CS 150 APL: Abstract Interpretation

Guannan Wei
Tufts University
Nov 13

Ensuring Correctness of Programs

- Empirical approaches
  - Testing
  - Debugging
  - Runtime monitoring
- Formal approaches
  - Deductive verification
    - Use theorem prover and program logic (eg Hoare logic, etc.), manual efforts
  - Model checking
    - Automated, precise, but usually finite models
  - Static analysis
    - Automated, deals with infinite models with abstraction, but can be imprecise

Static Analysis

- What can we say about programs without running the program?
- Examples
  - Buffer overflow
  - Division by 0
  - Null pointer
  - Control-flow and points-to information
  - Termination
  - ...

- Abstract interpretation: the theory behind static analysis

Abstract interpretation

ABSTRACT INTERPRETATION : A UNIFIED LATTICE MODEL FOR STATIC ANALYSIS
OF PROGRAMS BY CONSTRUCTION OR APPROXIMATION OF FIXPOINTS

Patrick Cousot* and Radhia Cousot**

Laboratoire d'Informatique, U.S.M.G., BP. 53
38041 Grenoble cedex, France

1. Introduction

A program denotes computations in some universe of objects. Abstract interpretation of programs consists in using that denotation to describe computations in another universe of abstract objects, so that the results of abstract execution give some informations on the actual computations. An intuitive example (which we borrow from Sintzoff [72]) is the rule of signs. The text -1515 * 17 may be understood to denote computations on the abstract universe {(+), (-), (±)} where the semantics of arithmetic operators is defined by the rule of signs. The abstract execution -1515 * 17 ==> -(+) * (+) ==> (-) * (+) ==> (-), proves that -1515 * 17 is a negative number. Abstract interpretation is concerned by a particular underlying structure of the usual universe of computations

Abstract program properties are modeled by a complete semilattice, Birkhoff[61]. Elementary program constructs are locally interpreted by order preserving functions which are used to associate a system of recursive equations with a program. The program global properties are then defined as one of the extreme fixpoints of that system, Tarski[55]. The abstraction process is defined in section 6. It is shown that the program properties obtained by an abstract interpretation of a program are consistent with those obtained by a more refined interpretation of that program. In particular, an abstract interpretation may be shown to be consistent with the formal semantics of the language. Levels of abstraction are formalized by showing that consistent abstract interpretations form a lattice (section 7). Section 8 gives a constructive definition of abstract properties of programs based on

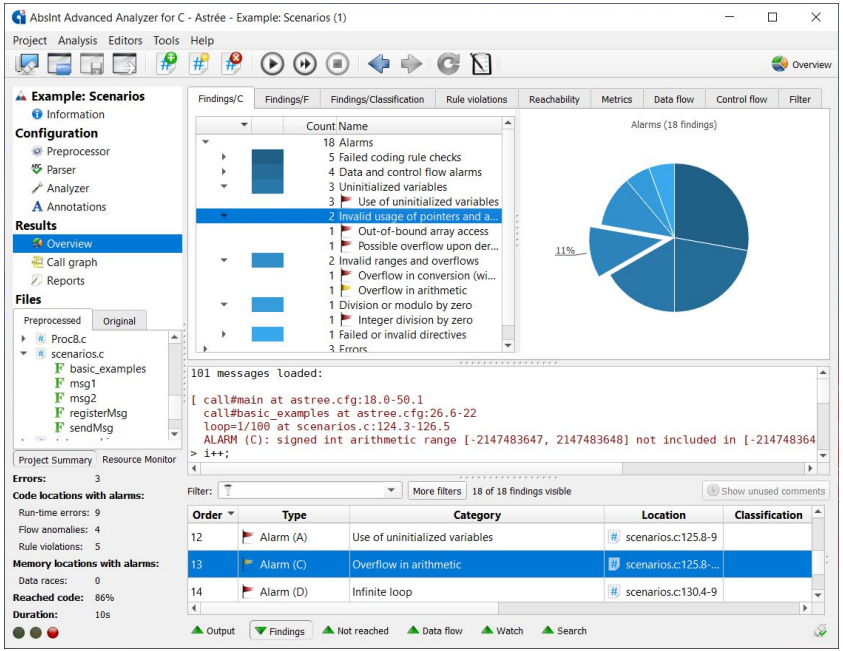
POPL 1977

Patrick Cousot

Radhia Cousot

Abstract Interpretation - Industry Adoption

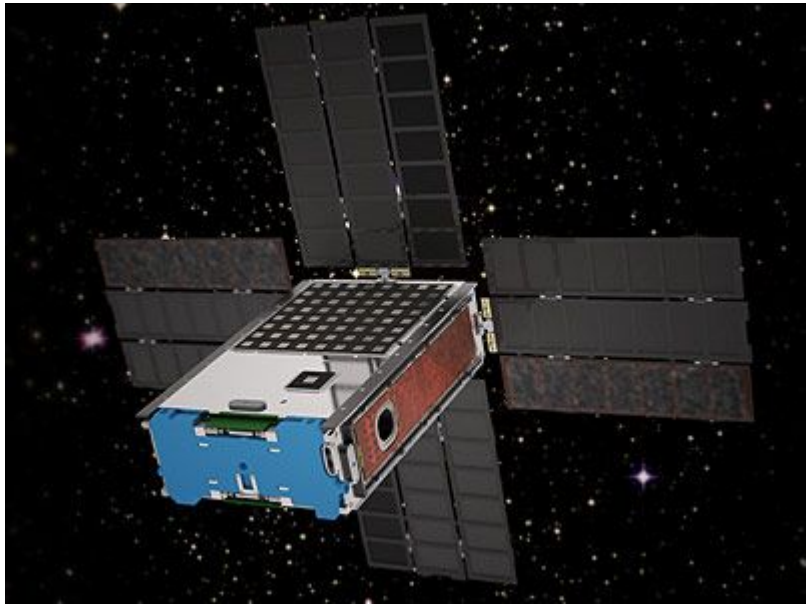
- Astrée - https://www.absint.com/astree/



Proved the absence of runtime errors for 132,000 lines of C code for Airbus A340 flight control software (2003).

Abstract Interpretation - Industry Adoption

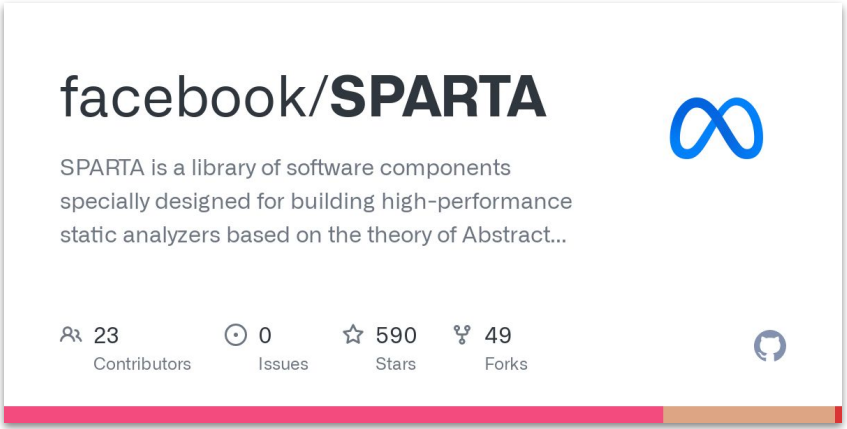
- NASA IKOS - https://github.com/NASA-SW-VnV/ikos

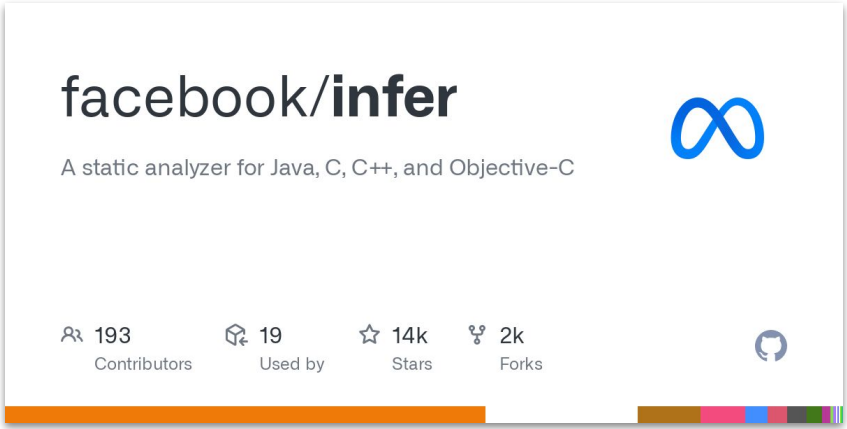


- BioSentinel spacecraft (launched in Nov 2022)
- IKOS found 17 real bugs in software running on BioSentinel

Abstract Interpretation - Industry Adoption

- Infer static analyzer
- SPARTA static analyzer
- Deployed with Meta’s continuous integration system





Undecidability and Approximation

- Halting problem:
  Can we have an analyzer that decides if a given program terminates?

Undecidability and Approximation

- Halting problem:
  Can we have an analyzer that decides if a given program terminates?

- Rice's theorem:

  All interesting *semantic* properties are *undecidable* (not computable).
  Proof: reduce to the Halting problem.

Undecidability and Approximation

- Halting problem:
  Can we have an analyzer that decides if a given program terminates?

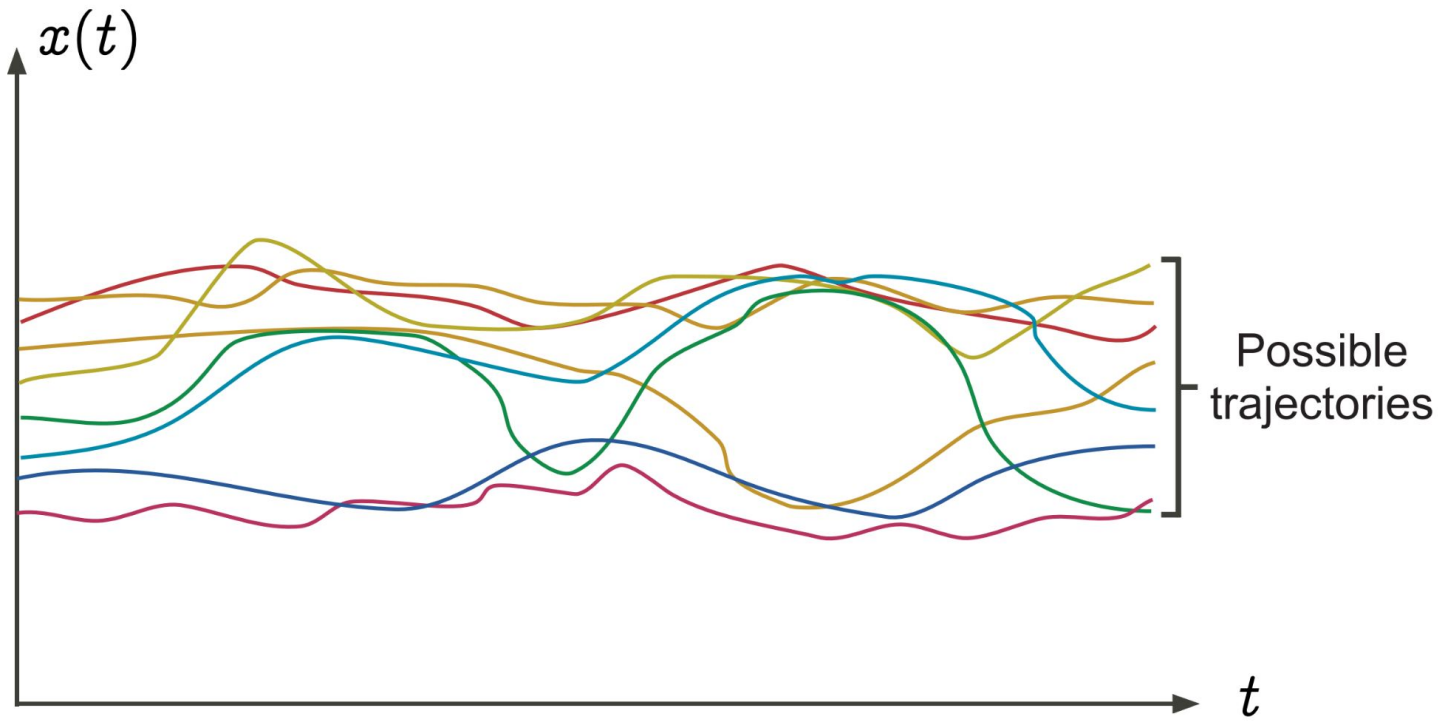
- Rice's theorem:

  All interesting *semantic* properties are *undecidable* (not computable).
  Proof: reduce to the Halting problem.

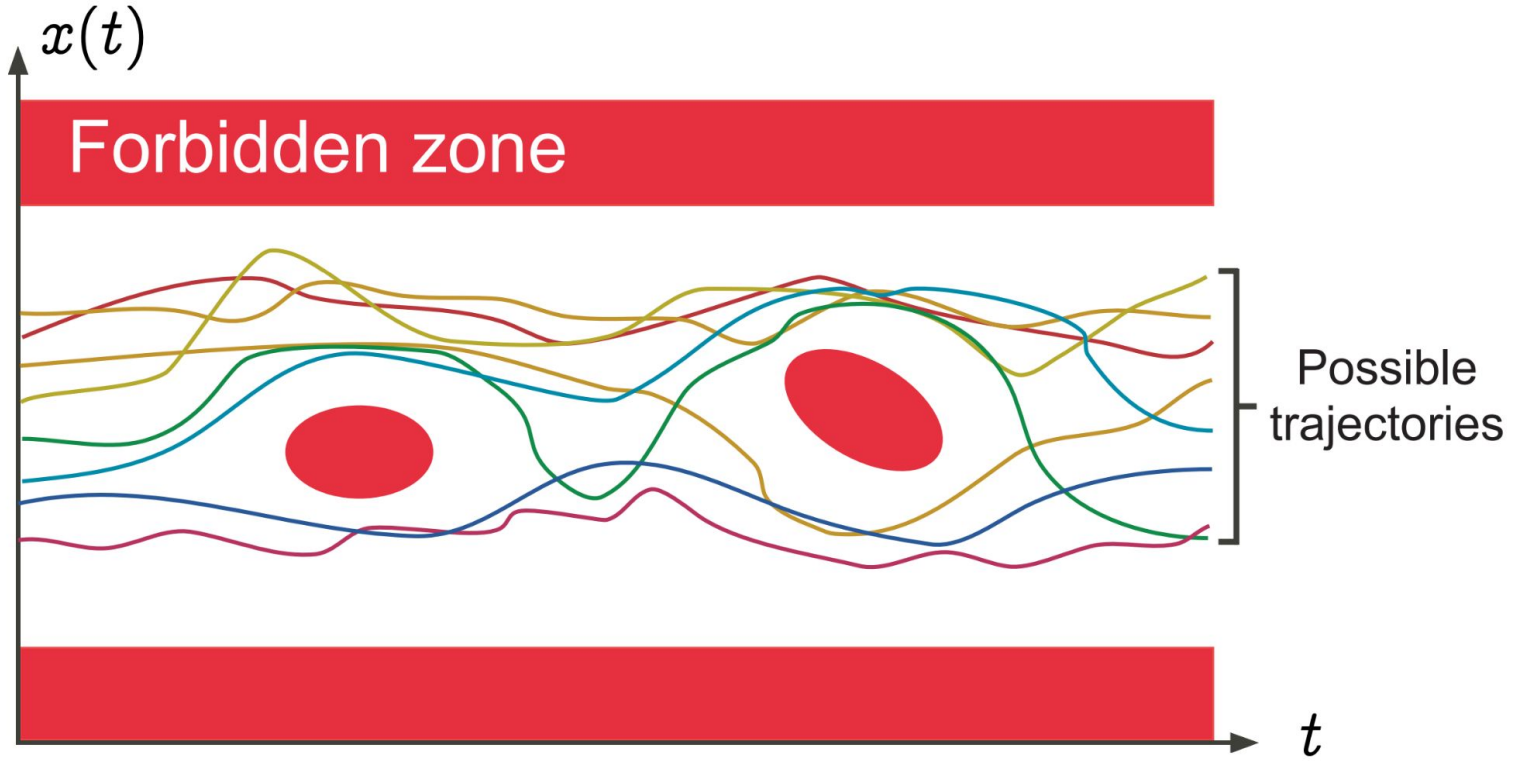
- Abstract interpretation:

  We can still predict the behavior of programs, but with some loss in precision (over-approximation).

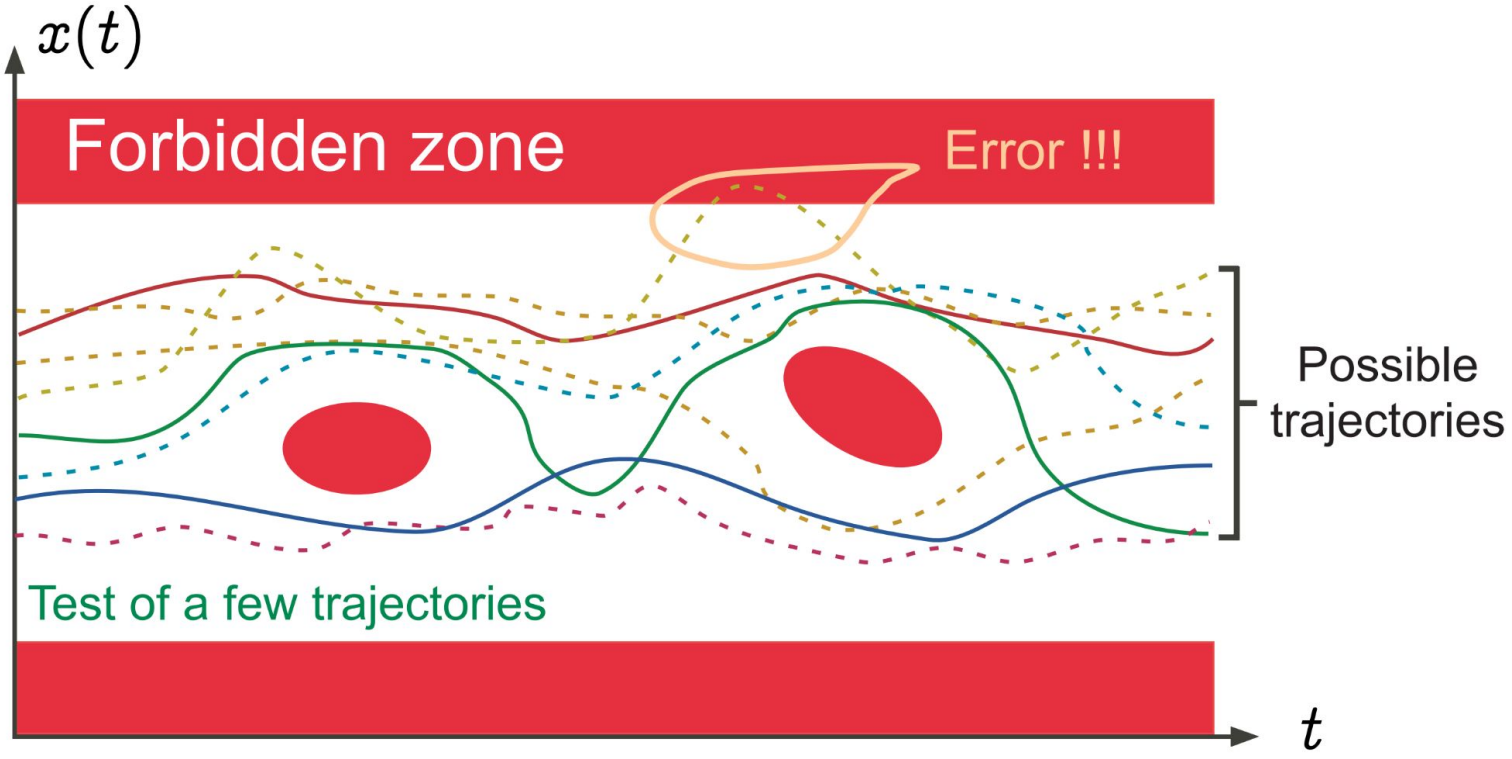
Concrete Behaviors (Possibly infinite)



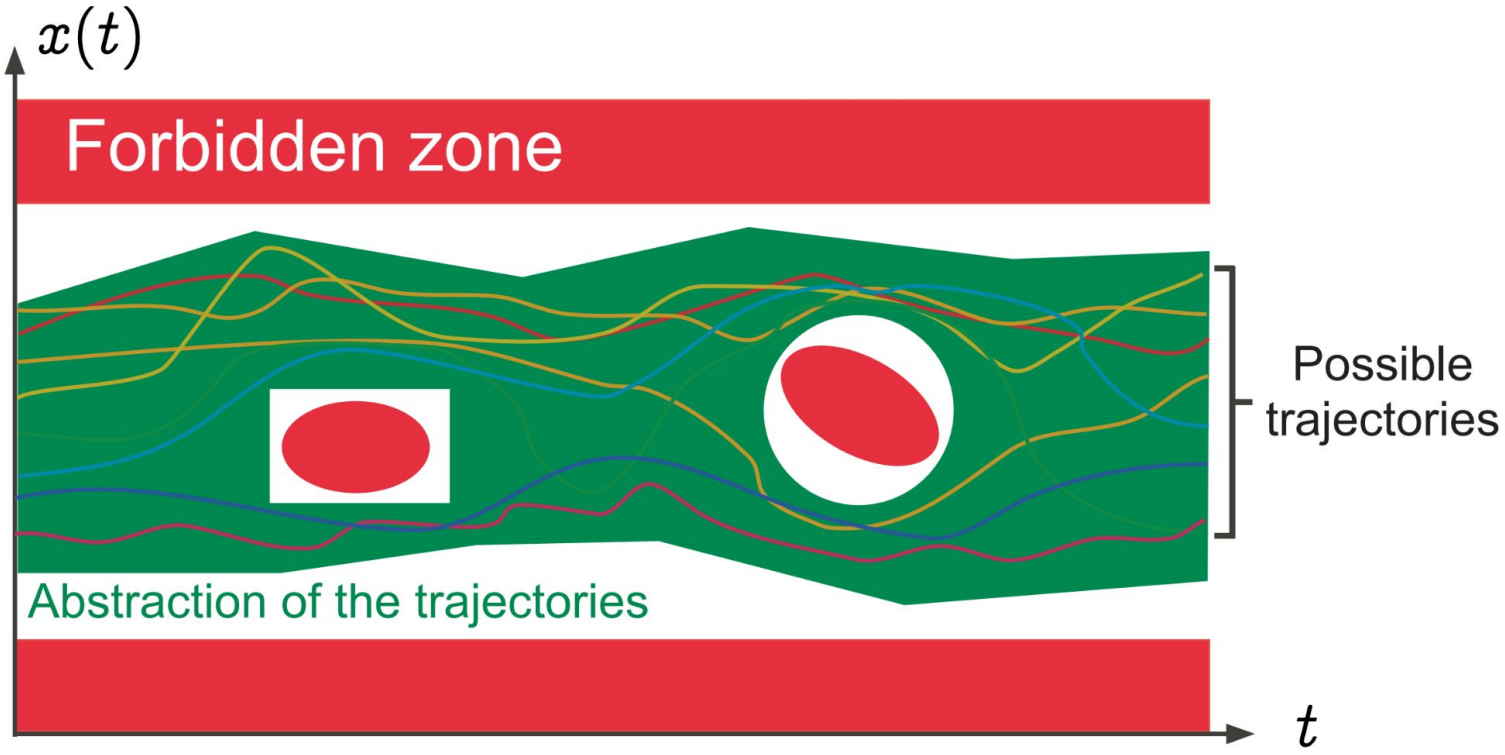
Safety Property: cannot reach into “forbidden zones”



Safety Property: Testing is not enough



Abstraction: over-approximate concrete behaviors



Specifying Abstract Interpretation

- **Step 1: define the meaning (concrete semantics) of the program**
- Step 2: define the abstraction that captures the property we want to analyze
- Step 3: define the abstract semantics using the abstraction

- Abstract interpretation guarantees *termination* and *soundness*
  - Analysis always terminates even the input program does not terminate.
  - All errors will be found by the analysis (but may contain false errors).

Concrete Semantics

- A simple arithmetic/imperative language

  e := $\mathbb{Z}$ | Var | e + e | e - e | e / e| input()

  s := Var ← e
  | skip | s; s
  | if0 e then s else s
  | while (e) s

- Analysis: range of numeric values

Concrete Semantics

- A simple arithmetic/imperative language

e := $\mathbb{Z}$ | Var | e + e | e - e | e / e | input()

s := Var ← e
| skip | s; s
| if0 e then s else s
| while (e) s

- Analysis: range of numeric values

Concrete Semantics

- Syntax:
  e := $\mathbb{Z}$ | Var | e + e | e - e | e / e | input()
- Domains:
  $\sigma \in \text{Var} \to \text{Value}$ $\quad\quad$ $v \in \text{Value} := \mathbb{Z}$
- Expression semantics: $\langle e, \sigma \rangle \Downarrow v$

$$\langle n, \sigma \rangle \Downarrow n$$

$$\langle x, \sigma \rangle \Downarrow \sigma(x)$$

$$\frac{\langle e1, \sigma \rangle \Downarrow v1 \quad \langle e2, \sigma \rangle \Downarrow v2}{\langle e1 + e2, \sigma \rangle \Downarrow v1 + v2}$$

$$\frac{\langle e1, \sigma \rangle \Downarrow v1 \quad \langle e2, \sigma \rangle \Downarrow v2}{\langle e1 - e2, \sigma \rangle \Downarrow v1 - v2}$$

$$\frac{\langle e1, \sigma \rangle \Downarrow v1 \quad \langle e2, \sigma \rangle \Downarrow v2}{\langle e1 \,/\, e2, \sigma \rangle \Downarrow v1 \,/\, v2}$$

$$\langle \texttt{input()}, \sigma \rangle \Downarrow \text{read\_stdin()}$$

Concrete Semantics

- Syntax:
  … and s := Var ← e | skip | s; s | ...
- Domains:
  $\sigma \in \text{Var} \to \text{Value}$ $\quad\quad$ $v \in \text{Value} := \mathbb{Z}$
- Statement semantics: $\langle s, \sigma \rangle \Downarrow \sigma$

$$\frac{\langle e, \sigma \rangle \Downarrow v}{\langle x \leftarrow e, \sigma \rangle \Downarrow \sigma[x \mapsto v]}$$

$$\langle \texttt{skip}, \sigma \rangle \Downarrow \sigma$$

$$\frac{\langle s1, \sigma \rangle \Downarrow \sigma1 \quad \langle s2, \sigma1 \rangle \Downarrow \sigma2}{\langle s1; s2, \sigma \rangle \Downarrow \sigma2}$$

Concrete Semantics - Example

- program: p = x ← x + 1; y ← x * 2
- initial store: σ = [x ↦ 42]

$$\dfrac{\dfrac{\dfrac{\langle x, \sigma\rangle \Downarrow 42 \quad \langle 1, \sigma\rangle \Downarrow 1}{\langle x + 1, \sigma\rangle \Downarrow 43}}{\langle x \leftarrow x + 1, \sigma\rangle \Downarrow [x \mapsto 43]} \quad \dfrac{\cdots}{\langle y \leftarrow x * 2, [x \mapsto 43]\rangle \Downarrow [x \mapsto 43, y \mapsto 86]}}{\langle p, \sigma\rangle \Downarrow [x \mapsto 43, y \mapsto 86]}$$

Specifying Abstract Interpretation

- Step 1: define the meaning (concrete semantics) of the program
- **Step 2: define the abstraction that captures the property we want to analyze**
- Step 3: define the abstract semantics using the abstraction

Abstractions of Integers

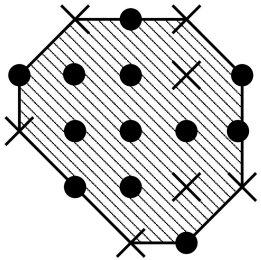
- Property: possible values of variables in the store
- Signs

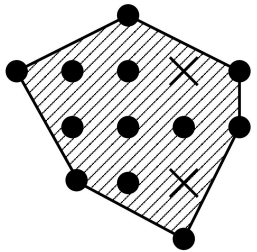
  $\mathbb{S}$ = { +, -, 0 }

- Intervals

  $[lb, rb]$ where $lb, ub \in \mathbb{Z} \cup \{-\infty, +\infty\}$

- Other domains:
  - octagons, polyhedra, etc.





The Sign Domain

- Signs $\mathbb{S}$ = { +, -, 0, $\top$, $\bot$ }
  - + for all positive integers
  - - for all negative integers
  - 0 for zero
  - $\top$ for all possible integers
  - $\bot$ for not-an-integers

The Sign Domain

- Signs $\mathbb{S}$ = { +, -, 0, $\top$, $\bot$ }

- Concrete domains:
  $\sigma \in \text{Var} \to \text{Value}$ $\quad\quad$ $v \in \text{Value} := \mathbb{Z}$

- Abstract domains:
  $\widehat{\sigma} \in \text{Var} \to \widehat{\text{Value}}$ $\quad\quad$ $\widehat{v} \in \widehat{\text{Value}} := \mathbb{S}$

The Sign Domain

- Signs $\mathbb{S}$ = { +, -, 0, $\top$, $\bot$ }
- Concrete domains:
  $\sigma \in \text{Var} \to \text{Value}$ $\quad$ $v \in \text{Value} := \mathbb{Z}$
- Abstract domains:
  $\widehat{\sigma} \in \text{Var} \to \widehat{\text{Value}}$ $\quad$ $\widehat{v} \in \widehat{\text{Value}} := \mathbb{S}$

The abstraction function

$\alpha : \mathbb{Z} \to \mathbb{S}$

$\alpha(n) = +$ if $n > 0$

$\alpha(n) = -$ if $n < 0$

$\alpha(n) = 0$ if $n \equiv 0$

The Sign Domain

- Signs $\mathbb{S}$ = { +, -, 0, $\top$, $\bot$ }

- Concrete domains:
  $\sigma \in \text{Var} \to \text{Value}$ $\quad$ $v \in \text{Value} := \mathbb{Z}$

- Abstract domains:
  $\widehat{\sigma} \in \text{Var} \to \widehat{\text{Value}}$ $\quad$ $\widehat{v} \in \widehat{\text{Value}} := \mathbb{S}$

The abstraction function

$\alpha : \mathbb{Z} \to \mathbb{S}$

$\alpha(n) = +$ if $n > 0$

$\alpha(n) = -$ if $n < 0$

$\alpha(n) = 0$ if $n \equiv 0$

- Define the abstract semantics for expressions and statements
  $\langle e, \widehat{\sigma} \rangle \Downarrow \widehat{v}$ $\quad$ $\langle s, \widehat{\sigma} \rangle \Downarrow \widehat{\sigma}$

Arithmetic on the Sign Domain

- Signs $\mathbb{S}$ = { +, -, 0, $\top$, $\bot$ }

$\oplus : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{S}$

$\oplus$	**+**	**-**	**0**	$\top$	$\bot$
+	+	$\top$	+	$\top$	$\bot$
-	$\top$	-	-	$\top$	$\bot$
0	+	-	0	$\top$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$\div : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{S}$

$\div$	**+**	**-**	**0**	$\top$	$\bot$
+	+	-	$\bot$	$\top$	$\bot$
-	-	+	$\bot$	$\top$	$\bot$
0	0	0	$\bot$	$\top$	$\bot$
$\top$	$\top$	$\top$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Arithmetic on the Sign Domain

- Signs $\mathbb{S}$ = { +, -, 0, $\top$, $\bot$ }

$\oplus : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$

$\oplus$	+	-	0	$\top$	$\bot$
+	+	$\top$	+	$\top$	$\bot$
-	$\top$	-	-	$\top$	$\bot$
0	+	-	0	$\top$	$\bot$
$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

$\div : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$

$\div$	+	-	0	$\top$	$\bot$
+	+	-	$\bot$	$\top$	$\bot$
-	-	+	$\bot$	$\top$	$\bot$
0	0	0	$\bot$	$\top$	$\bot$
$\top$	$\top$	$\top$	$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Specifying Abstract Interpretation

- Step 1: define the meaning (concrete semantics) of the program
- Step 2: define the abstraction that captures the property we want to analyze
- **Step 3: define the abstract semantics using the abstraction**

Specifying Abstract Interpretation

- Define the abstract semantics for expressions and statements

  $\langle e, \hat{\sigma} \rangle \Downarrow \hat{v}$ $\quad$ $\langle s, \hat{\sigma} \rangle \Downarrow \hat{\sigma}$

$$\langle n, \hat{\sigma} \rangle \Downarrow \alpha(n)$$

$$\langle x, \hat{\sigma} \rangle \Downarrow \hat{\sigma}(x)$$

$$\langle \texttt{input()}, \hat{\sigma} \rangle \Downarrow \top$$

$$\frac{\langle e1, \hat{\sigma} \rangle \Downarrow \hat{v1} \quad \langle e2, \hat{\sigma} \rangle \Downarrow \hat{v2}}{\langle e1 + e2, \hat{\sigma} \rangle \Downarrow \hat{v1} \oplus \hat{v2}}$$

$$\frac{\langle e1, \hat{\sigma} \rangle \Downarrow \hat{v1} \quad \langle e2, \hat{\sigma} \rangle \Downarrow \hat{v2}}{\langle e1 - e2, \hat{\sigma} \rangle \Downarrow \hat{v1} \ominus \hat{v2}}$$

$$\frac{\langle e1, \hat{\sigma} \rangle \Downarrow \hat{v1} \quad \langle e2, \hat{\sigma} \rangle \Downarrow \hat{v2}}{\langle e1 - e2, \hat{\sigma} \rangle \Downarrow \hat{v1} \div \hat{v2}}$$

$$\frac{\langle e, \hat{\sigma} \rangle \Downarrow \hat{v}}{\langle x \leftarrow e, \hat{\sigma} \rangle \Downarrow \sigma[x \mapsto \hat{v}]}$$

$$\langle \texttt{skip}, \hat{\sigma} \rangle \Downarrow \hat{\sigma}$$

$$\frac{\langle s1, \hat{\sigma} \rangle \Downarrow \hat{\sigma1} \quad \langle s2, \hat{\sigma1} \rangle \Downarrow \hat{\sigma2}}{\langle s1; s2, \hat{\sigma} \rangle \Downarrow \hat{\sigma2}}$$

Abstract Semantics - Example

- program: p = x ← input(); y ← 42 / x
- initial store: σ = []

$$\dfrac{\dfrac{}{\langle \text{input}(), [] \rangle \Downarrow \top}}{\langle x \leftarrow \text{input}(), [] \rangle \Downarrow [x \mapsto \top]} \qquad \dfrac{\dfrac{\langle 42, [x \mapsto \top] \rangle \Downarrow \alpha(42) = + \qquad \langle x, [x \mapsto \top] \rangle \Downarrow \top}{\langle 42 \,/\, x, [x \mapsto \top] \rangle \Downarrow + \div \top = \top}}{\langle y \leftarrow 42 \,/\, x, [x \mapsto \top] \rangle \Downarrow [x \mapsto \top, y \mapsto \top]}$$

$$\langle p, [] \rangle \Downarrow [x \mapsto \top, y \mapsto \top]$$

Orders and Lattices

- Partial Orders $\langle X, \sqsubseteq \rangle$
  - X is a set
  - $\sqsubseteq \in X \times X$ is a binary ordering relation that is
    - reflexive: $x \sqsubseteq x$
    - anti-symmetric: if $x \sqsubseteq y$ and $y \sqsubseteq x$ then $x = y$
    - transitive: if $x \sqsubseteq y$ and $y \sqsubseteq z$ then $x \sqsubseteq z$
- Instances:
  $\langle \mathbb{Z}, \leq \rangle$ integers and the less-than relation is a partial order

  $\langle \text{Prop}, \Rightarrow \rangle$ propositions and implication is a partial order

Orders and Lattices

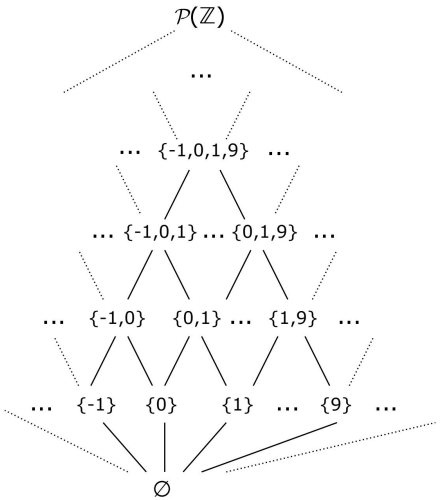
- Lattices $\langle X, \sqsubseteq, \sqcup, \sqcap \rangle$
  - $\langle X, \sqsubseteq \rangle$ is a partial order
  - $\sqcup : X \times X \to X$ is a binary operator to compute the least upper bound (LUB)
  - $\sqcap : X \times X \to X$ is a binary operator to compute the greatest lower bound (GLB)
- Instances:
  - $\langle \mathbb{Z}, \leq, \max, \min \rangle$
    integers, the less-than relation, the max, min operations
  - $\langle \text{Prop}, \Rightarrow, \bigwedge, \bigvee \rangle$
    propositions, implication, conjunction, and disjunction

Orders and Lattices

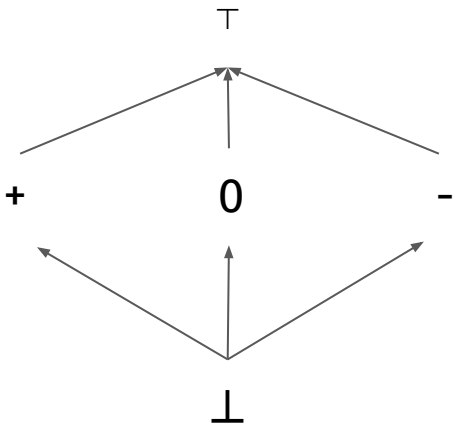
- Complete lattices $\langle X, \sqsubseteq, \sqcup, \sqcap, \top, \bot \rangle$
  - $\langle X, \sqsubseteq, \sqcup, \sqcap \rangle$ is a lattice
  - $\top$ is the upper bound for all elements in X
  - $\bot$ is the lower bound for all elements in X
- Instances:
  - $\langle \mathbb{Z}, \leq, \max, \min, ?, ? \rangle$
  - $\langle \text{Prop}, \Rightarrow, \wedge, \vee, \text{True}, \text{False} \rangle$
    propositions, implication, conjunction, and disjunction, True, False

Specifying Abstract Interpretation

- Step 1: define the concrete semantics and collecting semantics
- Step 2: define the abstraction
- Step 3: define the abstract semantics using the abstraction



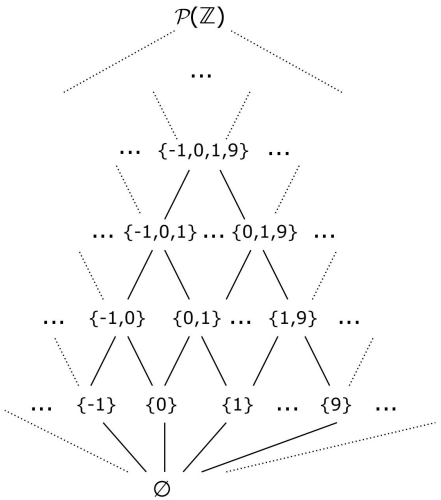
collecting semantics $\mathbb{P}(\mathbb{Z})$



abstract semantics $\mathbb{S}$

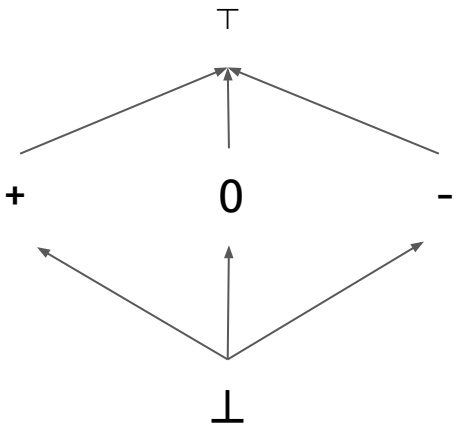
Specifying Abstract Interpretation

- Step 1: define the concrete semantics and collecting semantics
- Step 2: define the abstraction
- Step 3: define the abstract semantics using the abstraction



collecting semantics $\mathbb{P}(\mathbb{Z})$

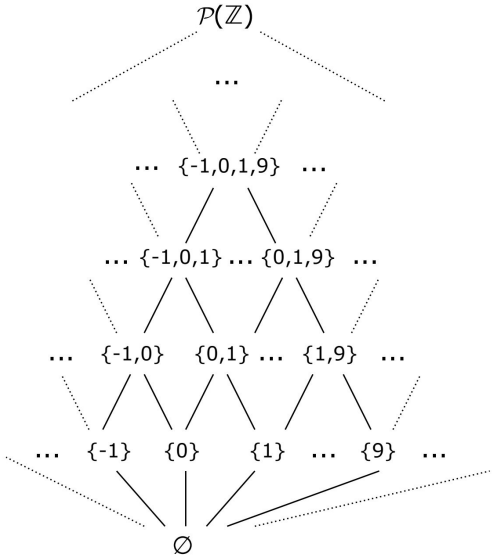
Galois connection



abstract semantics $\mathbb{S}$

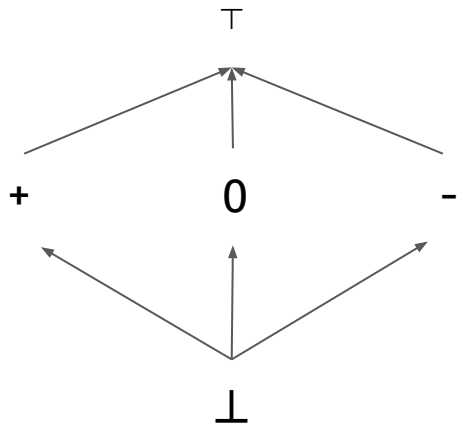
Derived Complete Lattices $\langle X, \sqsubseteq, \sqcup, \sqcap, \top, \bot \rangle$

- $\langle \mathbb{P}(X), \subseteq, \cup, \cap, \emptyset, X \rangle$
  For any set X, its powerset is a complete lattice, ordered by set-inclusion, LUB is set-union, GLB is set-intersection, bottom is the emptyset, and top is X.
- Example: $\mathbb{P}(\mathbb{Z})$



Signs as a Complete Lattice

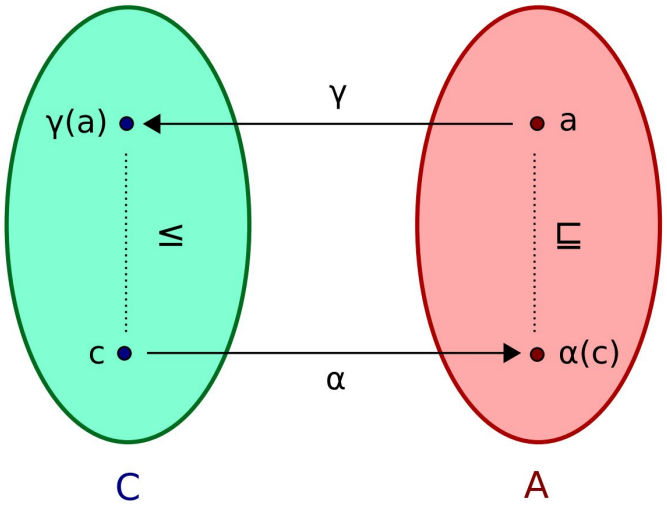
- Signs $\mathbb{S}$ = { +, -, 0, $\top$, $\bot$ }
- Forms a lattice $\langle \mathbb{S}, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$



Galois Connection

- Let $\langle C, \leq \rangle$, $\langle A, \sqsubseteq \rangle$ be two partial orders, the pair of functions $\alpha: C \to A$ and $\gamma: A \to C$ is a Galois connection if

$$\forall\ a \in A, c \in C, \qquad c \leq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$$



Galois Connection

- Let $\langle C, \leq \rangle$, $\langle A, \sqsubseteq \rangle$ be two partial orders, the pair of functions $\alpha: C \to A$ and $\gamma: A \to C$ is a Galois connection if

$$\forall\ a \in A, c \in C, \qquad c \leq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$$

- Example: conjunction and implication

Let $\langle C, \leq \rangle = \langle \text{Prop}, \Rightarrow \rangle$ and $\langle A, \sqsubseteq \rangle = \langle \text{Prop}, \Rightarrow \rangle$

Galois Connection

- Let $\langle C, \leq \rangle$, $\langle A, \sqsubseteq \rangle$ be two partial orders, the pair of functions $\alpha: C \to A$ and $\gamma: A \to C$ is a Galois connection if

$$\forall\ a \in A, c \in C, \qquad c \leq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$$

- Example: conjunction and implication

$$\text{Let } \langle C, \leq \rangle = \langle \text{Prop}, \Rightarrow \rangle \text{ and } \langle A, \sqsubseteq \rangle = \langle \text{Prop}, \Rightarrow \rangle$$

$$\alpha \triangleq (\cdot \wedge p) : \text{Prop} \to \text{Prop} \qquad \gamma \triangleq (p \Rightarrow \cdot) : \text{Prop} \to \text{Prop}$$

Galois Connection

- Let $\langle C, \leq \rangle$, $\langle A, \sqsubseteq \rangle$ be two partial orders, the pair of functions $\alpha: C \to A$ and $\gamma: A \to C$ is a Galois connection if

$$\forall\ a \in A, c \in C, \qquad c \leq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$$

- Example: conjunction and implication

$$\text{Let } \langle C, \leq \rangle = \langle \text{Prop}, \Rightarrow \rangle \text{ and } \langle A, \sqsubseteq \rangle = \langle \text{Prop}, \Rightarrow \rangle$$

$$\alpha \triangleq (\cdot \wedge p) : \text{Prop} \to \text{Prop} \qquad \gamma \triangleq (p \Rightarrow \cdot) : \text{Prop} \to \text{Prop}$$

$$\forall\ q \in \text{Prop}, r \in \text{Prop}, (r \leq \gamma(q)) \Leftrightarrow (\alpha(r) \sqsubseteq q)$$

Galois Connection

- Let $\langle C, \leq \rangle$, $\langle A, \sqsubseteq \rangle$ be two partial orders, the pair of functions $\alpha: C \to A$ and $\gamma: A \to C$ is a Galois connection if

$$\forall\ a \in A, c \in C, \qquad c \leq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$$

- Example: conjunction and implication

$$\text{Let } \langle C, \leq \rangle = \langle \text{Prop}, \Rightarrow \rangle \text{ and } \langle A, \sqsubseteq \rangle = \langle \text{Prop}, \Rightarrow \rangle$$

$$\alpha \triangleq (\cdot \wedge p) : \text{Prop} \to \text{Prop} \qquad \gamma \triangleq (p \Rightarrow \cdot) : \text{Prop} \to \text{Prop}$$

$$\forall\ q \in \text{Prop}, r \in \text{Prop}, (r \leq \gamma(q)) \Leftrightarrow (\alpha(r) \sqsubseteq q)$$

$$(r \Rightarrow \gamma(q)) \Leftrightarrow (\alpha(r) \Rightarrow q)$$

Galois Connection

- Let $\langle C, \leq \rangle$, $\langle A, \sqsubseteq \rangle$ be two partial orders, the pair of functions $\alpha: C \to A$ and $\gamma: A \to C$ is a Galois connection if

$$\forall\ a \in A, c \in C, \qquad c \leq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$$

- Example: conjunction and implication

$$\text{Let } \langle C, \leq \rangle = \langle \text{Prop}, \Rightarrow \rangle \text{ and } \langle A, \sqsubseteq \rangle = \langle \text{Prop}, \Rightarrow \rangle$$

$$\alpha \triangleq (\cdot \wedge p) : \text{Prop} \to \text{Prop} \qquad \gamma \triangleq (p \Rightarrow \cdot) : \text{Prop} \to \text{Prop}$$

$$\forall\ q \in \text{Prop}, r \in \text{Prop}, (r \leq \gamma(q)) \Leftrightarrow (\alpha(r) \sqsubseteq q)$$

$$(r \Rightarrow \gamma(q)) \Leftrightarrow (\alpha(r) \Rightarrow q)$$

$$\mathbf{(r \Rightarrow p \Rightarrow q) \Leftrightarrow (r \wedge p \Rightarrow q)}$$

Galois Connection is Everywhere

$$x \le r \times y \iff \lceil x \,/\, r \rceil \le y$$

Galois Connection is Everywhere

$$x \le r \times y \iff \lceil x \,/\, r \rceil \le y$$

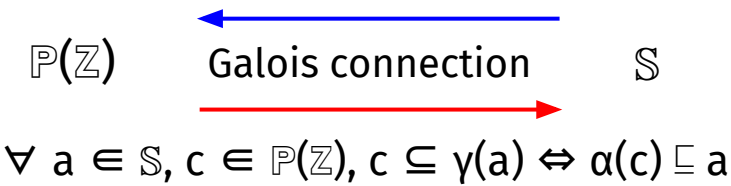
GC between $\alpha \triangleq \lceil \cdot \,/\, r \rceil$ and $\gamma \triangleq r \times \cdot$

$$x \le \gamma(y) \iff \alpha(x) \le y$$

Galois Connection between $\mathbb{P}(\mathbb{Z})$ and $\mathbb{S}$

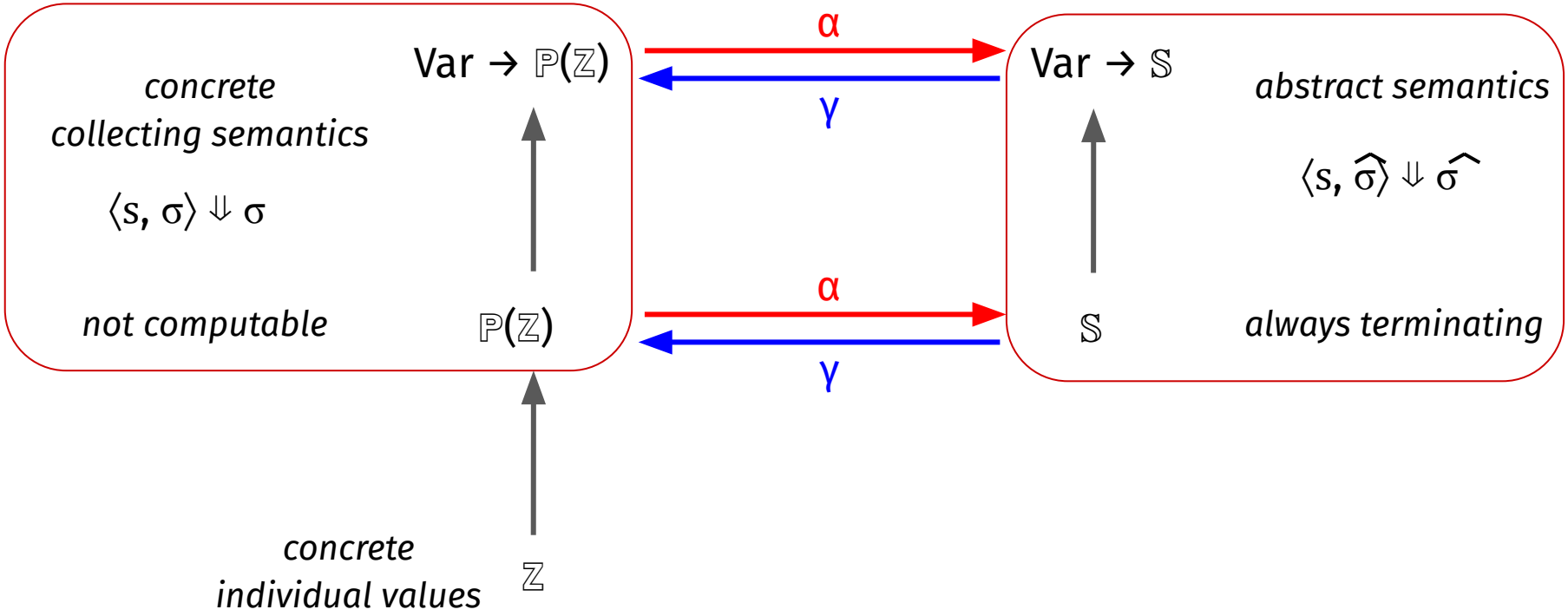
$\alpha : \mathbb{P}(\mathbb{Z}) \to \mathbb{S}$
$\alpha(\emptyset) = \bot$
$\alpha(\{0\}) = 0$
$\alpha(X) = +$ if $X \subseteq \{z \mid z > 0\}$
$\alpha(X) = -$ if $X \subseteq \{z \mid z < 0\}$
$\alpha(X) = \top$ otherwise

$\gamma : \mathbb{S} \to \mathbb{P}(\mathbb{Z})$
$\gamma(\top) = \mathbb{Z}$
$\gamma(+) = \{z \mid z > 0\}$
$\gamma(0) = \{0\}$
$\gamma(-) = \{z \mid z < 0\}$
$\gamma(\bot) = \emptyset$



$\forall\ a \in \mathbb{S}, c \in \mathbb{P}(\mathbb{Z}), c \subseteq \gamma(a) \Leftrightarrow \alpha(c) \sqsubseteq a$

Bigger Picture



Soundness of Abstract Interpretation

- Program behavior P, Abstracted behavior A, Specification S

- Soundness: $P \subseteq A$
  - Analysis always over-approximate actual program behavior
- Precise analysis: $A \subseteq S \Rightarrow P \subseteq S$
- May have false positive: $A \not\subseteq S$, but actually $P \subseteq S$

- Unsound: $A \subseteq S$, but actually $P \not\subseteq S$

Next Time

- How to abstractly interpret control structures?
  - Conditional
  - While loops
- Chains, Fixpoints, and Kleene Iterations