Iterative Program Analysis
Abstract Interpretation

Mooly Sagiv

http://www.cs.tau.ac.il/~msagiv/courses/pa12-13.html

Tel Aviv University

640-6706

Textbook: Principles of Program Analysis

Chapter 4

CC79, CC92

Specialized Chaotic Iterations System of Equations

$S =$

$$\left\{ \begin{array}{l} df_{entry}[s] = \iota \\ df_{entry}[v] = \sqcup\{f(u, v)\ (df_{entry}[u]) \mid (u, v) \in E\ \} \end{array} \right\}$$

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

$$F_S\ (X)[s] = \iota$$

$$F_S(X)[v] = \sqcup\{f(u, v)(X[u]) \mid (u, v) \in E\ \}$$

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

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 $df_{entry}[v] := \perp$

$df[s] = \iota$

WL = {s}

while (WL $\neq$ Ø) do

select and remove an element u $\in$ WL

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

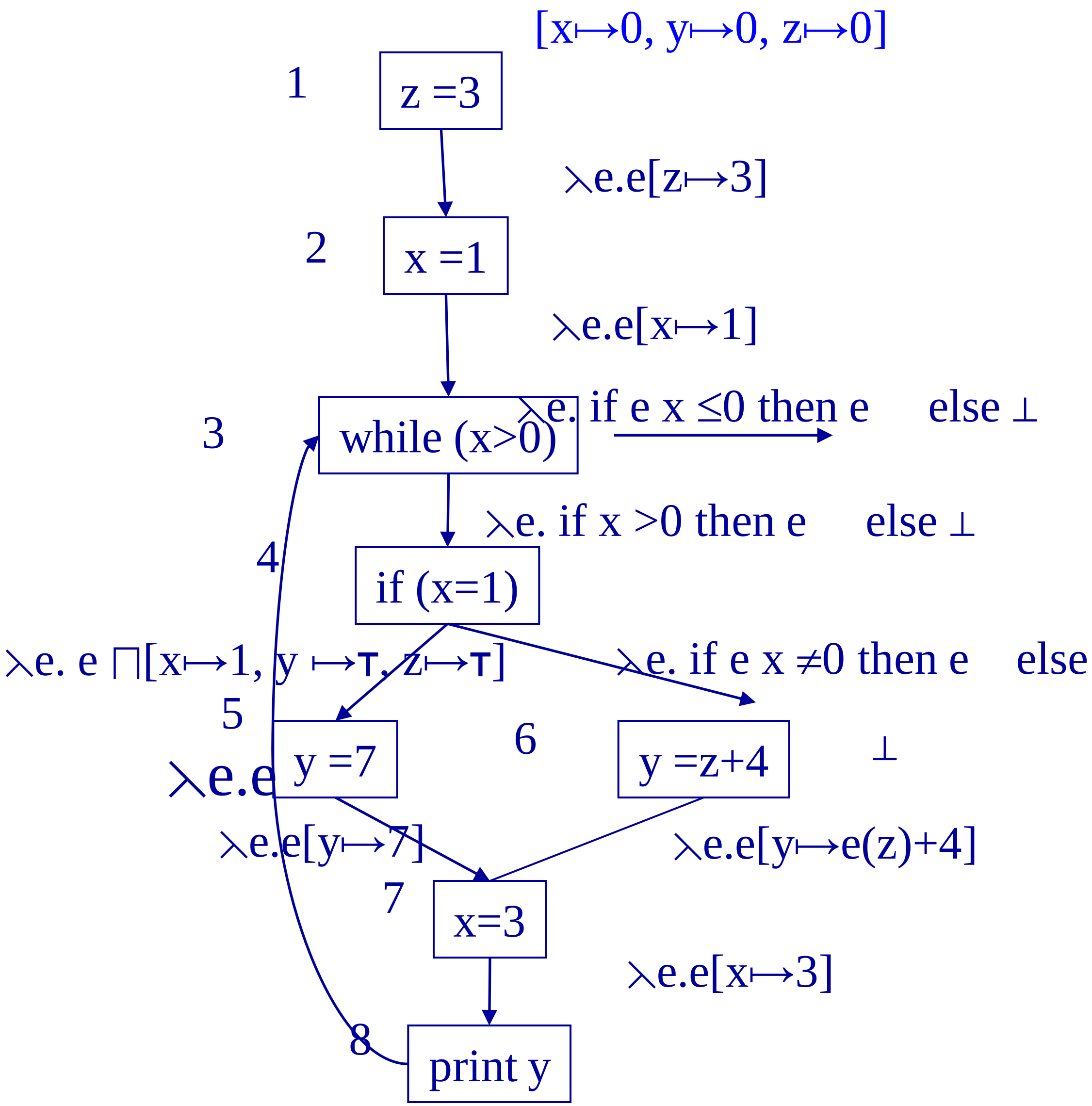
temp = $f(e)(df_{entry}[u])$

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

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

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

WL := WL $\cup$\{v\}

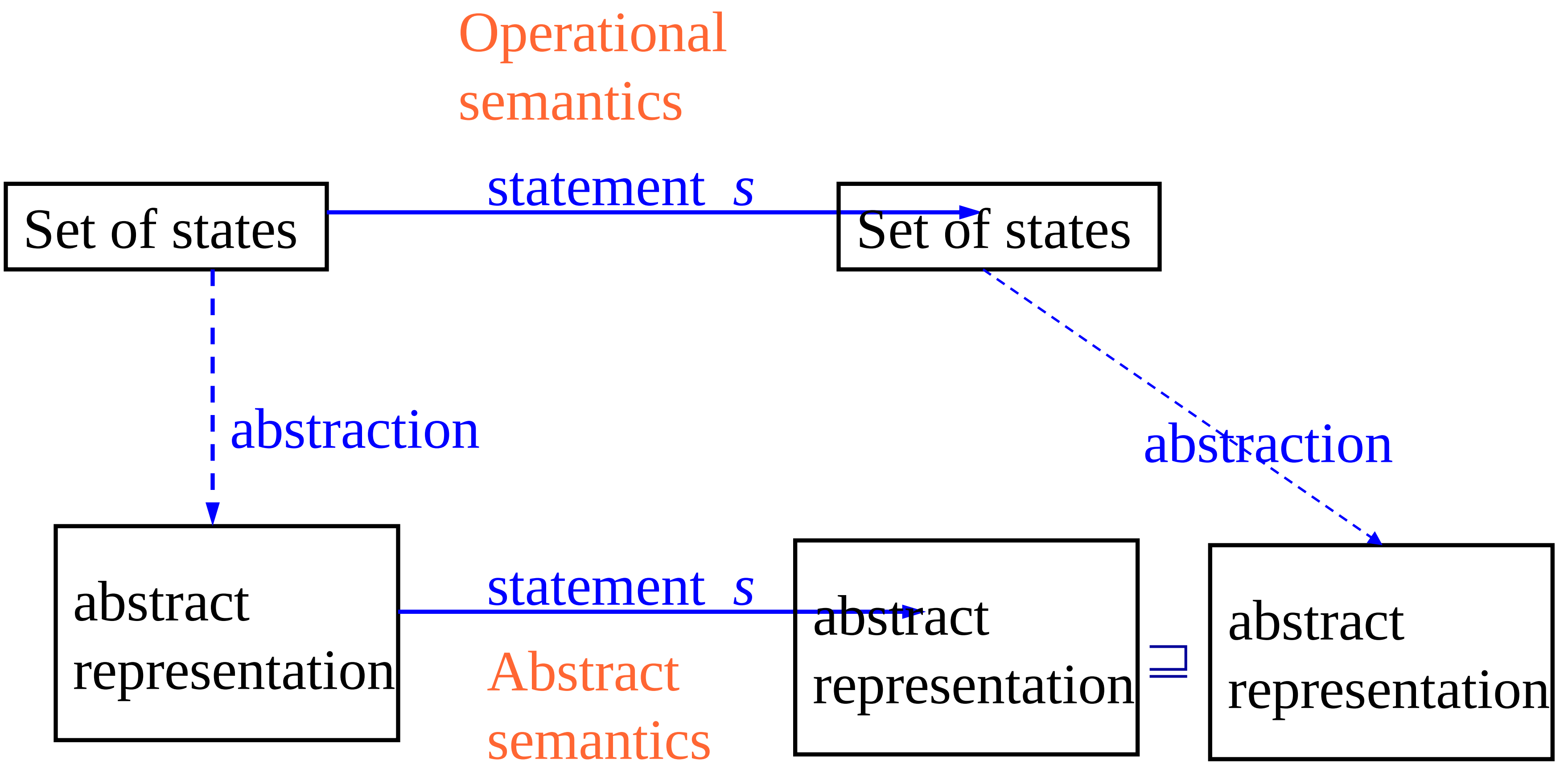


WL	$df_{entry}[v]$
{1}	
{2}	$df[2]:=[x \mapsto 0, y \mapsto 0, z \mapsto 3]$
{3}	$df[3]:=[x \mapsto 1, y \mapsto 0, z \mapsto 3]$
{4}	$df[4]:=[x \mapsto 1, y \mapsto 0, z \mapsto 3]$
{5}	$df[5]:=[x \mapsto 1, y \mapsto 0, z \mapsto 3]$
{7}	$df[7]:=[x \mapsto 1, y \mapsto 7, z \mapsto 3]$
{8}	$df[8]:=[x \mapsto 3, y \mapsto 7, z \mapsto 3]$
{3}	$df[3]:=[x \mapsto \top, y \mapsto \top, z \mapsto 3]$
{4}	$df[4]:=[x \mapsto \top, y \mapsto \top, z \mapsto 3]$
{5,6}	$df[5]:=[x \mapsto 1, y \mapsto \top, z \mapsto 3]$
{6,7}	$df[6]:=[x \mapsto \top, y \mapsto \top, z \mapsto 3]$
{7}	$df[7]:=[x \mapsto \top, y \mapsto 7, z \mapsto 3]$

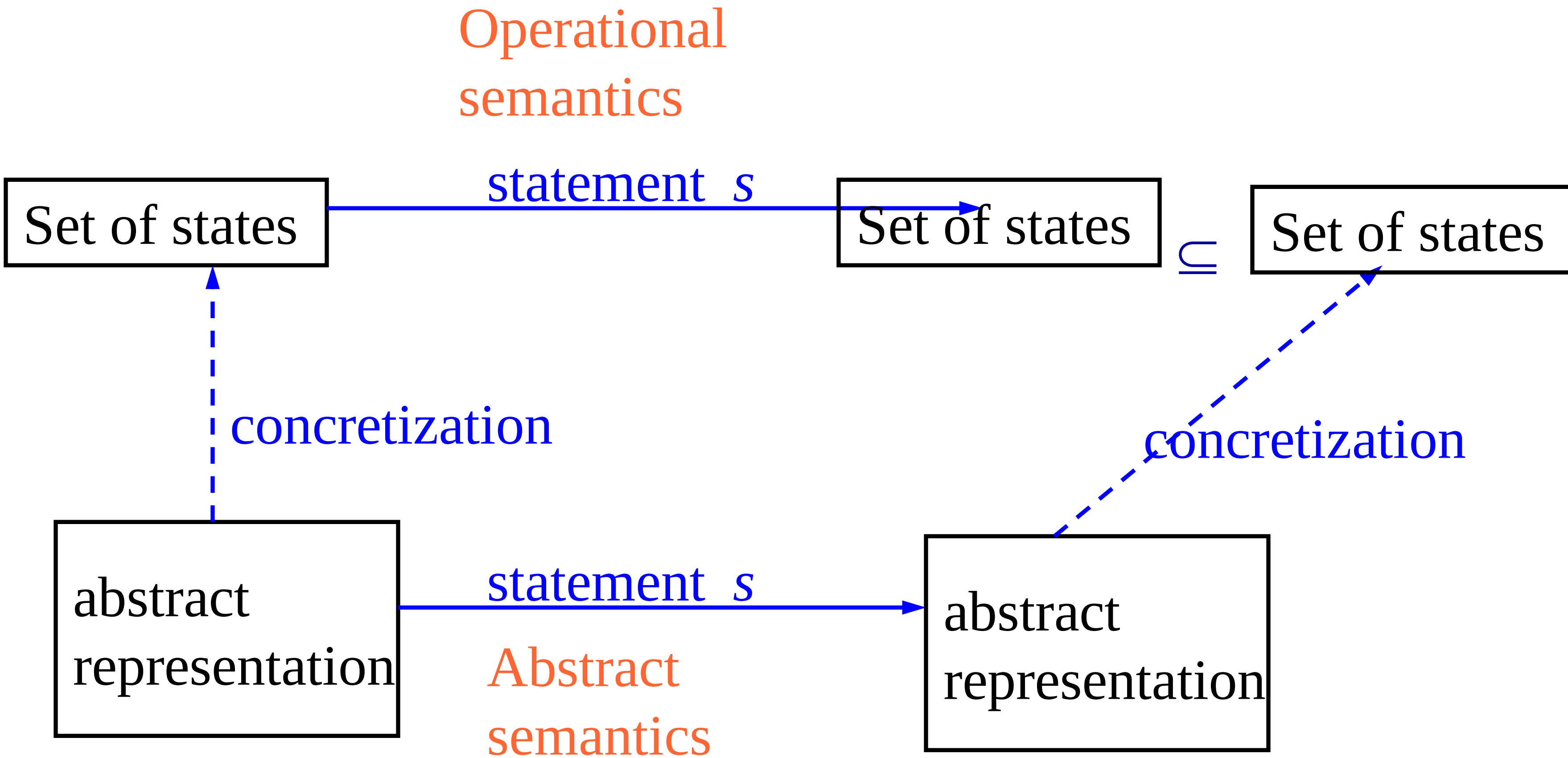
The Abstract Interpretation Technique (Cousot & Cousot)

- 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

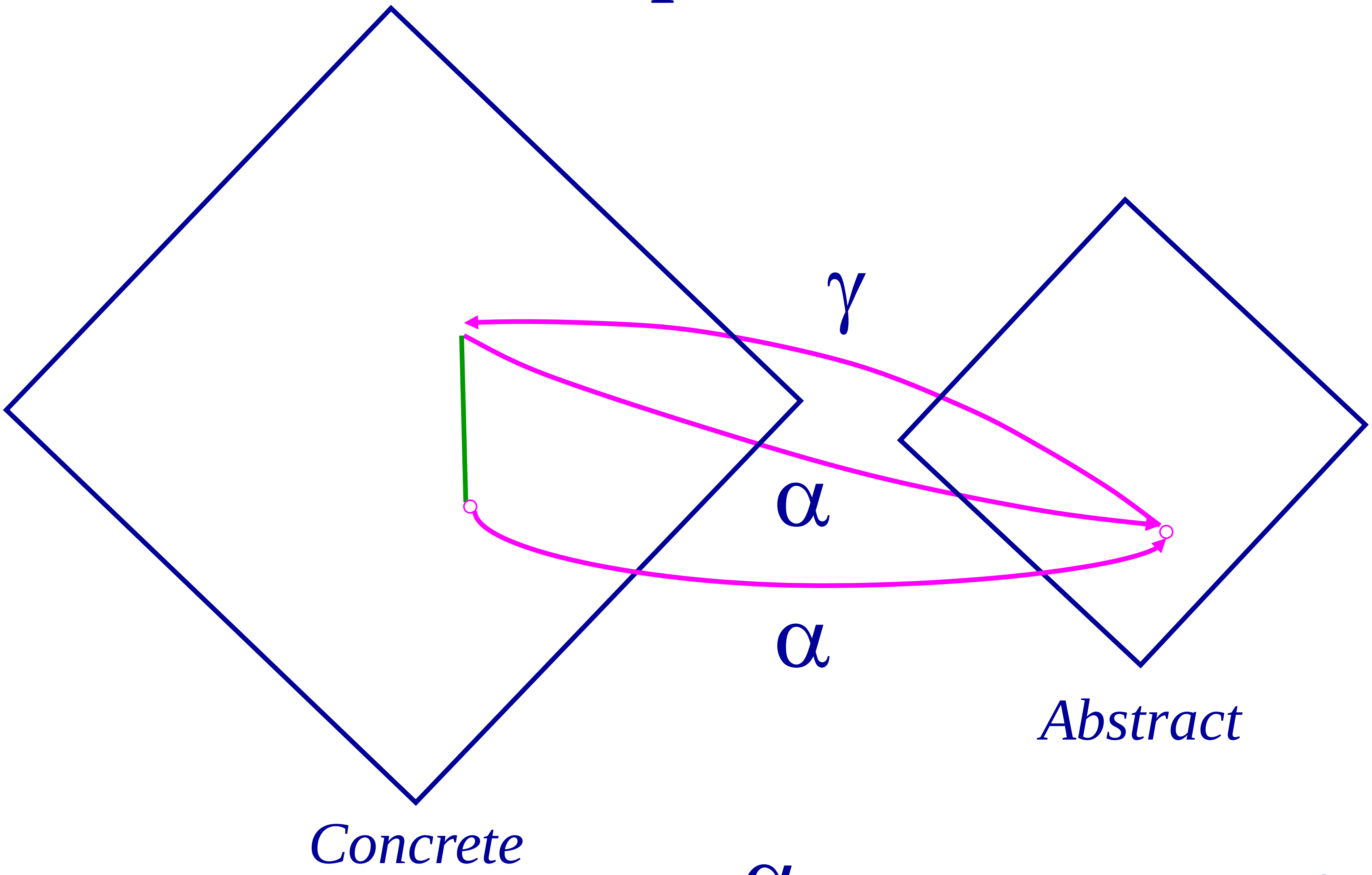
Abstract (Conservative) interpretation



Abstract (Conservative) interpretation



Abstract Interpretation



Sets of stores $\xrightarrow{\alpha}$ *Descriptors of sets of stores*

Galois Connections

- Lattices C and A and functions α: C $\rightarrow$A and γ: A $\rightarrow$C
- The pair of functions (α, γ) form
  Galois connection if
  - α and γ are monotone
  - $\forall$ a $\in$ A
    - $\alpha(\gamma(a)) \sqsubseteq a$
  - $\forall$ c $\in$ C
    - $c \sqsubseteq \gamma(\alpha(C))$
- Alternatively if:
  $\forall$ c $\in$ C
  $\forall$ a $\in$ A
  $$\alpha(c) \sqsubseteq a \text{ iff } c \sqsubseteq \gamma(a)$$
- α and γ uniquely determine each other

The Abstraction Function (CP)

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

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

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

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

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

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

The Concretization Function

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

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

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

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

Galois Connection Constant Propagation

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

Upper Closures

- Define abstractions on sets of concrete states
- $\uparrow: P(\Sigma) \rightarrow P(\Sigma)$ such that
  - $\uparrow$ is monotone, i.e., $X \subseteq Y \rightarrow \uparrow X \subseteq \uparrow Y$
  - $\uparrow$ is extensive, i.e., $\uparrow X \supseteq X$
  - $\uparrow$ is closure, i.e., $\uparrow(\uparrow X) = \uparrow X$
- Every Galois connection defines an upper closure

Proof of Soundness

- Define an “appropriate” operational semantics
- Define “collecting” operational semantics by pointwise extension
- 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

Collecting Semantics

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

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= 0
while (true) do
  x = x +1
```

An “Iterative” Definition

- 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]

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

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

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

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

Specialized Chaotic Iterations System of Equations (Collecting Semantics)

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

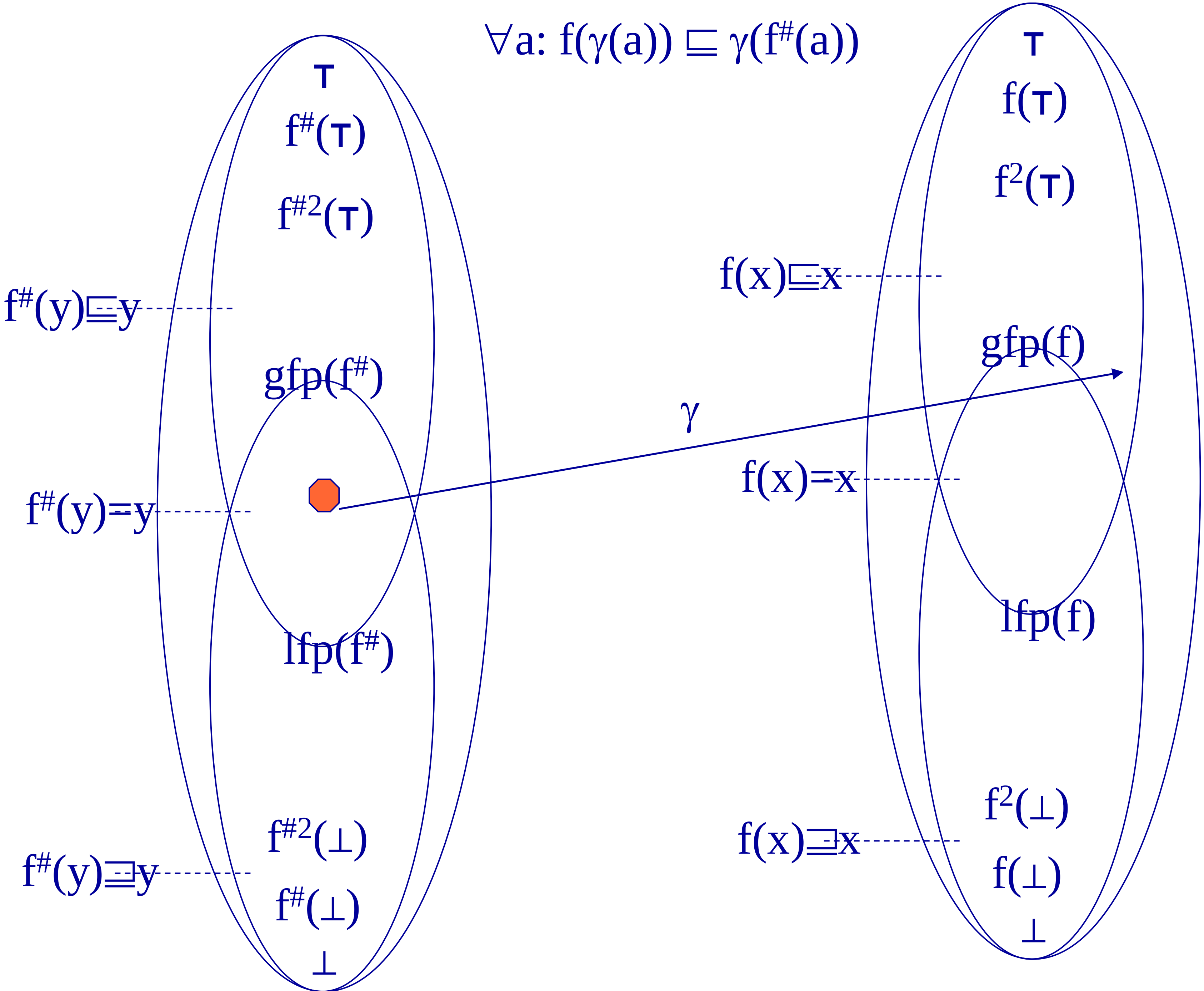
$$F_S : L^n \rightarrow L^n$$

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

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

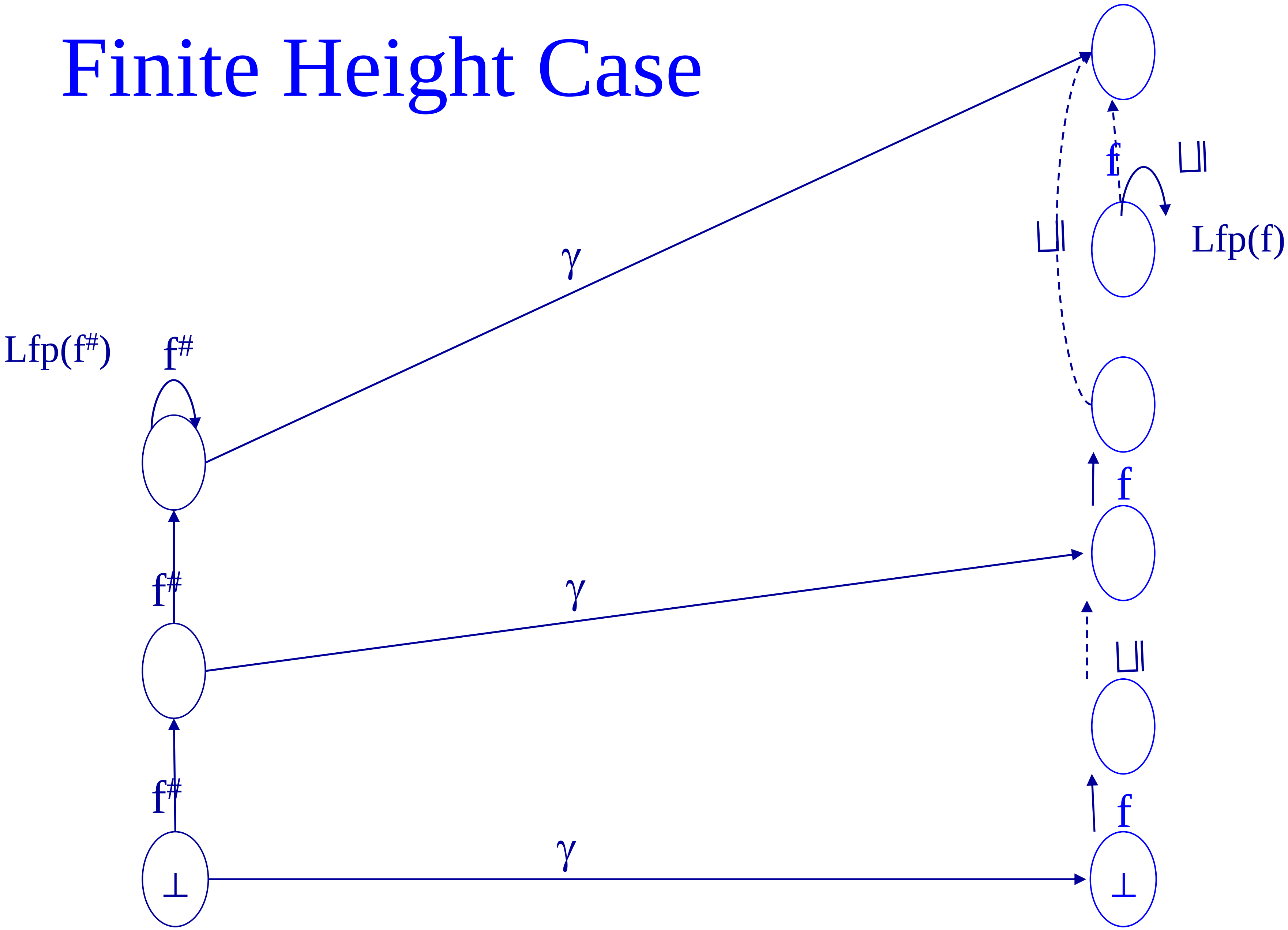
The Least Solution

- 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
- $CS_{entry}(v) = \{s | \exists s_0 | \langle P, s_0 \rangle \Rightarrow^* (S', s)),$
  $init(S')=v\}$
- Simplify the soundness criteria



∀a: f(γ(a)) ⊑ γ(f#(a))
⊤
f#(⊤)
f#2(⊤)
f#(y)⊑y
gfp(f#)
f#(y)=y
lfp(f#)
f#2(⊥)
f#(y)⊒y
f#(⊥)
⊥
γ
⊤
f(⊤)
f2(⊤)
f(x)⊑x
gfp(f)
f(x)=x
lfp(f)
f2(⊥)
f(x)⊒x
f(⊥)
⊥

Finite Height Case



Soundness Theorem(1)

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. $\forall a \in A$: $f(\gamma(a)) \sqsubseteq \gamma(f^{\#}(a))$

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

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

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. $\forall a \in A: \alpha(f(\gamma(a))) \sqsubseteq f^{\#}(a)$

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

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

Proof of Soundness (Summary)

- Define an “appropriate” structural operational semantics
- Define “collecting” structural operational semantics
- Establish a Galois connection between collecting states and reaching definitions
- (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

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

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

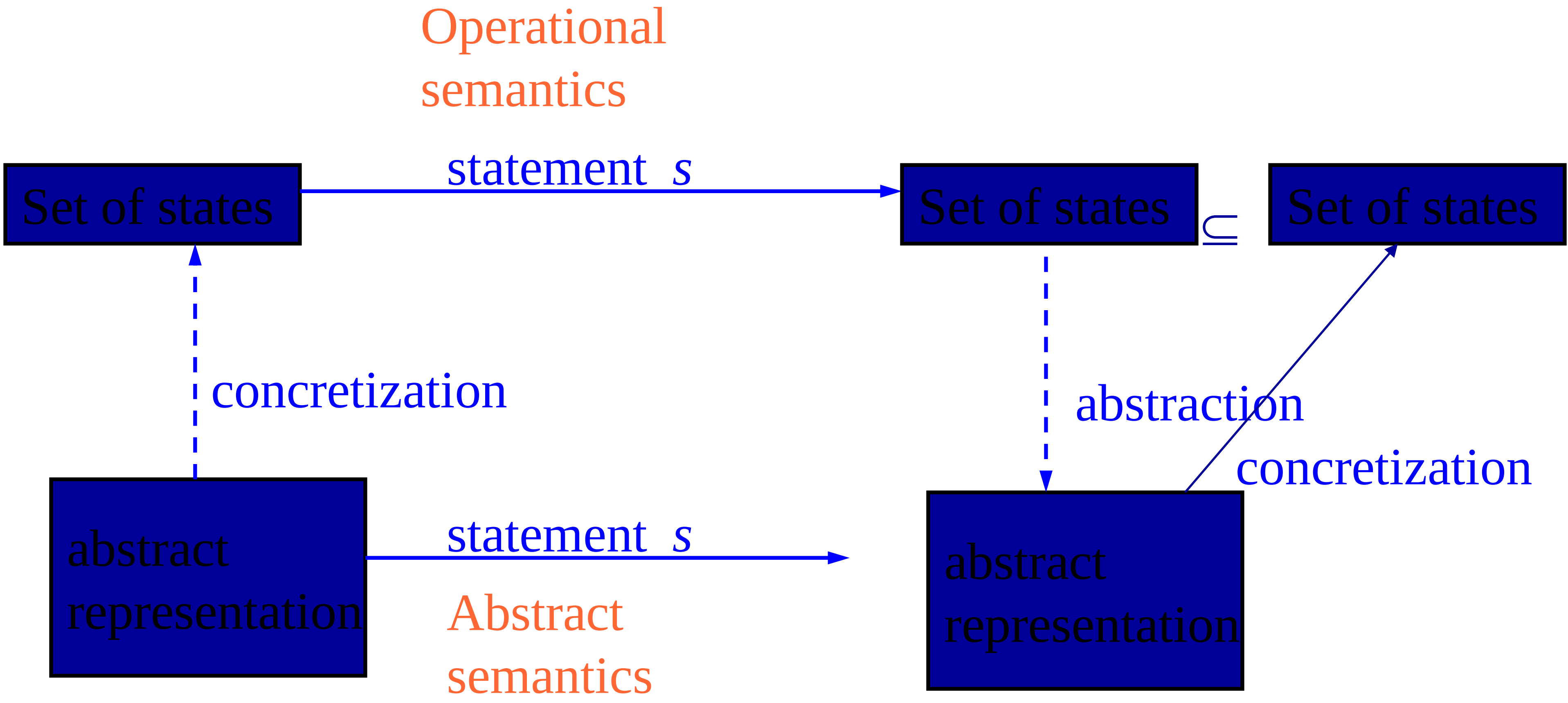
Constant Propagation

- β: [Var $\to$ Z] $\to$ [Var $\to$ Z$\cup\{\top, \perp\}$]
  - $\beta(\sigma) = (\sigma)$
- α: P([Var $\to$ Z]) $\to$ [Var $\to$ Z$\cup\{\top, \perp\}$]
  - $\alpha(X) = \sqcup \{\beta(\sigma) \mid \sigma \in X\} = \sqcup \{\sigma \mid \sigma \in X\}$
- γ:[Var $\to$ Z $\cup\{\top, \perp\}$] $\to$ P([Var $\to$ Z])
  - $\gamma(\sigma^{\#}) = \{\sigma \mid \beta(\sigma) \sqsubseteq \sigma^{\#} \} = \{\sigma \mid \sigma \sqsubseteq \sigma^{\#} \}$
- Local Soundness
  - $[\![st]\!]^{\#}(\sigma^{\#}) \sqsupseteq \alpha(\{ [\![st]\!]\ \sigma \mid \sigma \in \gamma(\sigma^{\#}) = \sqcup \{ [\![st]\!]\ \sigma \mid \sigma \sqsubseteq \sigma^{\#} \}$
- Optimality (Induced)
  - $[\![st]\!]^{\#}(\sigma^{\#}) = \alpha(\{ [\![st]\!]\ \sigma \mid \sigma \in \gamma(\sigma^{\#})\} = \sqcup \{ [\![st]\!]\ \sigma \mid \sigma \sqsubseteq \sigma^{\#} \}$
- Soundness
- Completeness

Proof of Soundness (Summary)

- Define an "appropriate" structural operational semantics
- Define "collecting" structural operational semantics
- Establish a Galois connection between collecting states and reaching definitions
- (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

Best (Conservative) interpretation



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

Example Dataflow Problem

- Formal available expression analysis
- Find out which expressions are available at a given program point
- Example program

```
x = y + t
z = y + r
while (...) {
      t = t + (y + r)
            }
```

- Lattice
- Galois connection
- Basic statements
- Soundness

Example: May-Be-Garbage

- A variable x may-be-garbage at a program point v if there exists a execution path leading to v in which x's value is unpredictable:
  - Was not assigned
  - Was assigned using an unpredictable expression
- Lattice
- Galois connection
- Basic statements
- Soundness

Points-To Analysis

- Determine if a pointer variable p may point to q
  on some path leading to a program point
- "Adapt" other optimizations
  - Constant propagation
    x = 5;
    *p = 7 ;
    … x …
- Pointer aliases
  - Variables p and q are may-aliases at v if the points-to set at v contains entries (p, x) and (q, x)
- Side-effect analysis
  *p = *q + * * t

The **PWhile** Programming Language
Abstract Syntax

$a := x \mid *x \mid \&x \mid n \mid a_1\ op_a\ a_2$

$b :=$ true | false | not b | $b_1\ op_b\ b_2 \mid a_1\ op_r\ a_2$

$S := x := a \mid *x := a \mid$ skip | S_1 ; S_2 |
if b **then** S_1 **else** S_2 | **while** b do S

Concrete Semantics 1 for PWhile

State1= [Loc→Loc∪Z]

For every atomic statement S

$[\![S]\!] : \text{States1} \to \text{States1}$

$[\![x := a]\!](\sigma)=\sigma[loc(x) \mapsto \mathbf{A}[\![a]\!]\ \sigma]$

$[\![x := \&y]\!](\sigma)$

$[\![x := *y]\!](\sigma)$

$[\![x := y]\!](\sigma)$

$[\![*x := y]\!](\sigma)$

Points-To Analysis

- Lattice L_{pt} =
- Galois connection

Abstract Semantics for PWhile

- For every atomic statement S

  $[\![S]\!]^{\#}: P(Var^{*} \times Var^{*}) \rightarrow P(Var^{*} \times Var^{*})$

  $[\![x := \&y]\!]^{\#}$

  $[\![x := {*y}]\!]^{\#}$

  $[\![x := y]\!]^{\#}$

  $[\![{*x} := y]\!]^{\#}$

```
t := &a;
y := &b;

z := &c;

 if x> 0;
         then p:= &y;
         else p:= &z;

 *p := t;
```

```
/* ∅ */ t := &a; /* {(t, a)}*/
 /* {(t, a)}*/  y := &b; /* {(t, a), (y, b) }*/

/* {(t, a), (y, b)}*/ z := &c; /* {(t, a), (y, b), (z, c) }*/

 if x> 0;
         then p:= &y; /* {(t, a),  (y, b), (z, c), (p, y)}*/


         else p:= &z; /* {(t, a),  (y, b), (z, c), (p, z)}*/
/* {(t, a),  (y, b), (z, c), (p, y), (p, z)}*/

 *p := t;

/* {(t, a), (y, b), (y, c), (p, y), (p, z), (y, a), (z, a)}*/
```

Flow insensitive points-to-analysis Steengard 1996

- Ignore control flow
- One set of points-to per program
- Can be represented as a directed graph
- Conservative approximation
  - Accumulate pointers
- Can be computed in almost linear time

```
t := &a;
y := &b;

z := &c;

 if x> 0;
          then p:= &y;
          else p:= &z;

 *p := t;
```

Precision

- We cannot usually have
  - $\alpha(CS) = DF$
    on all programs
- But can we say something about precision in all programs?

Summary

- Abstract interpretation Connects Abstract and Concrete Semantics
- Galois Connection
- Local Correctness
- Global Correctness