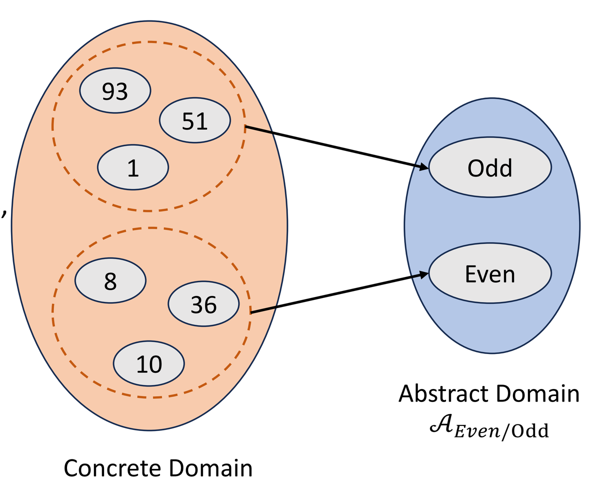
# Neural Abstract Interpretation

**Shaurya Gomber** (sgomber2@illinois.edu)
**Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA**

## ABSTRACT INTERPRETATION

Helps analyze programs, neural networks, and real-world systems by interpreting them in an abstract domain!

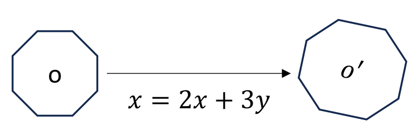



## ABSTRACT TRANSFORMERS

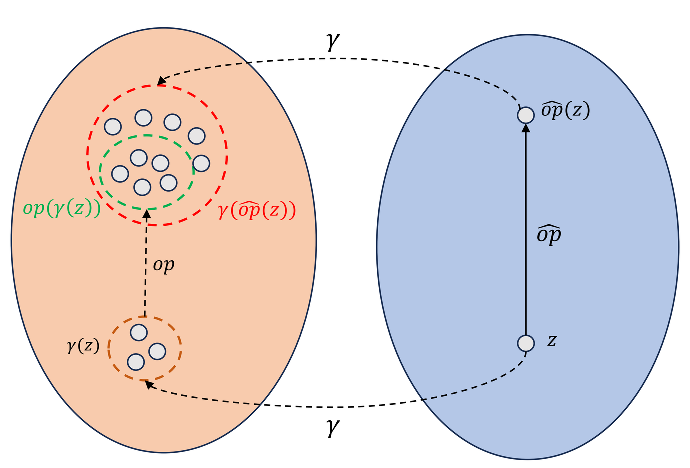
Operator $op$ in $\mathcal{C}$ $\rightarrow$ Need an equivalent $\widehat{op}$ in $\mathcal{A}$

x + y → [1, 2] $\hat{+}$ [3, 4]

Interval Add

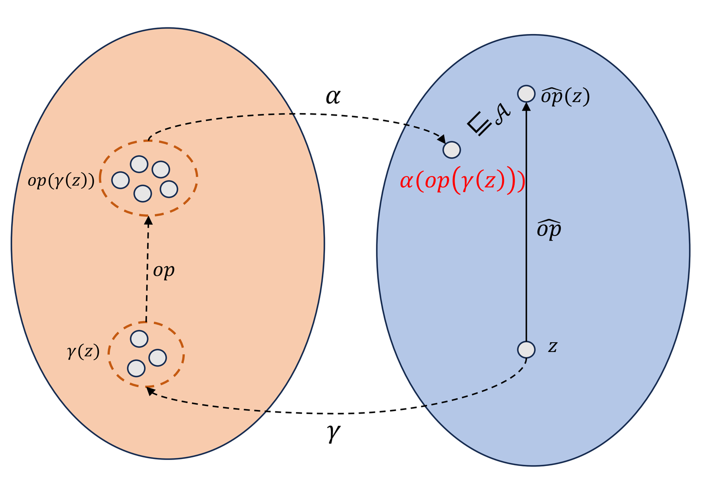



Octagon Affine Assignment




Soundness: $op(\gamma(z)) \sqsubseteq_C \gamma(\widehat{op}(z))$

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




Most-precise best transformer $\widehat{op}^{\#}$

$\alpha(op(\gamma(z)))$

## TRANSFORMERS: TEDIOUS TO IMPLEMENT!!

- Ensure soundness always.
- Most-precise implementations: **computationally expensive**.
  (join in Octagon domain is cubic and exponential in Polyhedra)
- Scalable implementations need **intricate optimizations**.
  (like octagon and polyhedral decomposition)
- Hand-crafted transformers can be **imprecise**.
  (like affine assignment in the octagon domain)
- Trade-off between Ease of Implementation, Soundness, Precision, and Efficiency.

MOTIVATES THE NEED TO LEARN SOUND AND PRECISE TRANSFORMERS AUTOMATICALLY!

## GENERAL LEARNING PROBLEM IS HARD

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

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

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

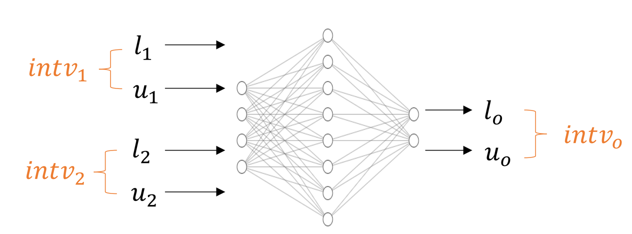
- $\mathcal{L}_S\big(op, a, f(a)\big)$ = 0 if $f(a)$ is a sound output of $f$ on $a$.
- Ensures f is sound on all $a \in \mathcal{A}$.

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

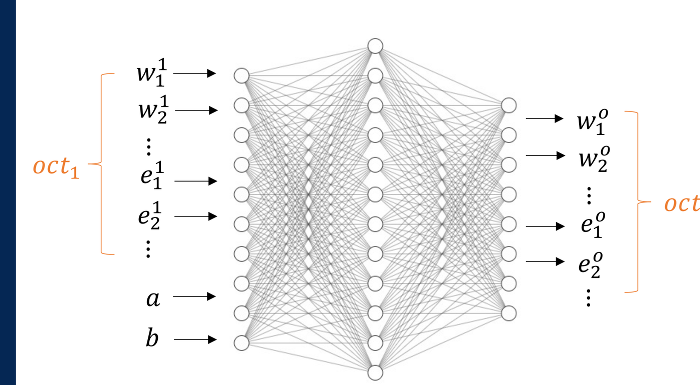
## NEURAL ABSTRACT TRANSFORMERS

As neural networks effectively approximate complex functions, we propose:

**NEURAL NETWORKS THAT SERVE AS ABSTRACT TRANSFORMERS!**




Interval Addition




Octagon Affine Assignment
(x = a*x + b*y)

Neural transformers allow for data-driven learning of transformers.

We propose **supervised and unsupervised relaxations** of the learning problem to train these transformers.

## KEY BENEFITS OF NEURAL TRANSFORMERS

1. Automatic generation of transformers with varying soundness and precision **eases development costs**.
2. Neural transformers are **differentiable**, which allows us to pose and solve interesting problems like invariant generation as gradient-guided learning methods.
3. Neural transformers can offer **faster alternatives** to computationally intensive transformers, such as the octagon join transformer, and **more precise alternatives** to transformers, like octagon affine assignment. (Unsound cases can be handled by resorting to hand-crafted transformers' outputs)

## SUPERVISED LEARNING OF TRANSFORMERS

$$\min_{\theta} \mathrm{E}_{(X_i, y_i) \sim \mathcal{D}}[\alpha \, * \mathcal{L}_S'\big(y_i, N_\theta(X_i)\big) + \beta \, * \mathcal{L}_P{}'\big(y_i, N_\theta(X_i)\big)]$$

### Soundness Enforcing Loss $\mathcal{L}_S'$

$\mathcal{L}_S'(a_1, a_2)$: Differentiable *proxy* for size of the set $\gamma(a_1) \setminus \gamma(a_2)$
$\mathcal{L}_S'(a_1, a_2) = 0$ implies $a_2$ over-approximates $a_1$: $\gamma(a_1) \sqsubseteq_C \gamma(a_2)$

**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 $\quad$ $\mathcal{L}_S'([1, 6], [2, 4])$ = (2-1) + (6-4) = 3 (Guides model's output to include [1, 6])

**Octagons:** $\mathcal{L}_S'(o_1, o_2) = \sum_{i,j} ite\,(c_{ij} - c_{ij}' > 0,\, c_{ij} - c_{ij}', 0)$ $\quad$ $o_1$: $\pm v_i \pm v_j \le c_{ij}$ $\quad$ $o_2$: $\pm v_i \pm v_j \le c_{ij}'$

$o_1$: $v_1 - v_2 \le 5$
$o_2$: $v_1 - v_2 \le 3$ $\quad$ $\mathcal{L}_S'(o_1, o_2)$ = 5 – 3 = 2

### Precision Enforcing Loss $\mathcal{L}_P'$

$\mathcal{L}_P'(a_1, a_2)$: Differentiable approximation of how "big" $a_2$ is as compared to $a_1$.
Reducing it leads to "smaller" more-precise outputs.

**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 (Guides model's output to match size of [1,2])

**Octagons:** $\mathcal{L}_P'(o_1, o_2) = \sum_{i,j} |c_{ij} - c_{ij}{}'|$ $\quad$ $o_1$: $\pm v_i \pm v_j \le c_{ij}$ $\quad$ $o_2$: $\pm v_i \pm v_j \le c_{ij}'$

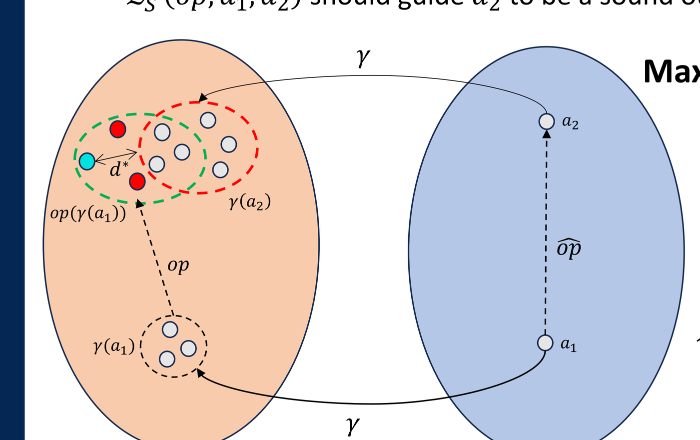
***α* and *β* are soundness and precision weights that manage the trade-off!**

## UNSUPERVISED LEARNING OF TRANSFORMERS

$$\min_{\theta} \mathrm{E}_{(X_i) \sim \mathcal{D}}[\alpha \, * \mathcal{L}_S''\big(op, X_i, N_\theta(X_i)\big) + \beta \, * \mathcal{L}_P{}''\big(N_\theta(X_i)\big)]$$

### Soundness Enforcing Loss $\mathcal{L}_S''$

$\mathcal{L}_S''(op, a_1, a_2)$ should guide $a_2$ to be a sound output of $op$ on $a_1$: $op(\gamma(a_1)) \sqsubseteq_C \gamma(a_2)$



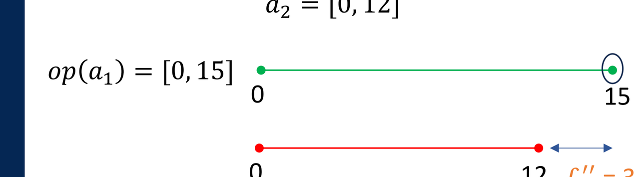

**Maximum Violating Concrete Point (MVCP)**

- Points that violate $op(\gamma(a_1)) \sqsubseteq_C \gamma(a_2)$
- Violating point farthest from $\gamma(a_2)$
  $MVCP(op, a_1, a_2)$

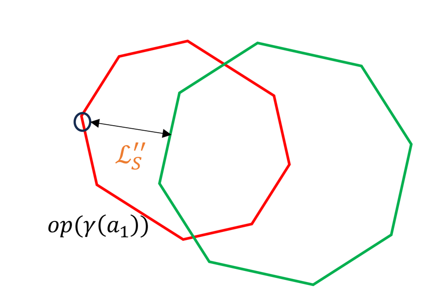
$\mathcal{L}_S''(op, a_1, a_2) = D(MVCP(op, a_1, a_2), \gamma(a_2))$
for distance metric $D(c, \gamma(a))$

**Intervals**

$op = abs$
$a_1 = [-10, 15]$
$a_2 = [0, 12]$




**Octagons**




### Precision Enforcing Loss $\mathcal{L}_P''$

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

**Intervals** $\quad$ $\mathcal{L}_P''([l, u]) = (u - l)$

**Octagons** $\quad$ $\mathcal{L}_P''(o) = \sum_{i,j} |c_{ij}|$ $\quad$ O: $\pm v_i \pm v_j \le c_{ij}$

## RESULTS

### Supervised Learning

Soundness increases

| Weights $(\alpha, \beta)$ | Interval Abs | | Interval Join | |
|---|---|---|---|---|
| (Soundness, Precision) | 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 |

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).

| Soundness Weight ($\alpha$) | Precision Weight ($\beta$) | 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

| Weights $(\alpha, \beta)$ | Interval Abs | | Interval Join | |
|---|---|---|---|---|
| (Soundness, Precision) | 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 Weight ($\alpha$) | Precision Weight ($\beta$) | 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 $x = ax + by$

### Benefits of Differentiability: Invariant Search

Consider a while program: $\mathcal{P} = < init > \; while(B) \; do \; C \; od$

Find $O_{inv}$ such that:

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

Training neural transformers $\hat{C}^*$ and $\hat{\cap}^*$ for C and ∩

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

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

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

1. $\{y \ge 65.51, \; x - y \le -49.95, -x - y \le 74.89\}$
2. $\{x - y \le 13.239\}$

ILLINOIS