[x \mapsto 1, y \mapsto 7, z \mapsto 3], [x \mapsto 3, y \mapsto 7, z \mapsto 3]\}$

else y = z + 4

$\{[x \mapsto 1, y \mapsto 7, z \mapsto 3], [x \mapsto 3, y \mapsto 7, z \mapsto 3]\}$

x = 3

$\{[x \mapsto 3, y \mapsto 7, z \mapsto 3]\}$

print y

$\{[x \mapsto 3, y \mapsto 7, z \mapsto 3]\}$

)

Another Example

$\{[x \mapsto 0]\}$

while (true) do

$\{[x \mapsto 0], \{[x \mapsto 1], \{[x \mapsto 2], \ldots \}$

x = x +1

$\{\{[x \mapsto 1], \{[x \mapsto 2], \ldots \}$

Global Soundness Theorem

- If the meaning of every statement is locally sound
- Then, the solution computed by the iterative algorithm overapproximates the collecting semantics
  - α (CS) $\sqsubseteq$ df
  - CS $\sqsubseteq \gamma$ (df)

Example

P {[x↦0]}

while (true) do

P {[x↦0], {[x↦1], {[x↦2], … }

x = x +1

P {{[x↦1], {[x↦2], … }

Bad Example

P $\{[x \mapsto 0]\}$

x = x -1

? $\{[x \mapsto -1]\}$

x = x + 1

? $\{[x \mapsto 0]\}$

An “Iterative” Definition of Collecting Semaics

- Generate a system of monotone equations
- The least solution is well-defined
- The least solution is the collecting interpretation
- But may not be computable

Equations Generated for Collecting Interpretation

- Equations for elementary statements
  - [skip]

    $CS_{exit}(1) = CS_{entry}(l)$
  - [b]

    $CS_{exit}(1) = \{\sigma: \sigma \in CS_{entry}(l), [\![b]\!]\sigma = tt\}$
  - [x := a]

    $CS_{exit}(1) = \{(s[x \mapsto \mathbf{A}[\![a]\!]s]) \mid s \in CS_{entry}(l)\}$
- Equations for control flow constructs

  $CS_{entry}(l) = \bigcup CS_{exit}(l')$ *l'* immediately precedes *l* in the control flow graph
- An equation for the entry

  $CS_{entry}(1) = \{\sigma \mid \sigma \in Var_* \rightarrow Z\}$

System of Equations (Collecting Semantics)

S =

$$\left\{ \begin{array}{l} CS_{entry}[s] = \{\sigma_0\} \\ CS_{entry}[v] = \cup\{f(e)(CS_{entry}[u]) \mid (u, v) \in E \} \\ \text{where } f(e) = \lambda X.\ \{[\![st(e)]\!]\ \sigma \mid \sigma \in X\} \text{ for atomic statements} \end{array} \right\}$$

$$f(e) = \lambda X.\{\sigma \mid [\![b(e)]\!]\ \sigma = tt\ \}$$

$$F_S : L^n \rightarrow L^n$$

$$F_s(X)[v] = \cup\{f(e)[u] \mid (u, v) \in E\ \}$$

$$lfp(S) = lfp(F_S)$$

The Least Solution

- 2n sets of equations
  $CS_{entry}(1), \ldots, CS_{entry}(n), CS_{exit}(1), \ldots, CS_{exit}(n)$
- Can be written in vectorial form
  $$\overrightarrow{CS} = F_{cs}(\overrightarrow{CS})$$
- The least solution $lfp(F_{cs})$ is well-defined
- Every component is minimal
- Since F_{cs} is monotone such a solution always exists
- $CS_{entry}(v) = \{s | \exists s_0 | <P, s_0> \Rightarrow^{*} (S', s)),$
  $init(S')=v\}$
- Simplify the soundness criteria

Example

$[x = 0]^0$

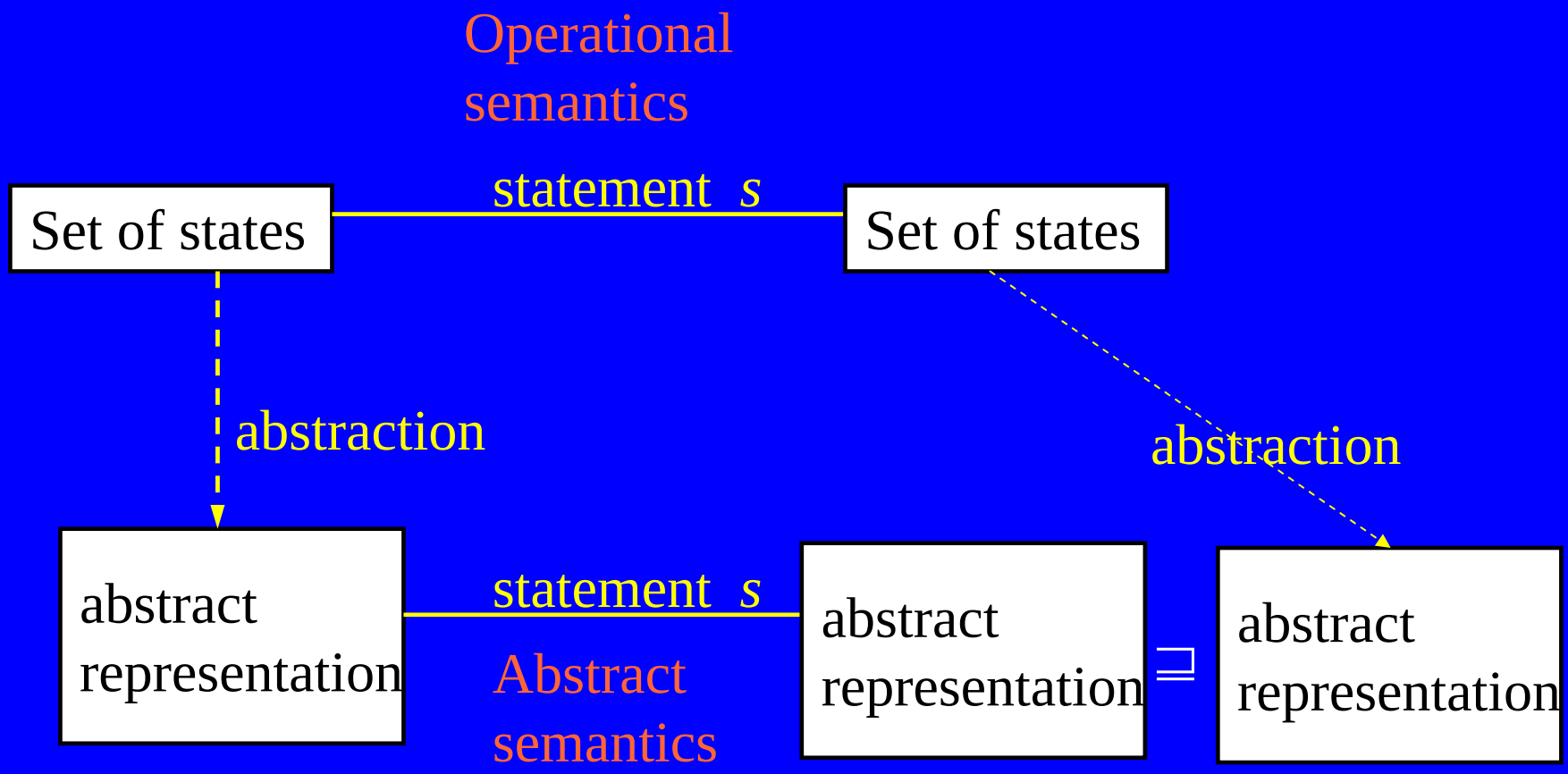
while $[(\text{true})]^1$ do

$[x = x + 1]^2$

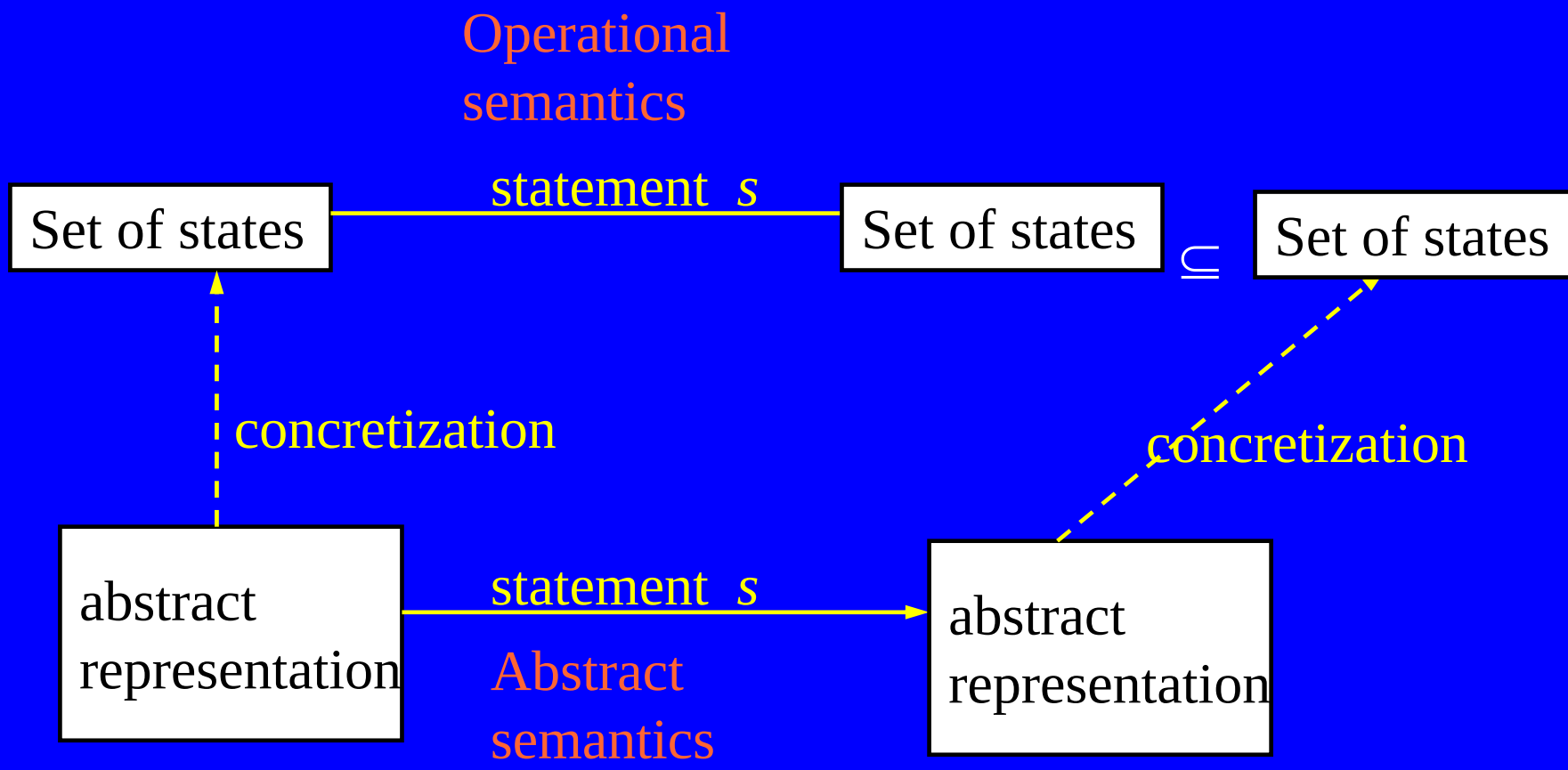
A Low Level View

- An infinite set of states Σ (including control)
- The meaning of the program (small step) is a transition relation $\tau \subseteq \Sigma \times \Sigma$
- Let Σ_s be the set of initial states
- The collecting interpretation system
  $F(CS) = \Sigma_s \cup \{ \sigma \mid \exists \sigma' : \sigma' \in CS \wedge \sigma' \rightarrow_\tau \sigma \}$
- Let A be an abstract domain (lattice)
- Let $a_s \in A$ be the initial abstract element
- Let $\gamma : A \rightarrow P(\Sigma)$ be the concretization
- The abstract meaning of the program (small step) is a transition relation $\tau^\# \subseteq A \times A$
- Local soundness:
- Global soundness:

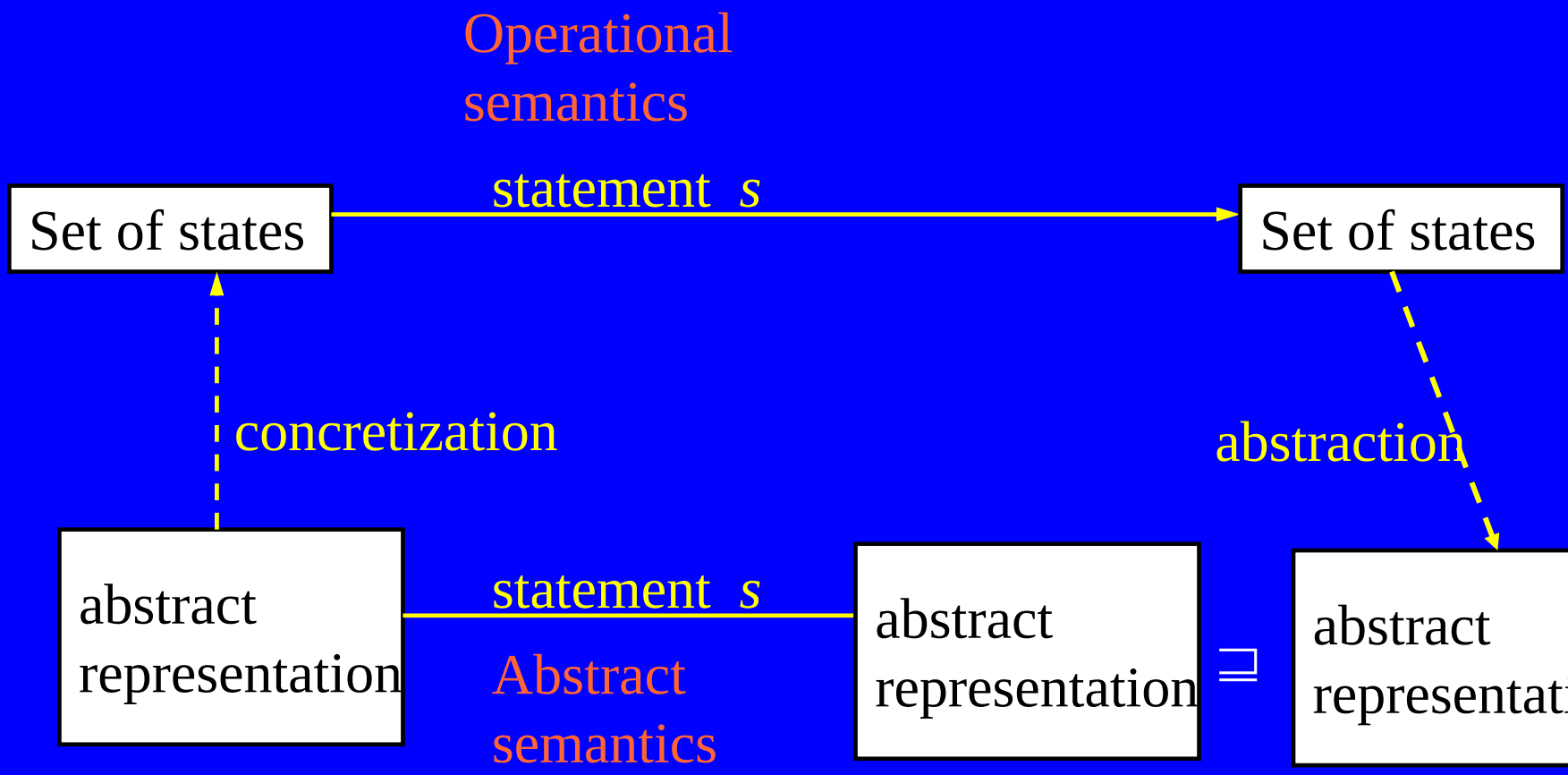
Abstract (Conservative) interpretation



Abstract (Conservative) interpretation



Abstract (Conservative) interpretation



Soundness Theorem(1)

1. Let (α, γ) form Galois connection from C to A
2. $f: C \to C$ be a monotone function
3. $f^{\#} : A \to A$ be a monotone function
4. $\forall a \in A: f(\gamma(a)) \sqsubseteq \gamma(f^{\#}(a))$

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

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

Soundness Theorem(2)

1. Let (α, γ) form Galois connection from C to A
2. $f: C \rightarrow C$ be a monotone function
3. $f^{\#} : A \rightarrow A$ be a monotone function
4. $\forall c \in C: \alpha(f(c)) \sqsubseteq f^{\#}(\alpha(c))$

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

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

Soundness Theorem(3)

1. Let (α, γ) form Galois connection from C to A
2. $f: C \rightarrow C$ be a monotone function
3. $f^{\#} : A \rightarrow A$ be a monotone function
4. $\forall a \in A: \alpha(f(\gamma(a))) \sqsubseteq f^{\#}(a)$

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

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

Local Concrete Semantics

- For every atomic statement S
  - $[\![S]\!] : [Var_* \to Z] \to [Var_* \to Z]$
  - $[\![x := a]\!]s = s[x \mapsto \mathbf{A}[\![a]\!]s]$
  - $[\![skip]\!]s = s$
- For Boolean conditions …

Local Abstract Semantics(CP)

- For every atomic statement S
  - $[\![S]\!]^{\#} : Var_* \rightarrow L \rightarrow Var_* \rightarrow L$
  - $[\![x := a]\!]^{\#}(e) = e[x \mapsto [\![a]\!]^{\#}(e)]$
  - $[\![skip]\!]^{\#}(e) = e$
- For Booleans …

Local Soundness (CP)

- For every atomic statement S show one of the following
  - $\alpha_{CP}(\{[\![S]\!]\sigma \mid \sigma \in CS\} \sqsubseteq [\![S]\!]^{\#}(\alpha_{CP}(CS))$
  - $\{[\![S]\!]\sigma \mid \sigma \in \gamma_{CP}(df)\} \subseteq \gamma_{CP}([\![S]\!]^{\#}(df))$
  - $\alpha(\{[\![S]\!]\sigma \mid \sigma \in \gamma_{CP}(df)\}) \sqsubseteq [\![S]\!]^{\#}(df)$
- The above condition implies global soundness [Cousot & Cousot 1976]

  $\alpha(CS_{entry}(l)) \sqsubseteq df_{entry}(l)$

  $CS_{entry}(l) \subseteq \gamma(df_{entry}(l))$

Lemma 1

Consider a lattice L.

f: $L \rightarrow L$ is monotone iff

for all $X \subseteq L$: $\sqcup\{f(z) \mid z \in X\} \sqsubseteq f(\sqcup\{z \mid z \in X\})$

Assignments in constant propagation

- Monotone
  - $df_1 \sqsubseteq df_2 \rightarrow [\![x := e]\!]^{\#}(df_1) \sqsubseteq [\![x := e]\!]^{\#}(df_2)$
- Local Soundness
  - $\alpha(\{[\![x := e]\!]\sigma \mid \sigma \in CS \} \sqsubseteq [\![x := e]\!]^{\#} (\alpha(CS))$

Proof of Soundness (Summary)

- Define an “appropriate” operational semantics
- Define “collecting” 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 result of the iterative analysis is sound w.r.t. the collecting semantics
- Can be applied between different abstractions

Induced Analysis (Relatively Optimal)

- It is sometimes possible to show that a given analysis is not only sound but optimal w.r.t. the chosen abstraction
  - but not necessarily optimal!
- Define
  $[\![S]\!]^{\#}(df) = \alpha(\{[\![S]\!]\sigma \mid \sigma \in \gamma(df)\})$
- But this $[\![S]\!]^{\#}$ may not be computable
- Derive (at compiler-generation time) an alternative form for $[\![S]\!]^{\#}$
- A useful measure to decide if the abstraction must lead to overly imprecise results

Notions of precision

- CS = γ (df)
- α(CS) = df
- Meet(Join) over all paths
- Using best transformers
- Good enough

Conclusions

- Abstract interpretation relates runtime semantics and static information
- The concrete semantics serves as a tool in designing abstractions
- Understanding concretization is a must
- Understand what is preserved/lost