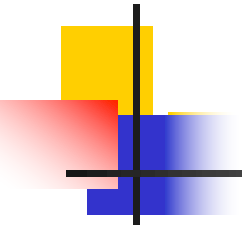
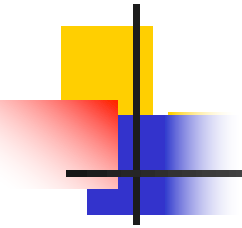
Abstract Interpretation, cont.



Announcements

- HW3 and HW4?

- HW5
  - Abstract interpretation and
  - Haskell
  - Download and get started with Haskell



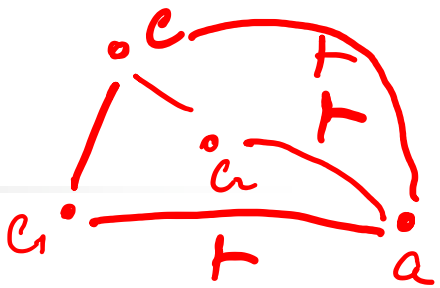
Outline

- Overview

- Semantics
- Notion of abstraction
- Concretization and abstraction functions
- Galois Connections

- Applications of abstract interpretation

Concretization Function



- Definition:

Concretization function γ : **A→C** (if it exists) maps **a** $\in$ **A** to the largest (most general) element **c** $\in$ **C** such that **c** $\vdash$ **a**

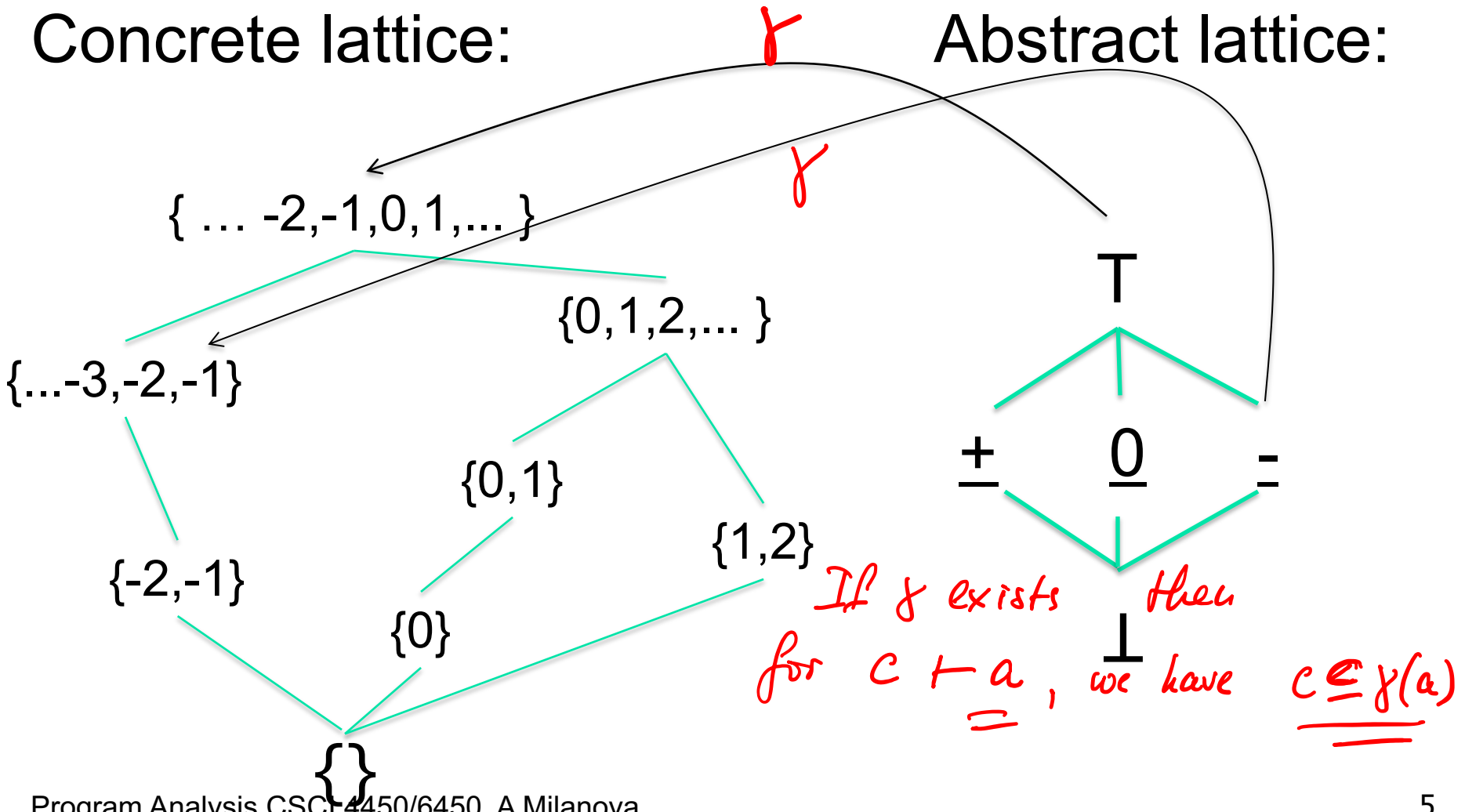
Note: γ(**a**) "covers" all concrete elements that are represented by **a**

- γ(**a**) returns the most general element **c** such that **c** is represented by **a**. This is called concretization

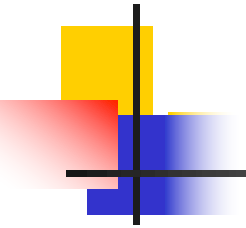
Gamma Examples

Concrete lattice:

Abstract lattice:



If γ exists then for $c \vdash a$, we have $c \sqsubseteq \gamma(a)$



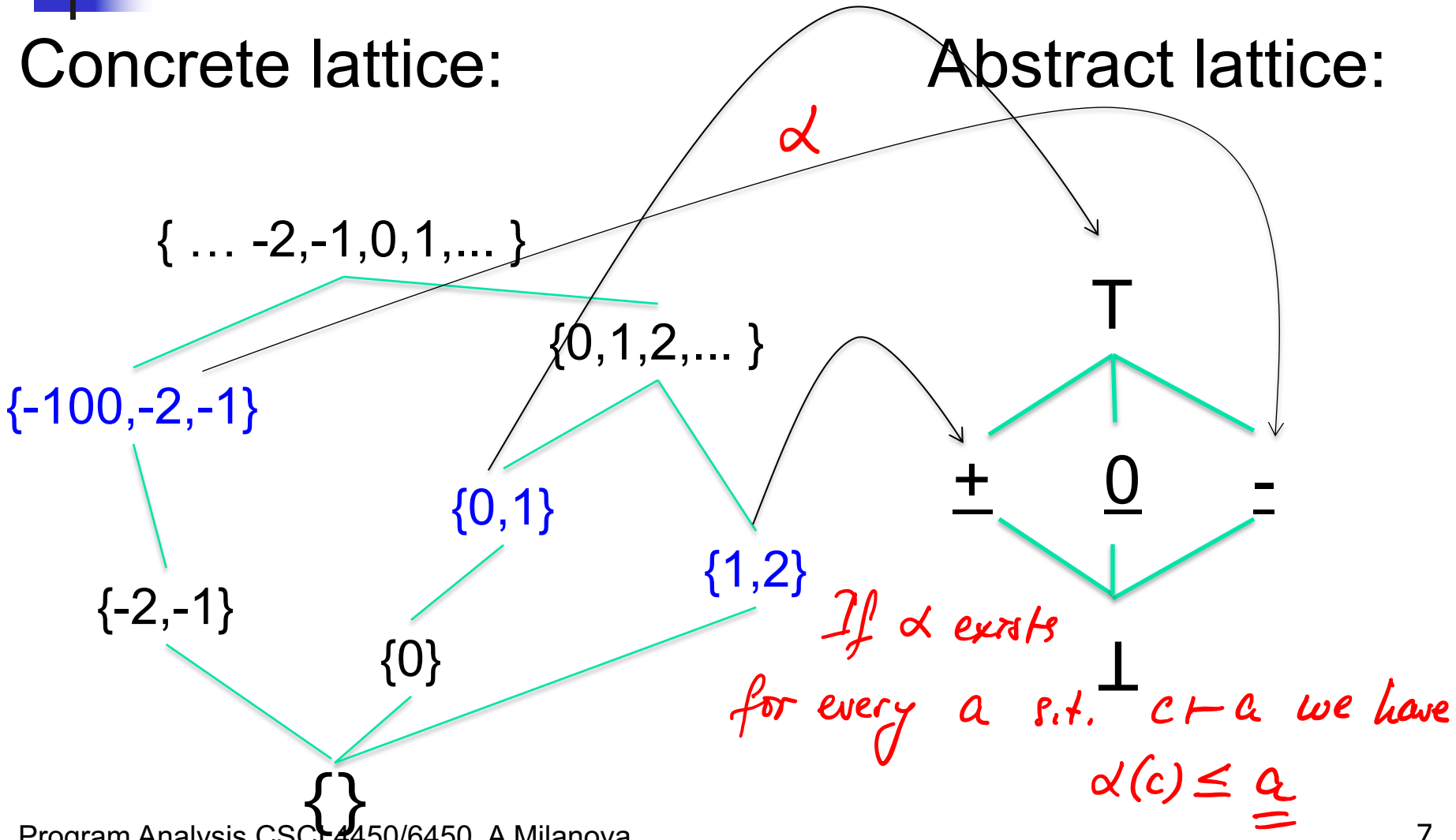
Abstraction Function

- Definition:

Abstraction function α : **C→A** (if it exists) maps **c**$\in$**C** to the smallest (most precise) element **a**$\in$**A** such that **c** $\vdash$ **a**

- α maps **c** to the most precise **a** such that **a** represents **c.** This is called best abstraction

Alpha Examples



Galois Connection

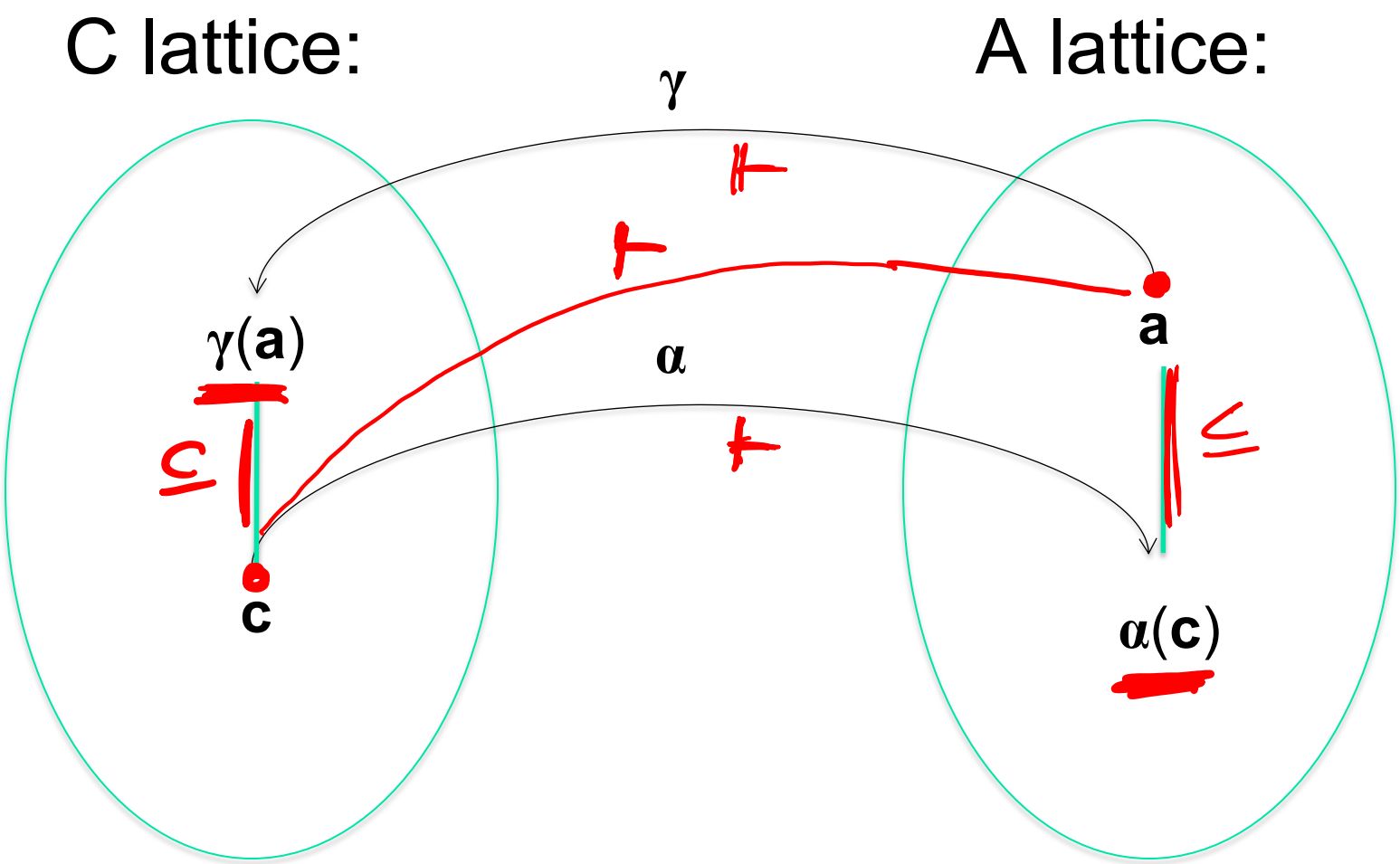
- A Galois Connection links α and γ. It captures that they represent the abstraction relation $\vdash$!
- Definition

A Galois connection is defined by concrete lattice (**C**,$\subseteq$), abstract lattice (**A**,$\leq$), an abstraction function α : **C**$\rightarrow$**A** and concretization function γ : **A**$\rightarrow$**C** such that

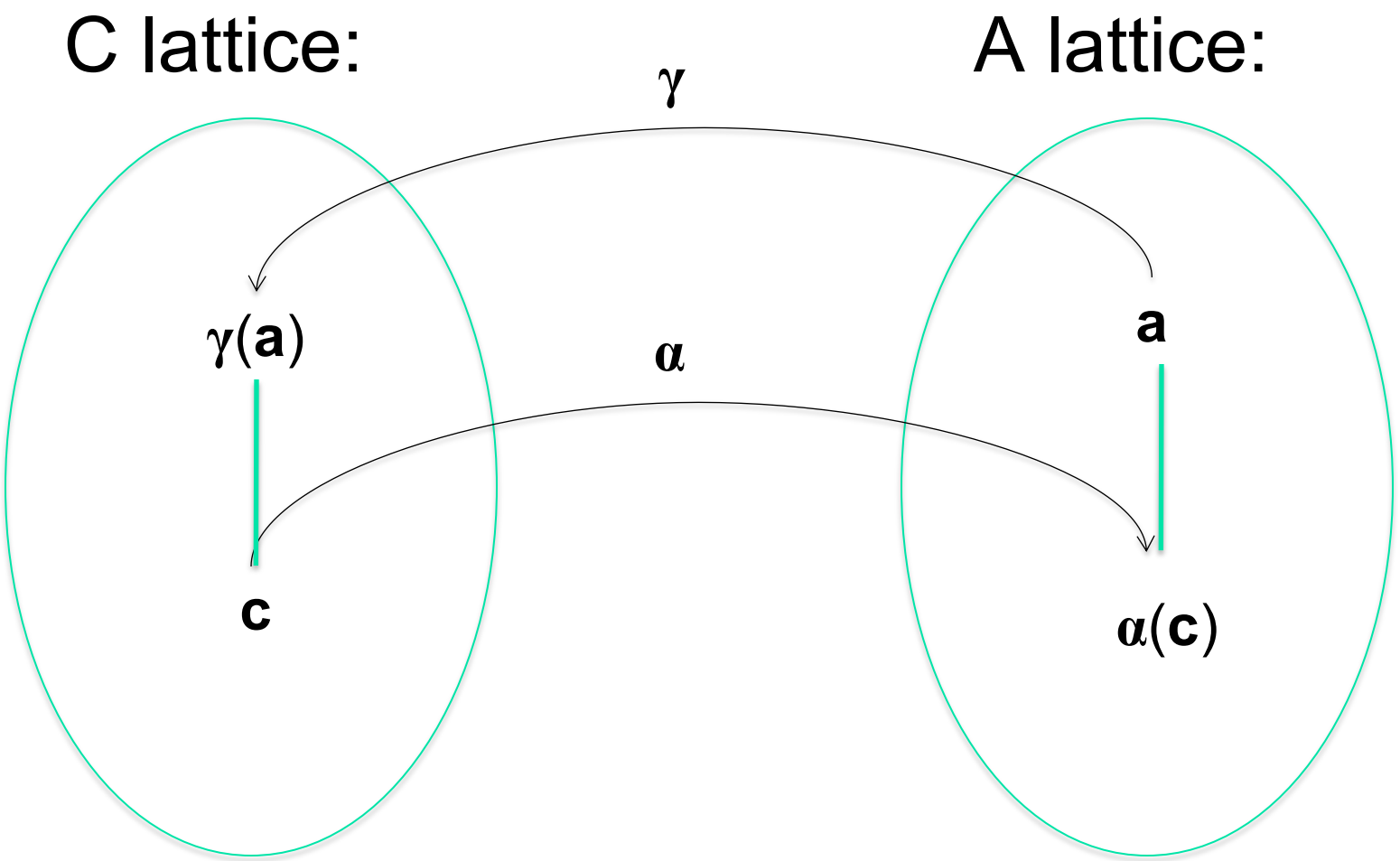
for every **a** $\in$ **A** and every **c** $\in$ **C**

c $\subseteq$ γ**(a)** if and only if α**(c)** $\leq$ **a**

Galois Connection

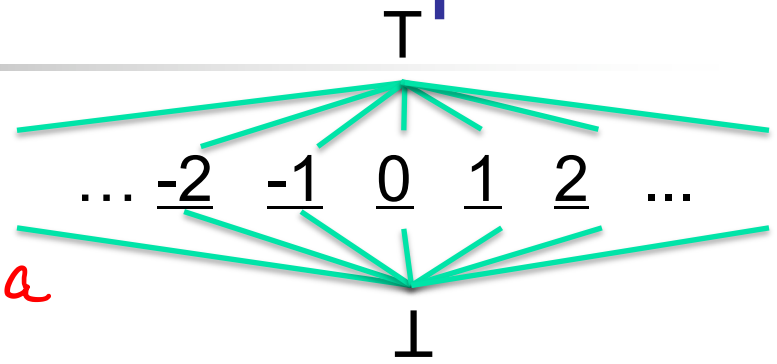


Galois Connection



Galois Connection Example

- Constants lattice



$c \subseteq \gamma(a) \rightleftharpoons \alpha(c) \leq a$

$\alpha_C(\mathbf{c}) \rightarrow \bot$ if $\mathbf{c} = \{\}$

$\alpha_C(\mathbf{c}) \rightarrow \underline{n}$ if $\mathbf{c} = \{n\}$

$\alpha_C(\mathbf{c}) \rightarrow$ T otherwise

$\gamma_C(T) \rightarrow \mathbf{Z}$

$\gamma_C(\underline{n}) \rightarrow \{n\}$

$\gamma_C(\bot) \rightarrow \{\}$

For each c and $a \in A$ s.t. $c \subseteq \gamma(a)$. We have to show $\alpha(c) \leq a$

Case 1: $c = \{\}$. For each $a \in A$ $c = \{\} \subseteq \gamma(a)$. We have $\alpha(\{\}) = \bot \leq a$ ✓

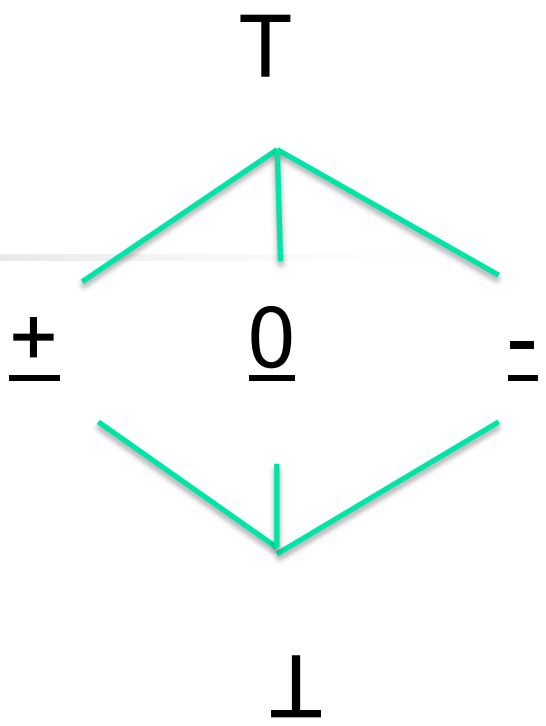
Case 2: $c = \{u\}$. $a = \underline{n}$ and $a = T$.

$a = \underline{u}$ $c = \{u\} \subseteq \gamma(\underline{u}) \Rightarrow \alpha(\{u\}) = \underline{u} \leq \underline{u}$ ✓

$a = T$ $c = \{u\} \subseteq \gamma(T) \Rightarrow \alpha(\{u\}) = \underline{u} \leq T$ ✓

Case 3: c = "otherwise", any set of 2 or more. $a = T$ $\alpha(c) = T \leq T$ ✓

Galois Connection Example



- Signs lattice

$\alpha_C(\mathbf{c})$ → ± if for every **n** in **c, n>0**

$\alpha_C(\mathbf{c})$ → - if for every **n** in **c, n<0**

$\alpha_C(\mathbf{c})$ → **0** if **c = {0}**

$\alpha_C(\mathbf{c})$ → ⊥ if **c = {}**

$\alpha_C(\mathbf{c})$ → T otherwise

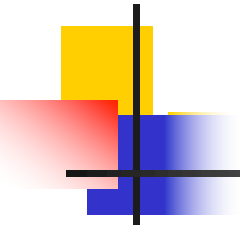
$\gamma_C(T)$ → **Z**

$\gamma_C(\pm)$ → **{1,2,...}**

$\gamma_C(0)$ → **{0}**

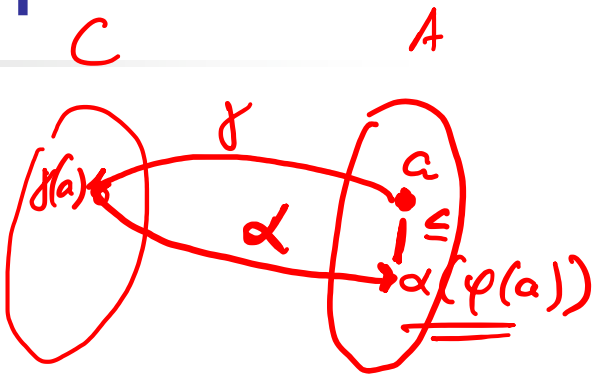
$\gamma_C(-)$ → **{…,-2,-1}**

$\gamma_C(\perp)$ → **{}**



Galois Connection Example

Galois Connection Properties



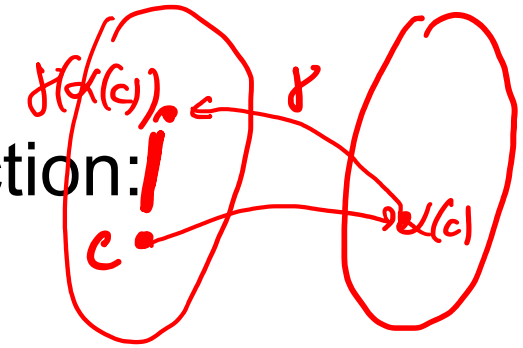
for every **a** ∈ **A** and every **c** ∈ **C**

c ⊆ **γ**(**a**) if and only if **α**(**c**) ≤ **a**

- Contractive and expansive:

α ° **γ** contracts: for every **a** ∈ **A** : **α** ° **γ** (**a**) ≤ **a**

γ ° **α** expands: for every **c** ∈ **C** : **γ** ° **α** (**c**) ⊇ **c**



Proof: from definition of Galois Connection:

γ(**a**) ⊆ **γ**(**a**) implies **α**(**γ**(**a**)) ≤ **a**

Galois Connection Properties

for every **a** $\in$ **A** and every **c** $\in$ **C**

c $\subseteq \gamma$**(a)** if and only if α**(c)** $\leq$ **a**

- Monotonicity:

α is monotone

γ is monotone

Proof: homework

Galois Connection Properties

for every **a** ∈ **A** and every **c** ∈ **C**

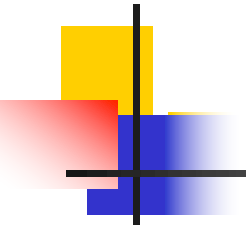
c ⊆ **γ(a)** if and only if **α(c)** ≤ **a**

- Repetition:

$\alpha \circ \gamma \circ \alpha = \alpha$ <= > $\alpha(\gamma(\alpha(c))) = \alpha(c)$

$\gamma \circ \alpha \circ \gamma = \gamma$

Proof: exercise



Outline

- Overview

- Semantics
- Notion of abstraction
- Concretization and abstraction functions
- Galois Connections

- Applications of abstract interpretation

Applications of Abstract Interpretation

- Deriving and reasoning about static analysis!
- Deriving a static analysis
- The hard (ad-hoc) way...
  - Starting from some concrete space and semantics
  - Define abstract space and abstract semantics (i.e., transfer functions)
  - Guess an invariant implying correctness conclusion
  - Make an inductive argument that each transfer function preserves the invariant

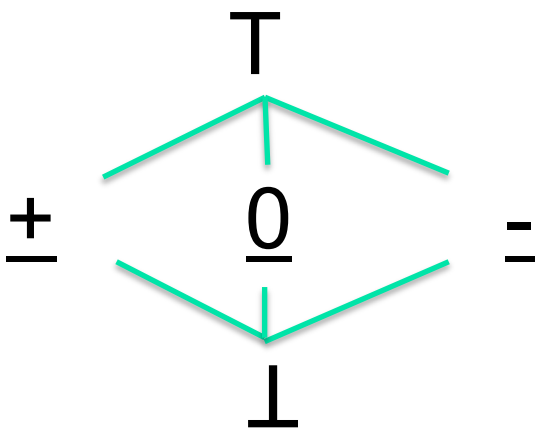
Remember Constant Propagation

Applications of Abstract Interpretation

- An easier (principled) way...
  - Formalize concrete domain ($\mathbf{C}, \subseteq$) and semantics
  - Construct an abstract space ($\mathbf{A}, \leq$)
  - Set up α and γ, Galois Connection, if it exists
  - Derive abstract semantics
    - "Principled" transfromration:
      - Use α when it exists
      - Use γ otherwise (more advanced abstract interpretation)
  - Correctness (soundness) holds!

An Example

- We will derive a static analysis that computes signs (**+,-,0**) using a collecting semantics and the signs abstraction:



- Correctness property (simplified):
  - x at ℓ is $\underline{+}$ only if $\sigma(x) > 0$ for every σ collected at ℓ
  - x at ℓ is $\underline{-}$ only if $\sigma(x) < 0$ for every σ collected at ℓ
  - x at ℓ is $\underline{0}$ only if $\sigma(x) = 0$ for every σ collected at ℓ

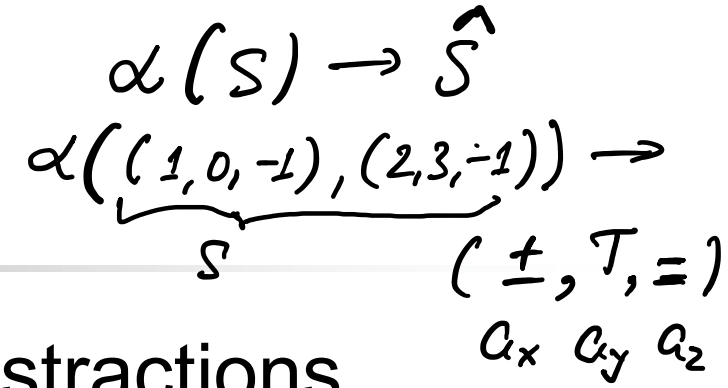
The Collecting Semantics

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

$[\![x = y]\!](S)$ COLLECTING

- Collecting semantics (partial)
  - Assume fixed set of integer variables: **x,y,z**
  - $\sigma = (n_x, n_y, n_z)$ $\quad \sigma = [x \to n_x, y \to n_y, z \to n_z]$
    - **σ** is a mapping from integer variables to values in **Z**
  - Concrete state **S** is a set of **σ**'s
  - $[\![x=n]\!](S) = \{ \sigma[x \leftarrow n] \mid \sigma \in S \}$
  - $[\![x=y+z]\!](S) = \{ \sigma[x \leftarrow [\![y]\!](\sigma) + [\![z]\!](\sigma)] \mid \sigma \in S \}$
  - $[\![x=y \bullet z]\!](S) = \{ \sigma[x \leftarrow [\![y]\!](\sigma) \bullet [\![z]\!](\sigma)] \mid \sigma \in S \}$

The Abstraction



- α is a composition of two abstractions
  - Collecting abstraction: collects the values of a variable across all σ's into one set
  - Signs abstraction: our running example
- $\alpha : \mathbf{S} \rightarrow (\alpha_S(\{ n_x \mid (n_x,\_,\_) \in \mathbf{S} \}),$
  $\alpha_S(\{ n_y \mid (\_,n_y,\_) \in \mathbf{S} \}),$
  $\alpha_S(\{ n_z \mid (\_,\_n_z) \in \mathbf{S} \}))$

The Abstraction

$\alpha : S \rightarrow (\alpha_S(\{ n_x \mid (n_x,\_,\_) \in S \}),$
$\alpha_S(\{ n_y \mid (\_,n_y,\_) \in S \}),$
$\alpha_S(\{ n_z \mid (\_,\_n_z) \in S \}))$

- E.g. **S** = { **(1,2,0), (-1,3,0)** } $\alpha(S) = (\top, \pm, \underline{0})$

$\alpha(S) = (\alpha_S(\{1,-1\}), \alpha_S(\{2,3\}), \alpha_S(\{0\})) = (\top, \pm, \underline{0})$

I.e., in the abstract, **x** is $\top$, **y** is $\pm$, **z** is $\underline{0}$

The Abstraction

S-hat is standard notation for abstract state

$\gamma : \hat{S} \rightarrow S$

(a_x, a_y, a_z)

- Abstract state $\hat{\mathbf{S}}$ is a tuple $(\mathbf{a_x}, \mathbf{a_y}, \mathbf{a_z})$
- $\gamma : \hat{\mathbf{S}} \rightarrow \{ (\mathbf{n_x}, \mathbf{n_y}, \mathbf{n_z}) \mid \mathbf{n_x} \in \gamma_S(\mathbf{a_x}),$
  $\mathbf{n_y} \in \gamma_S(\mathbf{a_y}), \mathbf{n_z} \in \gamma_S(\mathbf{a_z}) \}$

The Abstraction

S-hat is standard notation for abstract state

$\hat{S}$ S-hat

$\gamma : \hat{\mathbf{S}} \rightarrow \{ (n_x, n_y, n_z) \mid n_x \in \gamma_S(a_x),$

$n_y \in \gamma_S(a_y),\ n_z \in \gamma_S(a_z) \}$

- E.g., from previous slide:

S = { **(1,2,0), (-1,3,0)** }

$\alpha(\mathbf{S}) = (\alpha_S(\{1,-1\}), \alpha_S(\{2,3\}), \alpha_S(\{0\})) = (T, \underline{\pm}, \underline{0})$

$\gamma(\hat{\mathbf{S}}) = ?$

$\hat{\mathbf{S}} = (T, \underline{\pm}, \underline{0}), \gamma(\hat{\mathbf{S}})$ = { **(1,2,0), (-1,2,0), (-2,3,0)**,etc. }

- Exercise: Prove α, γ form Galois connection

Deriving the Abstract Semantics

- We can derive an abstract semantics by "principled" transformation: $[\![x = y + z]\!](S)$
  - Consider concrete transfer function $[\![\textbf{Stmt}]\!](\textbf{S})$
  - We derive a corresponding abstract transfer function $\widehat{[\![\textbf{Stmt}]\!]}(\hat{\textbf{S}})$
  - We then show $\alpha([\![\textbf{Stmt}]\!](\textbf{S})) = \widehat{[\![\textbf{Stmt}]\!]}(\alpha(\textbf{S}))$
    - Implies static analysis is correct (i.e., over-approximation of all sigmas)
  - Exact α-based transformation does not always work

Deriving the Abstract Semantics

$\S = (\tau, \tau, \tau)$

- Abstracting result of concrete transfer function:

$$\alpha\,(\,[\![\mathbf{Stmt}]\!](\mathbf{S})\,) = (\alpha_S(\{\,\mathbf{n_x} \mid (\mathbf{n_x},\_,\_) \in [\![\mathbf{Stmt}]\!](\mathbf{S})\,\}),$$
$$\alpha_S(\{\,\mathbf{n_y} \mid (\_,\mathbf{n_y},\_) \in [\![\mathbf{Stmt}]\!](\mathbf{S})\,\}),$$
$$\alpha_S(\{\,\mathbf{n_z} \mid (\_,\_\mathbf{n_z}) \in [\![\mathbf{Stmt}]\!](\mathbf{S})\,\})\,)$$

Read: apply concrete transfer function $[\![\mathbf{Stmt}]\!](\mathbf{S})$ resulting in a new set of $\boldsymbol{\sigma} = (\mathbf{n_x},\mathbf{n_y},\mathbf{n_z})$'s. Then

(1) apply collecting abstraction: collect all values of **x**, all values of **y**, and all values of **z** into respective sets

(2) apply sign abstraction on each set

Deriving the Abstract Semantics

- Concreate transfer function: **[[x=n]](S)**
- Abstract transfer function: **[[x=n]](Ŝ) = Ŝ[x←n]**

sign(u)

$\alpha([\![x=n]\!](S)) = ?\ [\![x=n]\!](\alpha(S))$

new set of signs S'

$\alpha_S(\{n\}) = sign(n)$

$\alpha([\![x=u]\!](S)) = (\alpha_S(\{u_x \mid (u_x, u_y, u_2) \in S'\}),$
$\alpha_S(\{u_y \mid (u_x, u_y, u_2) \in S'\}),$
$\alpha_S(\{u_2 \mid (u_x, u_y, u_2) \in S'\}))$

// Important: values for y and z are the same in S and in S'.

$\alpha_S(\{u_y \mid (u_x, u_y, u_2) \in S\})$
$= \alpha_S(\{u_y \mid (u_x, u_y, u_2) \in S'\})$

$= (sign(u), \alpha_S(\{\ldots \in S\}), \alpha_S(\{\ldots \in S\}))$

a_x a_y a_z

Deriving the Abstract Semantics

- Concreate transfer function: $|[\mathbf{x=n}]|(\mathbf{S})$
- Abstract transfer function: $\widehat{|[\mathbf{x=n}]|}(\hat{\mathbf{S}}) = \hat{\mathbf{S}}[\mathbf{x} \leftarrow \mathbf{sign}(\mathbf{n})]$

We showed $\alpha([\![x=n]\!](S)) = (sign(n), \alpha_S(\{n_y \mid (n_x, n_y, n_z) \in S\}), \alpha_S(\{n_z \mid (n_x, n_y, n_z) \in S\}))$ (1)

We now consider $\widehat{[\![x=n]\!]}(\alpha(S))$

$\widehat{[\![x=n]\!]}(\alpha(S)) = \widehat{[\![x=n]\!]}(\alpha_S(\{n_x \mid (n_x, n_y, n_z) \in S\}), \alpha_S(\{n_y \mid (n_x, n_y, n_z) \in S\}), \alpha_S(\{n_z \mid (n_x, n_y, n_z) \in S\}))$

$= (sign(n), \alpha_S(\{n_y \mid (n_x, n_y, n_z) \in S\}), \alpha_S(\{n_z \mid (n_x, n_y, n_z) \in S\}))$ (2)

// Just applied $\hat{S}[x \leftarrow sign(n)]$.

(1) and (2) are the same! Thus $\alpha([\![x=n]\!](S)) = \widehat{[\![x=n]\!]}(\alpha(S))$.

Deriving the Abstract Semantics

- Concreate transfer function: $|[\mathbf{x=n}]|(\mathbf{S})$
- Abstract transfer function: $\widehat{|[\mathbf{x=n}]|}(\hat{\mathbf{S}}) = \hat{\mathbf{S}}[\mathbf{x} \leftarrow \mathbf{n}]$

α ($|[\mathbf{x=n}]|(\mathbf{S})$) =

($\alpha_S(\{ \mathbf{n_x} \mid (\mathbf{n_x},\mathbf{n_y},\mathbf{n_z}) \in \{ \boldsymbol{\sigma}[\mathbf{x} \leftarrow \mathbf{n}] \mid \boldsymbol{\sigma} \in \mathbf{S} \} \})$,

$\alpha_S(\{ \mathbf{n_y} \mid (\mathbf{n_x},\mathbf{n_y},\mathbf{n_z}) \in \{ \boldsymbol{\sigma}[\mathbf{x} \leftarrow \mathbf{n}] \mid \boldsymbol{\sigma} \in \mathbf{S} \} \})$,

$\alpha_S(\{ \mathbf{n_z} \mid (\mathbf{n_x},\mathbf{n_y},\mathbf{n_z}) \in \{ \boldsymbol{\sigma}[\mathbf{x} \leftarrow \mathbf{n}] \mid \boldsymbol{\sigma} \in \mathbf{S} \} \})$) =

$(\alpha_S(\{\mathbf{n}\}), \alpha_S(\{ \mathbf{n_y} \mid (\_,\mathbf{n_y},\_) \in \mathbf{S} \}), \alpha_S(\{ \mathbf{n_z} \mid (\_,\_,\mathbf{n_z}) \in \mathbf{S} \}))$

= $\alpha(\mathbf{S})[\mathbf{x} \leftarrow \alpha_S(\{\mathbf{n}\})]$ which is

$\widehat{|[\mathbf{x=n}]|}(\hat{\mathbf{S}}) = \hat{\mathbf{S}}[\mathbf{x} \leftarrow \alpha_S(\{\mathbf{n}\})]$

The Abstract Semantics

- $\widehat{|[x=n]|}(\hat{S}) = \hat{S}[x \leftarrow \underline{n}]$
- $\widehat{|[x=y+z]|}(\hat{S}) = \hat{S}[x \leftarrow |[y]|(\hat{S}) \oplus |[z]|(\hat{S})]$
- $|[x=y \bullet z]|(\hat{S}) = \hat{S}[x \leftarrow |[y]|(\hat{S}) \otimes |[z]|(\hat{S})]$

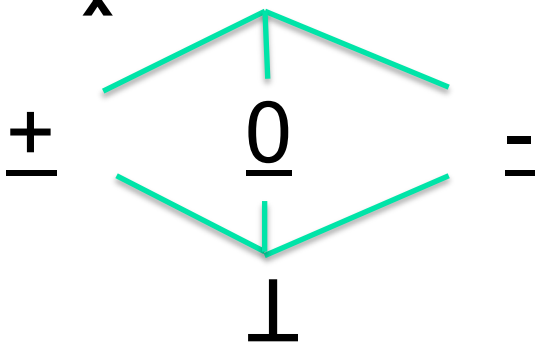
$\oplus$	$\bot$	$\underline{+}$	$\underline{0}$	$\underline{-}$	T
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\underline{+}$	$\bot$	$\underline{+}$	$\underline{+}$	T	T
$\underline{0}$	$\bot$	$\underline{+}$	$\underline{0}$	$\underline{-}$	T
$\underline{-}$	$\bot$	T	$\underline{-}$	$\underline{-}$	T
T	$\bot$	T	T	T	T

Prove: applying abstract transfer function on abstract value $\alpha(\mathbf{S})$ yields same result as applying concrete transfer function on **S**, then applying α on the result: $\widehat{|[Stmt]|} \circ \alpha = \alpha \circ |[Stmt]|$

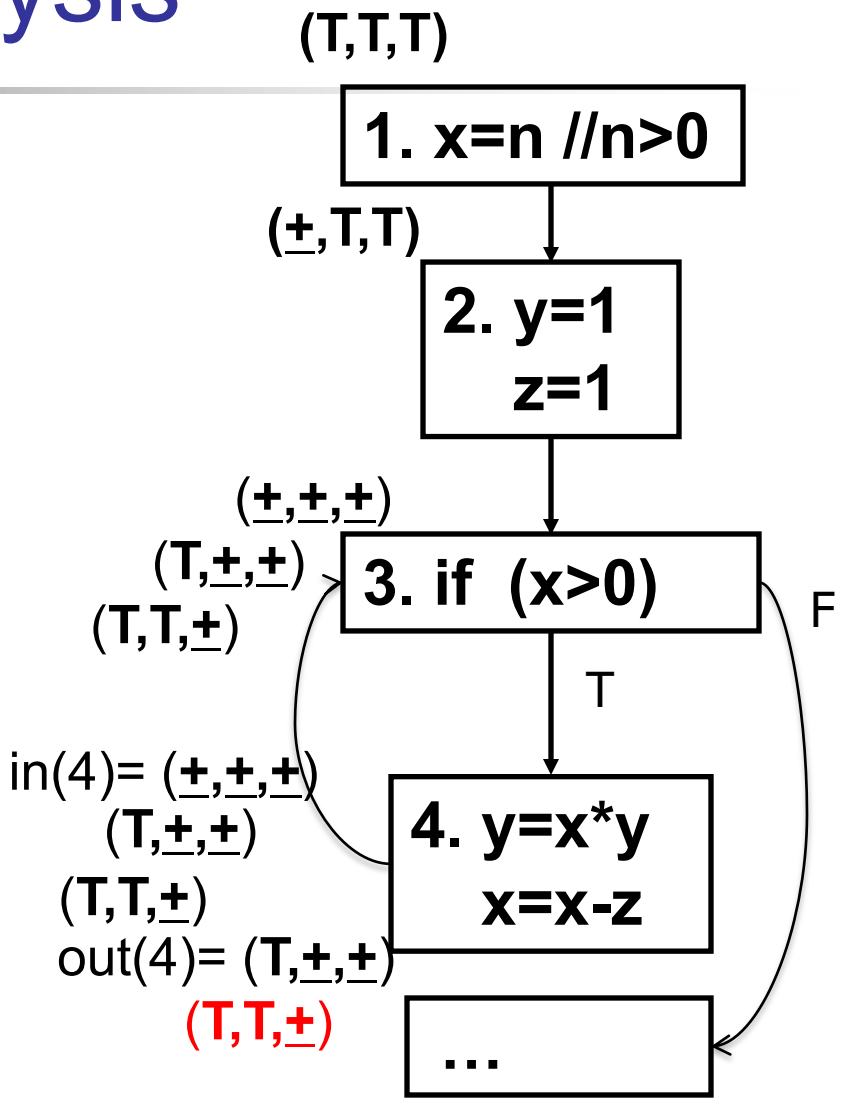
Guarantees soundness!

Dataflow Analysis

- Factorial
- L_x:



- Define transfer functions, then apply fixpoint iteration

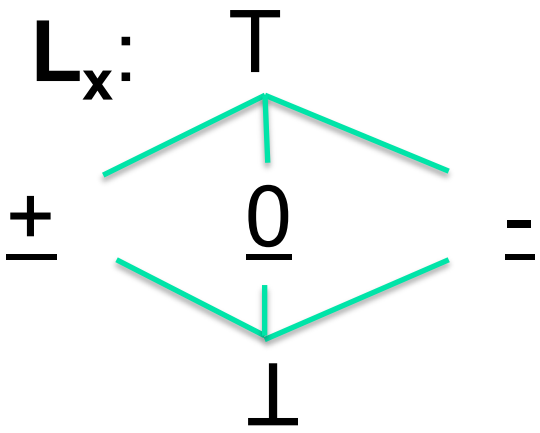


The Abstract Semantics

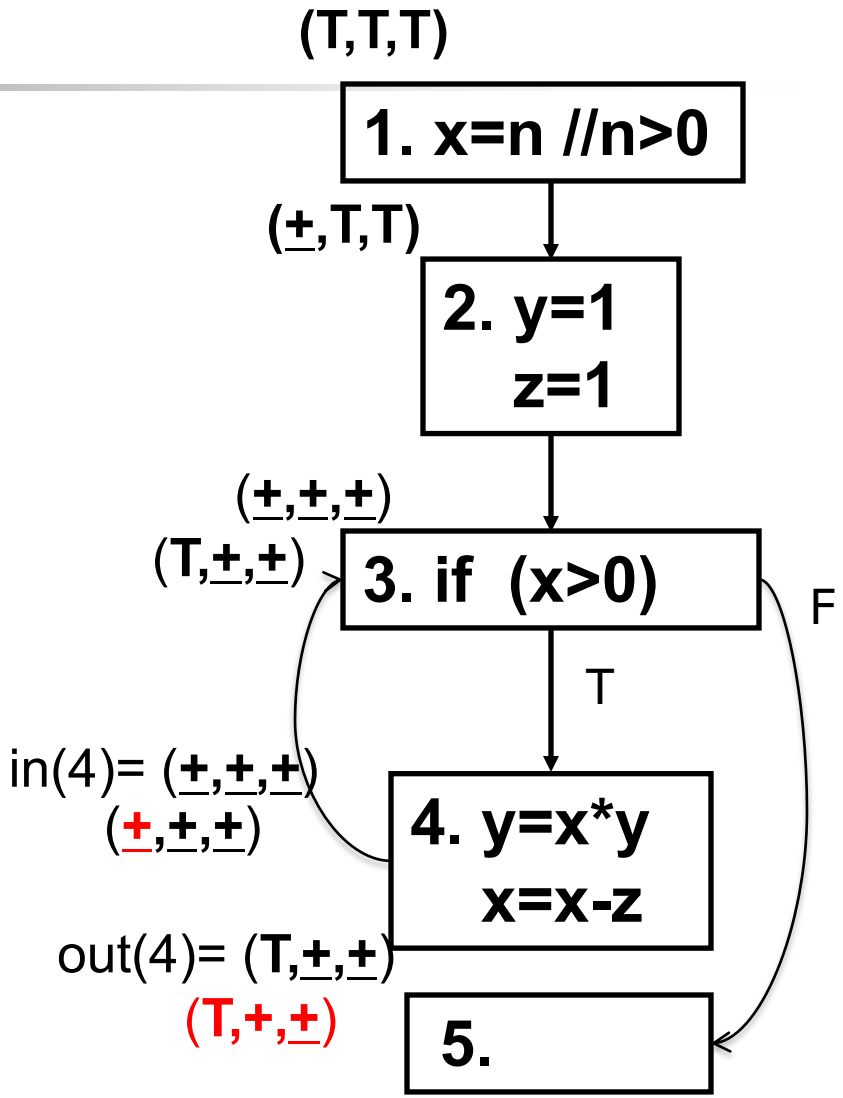
- Abstract interpretation does better than that: it symbolically executes abstract transfer functions
- Defines abstract semantics for
  - |[**if (b) then** Seq_1 **else** Seq_2]|(**S**)
  - |[**while** (**b**) *Seq*]|(**S**)
- Compositional semantics
  - $\widehat{|[\ Seq\ ]|}$ is a composition of abstract transfer functions
- Takes into account conditionals
  - E.g., in if-then-else applies $\widehat{|[\ Seq_1\ ]|}$ on abstract state augmented with **b = true**

Abstract Interpretation Does Better

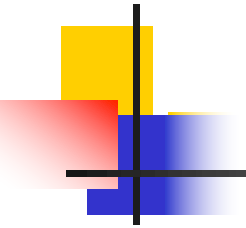
- Factorial
- L_x:



- E.g., applies f_4 on **(±,±,±)**
- in(5)=**(T,±,±)**



Abstract Interpretation, Conclusion



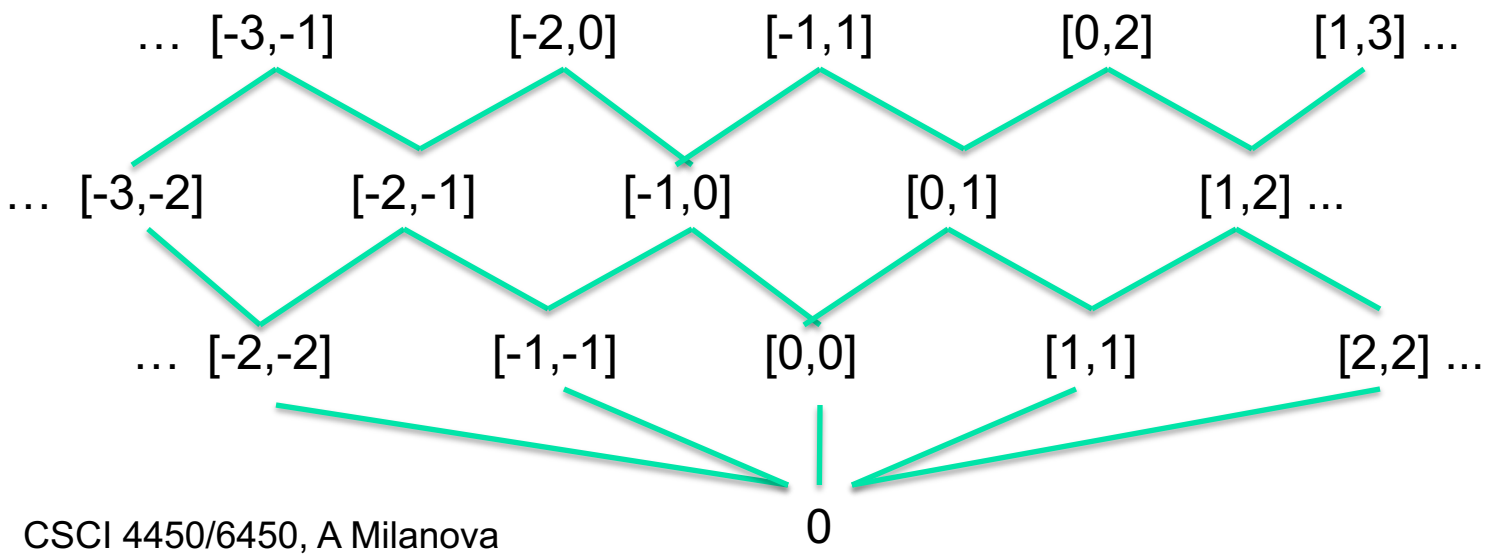
- A general framework
  - Building static analyses
  - Reasoning about correctness of static analysis
  - Comparing static analyses
  - Has applications in different areas, including reasoning about robustness of neural networks
- A lot more, it is an active area of research

Abstract Interpretation is Even More General!

- $|\widehat{[\textbf{Stmt}]}|$ s.t. $\alpha \circ |[\textbf{Stmt}]| = |\widehat{[\textbf{Stmt}]}| \circ \alpha$ may not exist for some $|[\textbf{Stmt}]|$
  - For some abstract domains, α does not exist
  - Abstract Interpretation allows building analyses even in cases like these. Uses γ:
    - $|[\textbf{Stmt}]| \circ \gamma(\hat{\textbf{S}}) \subseteq \gamma \circ |\widehat{[\textbf{Stmt}]}|(\hat{\textbf{S}})$
    - Sound (over-approximation) but less precise

Widening

- What if the abstract lattice does not have finite height?
- E.g., lattice of intervals, a popular abstract domain



Widening ∇

- Widening:
  - Over-approximates join **V** (for correctness)
  - Guarantees termination and faster convergence on programs with loops

```
x = 0;
while (true) {
  if (x < 9999)
    x = x+1;
  else
    x = -x;
}
```

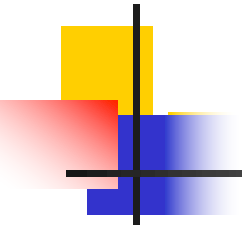
Typical widening:

$[a,b_0] \nabla [a,b_1] = [a,+\infty]$ if $b_0 < b_1$

$[a,b_0] \nabla [a,b_1] = [a,b_0]$ otherwise

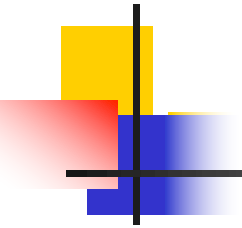
(Read: if interval grows with next iteration through the loop, widen to infinity.)

Abstract Interpretation, Conclusion



- A general framework
  - Building static analyses!
  - Reasoning about correctness of static analysis
  - Comparing static analyses

Abstract Interpretation, Conclusion



- Active area of research

- New applications of abstract interpretation
- Proof assistants
- New abstract domains
- Faster analysis techniques