

Neural Abstract Interpretation

Shaurya Gomber, Gagandeep Singh

FM/SE Seminar, Spring 2024
UIUC

Presented by:

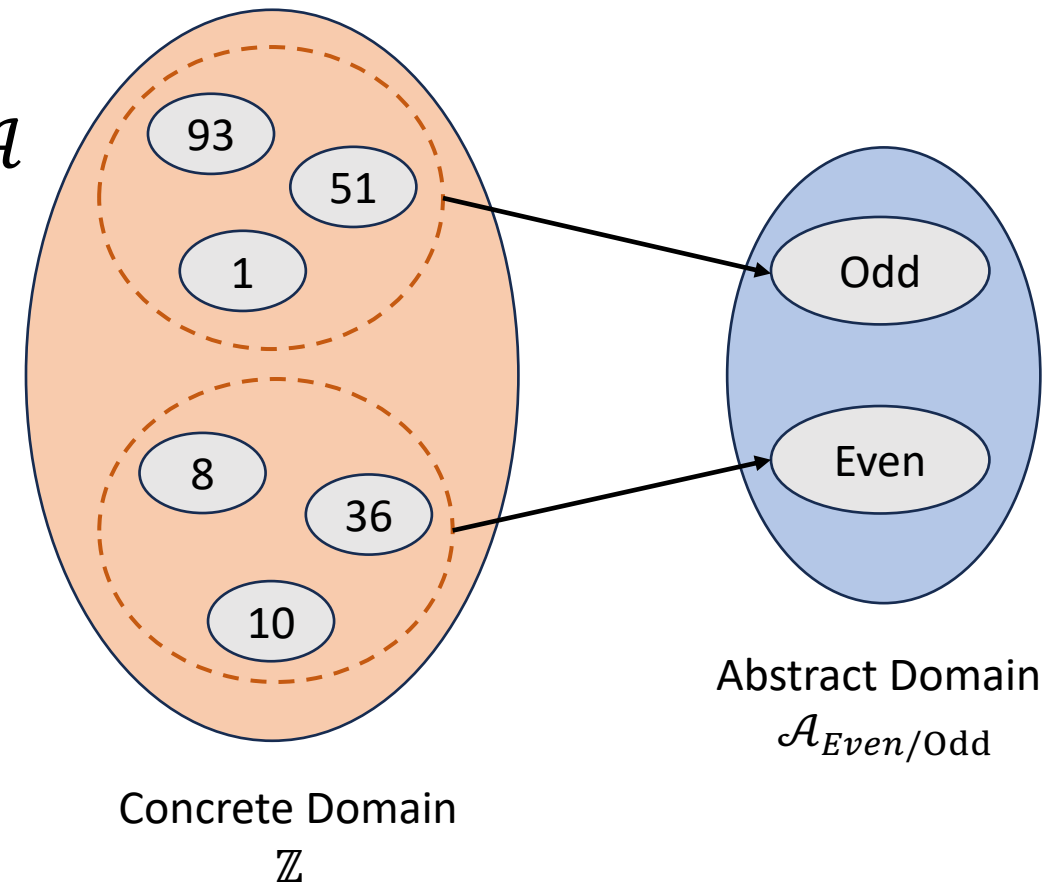
Shaurya Gomber (sgomber2@illinois.edu)

2nd yr MS CS, UIUC



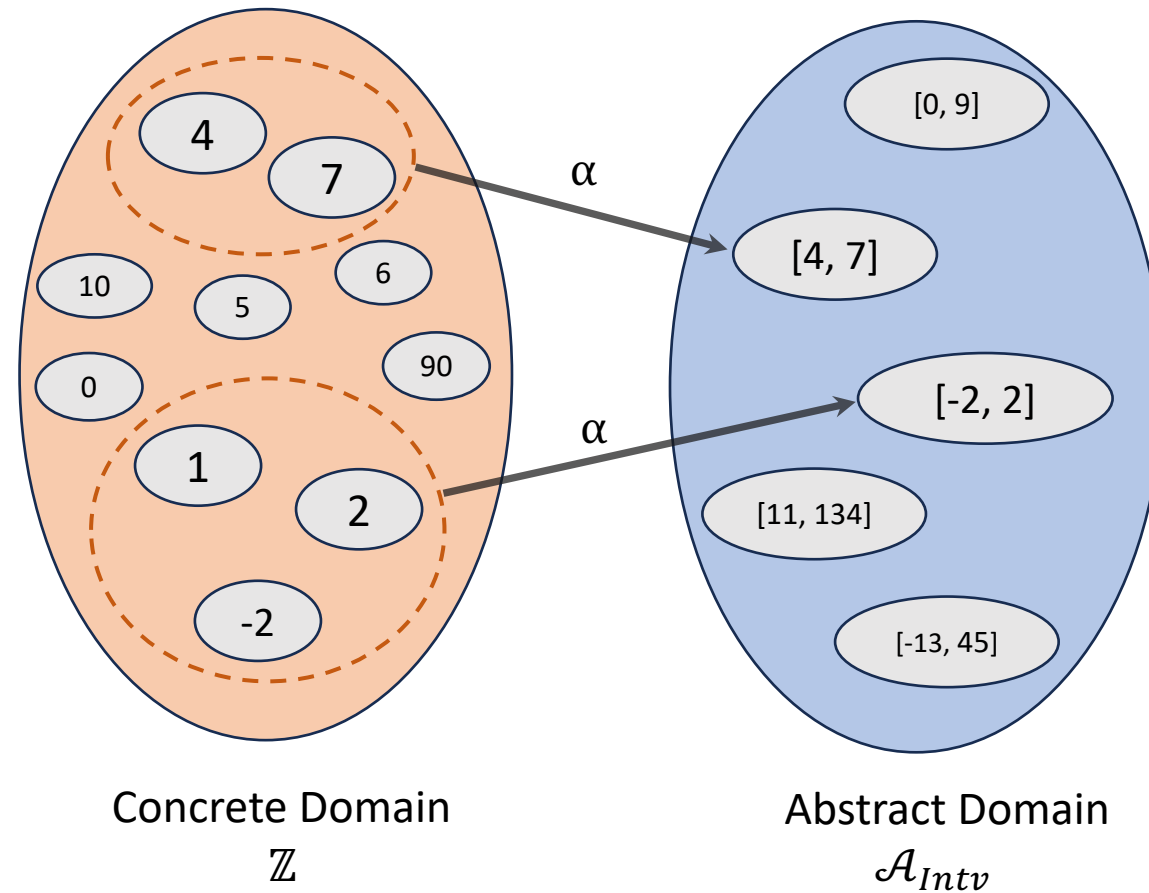
Abstract Interpretation

- Widely used to analyze programs, neural networks, and real-world systems.
- Concrete Domain $\mathcal{C} \rightarrow$ Abstract Domain \mathcal{A}
- “Suitable finiteness” of \mathcal{A} makes analysis tractable



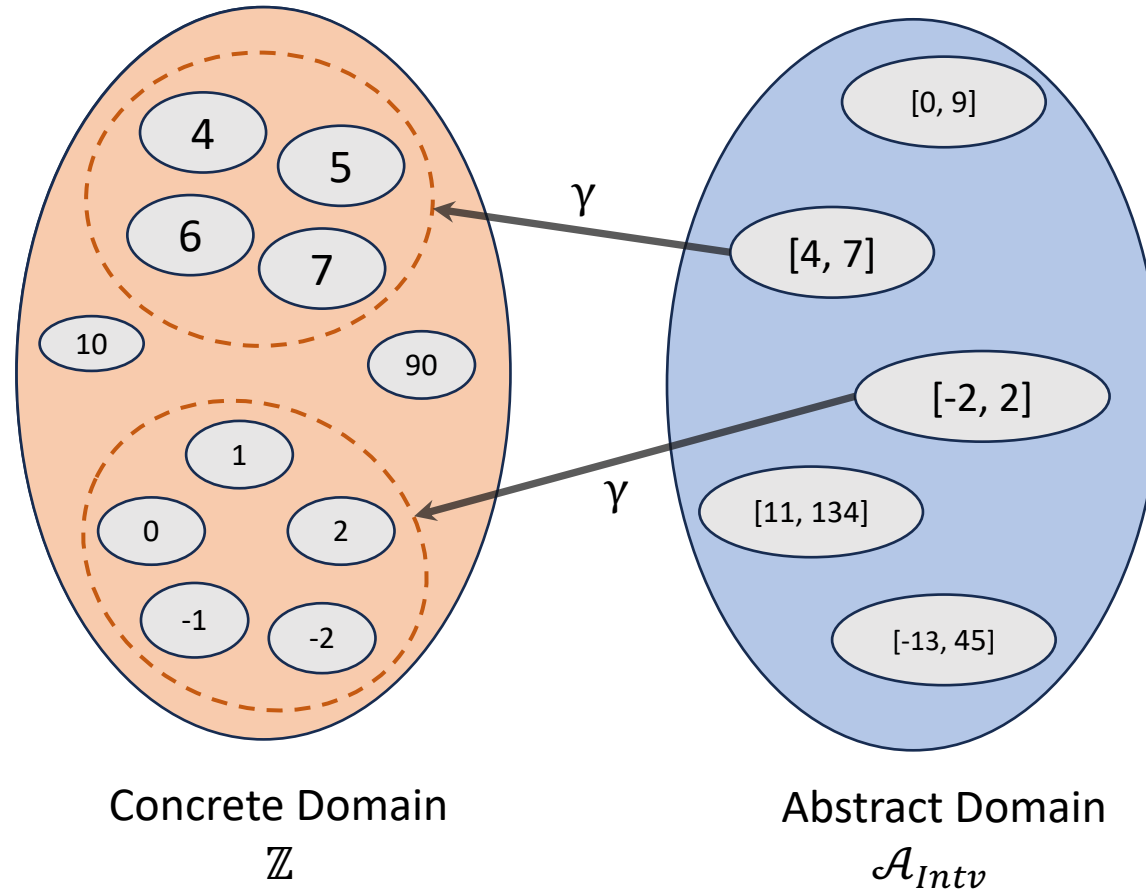
Abstraction Function

Abstraction Function (α): $\mathcal{P}(\mathcal{C}) \rightarrow \mathcal{A}$



Concretization Function

Concretization Function (α): $\mathcal{A} \rightarrow \mathcal{P}(\mathcal{C})$

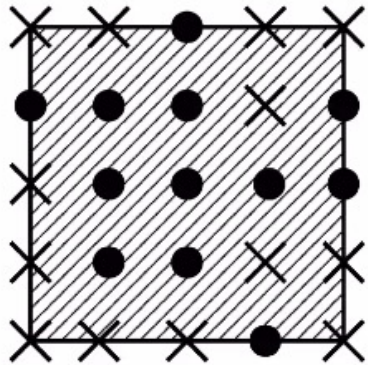


Abstract Domains

Intervals

$$x \in [-1.2, 3.4]$$

$$y \in [3, 8.2]$$

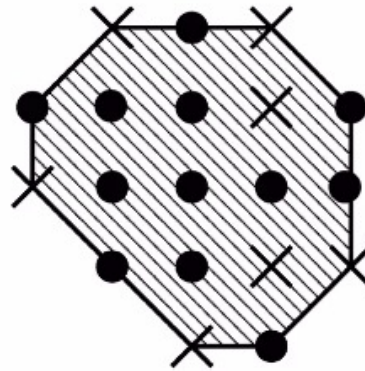


Octagon

$$\pm x \pm y \leq c_i$$

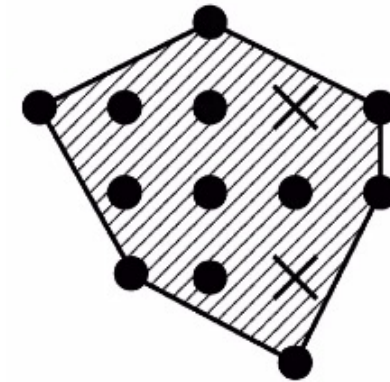
$$\pm x \leq d_i$$

$$\pm y \leq e_i$$



Polyhedra

$$c_1x_1 + c_2x_2 + \dots + c_nx_n \leq p$$



Precision increases but so does complexity

Figures: [A Miné 2006](#)

1. [A. Mine, "The octagon abstract domain."](#)
2. [P. Cousot and N. Halbwachs, "Automatic discovery of linear restraints among variables of a program,"](#)

Abstract Transformers

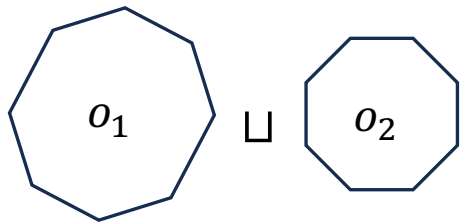
Operator op in \mathcal{C} \rightarrow Need an equivalent \widehat{op} in \mathcal{A}

$$x + y \rightarrow [1, 2] \hat{+} [3, 4]$$

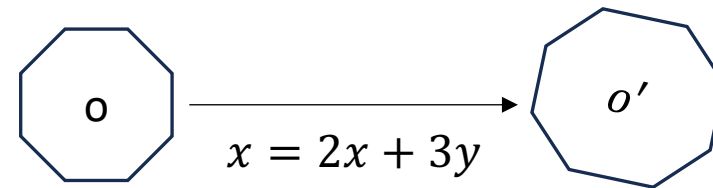
Interval Add

$$z = abs(x) \rightarrow \hat{x} = [-9, 15] \rightarrow \hat{z} = \widehat{abs}(\hat{x})$$

Interval Abs



Octagon Join

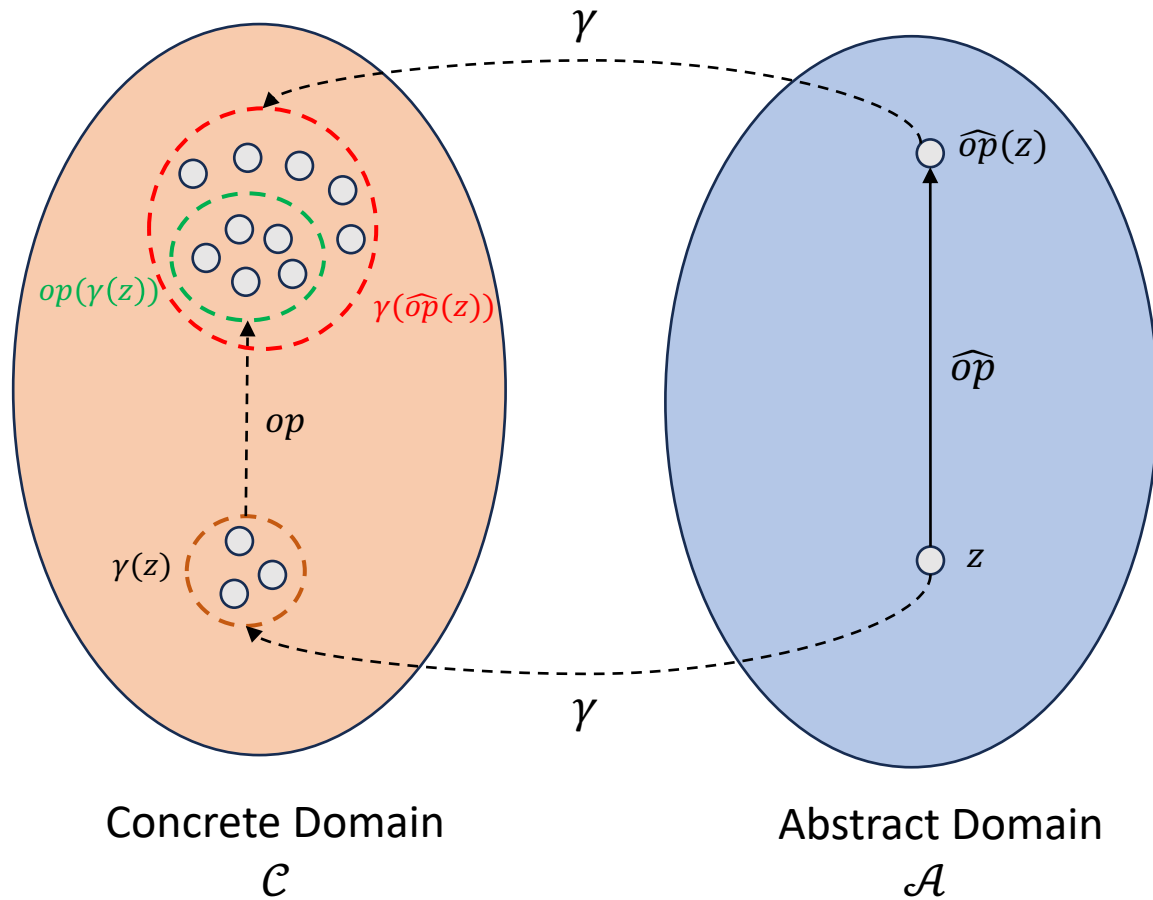


Octagon Affine Assignment

Abstract Transformers transform abstract elements as result of operations in the concrete domain!

\widehat{op} denotes the Abstract Transformer for operator op .

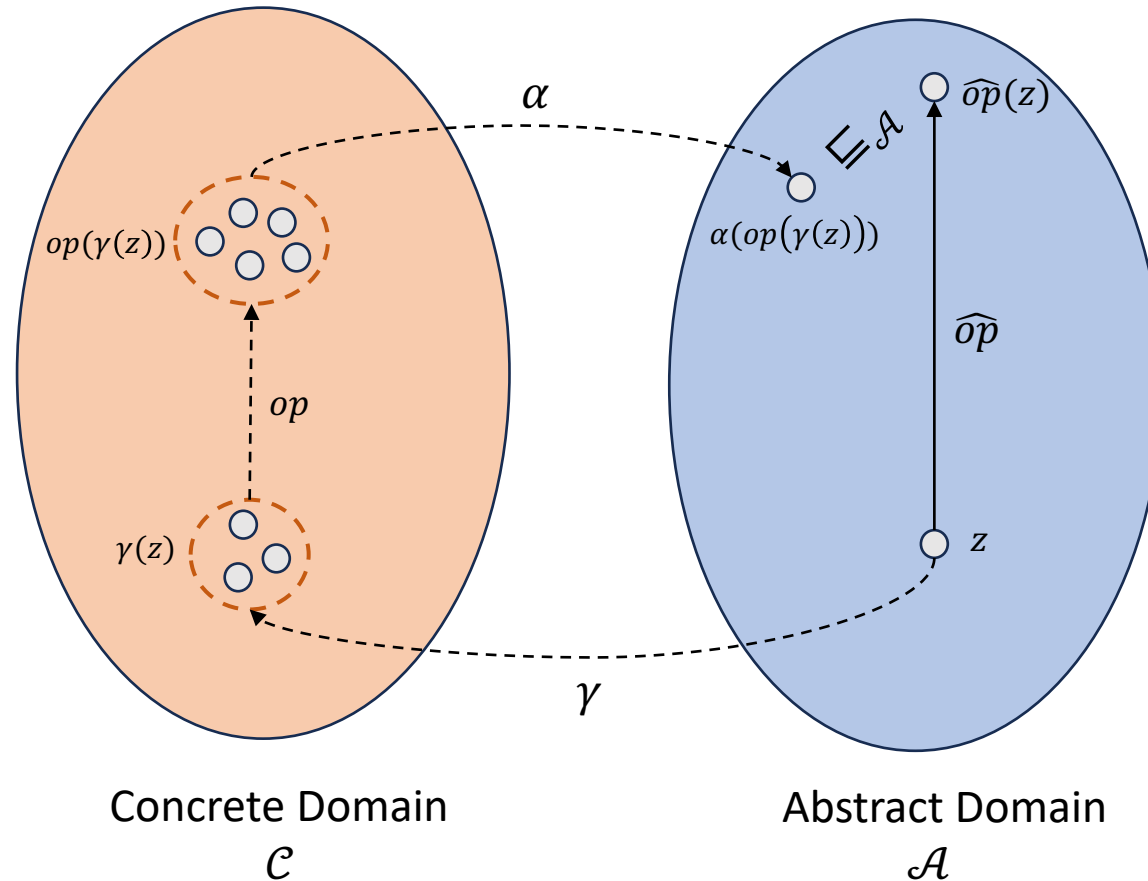
Soundness and Precision of Transformers



Soundness: $op(\gamma(z)) \sqsubseteq_{\mathcal{C}} \gamma(\widehat{op}(z))$

Precision: $\gamma(\widehat{op}(z)) \setminus op(\gamma(z))$

Most-precise (“Best”) Transformer $\widehat{op}^\#$



- $\alpha(op(\gamma(z)))$ is known as the best transformer.
- $\alpha(op(\gamma(z)))$ can not be directly computed using z .

Issues with current abstract transformers

Tedious to implement as:

- Need to ensure soundness always.
- Most precise implementations are computationally expensive: Octagon Join (cubic), Polyhedra Join (exponential)
- Intricate optimizations like octagon decompositions [1] are needed to make them scalable.
- Some hand-crafted transformers are imprecise: affine assignment in the Octagon domain.
- A fundamental trade-off between soundness, precision, and cost of the transformers.

Motivates the need to automatically learn sound and precise abstract transformers to ease development effort!

General Optimization Problem

Find the sound and most-precise abstract transformer \widehat{op} for op from a set of functions \mathcal{F}

$$\widehat{op} = \min_{f \in \mathcal{F}} \sum_{a \in \mathcal{A}} \mathcal{L}_P(\widehat{op}^\#(a), f(a)) \quad \text{s.t.} \quad \sum_{a \in \mathcal{A}} \mathcal{L}_S(op, a, f(a)) = 0$$

PRECISION

- $\mathcal{L}_P(a_1, a_2)$ measures how “big” a_2 is as compared to a_1
- Ensures that learned transformer is closest to $\widehat{op}^\#$ in precision.

SOUNDNESS

- $\mathcal{L}_S(op, a, f(a)) = 0$ if $f(a)$ is a sound output of f on a .
- Ensures f is sound on all $a \in \mathcal{A}$

This problem is hard to solve because:

- \mathcal{A} has an infinite size!
- $\widehat{op}^\#$ just a specification $\rightarrow \mathcal{L}_P(a_1, a_2)$ thus hard to compute!
- $\mathcal{L}_S(op, a, f(a))$ checks for soundness: $op(\gamma(a)) \sqsubseteq_C \gamma(f(a))$
 - \rightarrow via SMT
 - \rightarrow expensive and not differentiable!

Neural Abstract Transformers

To make the solving feasible, we propose supervised and unsupervised relaxations of the above problem.

Neural Abstract Transformers: neural networks that serve as abstract transformers.

- Data-driven approach to learn transformers.
- Ability of neural networks to effectively approximate complex functions [1] allows us to learn sound and precise transformers easily.

This also separates us from the present body of works like [2].
These use symbolic methods to learn transformers from a DSL.

May not be able to learn complex transformers like affine assignment in octagon domain.

1. [K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators](#)

2. [Pankaj Kumar Kalita, Sujit Kumar Muduli, Loris D'Antoni, Thomas Reps, and Subhajit Roy. 2022. Synthesizing abstract transformers](#)

Benefits of Neural Transformers

Neural Abstract Transformers generated by our supervised and unsupervised approaches have the following benefits:

1. Automatic generation of transformers with varying soundness and precision **eases development costs**.
2. The neural transformers can act as **fast and sometimes even more precise** replacements for the current hand-crafted transformers. Unsound cases can be handled by *resorting to hand-crafted transformers' outputs*.
3. Neural transformers being **differentiable** allow for **differentiable abstract interpretation**. This allows us to pose and solve interesting problems like invariant generation as gradient-guided learning methods.

Now, we describe the supervised and unsupervised approaches to learn these transformers

Supervised Learning Relaxation

- Dataset $\mathcal{D} = \{X_i, y_i\}$ for an abstract transformer \widehat{op}
- Use current hand-crafted transformers to collect data.

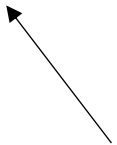
Neural Abstract Transformer $\widehat{op}^*(X_i; \theta)$ can be learned using:

$$\min_{\theta} \mathbb{E}_{(X_i, y_i) \sim \mathcal{D}} [\alpha * \mathcal{L}'_S(y_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}'_P(y_i, \widehat{op}^*(X_i; \theta))]$$

Soundness Ensuring Loss



Precision Ensuring Loss



Supervised Learning Relaxation (Soundness)

$$\min_{\theta} \mathbb{E}_{(X_i, y_i) \sim \mathcal{D}} [\alpha * \mathcal{L}'_S(y_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}'_P(y_i, \widehat{op}^*(X_i; \theta))]$$

- $\mathcal{L}'_S(a_1, a_2) = 0$ implies a_2 over-approximates a_1 , i.e. $a_2 \sqsubseteq_A a_1$ or $\gamma(a_2) \sqsubseteq_C \gamma(a_1)$
- Otherwise $\mathcal{L}'_S(a_1, a_2)$ gives a differentiable *proxy* for size of the set $\gamma(a_2) \setminus \gamma(a_1)$
- Guides model's output $\widehat{op}^*(X_i; \theta)$ to over-approximate ground truth (y_i) which ensures soundness!

Intervals: $\mathcal{L}'_S([l_1, u_1], [l_2, u_2]) = \max(l_2 - l_1, 0) + \max(u_1 - u_2, 0)$

$$\mathcal{L}'_S([1, 2], [0, 5]) = 0$$

$$\mathcal{L}'_S([1, 6], [2, 4]) = (2-1) + (6-4) = 3$$

Guides model to expand $[2, 4]$ to fit $[1, 6]$ inside it

Supervised Learning Relaxation (Precision)

$$\min_{\theta} \mathbb{E}_{(X_i, y_i) \sim \mathcal{D}} [\alpha * \mathcal{L}'_S(y_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}'_P(y_i, \widehat{op}^*(X_i; \theta))]$$

- $\mathcal{L}'_P(a_1, a_2)$ gives a differentiable approximation of how “big” a_2 is as compared to a_1 .
- Needed to maintain precision; only soundness loss can lead to results like $[-\infty, \infty]$

Intervals: $\mathcal{L}'_p([l_1, u_1], [l_2, u_2]) = (u_2 - l_2) - (u_1 - l_1)$

$$\mathcal{L}'_p([1, 2], [0, 5]) = 5 - 1 = 4$$

[0,5] a sound approximation for [1,2], but is bigger!

Guides model to reduce [0, 5] to match size of [1, 2]

Supervised Learning Relaxation (Tradeoff)

$$\min_{\theta} \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}} [\alpha * \mathcal{L}'_S(y_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}'_P(y_i, \widehat{op}^*(X_i; \theta))]$$

↓
Soundness Weight

↓
Precision Weight

Soundness-Precision Tradeoff

Soundness Loss tries to increase the output size.

Precision Loss tries to decrease the output size.

Multi-objective learning problem with conflicting objectives!

Weights can be tweaked to get neural transformers with varying soundness and precision!

Supervised Learning Results (Interval)

Weights (α, β) (Soundness, Precision)	Interval Abs		Interval Join	
	Soundness (%)	Imprecision	Soundness (%)	Imprecision
(-, -)	20.03	4.44	3.88	0.16
(1, 1)	26.39	1.57	32.34	40.16
(2, 1)	47.43	5.74	40.53	25.70
(5, 1)	66.88	11.70	63.10	43.58
(7, 1)	84.02	10.39	73.07	113.78
(10, 1)	97.72	18.41	89.24	116.31
(50, 1)	99.99	40.63	99.57	191.72



Soundness increases
Precision decreases

Soundness %: Percentage of sound outputs on a test set of 10,000 input-output pairs.

Imprecision: The avg. difference in sizes of intervals produced by the model and the ground truth (for SOUND cases).

Supervised Learning: Octagons

$$o_1: \pm v_i \pm v_j \leq c_{ij}$$

$$o_2: \pm v_i \pm v_j \leq c_{ij}'$$

$$\mathcal{L}'_S(o_1, o_2) = \sum_{i,j} \text{ite}(c_{ij} - c_{ij}' > 0, c_{ij} - c_{ij}', 0)$$

$$\begin{aligned} o_1: v_1 - v_2 &\leq 5 \\ o_2: v_1 - v_2 &\leq 3 \end{aligned} \quad \mathcal{L}'_S(o_1, o_2) = 5 - 3 = 2$$

Enforces constants of o_2 to be bigger than o_1 ;
sufficient for soundness

$$\mathcal{L}'_P(o_1, o_2) = \sum_{i,j} |c_{ij} - c_{ij}'|$$

Enforces constants of o_2 are closer to o_1 ;
maintains precision

Soundness Weight (α)	Precision Weight (β)	Soundness Measure (%)	Imprecision Measure
1	1	9.6	49.96
10	1	36.6	85.30
20	1	49.0	110.51
50	1	64.5	129.26
100	1	79.0	184.58
250	1	86.3	291.23

Neural Octagon Join (3 variables)

Soundness %: on a test set of 1000 input-output pairs.

Imprecision: average difference between the inequality constants of the model's output and the ground truth.


Unsupervised Learning Relaxation

- Dataset $\mathcal{D} = \{X_i\}$ for an abstract transformer \widehat{op} for an operator op

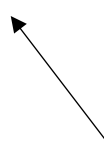
Neural Abstract Transformer $\widehat{op}^*(X_i; \theta)$ can be learned using:

$$\min_{\theta} \mathbb{E}_{(X_i) \sim \mathcal{D}} [\alpha * \mathcal{L}_S''(op, X_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}_P''(\widehat{op}^*(X_i; \theta))]$$

Soundness Ensuring Loss



Precision Ensuring Loss

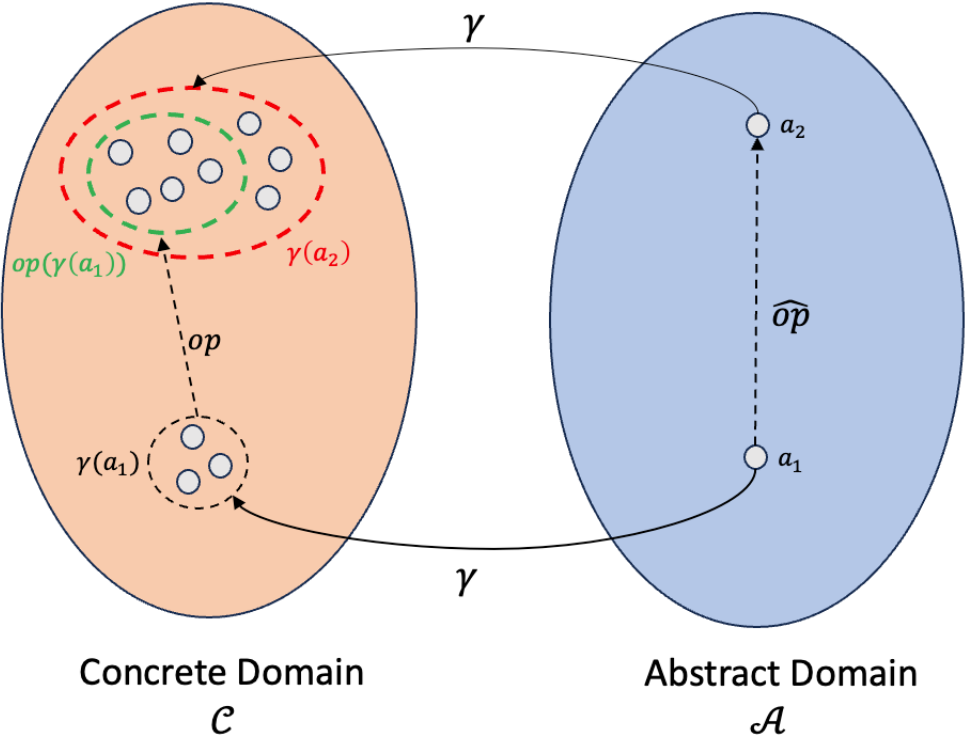


Unsupervised Learning Relaxation (Soundness)

$$\min_{\theta} \mathbb{E}_{(X_i) \sim \mathcal{D}} [\alpha * \mathcal{L}_S''(op, X_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}_P''(\widehat{op}^*(X_i; \theta))]$$

Main challenge here is that we do not have ground truth for reference (and over-approximating)

We want that $\mathcal{L}_S''(op, a_1, a_2)$ should give 0 if a_2 is a sound output with respect to op on a_1



$$op(\gamma(a_1)) \sqsubseteq_C \gamma(a_2)$$

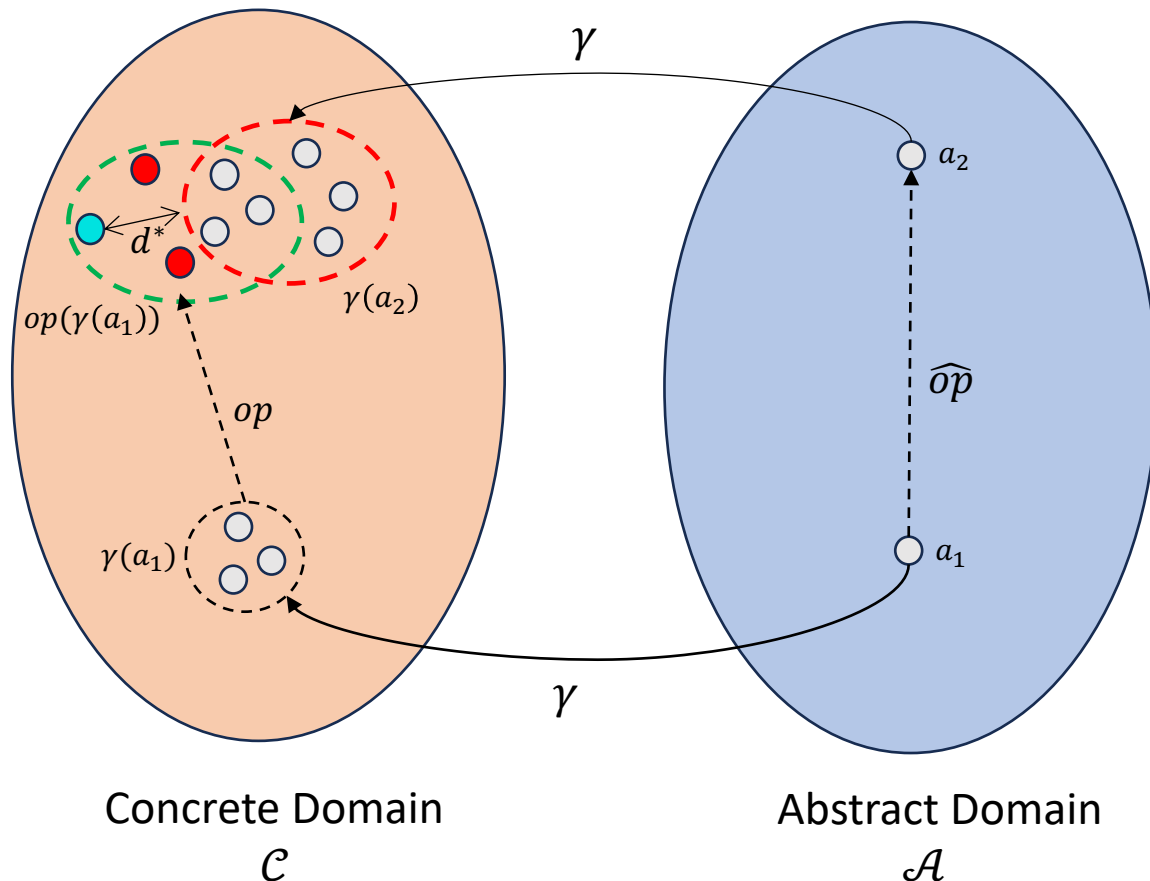
Else, we want $\mathcal{L}_S''(op, a_1, a_2)$ to guide model towards a_2 which ensures the above condition!

The above condition can be checked by an SMT solver but the 0/1 returned by the solver can not be used to guide the learning.

Unsupervised Learning Relaxation (Soundness)

$$\min_{\theta} \mathbb{E}_{(X_i) \sim \mathcal{D}} [\alpha * \mathcal{L}_S''(op, X_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}_P''(\widehat{op}^*(X_i; \theta))]$$

To tackle this, we introduce the notion of **Maximum Violating Concrete Point (MVCP)**



● Points that violate $op(\gamma(a_1)) \in_C \gamma(a_2)$

● Violating point farthest from $\gamma(a_2)$

$$MVCP(op, a_1, a_2)$$

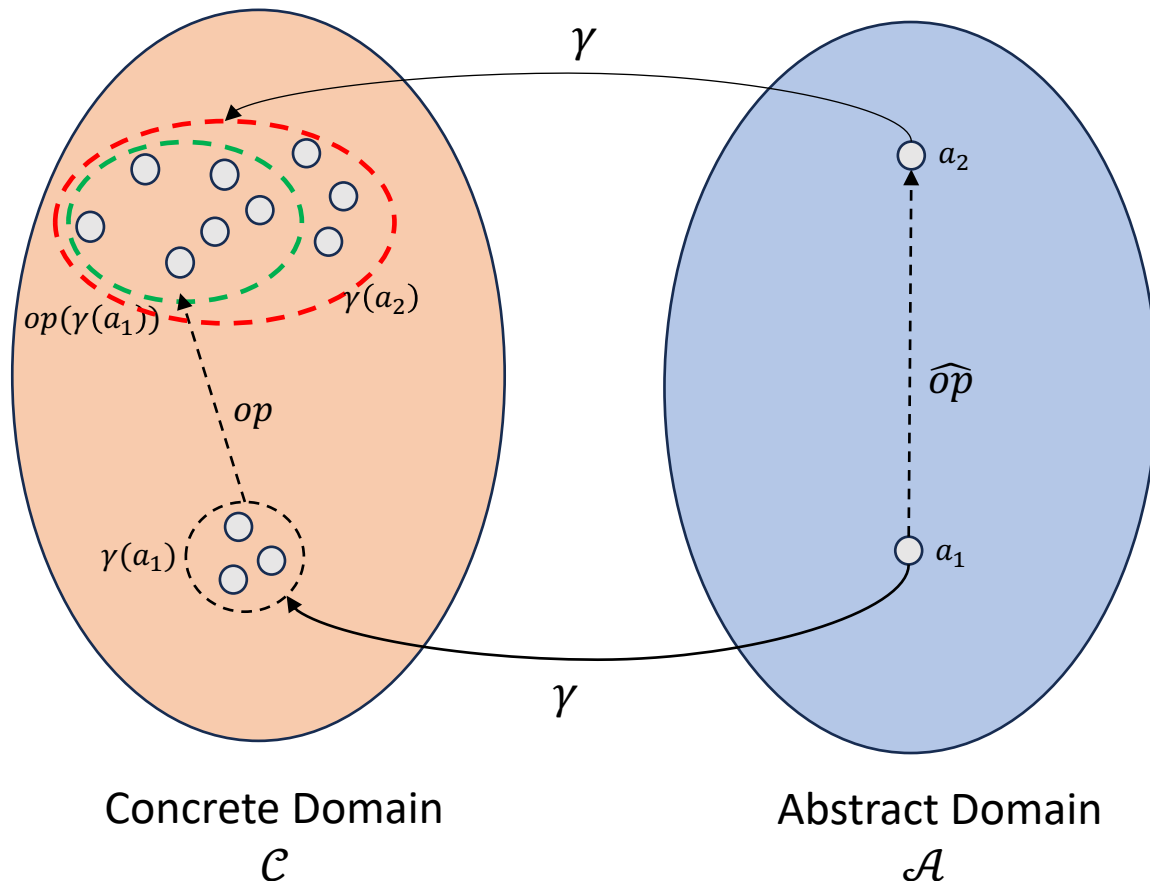
$$\mathcal{L}_S''(op, a_1, a_2) = D(MVCP(op, a_1, a_2), \gamma(a_2))$$

for distance metric $D(c, \gamma(a))$

Unsupervised Learning Relaxation (Soundness)

$$\min_{\theta} \mathbb{E}_{(X_i) \sim \mathcal{D}} [\alpha * \mathcal{L}_S''(op, X_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}_P''(\widehat{op}^*(X_i; \theta))]$$

To tackle this, we introduce the notion of **Maximum Violating Concrete Point (MVCP)**



● Points that violate $op(\gamma(a_1)) \in_C \gamma(a_2)$

● Violating point farthest from $\gamma(a_2)$

$$MVCP(op, a_1, a_2)$$

$$\mathcal{L}_S''(op, a_1, a_2) = D(MVCP(op, a_1, a_2), \gamma(a_2))$$

for distance metric $D(c, \gamma(a))$

Unsupervised Learning Relaxation (Soundness)

$$\min_{\theta} \mathbb{E}_{(X_i) \sim \mathcal{D}} [\alpha * \mathcal{L}_S''(op, X_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}_P''(\widehat{op}^*(X_i; \theta))]$$

MVCP and \mathcal{L}_S'' example for Interval domain

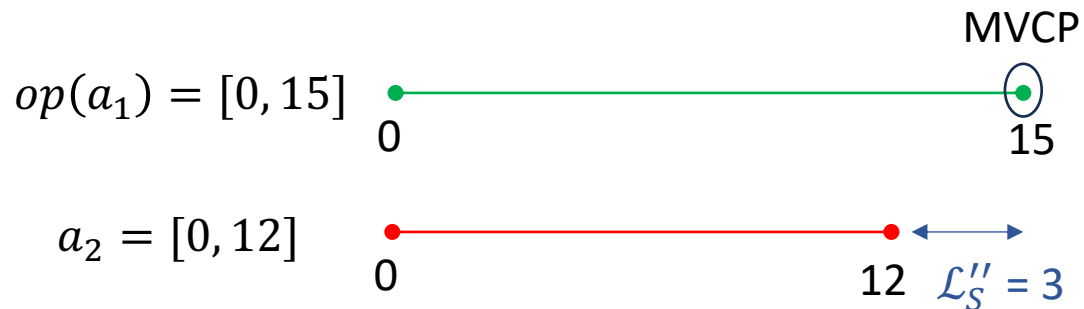
Consider learning transformer for the *abs* function.

$$\begin{aligned} op &= abs \\ a_1 &= [-10, 15] \\ a_2 &= [0, 12] \end{aligned}$$

$$D(c, [l_1, u_1]) = \begin{cases} l_1 - c & \text{if } c < l_1 \\ c - u_1 & \text{if } c > u_1 \\ 0 & \text{otherwise} \end{cases}$$

MVCP Query to solver:

$$\begin{aligned} &\operatorname{argmax}_c D(c, [0, 12]) \\ &s.t. \exists x. -10 \leq x \leq 15 \wedge c = |x| \end{aligned}$$



Once MVCPs are computed for a batch, \mathcal{L}_S'' can be computed as $D(mvcp, \gamma(a_2))$, which is differentiable!

Unsupervised Learning Relaxation (Precision)

$$\min_{\theta} \mathbb{E}_{(X_i) \sim \mathcal{D}} [\alpha * \mathcal{L}_S''(op, X_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}_P''(\widehat{op}^*(X_i; \theta))]$$

$\mathcal{L}_P''(a)$ gives a differentiable approximation of the size of a .

Helps maintain precision by avoiding results like $[-\infty, \infty]$.

Intervals: $\mathcal{L}_P''([l_1, u_1]) = u_1 - l_1$

Unsupervised Learning Relaxation (Precision)

$$\min_{\theta} \mathbb{E}_{(X_i) \sim \mathcal{D}} [\alpha * \mathcal{L}_S''(op, X_i, \widehat{op}^*(X_i; \theta)) + \beta * \mathcal{L}_P''(\widehat{op}^*(X_i; \theta))]$$

↓
Soundness Weight

↓
Precision Weight

Weights can be tweaked to get neural transformers with varying soundness and precision!

Unsupervised Learning Results (Interval)

Weights (α, β) (Soundness, Precision)	Interval Abs		Interval Join	
	Soundness (%)	Imprecision	Soundness (%)	Imprecision
(-, -)	20.03	4.44	3.91	26.80
(20, 10)	25.04	4.29	38.99	164.61
(30, 10)	63.04	25.86	53.65	219.08
(50, 10)	85.96	36.95	93.03	255.93
(75, 10)	100	73.17	97.95	277.70

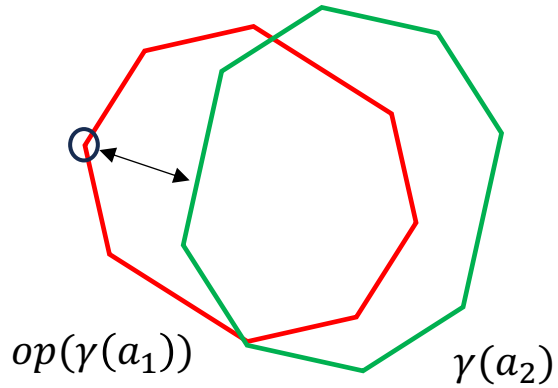


Soundness increases
Precision decreases

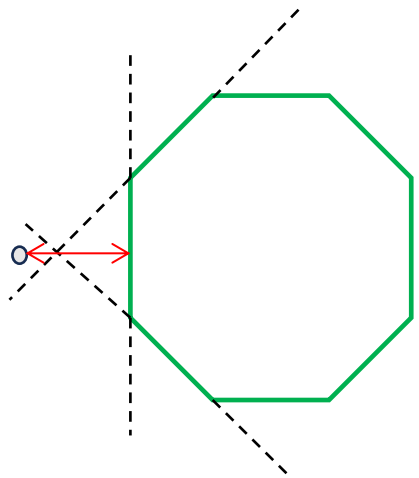
Soundness %: Percentage of sound outputs on a test set of 10,000 input-output pairs.

Imprecision: The avg. difference in sizes of intervals produced by the model and the ground truth (for SOUND cases).

Unsupervised Learning: Octagons



MVCP can be found by encoding the octagon constraints and notion of D in SMT solver



$D(c, o)$: maximum perpendicular distance from non-satisfying constraints.

Soundness Weight (α)	Precision Weight (β)	Soundness Measure (%)	Imprecision Measure
10	10	20.6	85.30
20	10	35.1	150.51
50	10	57.1	229.26
100	10	80.0	384.58

Octagon affine assignment $x = ax + by$

What next?

Our results show the efficacy of our methods to learn neural abstract transformers.

They are generated automatically and can be plugged into downstream tasks in place of hand-crafted transformers.

However, using neural transformers opens up one more avenue: **differentiable abstract interpretation.**

Differentiability use case: Differentiable Learning of Loop Invariants

Consider a while program: $\mathcal{P} = \text{while}(B) \text{ do } C \text{ od}$

Say we need to find octagonal invariants (of the form $\pm x \pm y \leq c$)

O_{inv} is a valid octagon loop invariant if:

$$(O_{init} \in O_{inv}) \wedge (\hat{C}(O_{inv} \hat{\cap} B) \in O_{inv})$$

\hat{C} and $\hat{\cap}$ are abstract transformers for C and \cap (*Conjunction with guard*)

Now, say we train neural transformers \hat{C}^* and $\hat{\cap}^*$ for C and \cap

We can initialize invariant as a random octagon o and do gradient descent using:

$$\mathcal{L}'_S(O_{init}, o) + \mathcal{L}'_S(\hat{C}^*(o \hat{\cap}^* B), o)$$

$$\mathcal{L}'_S(o_1, o_2) = \sum_{i,j} \text{ite}(c_{ij} - c'_{ij} > 0, c_{ij} - c'_{ij}, 0)$$

Differentiability use case: Differentiable Learning of Loop Invariants

The differentiable search succeeds in finding valid invariants!

```
x = 100;  
y = 150;  
while (y <= 600)  
{  
  x = x + y;  
  y = 2*y;  
}
```

Some generated invariants:

1. $y \geq 65.51$, $x - y \leq -49.95$, $-x - y \leq 74.89$
2. $x - y \leq 13.239$

Note that if x_1, y_1 enter the loop body

$$x_2 = x_1 + y_1 \text{ and } y_2 = 2 * y_1$$

$$x_2 - y_2 = x_1 - y_1$$

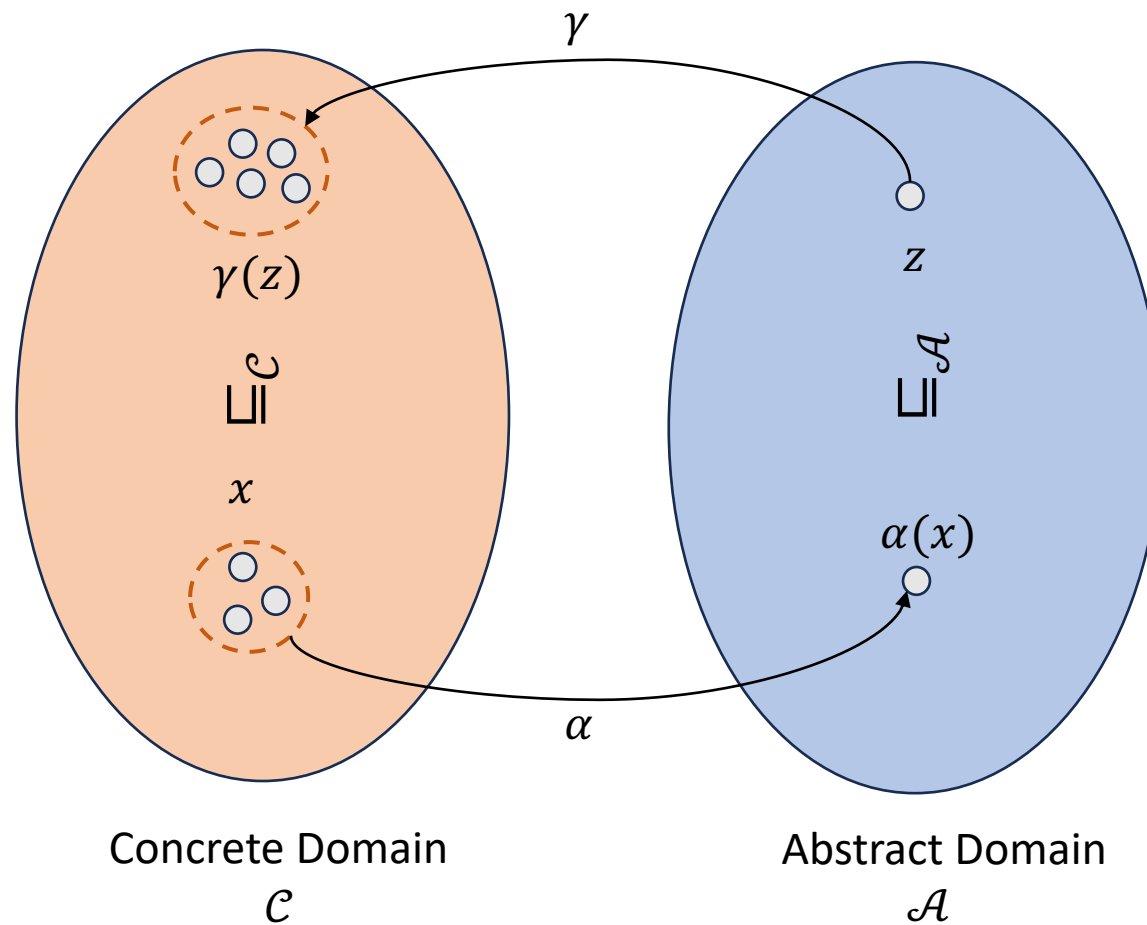
Future Works

- Better representation for octagon (like graphs and using GNNs)
- Evaluate the speed and precision benefits of neural transformers.
- Explore other avenues to use differentiable abstract interpretation.

Questions?

Backup Slides

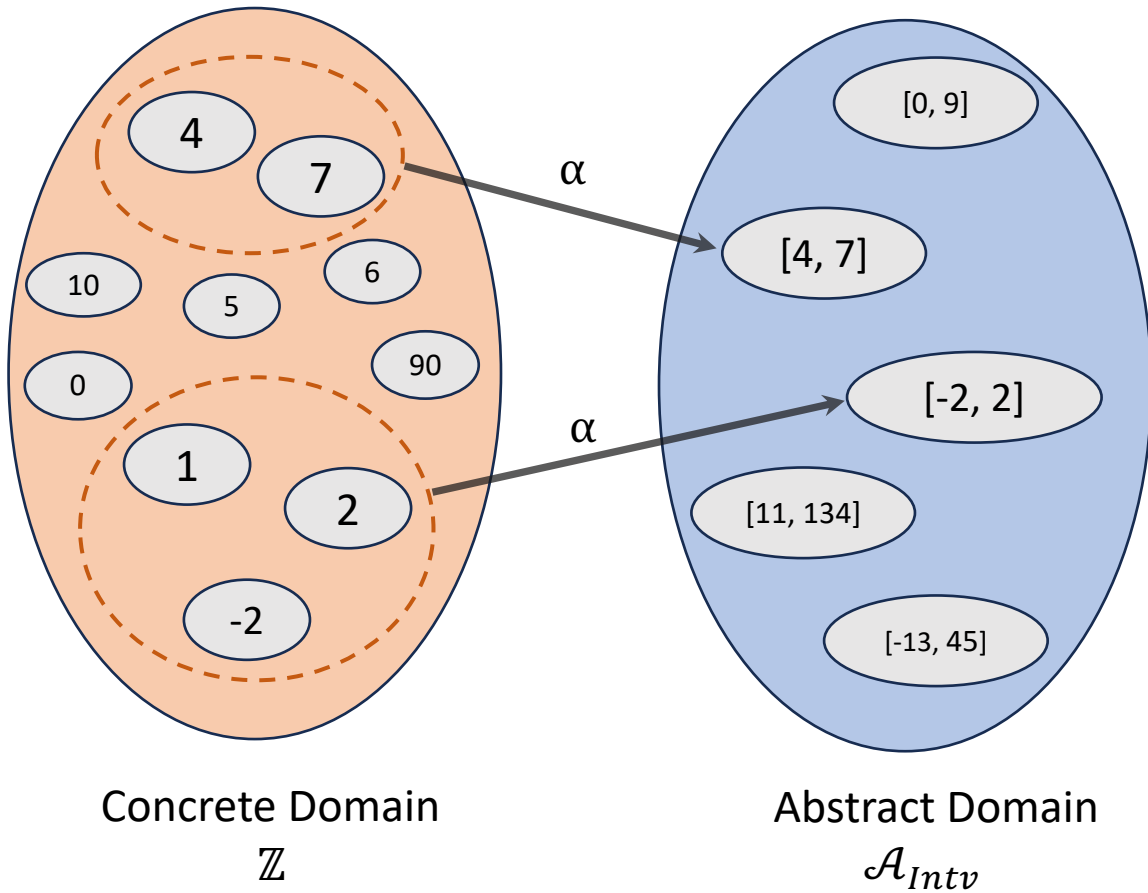
Galois Connection



$$\alpha(x) \sqsubseteq_{\mathcal{A}} z \iff x \sqsubseteq_{\mathcal{C}} \gamma(z)$$

Abstraction introduces imprecision

Abstraction Function (α): $\mathcal{P}(\mathbb{Z}) \rightarrow \mathcal{A}$



Concretization Function (γ): $\mathcal{A} \rightarrow \mathcal{P}(\mathbb{Z})$

