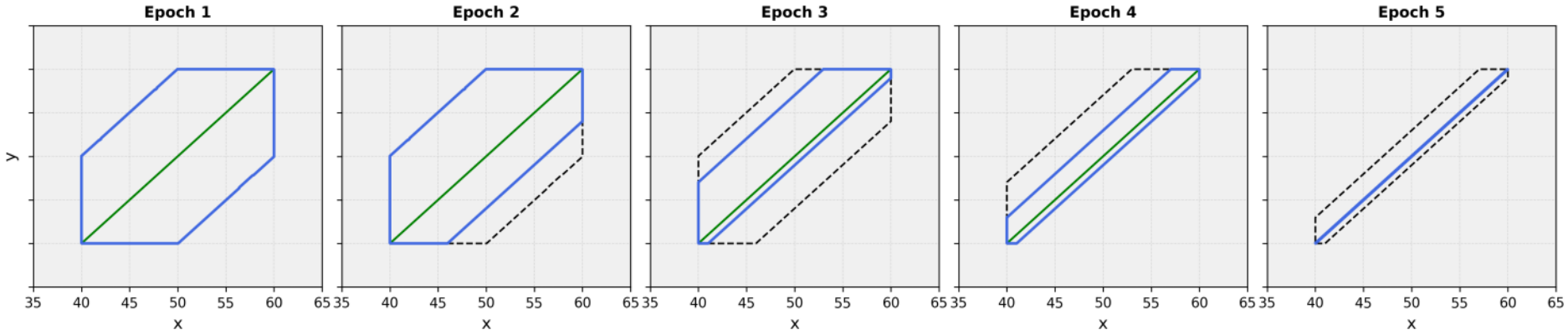
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

PLDI BOULDER 2026

Evolving Abstract Transformers for Gradient-Guided, Adaptable Abstract Interpretation

Shaurya Gomber, Debangshu Banerjee, Gagandeep Singh

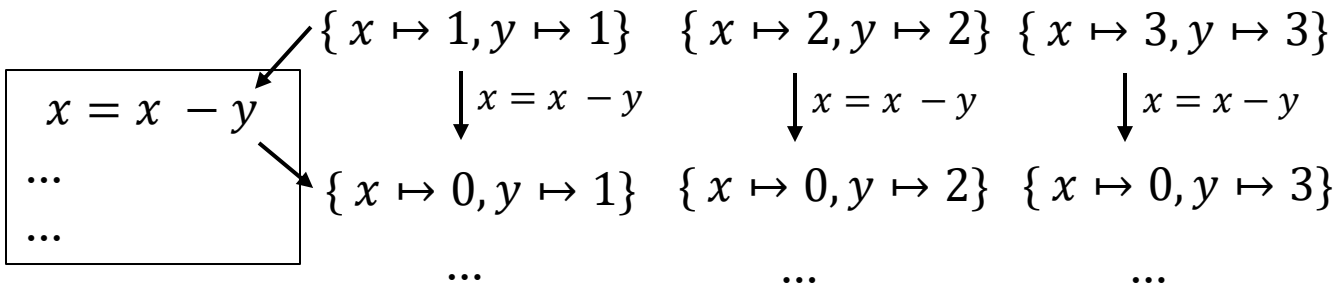


Abstract Interpretation

Automatic way to statically analyze programs, DNNs, real-world systems, etc.

Capture all program behaviors using an abstract domain.

All states reachable at a point are represented by an abstract element.

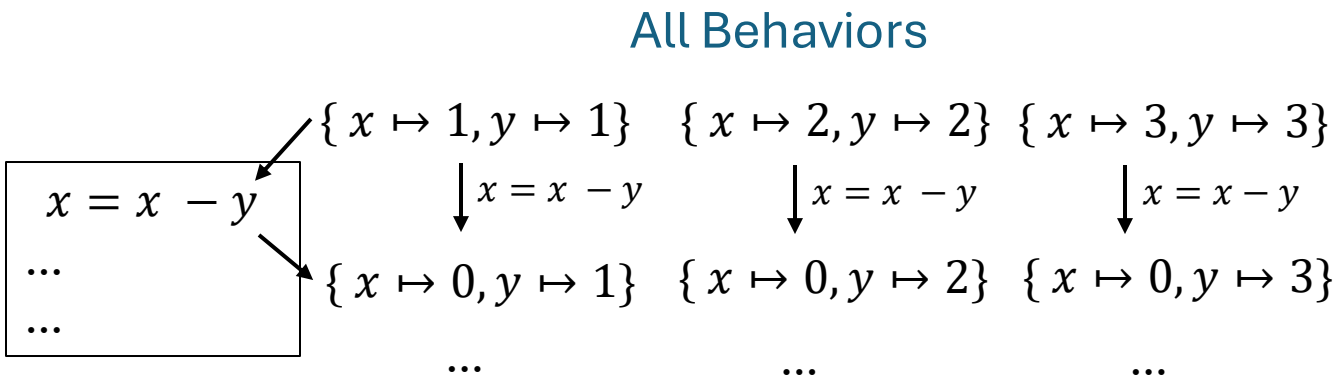


Abstract Interpretation

Automatic way to statically analyze programs, DNNs, real-world systems, etc.

Capture all program behaviors using an abstract domain.

All states reachable at a point are represented by an abstract element.

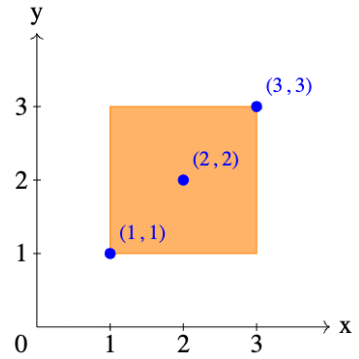


Abstract Interpretation

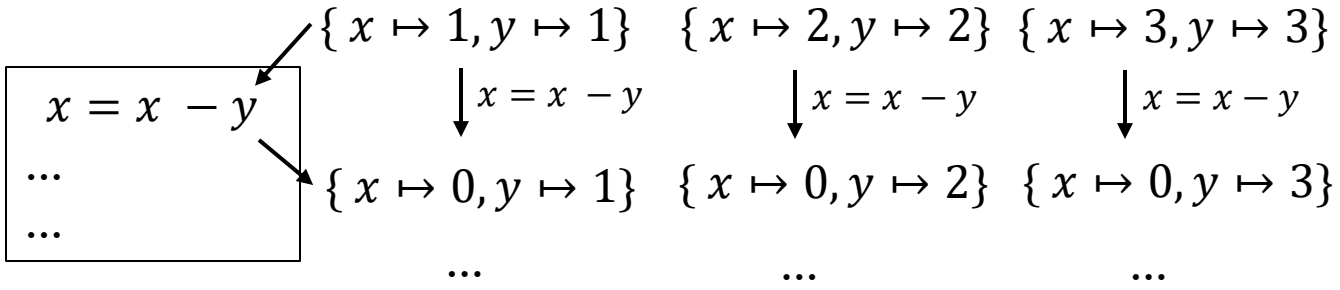
Automatic way to statically analyze programs, DNNs, real-world systems, etc.

Capture all program behaviors using an abstract domain.

All states reachable at a point are represented by an abstract element.



All Behaviors

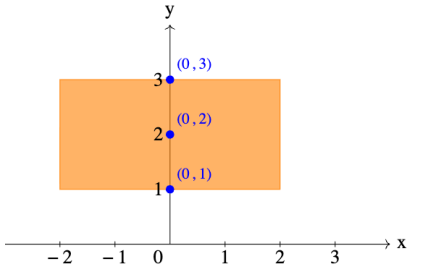


Abstract Domain: Intervals

$\{x \mapsto [1,3], y \mapsto [1,3]\}$

$(x = x - y)^{\#}$

$\{x \mapsto [-2,2], y \mapsto [1,3]\}$

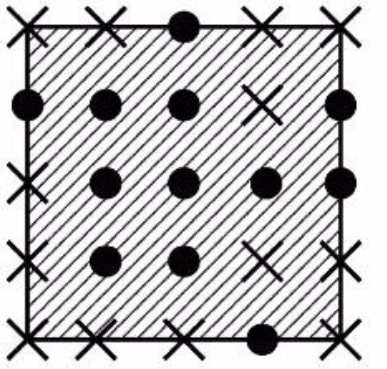


Abstract elements are propagated by applying abstract transformers ($op^{\#}$)

Abstract Domains

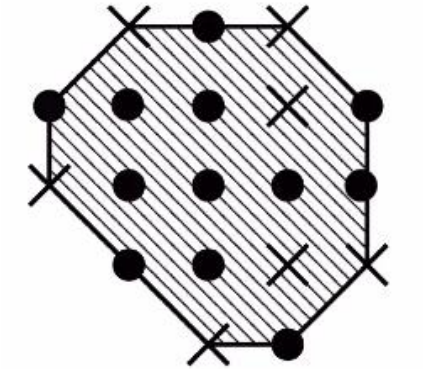
Intervals

$x \in [-1.2, 3.4]$
$y \in [3, 8.2]$



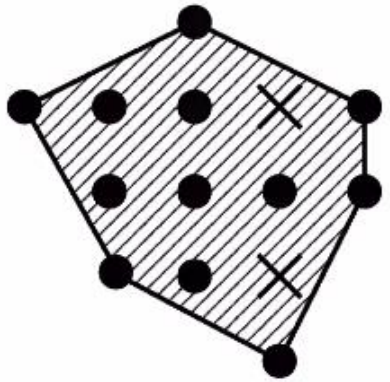
Octagon

$\pm x \pm y \leq c_i$
$\pm x \leq d_i$
$\pm y \leq e_i$



Polyhedra

$c_1 x_1 + c_2 x_2 + \cdots c_n x_n \leq p$



Precision increases but so does complexity

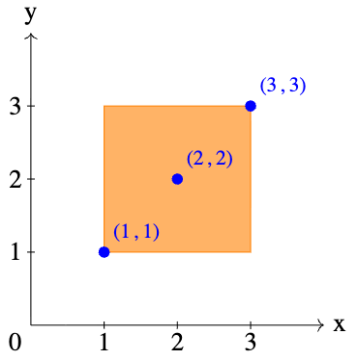
References:

A. Mine, “The octagon abstract domain,”
P. Cousot and N. Halbwachs, “Automatic discovery of linear restraints among variables of a program,”

Abstract Domains

Intervals

$\{ x \mapsto [1,3], y \mapsto [1,3] \}$



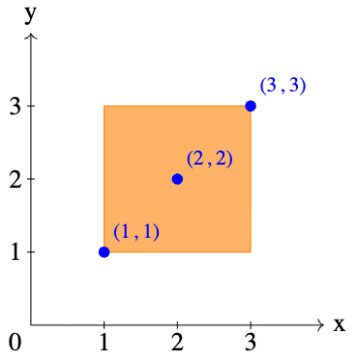
Octagons

$? \leq x \leq ?$
$? \leq y \leq ?$
$? \leq x - y \leq ?$
$? \leq x + y \leq ?$

Abstract Domains

Intervals

$\{ x \mapsto [1,3], y \mapsto [1,3] \}$



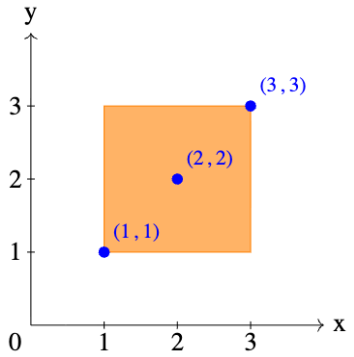
Octagons

$1 \leq x \leq 3$
$1 \leq y \leq 3$
$? \leq x - y \leq ?$
$? \leq x + y \leq ?$

Abstract Domains

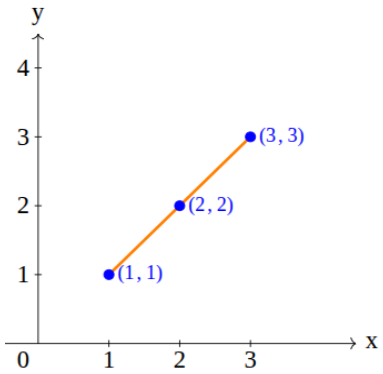
Intervals

$\{ x \mapsto [1,3], y \mapsto [1,3] \}$



Octagons

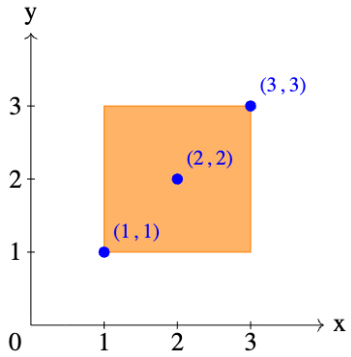
$$
\begin{aligned}
1 &\leq x \leq 3 \\
1 &\leq y \leq 3 \\
0 &\leq x - y \leq 0 \\
2 &\leq x + y \leq 6
\end{aligned}
$$



Abstract Domains

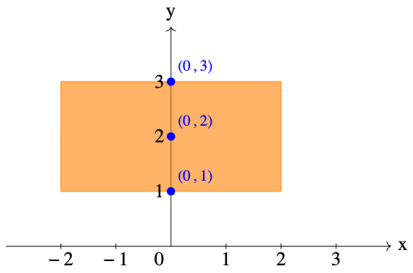
Intervals

$\{ x \mapsto [1,3], y \mapsto [1,3] \}$



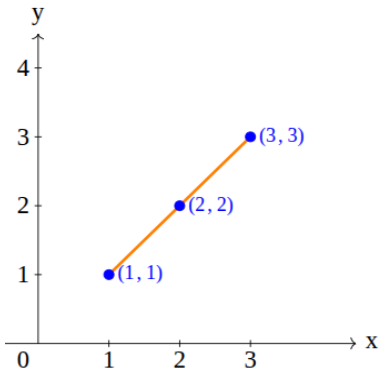
$(x = x \ - y)^{\#}$

$\{ x \mapsto [-2,2], y \mapsto [1,3] \}$



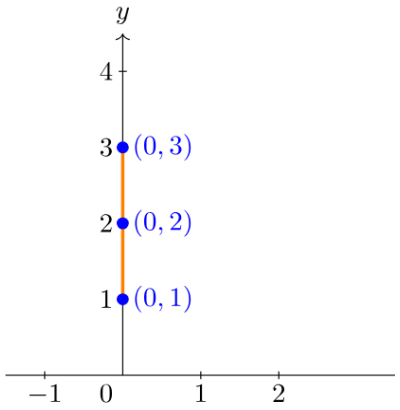
Octagons

$1 \leq x \leq 3$
$1 \leq y \leq 3$
$0 \leq x - y \leq 0$
$2 \leq x + y \leq 6$

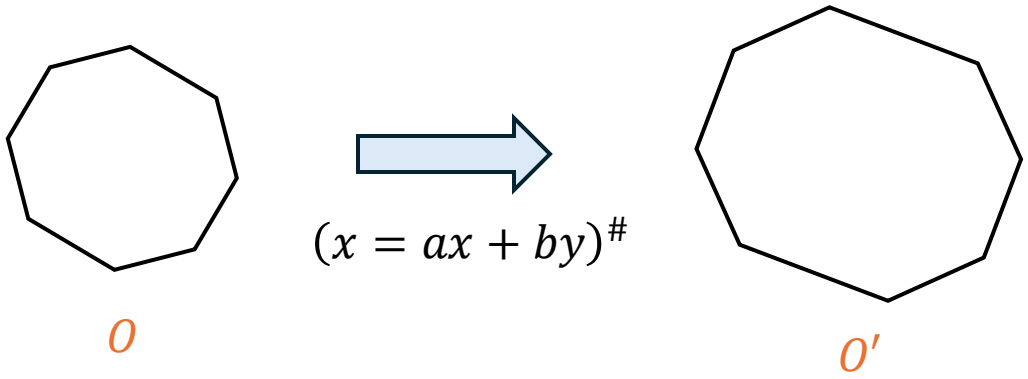


$(x = x \ - y)^{\#}$

$0 \leq x \leq 0$
$1 \leq y \leq 3$
$-3 \leq x - y \leq -1$
$1 \leq x + y \leq 3$

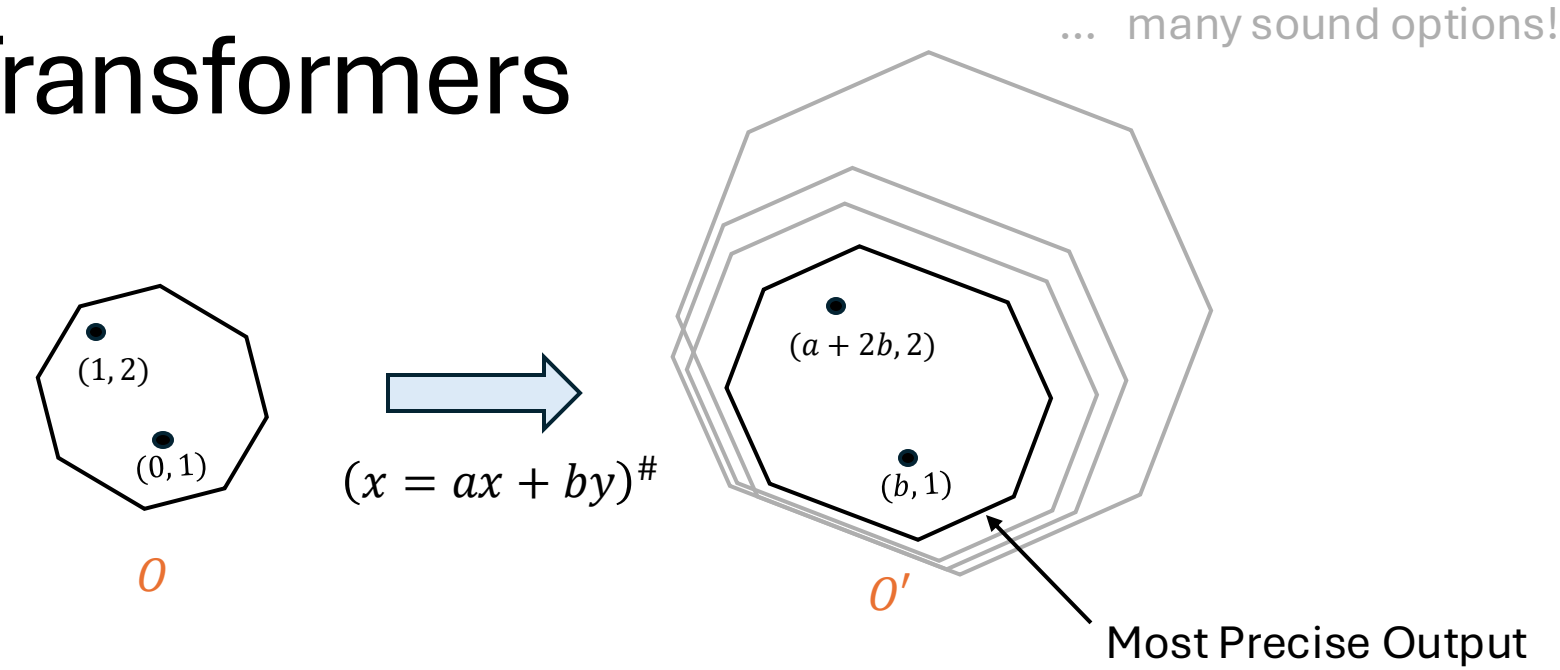


Abstract Transformers



Soundness: Output should cover all possible new states.

Abstract Transformers



Soundness: Output should cover all possible new states.

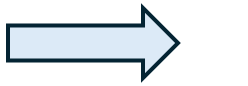
$$\forall\, (x, y) \in O, (ax + by, y) \in O'$$

Precision: How "tight" the sound outputs are!

Limitation 1 - Transformers are imprecise

Computing the most precise outputs is time-consuming!!

$$x \leq 10$$
$$1 \leq y \leq 11$$
$$x - y \leq -1$$
$$x + y \leq 21$$



$(x = x - y)^{\#}$

$$? \leq x \leq ?$$
$$? \leq y \leq ?$$
$$? \leq x - y \leq ?$$
$$? \leq x + y \leq ?$$

$$x_{new} - y_{new} = (x_{old} - y_{old}) - y_{old} = \mathrm{x_{old}} - 2 * \mathrm{y_{old}}$$

$$\max\ x - 2y \quad s.t. \quad \begin{array}{l} x \leq 10 \\ 1 \leq y \leq 11 \\ x - y \leq -1 \\ x + y \leq 21 \end{array}$$

-2

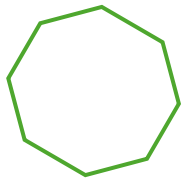
Limitation 1 - Transformers are imprecise

Computing the most precise outputs is time-consuming!!

$$x \leq 10$$
$$1 \leq y \leq 11$$
$$x - y \leq -1$$
$$x + y \leq 21$$

$$(x = x \ - y)^{\#}$$

$$-\infty \leq x \leq -1$$
$$1 \ \leq y \leq 11$$
$$-\infty \leq x - y \leq -2$$
$$-\infty \ \leq x + y \leq 10$$



Precise but expensive as O(V^2) solver calls!

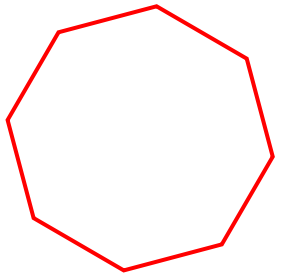
Libraries use cheaper (but less precise) heuristics like Interval Relaxation.

$$x \leq 10$$
$$1 \leq y \leq 11$$
$$x - y \leq -1$$
$$x + y \leq 21$$

$$(x = x \ - y)^{\#}$$

$$x_{old} \in [-\infty, 10]$$
$$y_{old} \in [1, 11]$$
$$x_{new} \ - y_{new} = x_{old} \ - 2 * y_{old}$$
$$= [-\infty, 8]$$

$$-\infty \leq x \leq 9$$
$$1 \ \leq y \leq 11$$
$$-\infty \leq x - y \leq 8$$
$$-\infty \ \leq x + y \leq 10$$



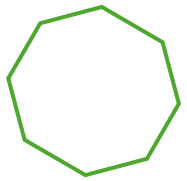
Limitation 1 - Transformers are imprecise

Computing the most precise outputs is time-consuming!!

$$x \leq 10$$
$$1 \leq y \leq 11$$
$$x - y \leq -1$$
$$x + y \leq 21$$

$\Rightarrow$ $(x = x\ -y)^{\#}$

$$-\infty \leq x \leq -1$$
$$1\ \leq y \leq 11$$
$$-\infty \leq x - y \leq -2$$
$$-\infty\ \leq x + y \leq 10$$



Precise but expensive as O(V^2) solver calls!

Libraries use cheaper (but less precise) heuristics like Interval Relaxation.

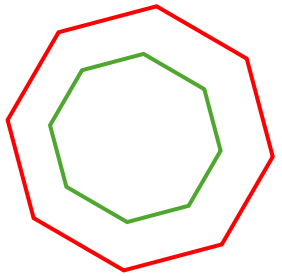
$$x \leq 10$$
$$1 \leq y \leq 11$$
$$x - y \leq -1$$
$$x + y \leq 21$$

$\Rightarrow$ $(x = x\ -y)^{\#}$

$$x_{old} \in [-\infty, 10]$$
$$y_{old} \in [1, 11]$$
$$x_{new}\ - y_{new} = x_{old}\ - 2 * y_{old}$$
$$= [-\infty, 8]$$

$\Rightarrow$

$$-\infty \leq x \leq 9$$
$$1\ \leq y \leq 11$$
$$-\infty \leq x - y \leq 8$$
$$-\infty\ \leq x + y \leq 10$$



Sound but less precise!

Limitation 1 - Transformers are imprecise

Computing the most precise outputs is time-consuming!!

$$x \leq 10,\ 1 \leq y \leq 11,\ x - y \leq -1,\ x + y \leq 21 \quad \Longrightarrow \quad (x = x - y)^{\#} \quad \Longrightarrow \quad -\infty \leq x \leq -1,\ 1 \leq y \leq 11,\ -\infty \leq x - y \leq -2,\ -\infty \leq x + y \leq 10$$

Precise but expensive as $O(V^2)$ solver calls!

PRECISION-EFFICIENCY TRADEOFF

Libraries use cheaper (but less precise) heuristics like Interval Relaxation.

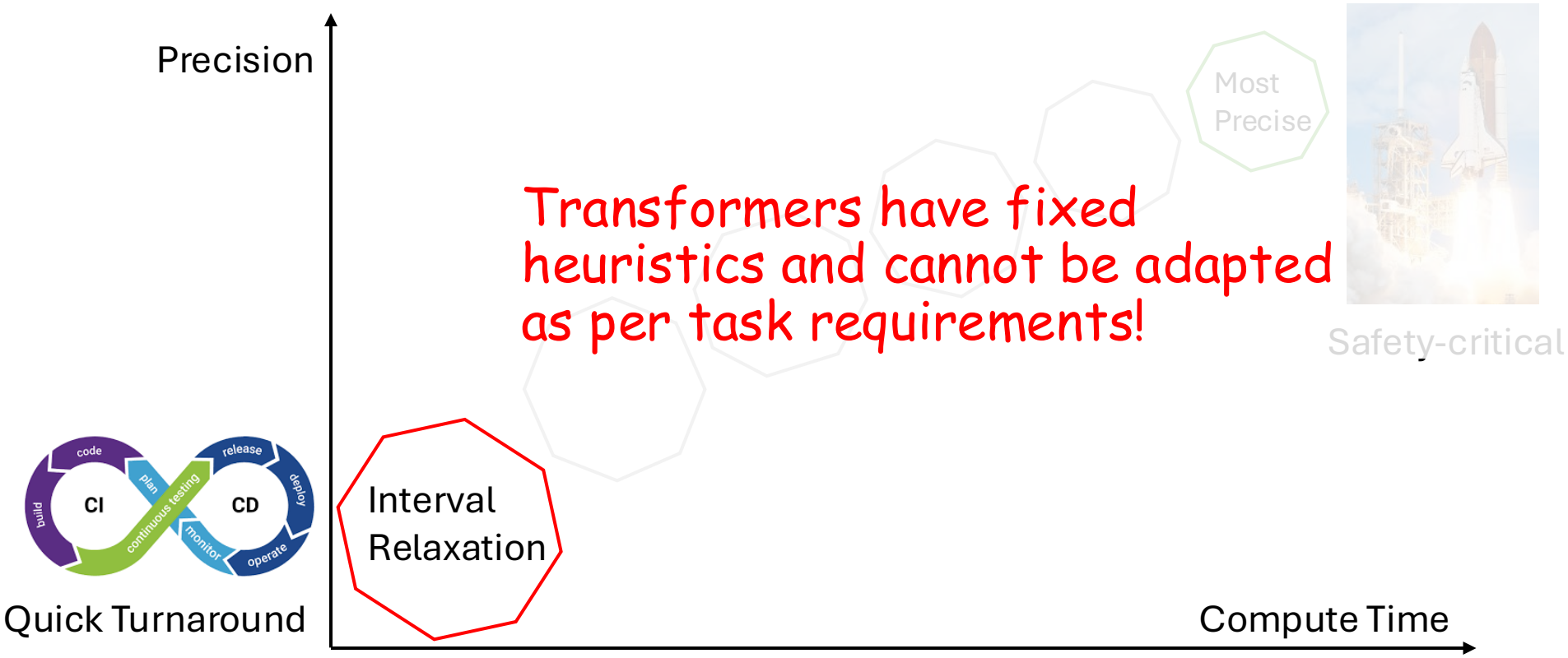
$$x \leq 10,\ 1 \leq y \leq 11,\ x - y \leq -1,\ x + y \leq 21 \quad \Longrightarrow \quad (x = x - y)^{\#}$$

$$x_{old} \in [-\infty, 10],\ y_{old} \in [1, 11],\ x_{new} - y_{new} = x_{old} - 2 * y_{old} = [-\infty, 8] \quad \Longrightarrow \quad -\infty \leq x \leq 9,\ 1 \leq y \leq 11,\ -\infty \leq x - y \leq 8,\ -\infty \leq x + y \leq 10$$

Sound but less precise!

Limitation 2 - Transformers are ‘rigid’

Different tasks have different precision-efficiency requirements.



Key Idea 1 – Sound Space of Outputs

Instead of computing a fixed output, capture a **parametric space of sound outputs**!

$x \le 10$
$1 \le y \le 11$
$x - y \le -1$
$x + y \le 21$

$(x = x\ -y)^{\#}$

$-\infty \le x \le -1$
$1 \le y \le 11$
$-\infty \le x - y \le -2$
$-\infty \le x + y \le 10$

or

$-\infty \le x \le 9$
$1 \le y \le 11$
$-\infty \le x - y \le 8$
$-\infty \le x + y \le 10$

or ...

$x \le 10$
$1 \le y \le 11$
$x - y \le -1$
$x + y \le 21$

$(x = x\ -y)^{\#}$

$$O'(\theta) = \begin{array}{l} f_1(\theta) \le x \le f_2(\theta) \\ f_3(\theta) \le y \le f_4(\theta) \\ f_5(\theta) \le x - y \le f_6(\theta) \\ f_7(\theta) \le x + y \le f_8(\theta) \end{array}$$

Parameter space $\Theta = \{A\theta \le b\}$

Various $\theta \in \Theta$ provide various sound outputs!

$O'(\theta_1)$
$O'(\theta_2)$
$O'(\theta_3)$
$O'(\theta_4)$
...

Tasks can adapt and pick the outputs that suit them!

Key Idea 1 – Sound Space of Outputs

How to construct this sound space?

$$\begin{array}{l} x \leq 10 \\ 1 \leq y \leq 11 \\ x - y \leq -1 \\ x + y \leq 21 \end{array} \quad \xrightarrow{(x = x\ -y)^{\#}} \quad \begin{array}{l} ? \leq x \leq ? \\ ? \leq y \leq ? \\ ? \leq x - y \leq ? \\ ? \leq x + y \leq ? \end{array}$$

$$c^{\#} = \max\ x\ - 2y\ \ s.t. \begin{array}{l} x \leq 10 \\ 1 \leq y \leq 11 \\ x - y \leq -1 \\ x + y \leq 21 \end{array} = -2$$

$x\ - y \leq c^{\#}$ is most precise.
But **any bound more than $c^{\#}$ is sound**!

Example: $x_{old} \in [-\infty, 10]\ y_{old} \in [1, 11]$

$$x_{new}\ - y_{new} = x_{old}\ - 2 * y_{old} = [-\infty, 8]$$

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$?

$$c^{\#} = \max\ x\ -2y\quad s.t.\ x \in [-\infty, 10], y \in [1, 11], x - y \leq -1, x + y \leq 21$$

Duality!

Primal Problem: $\mathrm{p}^* = \max_{\mathrm{x}} f(x) \;\; s.t \;\; Ax \leq b$

Lagrangian: $L(x, \lambda) = f(x) \; - \lambda(Ax \; - b)$

Lagrangian Dual: $\max_{\mathrm{x}} L(x, \lambda) = \max_{\mathrm{x}} \left(f(x) \; - \lambda(Ax \; - b)\right) = g(\lambda)$

Weak Duality Theorem: $\forall \lambda \geq 0, g(\lambda) \geq p^*$

Can be used to capture the space of bounds more than p^* !

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

$$c^{\#} = \max\ x\ - 2y\ \ s.t.\ x \in [-\infty, 10], y \in [1, 11], x - y \leq -1, x + y \leq 21$$

$$L(x, \lambda) = x\ - 2y + \lambda_1 * (y\ - x\ - 1) + \lambda_2\ * (21\ - x\ - y)$$

Lagrangian Function!

$$g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10], y \in [1, 11]} [x\ * (1\ - \lambda_1\ - \lambda_2) + y\ * (-2 + \lambda_1\ - \lambda_2) + (21 * \lambda_2\ - \lambda_1)]$$

By Weak Duality Theorem:

$$\forall\ \lambda_1, \lambda_2 \geq 0, g(\lambda_1, \lambda_2) \geq c^{\#}$$

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

By Weak Duality Theorem: $\forall\, \lambda_1, \lambda_2 \geq 0, g(\lambda_1, \lambda_2) \geq c^{\#}$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10], y \in [1,11]} [x\ *(1\ -\lambda_1\ -\lambda_2) + y\ *(-2 + \lambda_1\ -\lambda_2) + (21 * \lambda_2\ -\lambda_1)]$$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10]} [x\ *(1\ -\lambda_1\ -\lambda_2)] + \max_{y \in [1,11]} [y\ *(-2 + \lambda_1\ -\lambda_2)] + (21 * \lambda_2\ -\lambda_1)]$$

If $(1\ -\lambda_1\ -\lambda_2) < 0,$ then
this goes to ∞.
Sound, but trivial!

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

By Weak Duality Theorem: $\forall\, \lambda_1, \lambda_2 \geq 0, g(\lambda_1, \lambda_2) \geq c^{\#}$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10], y \in [1,11]} [x\ *(1\ -\lambda_1\ -\lambda_2) + y\ *(-2 + \lambda_1\ -\lambda_2) + (21 * \lambda_2\ -\lambda_1)]$$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10]} [x\ *(1\ -\lambda_1\ -\lambda_2)] + \max_{y \in [1,11]} [y\ *(-2 + \lambda_1\ -\lambda_2)] + (21 * \lambda_2\ -\lambda_1)]$$

Constraint the space by adding $(1\ -\lambda_1\ -\lambda_2) \geq 0$

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

By Weak Duality Theorem: $\forall\, \lambda_1, \lambda_2 \geq 0, g(\lambda_1, \lambda_2) \geq c^{\#}$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10], y \in [1,11]} [x\ *(1\ -\lambda_1\ -\lambda_2) + y\ *(-2 + \lambda_1\ -\lambda_2) + (21 * \lambda_2\ -\lambda_1)]$$

$$\{\lambda_1, \lambda_2 \geq 0,\quad g(\lambda_1, \lambda_2) = [10\ *(1\ -\lambda_1\ -\lambda_2)] + \max_{y \in [1,11]} [y\ *(-2 + \lambda_1\ -\lambda_2)] + (21 * \lambda_2\ -\lambda_1)]$$
$$\lambda_1 + \lambda_2 \leq 1\},$$

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

By Weak Duality Theorem: $\forall\, \lambda_1, \lambda_2 \geq 0, g(\lambda_1, \lambda_2) \geq c^{\#}$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10], y \in [1,11]} [x\ *(1\ -\lambda_1\ -\lambda_2) + y\ *(-2+\lambda_1\ -\lambda_2) + (21 * \lambda_2\ -\lambda_1)]$$

$$\{\lambda_1, \lambda_2 \geq 0,\quad g(\lambda_1, \lambda_2) = [10\ *(1\ -\lambda_1\ -\lambda_2)] + \max_{y \in [1,11]} [y\ *(-2+\lambda_1\ -\lambda_2)] + (21 * \lambda_2\ -\lambda_1)]$$
$$\lambda_1 + \lambda_2 \leq 1\},$$

$$\begin{cases} 11(-2+\lambda_1-\lambda_2) & \text{if } -2+\lambda_1-\lambda_2 > 0 \\ 1(-2+\lambda_1-\lambda_2) & \text{if } -2+\lambda_1-\lambda_2 \leq 0 \end{cases}$$

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

By Weak Duality Theorem: $\forall\, \lambda_1, \lambda_2 \geq 0, g(\lambda_1, \lambda_2) \geq c^{\#}$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10], y \in [1,11]} [x\ *(1\ -\lambda_1\ -\lambda_2) + y\ *(-2+\lambda_1\ -\lambda_2) + (21 * \lambda_2\ -\lambda_1)]$$

$$\{\lambda_1, \lambda_2 \geq 0,\quad g(\lambda_1, \lambda_2) = [10\ *(1\ -\lambda_1\ -\lambda_2)] + \max_{y \in [1,11]} [y\ *(-2+\lambda_1\ -\lambda_2)] + (21 * \lambda_2\ -\lambda_1)]$$
$$\lambda_1 + \lambda_2 \leq 1\},$$

$$\begin{cases} 11(-2+\lambda_1-\lambda_2) & \text{if } -2+\lambda_1-\lambda_2 > 0 \\ 1(-2+\lambda_1-\lambda_2) & \text{if } -2+\lambda_1-\lambda_2 \leq 0 \end{cases}$$

$$\max_{\{y \in [l,u]\}} y \cdot c = \left(\frac{l+u}{2}\right) * c + \left(\frac{u\ -l}{2}\right)\ * |c|$$

$$6(-2+\lambda_1-\lambda_2) + 5\,|-2+\lambda_1-\lambda_2|$$

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

By Weak Duality Theorem: $\forall\, \lambda_1, \lambda_2 \geq 0, g(\lambda_1, \lambda_2) \geq c^{\#}$

$$\{\lambda_1, \lambda_2 \geq 0\},\ g(\lambda_1, \lambda_2) = \max_{x \in [-\infty, 10], y \in [1,11]} [x\ *(1\ -\lambda_1\ -\lambda_2) + y\ *(-2 + \lambda_1\ -\lambda_2) + (21 * \lambda_2\ -\lambda_1)]$$

$$\{\lambda_1, \lambda_2 \geq 0,\quad g(\lambda_1, \lambda_2) = [10\ *(1\ -\lambda_1\ -\lambda_2)] + [-12 + 6\lambda_1\ -6\lambda_2 + 5|-2 + \lambda_1 - \lambda_2|] + (21 * \lambda_2\ -\lambda_1)]$$
$$\lambda_1 + \lambda_2 \leq 1\},$$

Key Idea 1 – Sound Space of Outputs

How to capture a space of bounds more than (or equal to) $c^{\#}$? **Duality!**

$$\Theta = \{\lambda_1, \lambda_2 \geq 0,\ \lambda_1 + \lambda_2 \leq 1\} \qquad g(\lambda_1, \lambda_2) = -2 - 5\lambda_1 + 5\lambda_2 + 5|-2 + \lambda_1 - \lambda_2|$$

$$x \leq 10$$
$$1 \leq y \leq 11$$
$$x - y \leq -1$$
$$x + y \leq 21$$

$$(x = x - y)^{\#}$$

$$? \leq x \leq ?$$
$$? \leq y \leq ?$$
$$? \leq x - y \leq g(\lambda_1, \lambda_2)$$
$$? \leq x + y \leq ?$$

$(\lambda_1, \lambda_2) = (0, 0) \Rightarrow g(\lambda_1, \lambda_2) = 8$ (Interval Analysis)

$(\lambda_1, \lambda_2) = (0.4, 0) \Rightarrow g(\lambda_1, \lambda_2) = 4$

$(\lambda_1, \lambda_2) = (0, 0.4) \Rightarrow g(\lambda_1, \lambda_2) = 12$

$(\lambda_1, \lambda_2) = (1, 0) \Rightarrow g(\lambda_1, \lambda_2) = -2$ ($c^{\#}$ Most Precise)

$(\lambda_1, \lambda_2) = (0, 1) \Rightarrow g(\lambda_1, \lambda_2) = 18$

Key Idea 1 – Sound Space of Outputs

The space of outputs is sound.

THEOREM 4.1 (SOUNDNESS OF THE CONSTRUCTED PARAMETRIC SPACE). *Let $\mathcal{S}_{\mathcal{T}}(\overline{\mathcal{M}})$ be the parametric space of elements constructed by the UPOSE algorithm for an input element a_{in}, QGO operator O and template $\mathcal{T}$, where $\overline{\mathcal{M}} = [\mathcal{M}_1, \ldots, \mathcal{M}_t]$ is the collection of PSMs $\mathcal{M}_i$ obtained by lower bounding effective objectives $\hat{F}_i$. Then, $\mathcal{S}_{\mathcal{T}}(\overline{\mathcal{M}})$ is a sound space of outputs (Def. 4.4) for a_{in} and O.*

PROOF. Assume that $\mathcal{S}_{\mathcal{T}}(\overline{\mathcal{M}})$ is not a sound space of outputs for a_{in} and O. Then, by Definition 4.4,

The space of outputs contains the most precise outputs for linear operators.

THEOREM 4.2 (MOST PRECISE OUTPUT FOR LINEAR CASE). *For QGO operators O with linear $\sigma_{\leq 2}$, the space $\mathcal{S}_{\mathcal{T}}(\overline{\mathcal{M}})$ contains the most precise abstract output. That is, for each PSM $\mathcal{M}_i$, there exists a parameter instantiation that yields the most precise bound $c_i^{\#}$.*

PROOF. This follows by Theorem C.2 in Appendix, which uses Strong Duality [7] to show that, for

Output with $\theta = 0$ is the interval relaxation output.

Interval Relaxation at $\vec{0}$: The abstract element $\mathcal{S}_{\mathcal{T}}(\overline{\mathcal{M}}, \vec{0})$, obtained by setting $\boldsymbol{\theta} = \vec{0}$, corresponds to the output of the *interval relaxation* heuristic used commonly in existing libraries, which computes bounds by simply discarding inter-variable constraints and only using the interval bounds of the variables (By Theorem C.2 proved in Appendix).

How to traverse the space of outputs?

$$x \leq 10$$
$$1 \leq y \leq 11$$
$$x - y \leq -1$$
$$x + y \leq 21$$

$$(x = x \ - y)^{\#}$$

$$O'(\theta) = \begin{matrix} f_1(\theta) \leq x \leq f_2(\theta) \\ f_3(\theta) \leq y \leq f_4(\theta) \\ f_5(\theta) \leq x - y \leq f_6(\theta) \\ f_7(\theta) \leq x + y \leq f_8(\theta) \end{matrix}$$

Parameter space $\Theta = \{\boldsymbol{A}\theta \leq b\}$

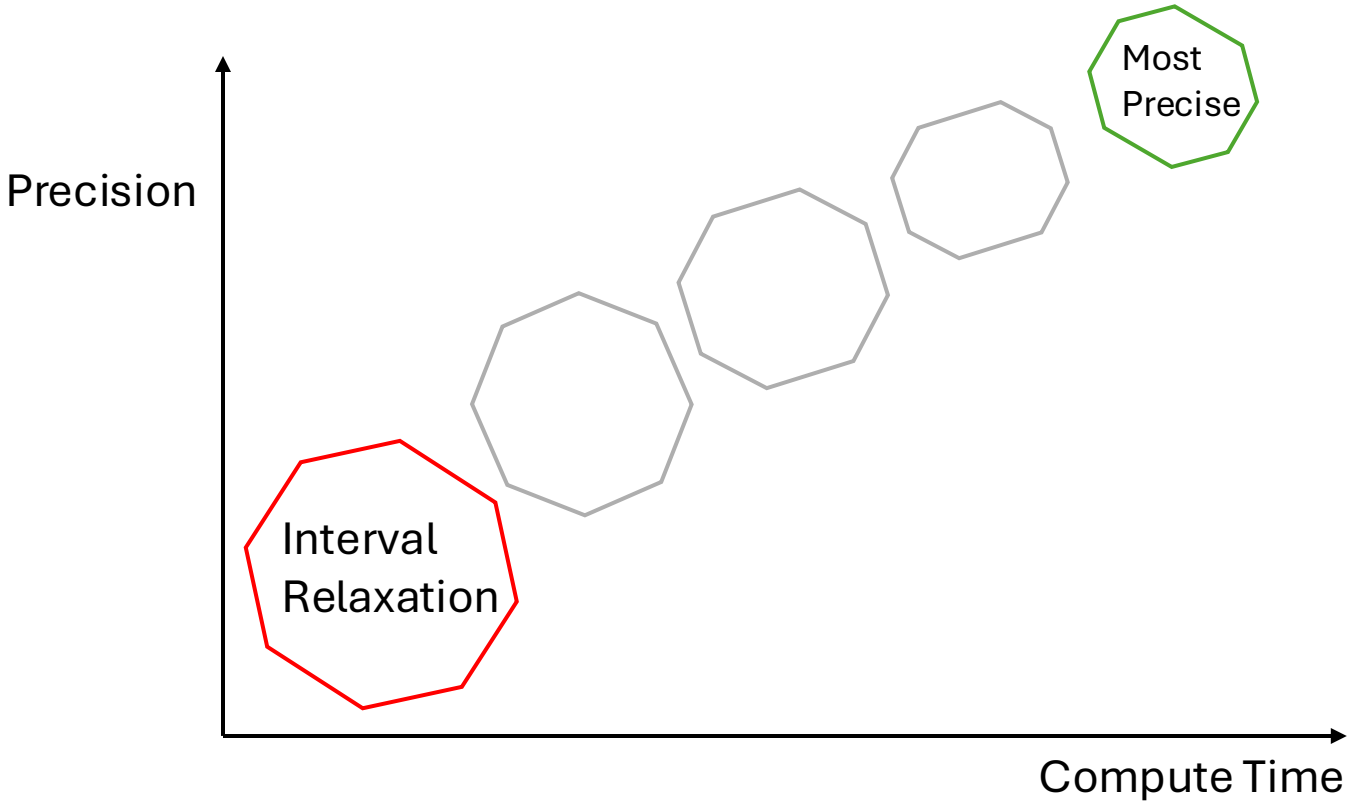
- For analysis to proceed, we need a concrete output (choose some $\theta^* \in \Theta$ to initialize)!
- Ideally, find the most precise in the space of outputs but not trivial.
- For allowing precision-efficiency adaptability, an ideal search procedure should:
  - Allow downstream tasks to specify some runtime budget R.
  - Should ensure that more budget implies more precision.

Key Idea 2 – Gradient-Guided Adaptable Search

- Use gradient descent to make the bounds precise!
- Analysis can specify the number of gradient steps R to perform.
- More gradient steps allow the analysis to infer more precise bounds.
- Traversal is started at $\theta = 0$ to always be as precise as interval relaxation.

Key Idea 2 – Gradient-Guided Adaptable Search

$\Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$ $g(\lambda_1, \lambda_2) = -2 - 5\lambda_1 + 5\lambda_2 + 5|-2 + \lambda_1 - \lambda_2|$ $x - y \leq g(\lambda_1, \lambda_2)$



$(\lambda_1, \lambda_2) = (0, 0) \Rightarrow x - y \leq 8$

⇩ Gradient Step

$(\lambda_1, \lambda_2) = (0.3, 0) \Rightarrow x - y \leq 5$

⇩ Gradient Step

$(\lambda_1, \lambda_2) = (0.6, 0) \Rightarrow x - y \leq 2$

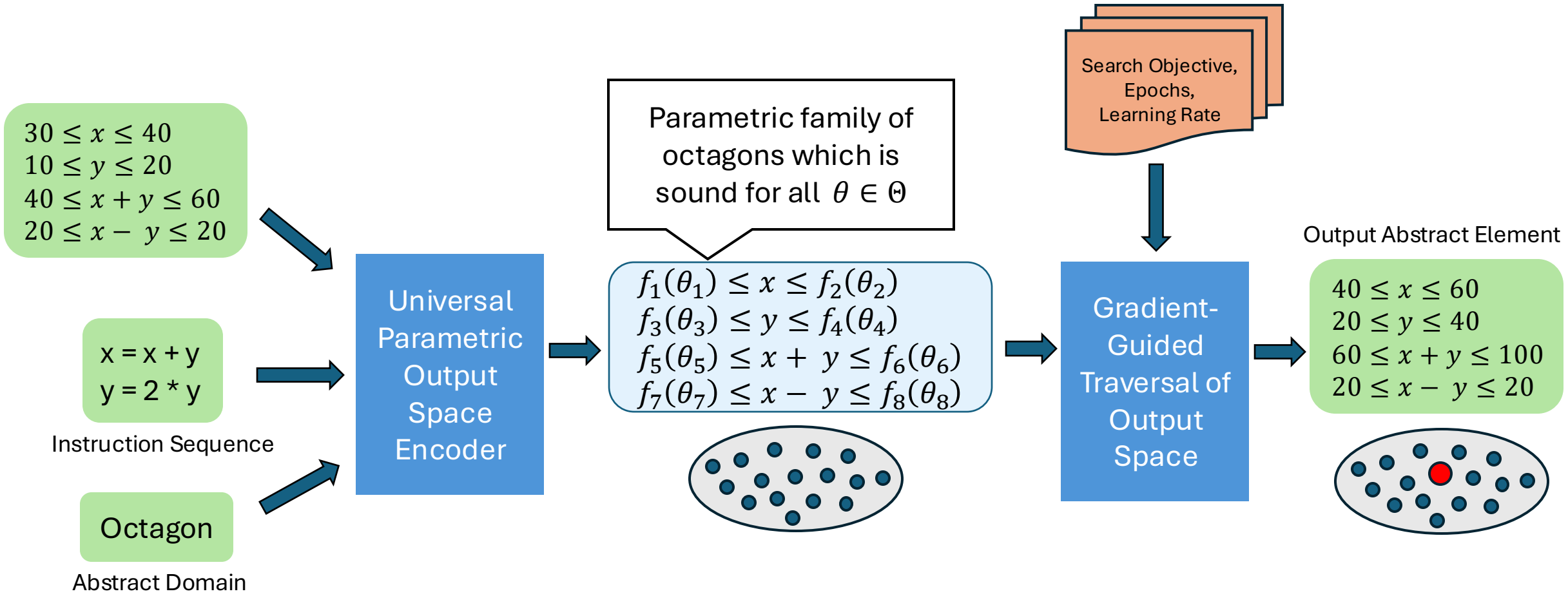
⇩ Gradient Step

$(\lambda_1, \lambda_2) = (0.9, 0) \Rightarrow x - y \leq -1$

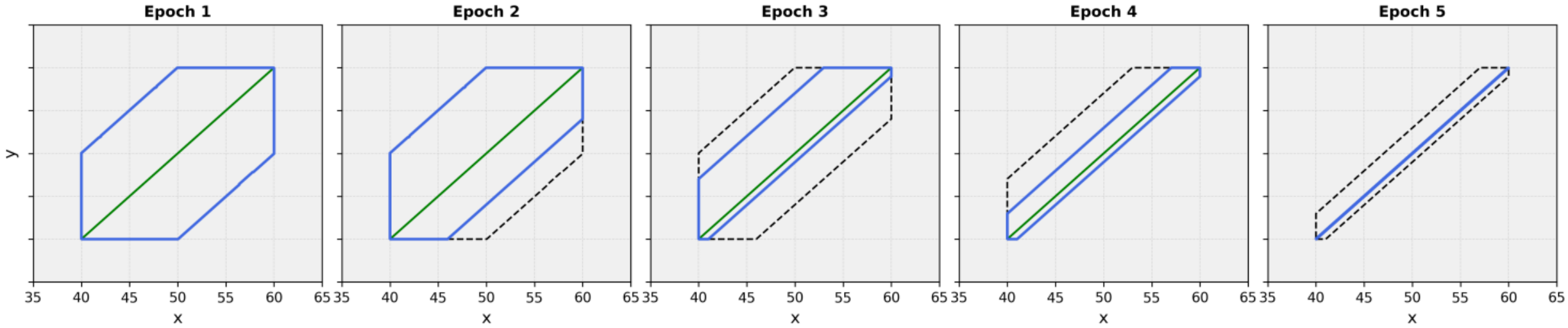
⇩ Gradient Step

$(\lambda_1, \lambda_2) = (1, 0) \Rightarrow x - y \leq -2$

AbsEvolve Overview

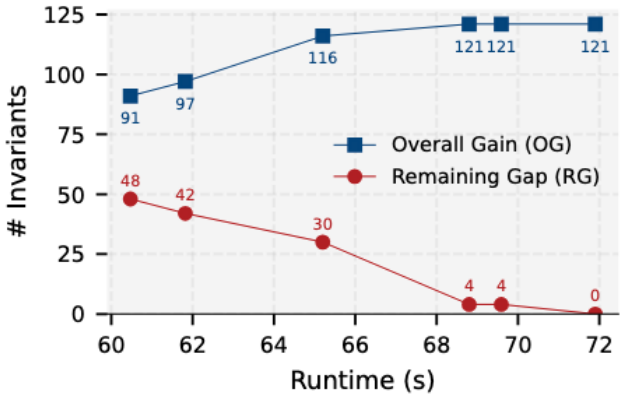


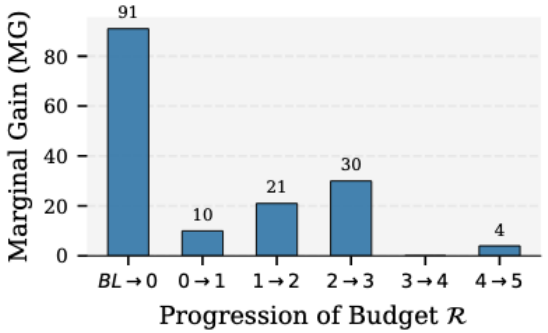
AbsEvolve In Action



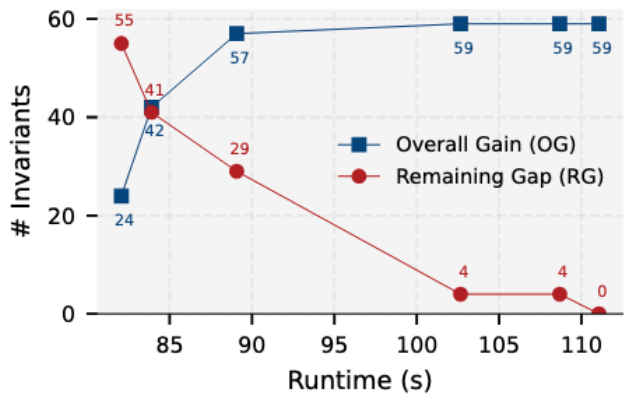
Evaluation

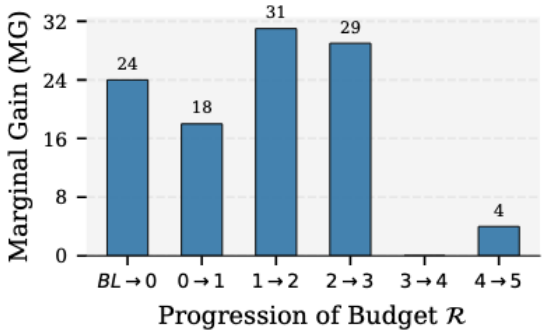
- NLA-Digbench benchmarks:
  55 programs, and 237 invariants across all programs.
- AbsEvolve run with increasing number of gradient steps R
- Overall Gain (OG): # of invariants strengthened over those computed by state-of-the-art library ELINA.
- Remaining Gap (RG): # of invariants still weaker than those computed by most precise LP solver baseline (takes 230s and 350s for Zones and Octagons).
- Marginal Gain (MG): # of invariants strengthened compared to the last setting of R.





(a) Zones

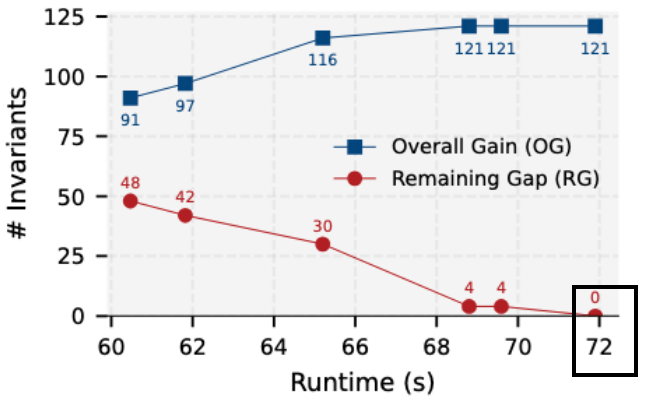


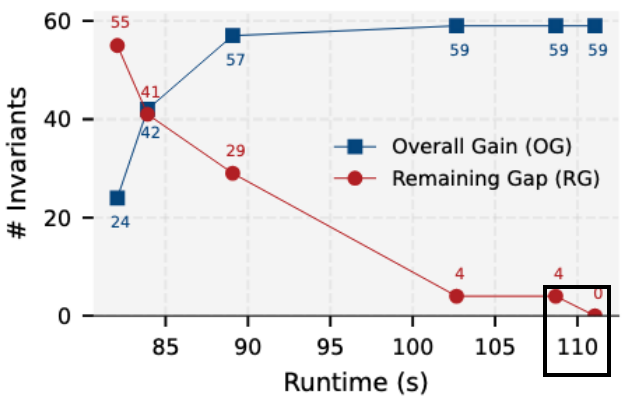


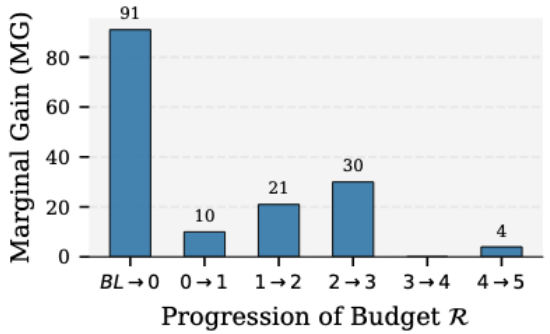
(b) Octagons

Evaluation

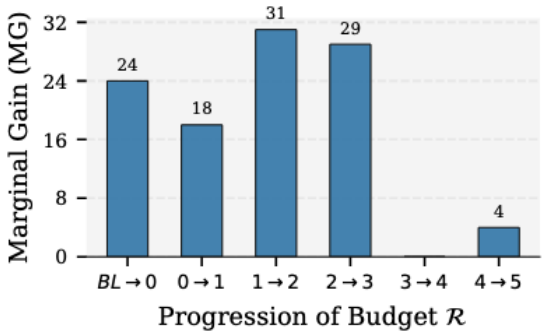
- NLA-Digbench benchmarks:
  55 programs, and 237 invariants across all programs.
- AbsEvolve run with increasing number of gradient steps R
- Overall Gain (OG): # of invariants strengthened over those computed by state-of-the-art library ELINA.
- Remaining Gap (RG): # of invariants still weaker than those computed by most precise LP solver baseline (takes **230s** and **350s** for Zones and Octagons).
- Marginal Gain (MG): # of invariants strengthened compared to the last setting of R.







(a) Zones



(b) Octagons

About 3.2x faster convergence to most-precise!

Evaluation

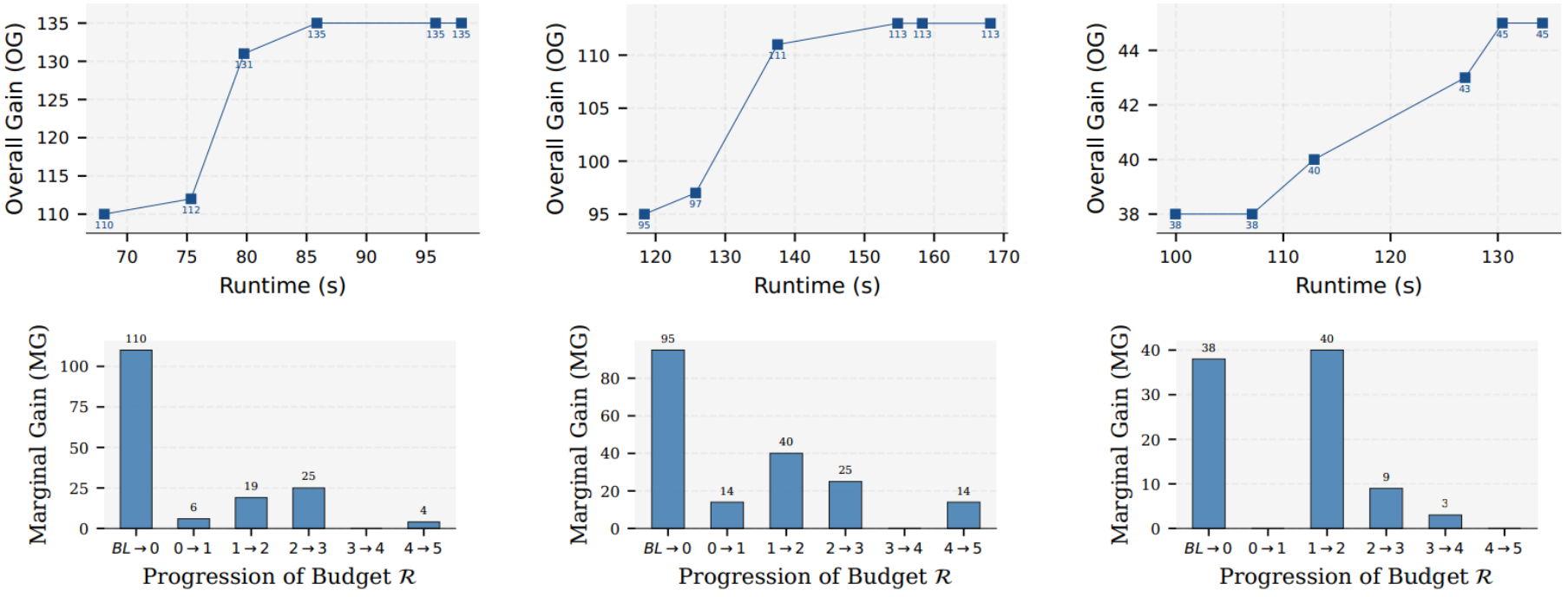
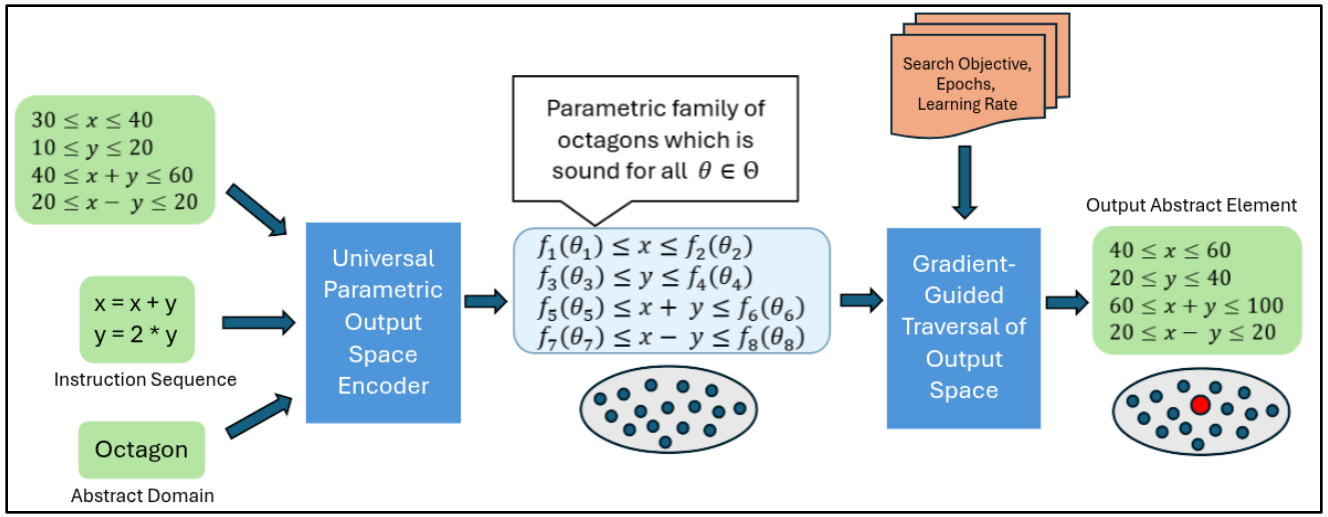
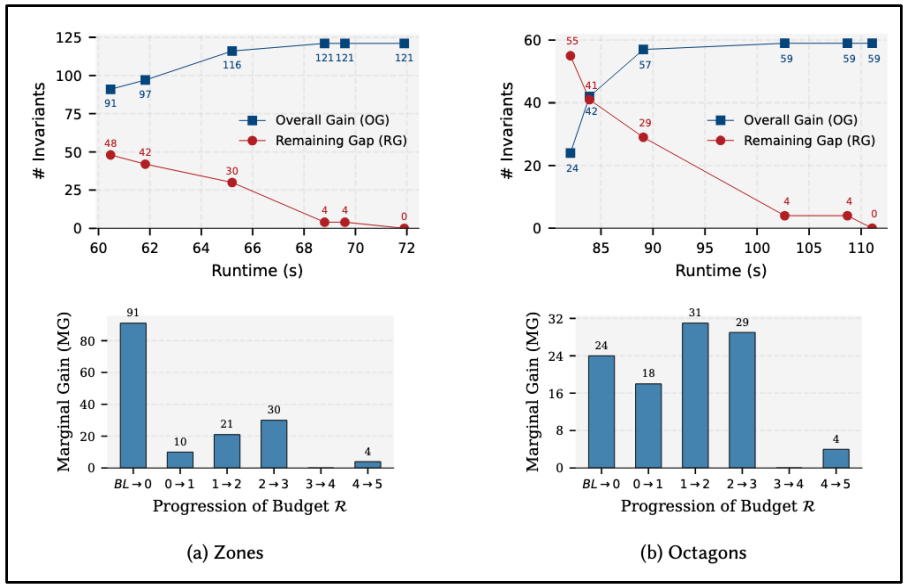


Fig. 7. Evolution of invariant strengthening while handling quadratic assignments across domains.

Future Possibilities

- Can enable GPU-driven large-scale program analysis.
- Other analysis tasks like finding an octagon within a target one
  - $x - y \leq g(\lambda_1, \lambda_2) \subseteq x - y \leq 10$ => $\text{Descent on } \max(10 - g(\lambda_1, \lambda_2), 0)$
- Directly use a NN or LLM to learn/predict $\theta \in \Theta$ for various tasks.
  - Soundness guarantee always there!
- While program verification, if property unproved, choices of θ can be refined!





Thanks for the attention!

Questions?

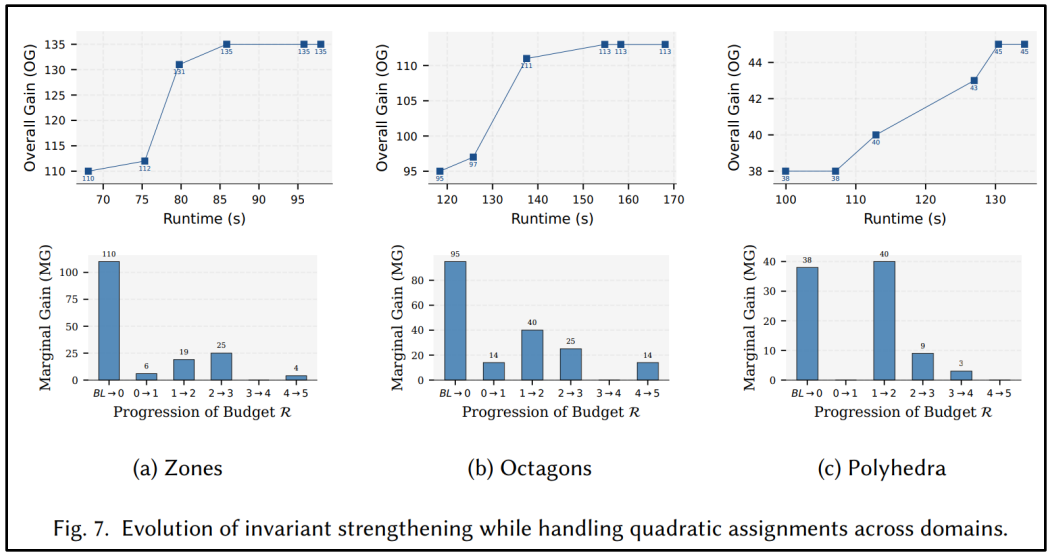


Fig. 7. Evolution of invariant strengthening while handling quadratic assignments across domains.