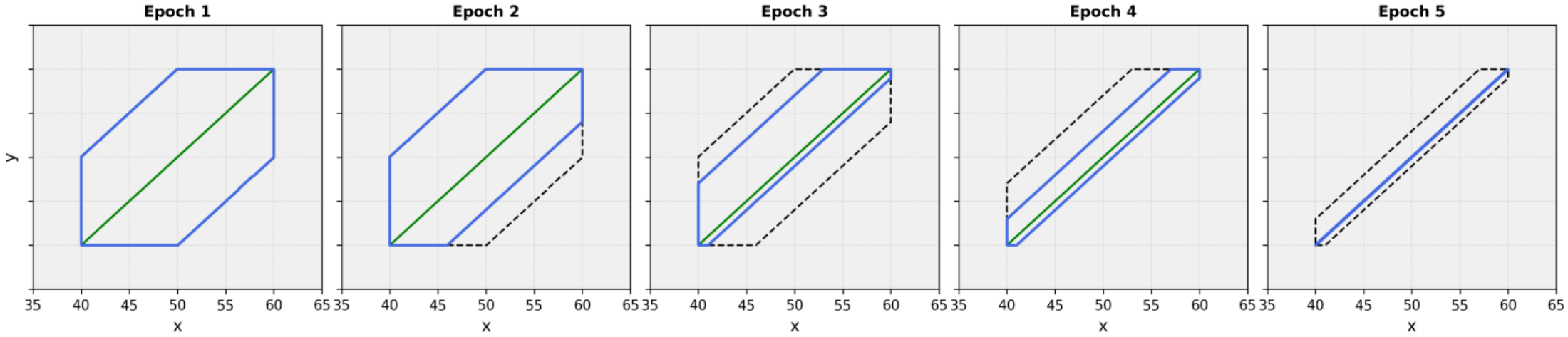
Evolving Abstract Transformers for Gradient-Guided, Adaptable Abstract Interpretation

Shaurya Gomber, Debangshu Banerjee, Gagandeep Singh
University of Illinois Urbana-Champaign



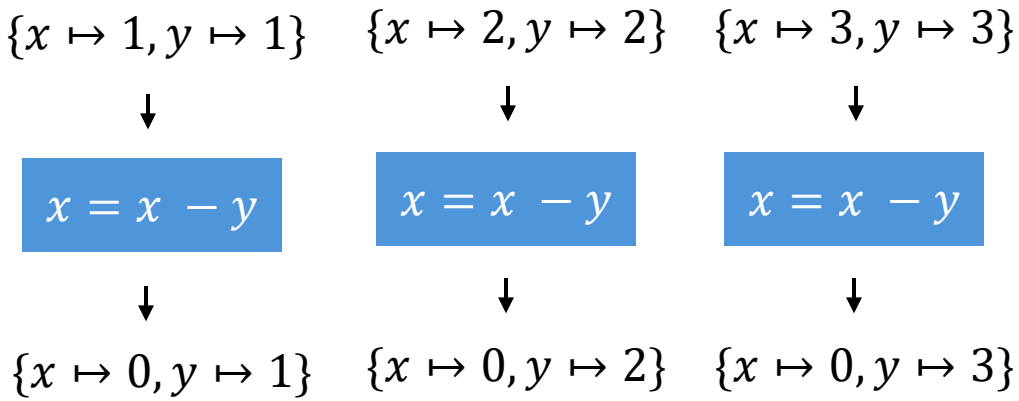
Abstract Interpretation

$x = x - y$

...

...

Concrete Executions



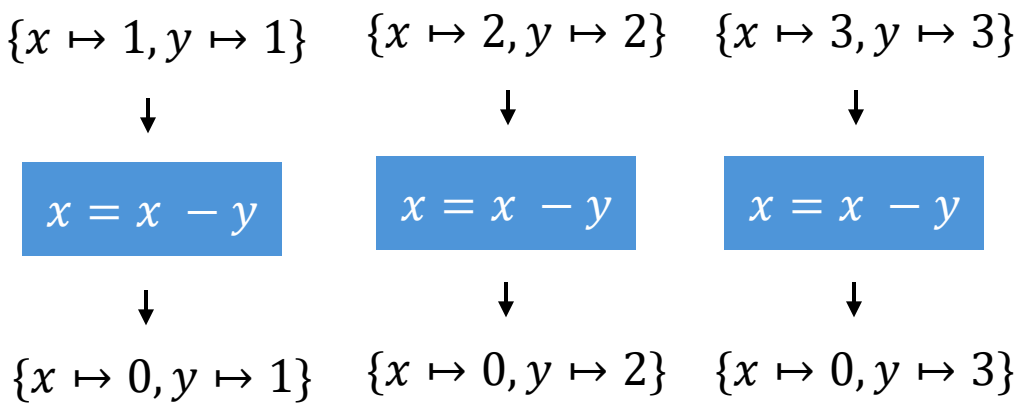
Abstract Interpretation

$x = x - y$

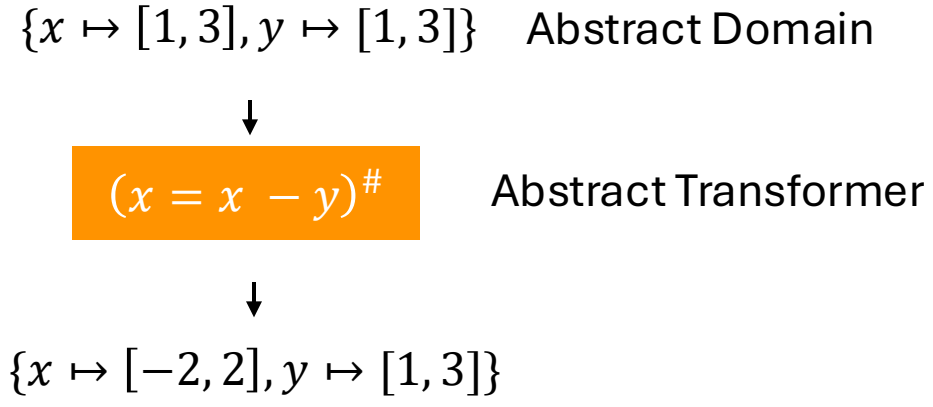
...

...

Concrete Executions



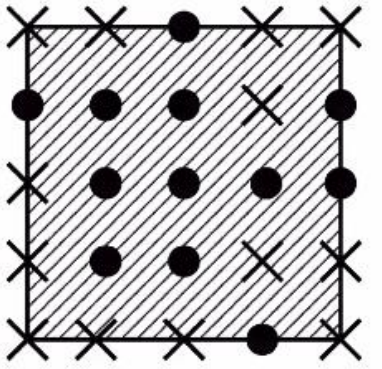
Abstract Interpretation



Abstract Domains

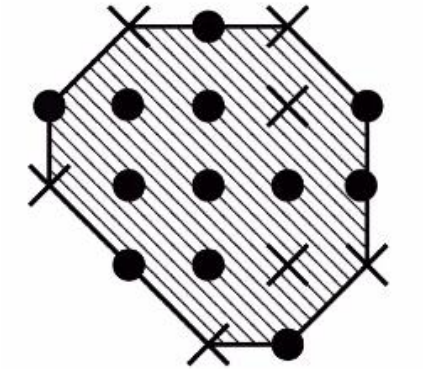
Intervals

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



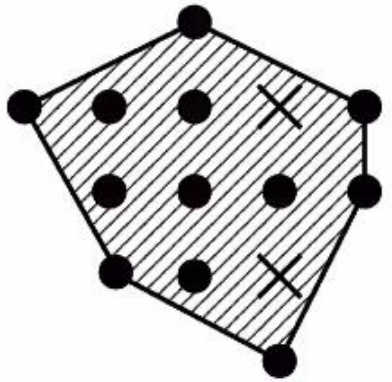
Octagons

$$\begin{aligned} \pm x \pm y &\leq c_i \\ \pm x &\leq d_i \\ \pm y &\leq e_i \end{aligned}$$



Polyhedra

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



Expressivity increases. So does transformer complexity.

References:
A. Mine. 2001. The octagon abstract domain. In Proceedings Eighth Working Conference on Reverse Engineering
Patrick Cousot and Nicolas Halbwachs. 1978. Automatic discovery of linear restraints among variables of a program, POPL'78

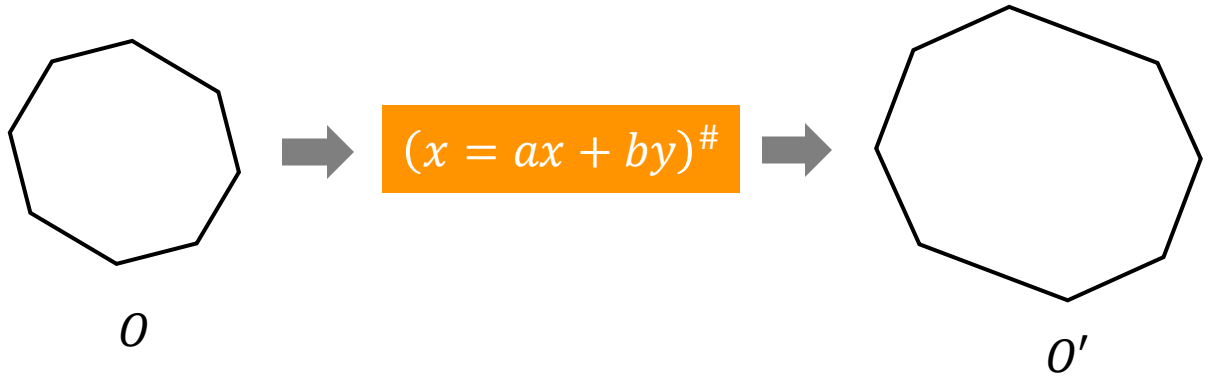
Abstract Transformers

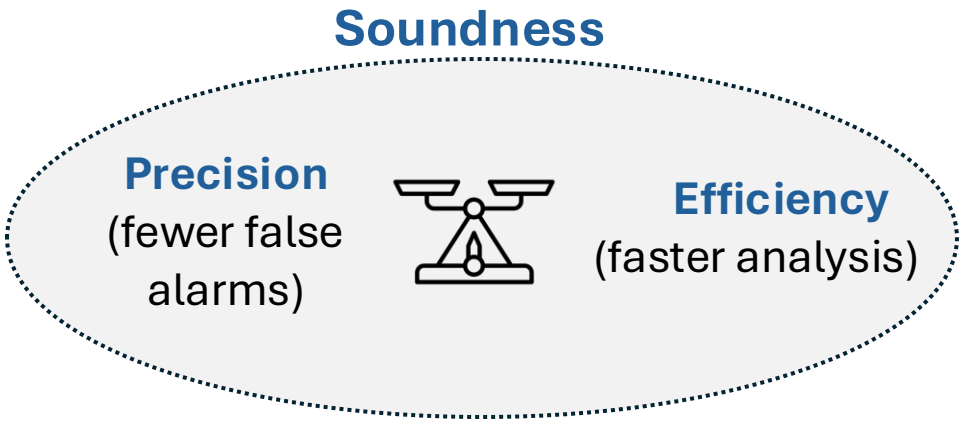
1. **Soundness:** Output should cover all possible new states.

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

2. **Precision:** Measure of how "small" the sound outputs are.

3. **Efficiency:** Computation time of the transformer.





Limitations of Current Transformers Design

Per-Domain, Per-Instruction

Instruction-Wise Reasoning

Fixed Precision-Efficiency Tradeoff

Limitations of Current Transformers Design

Per-Domain, Per-Instruction

Instruction-Wise Reasoning

Fixed Precision-Efficiency Tradeoff

Limitation 1: Per-Domain, Per-Instruction

Abstract Domains

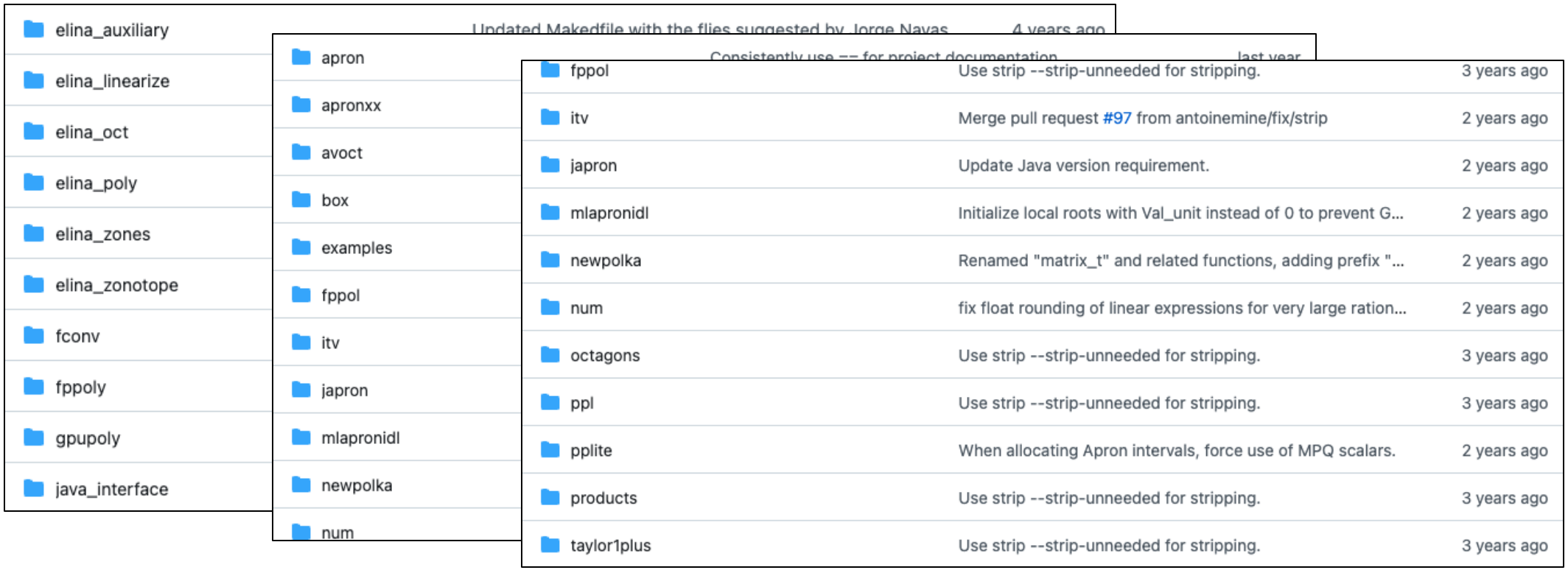
Interval
Zones
Octagons
Octahedron
Polyhedra
…

Concrete Instructions

Assignments
Meet with guards
Guarded assignments
…

Transformer needs to be implemented for each (domain, instruction) pair!

Limitation 1: Per-Domain, Per-Instruction



Thousands of lines of complex implementation!

Limitations of Current Transformers Design

Per-Domain, Per-Instruction

Instruction-Wise Reasoning

Fixed Precision-Efficiency Tradeoff

Limitation 2: Instruction-Wise Reasoning

- Transformers are implemented for each instruction and then composed.

$$x = x + y$$
$$y = 2 * y$$

$$O \rightarrow (x = x + y)^{\#} \rightarrow O' \rightarrow (y = 2 * y)^{\#} \rightarrow O''$$

Precision is non-compositional!

- $x - y$ is preserved across the block, but not captured instruction-wise.

- Need sequence-aware transformers. More complex and many possible sequences!

$$O \rightarrow \begin{pmatrix} x = x + y; \\ y = 2 * y; \end{pmatrix}^{\#} \rightarrow O'$$

Limitations of Current Transformers Design

Per-Domain, Per-Instruction

Instruction-Wise Reasoning

Fixed Precision-Efficiency Tradeoff

Limitation 3: Fixed Precision-Efficiency Tradeoff

```
int x = 0, y = 0;

while (y <= 10)
{
   x = x + y;
   y = y + 1;
   x = x - y;
}

assert(x <= 0);
```

Limitation 3: Fixed Precision-Efficiency Tradeoff

```
int x = 0, y = 0;

while (y <= 10)
{
   x = x + y;
   y = y + 1;
   x = x - y;
}

assert(x <= 0);
```

Initial value of x is 0

Limitation 3: Fixed Precision-Efficiency Tradeoff

```
int x = 0, y = 0;

while (y <= 10)
{
   x = x + y;
   y = y + 1;
   x = x - y;
}

assert(x <= 0);
```

Initial value of x is 0

$x' = x + y - (y + 1) = x - 1$

(x decreases by one in each iteration)

Limitation 3: Fixed Precision-Efficiency Tradeoff

```
int x = 0, y = 0;

while (y <= 10)
{
   x = x + y;
   y = y + 1;
   x = x - y;
}

assert(x <= 0);
```

Initial value of x is 0

$x' = x + y - (y + 1) = x - 1$

(x decreases by one in each iteration)

$x \leq 0$ is a valid loop invariant!

Limitation 3: Fixed Precision-Efficiency Tradeoff

```
int x = 0, y = 0;

while (y <= 10)
{
    x = x + y;
    y = y + 1;
    x = x - y;
}

assert(x <= 0);
```

Initial value of x is 0

$x' = x + y - (y + 1) = x - 1$

(x decreases by one in each iteration)

$x \leq 0$ is a valid loop invariant!

Octagon analysis with state-of-the-art libraries fail to prove this!

Limitation 3: Fixed Precision-Efficiency Tradeoff

Computing the most precise outputs is expensive!

$$\begin{array}{l} -\infty \le \quad x \quad \le 10 \\ 1 \quad \le \quad y \quad \le 11 \\ -\infty \le x - y \\ \le -1 \\ -\infty \le x + y \le 21 \end{array} \rightarrow \boxed{(x = x - y)^{\#}} \rightarrow \begin{array}{l} ? \le \quad x \quad \le ? \\ ? \le \quad y \quad \le ? \\ ? \le x - y \le \boxed{?} \\ ? \le x + y \le ? \end{array}$$

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

$$\max\ x - 2y \quad s.t. \quad \begin{array}{l} -\infty \le \quad x \quad \le 10 \\ 1 \quad \le \quad y \quad \le 11 \\ -\infty \le x - y \le -1 \\ -\infty \le x + y \le 21 \end{array} \rightarrow \boxed{\text{LP Solver}} \rightarrow \text{-2}$$

Limitation 3: Fixed Precision-Efficiency Tradeoff

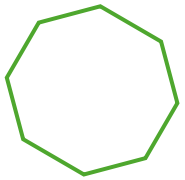
Computing the most precise outputs is expensive!

$$
\begin{aligned}
-\infty \leq\ & x & \leq 10\\
1 \leq\ & y & \leq 11\\
-\infty \leq\ & x - y & \\
\leq\ & -1 & \\
-\infty \leq\ & x + y & \leq 21
\end{aligned}
$$

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

$$
\begin{aligned}
-\infty \leq\ & x & \leq -1\\
1 \leq\ & y & \leq 11\\
-\infty \leq\ & x - y & \leq -2\\
-\infty \leq\ & x + y & \leq 10
\end{aligned}
$$

Proves the assertion!



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

Libraries use cheaper (but less precise) heuristics like Interv

$$
\begin{aligned}
-\infty \leq\ & x & \leq 10\\
1 \leq\ & y & \leq 11\\
-\infty \leq\ & x - y & \\
\leq\ & -1 & \\
-\infty \leq\ & x + y & \leq 21
\end{aligned}
$$

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

$x_{old} \in [-\infty, 10]$

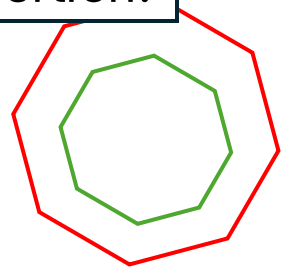
$y_{old} \in [1, 11]$

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

$\Rightarrow$

$$
\begin{aligned}
-\infty \leq\ & x & \leq 9\\
1 \leq\ & y & \leq 11\\
-\infty \leq\ & x - y & \leq 8\\
-\infty \leq\ & x + y & \leq 10
\end{aligned}
$$

Leads to failure in proving the assertion!



Sound but less precise!

Limitation 3: Fixed Precision-Efficiency Tradeoff

Computing the most precise outputs is expensive!

Proves the assertion!

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

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

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

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

Libraries prioritize efficiency over precision!

Libraries use cheaper (but less precise) heuristics like Interv

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

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

$x_{old} \in [-\infty, 10]$

$y_{old} \in [1, 11]$

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

Leads to failure in proving the assertion!

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

Sound but less precise!

Limitation 3: Fixed Precision-Efficiency Tradeoff

Different tasks have different precision-efficiency requirements.

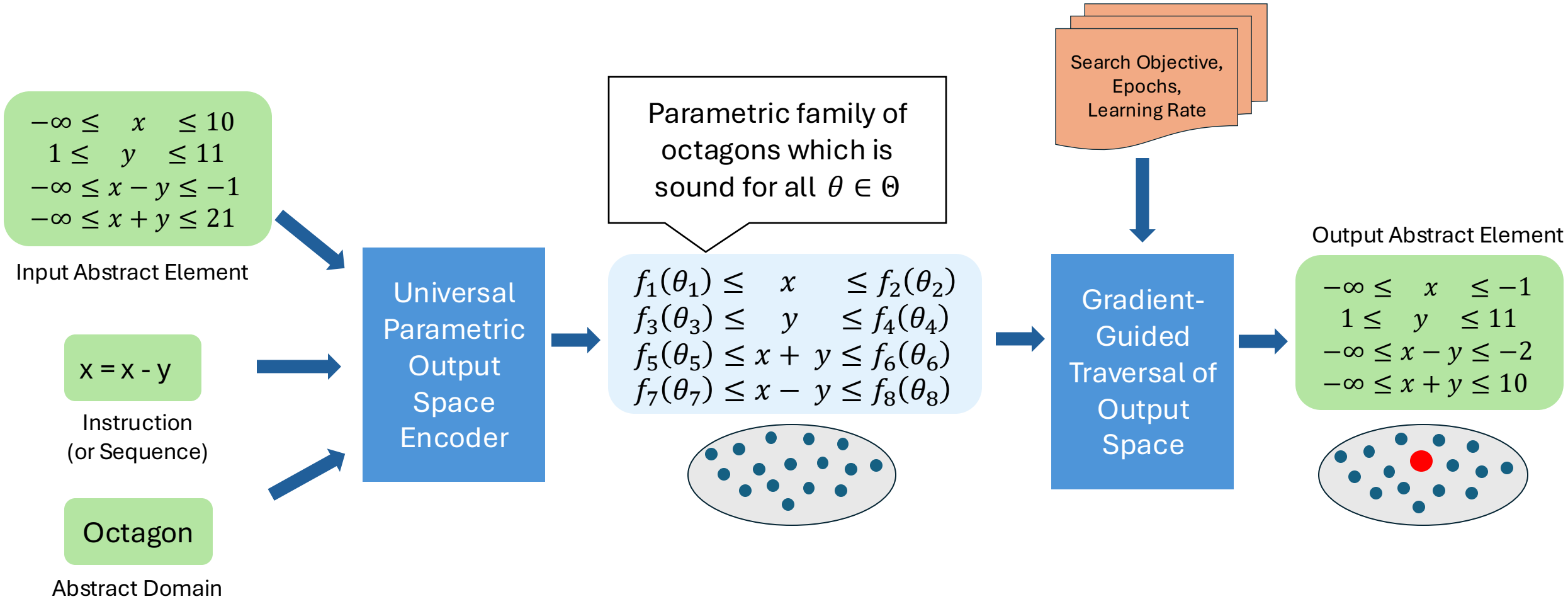
What Do We Need?

Per-Domain, Per-Instruction	Works across domains and instructions
Instruction-Wise Reasoning	Supports sequences
Fixed Precision-Efficiency Tradeoff	Adaptable Precision-Efficiency

Evolving Abstract Transformers

Evolving Abstract Transformers provide a **unified framework** across various domains, instructions, and instruction sequences and work by constructing a **sound space of outputs** and **evolving** within it to allow precision-efficiency adaptability.

Evolving Abstract Transformers



Evolving Abstract Transformers

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

Input Abstract Element

Supports Polyhedral domains ($Ax \leq b$) like Intervals, Octagons, Polyhedra etc.

Octagon

Abstract Domain

Universal
etric
ut
ce
der

Parametric family of octagons which is sound for all $\theta \in \Theta$

$$f_1(\theta_1) \leq x \leq f_2(\theta_2)$$
$$f_3(\theta_3) \leq y \leq f_4(\theta_4)$$
$$f_5(\theta_5) \leq x + y \leq f_6(\theta_6)$$
$$f_7(\theta_7) \leq x - y \leq f_8(\theta_8)$$

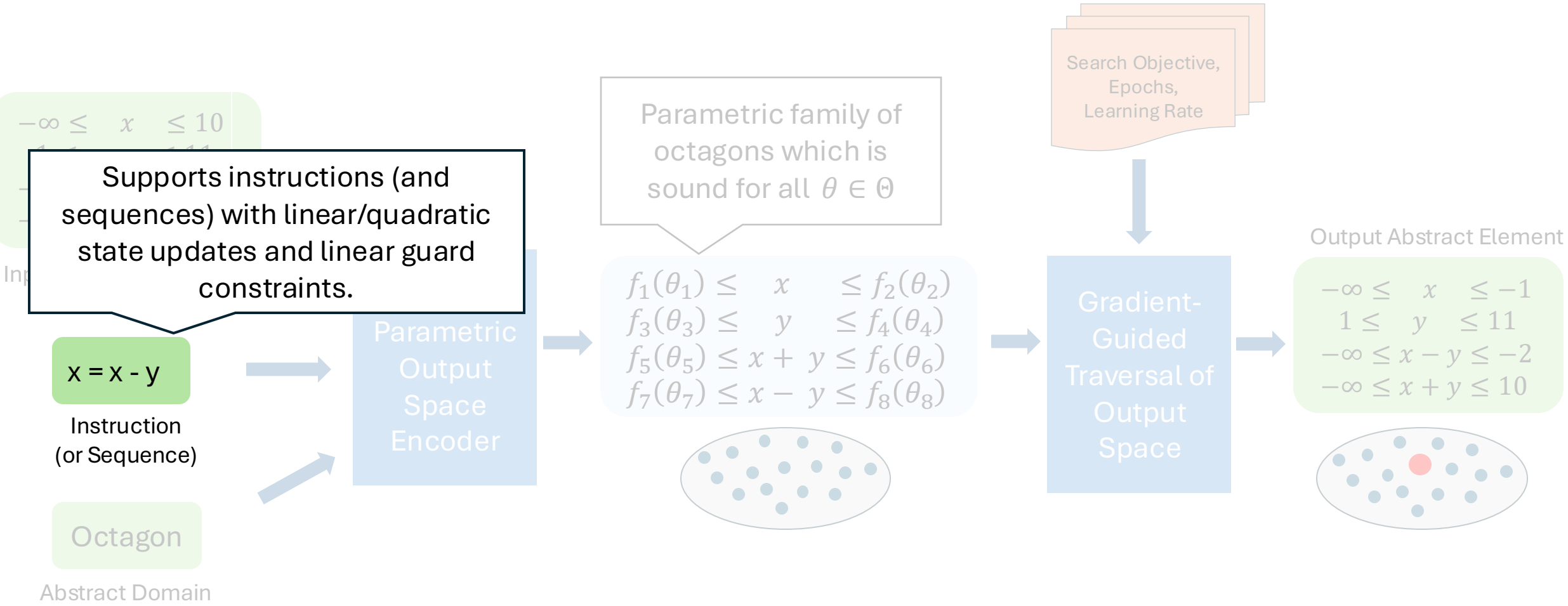
Search Objective, Epochs, Learning Rate

Gradient-Guided Traversal of Output Space

Output Abstract Element

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

Evolving Abstract Transformers



Evolving Abstract Transformers

Per-Domain, Per-Instruction

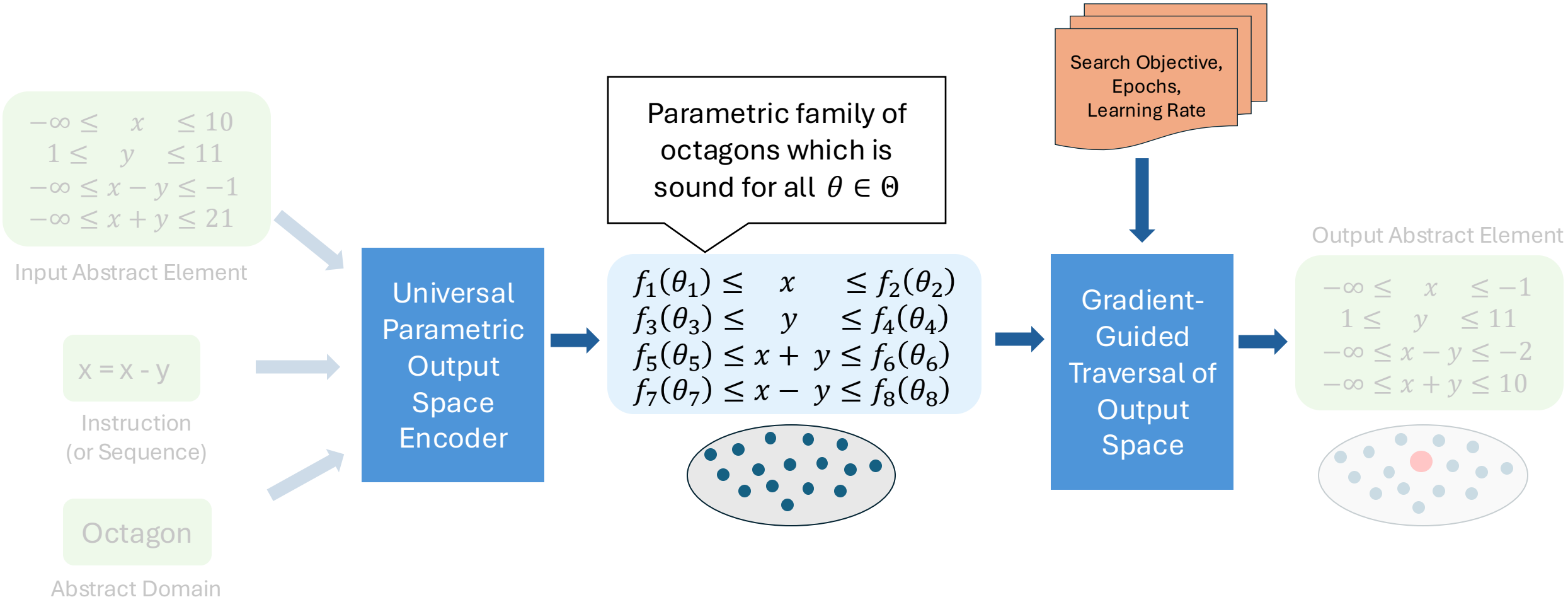
Works across domains and instructions ✅

Instruction-Wise Reasoning

Supports sequences ✅

Fixed Precision-Efficiency Tradeoff

Evolving Abstract Transformers



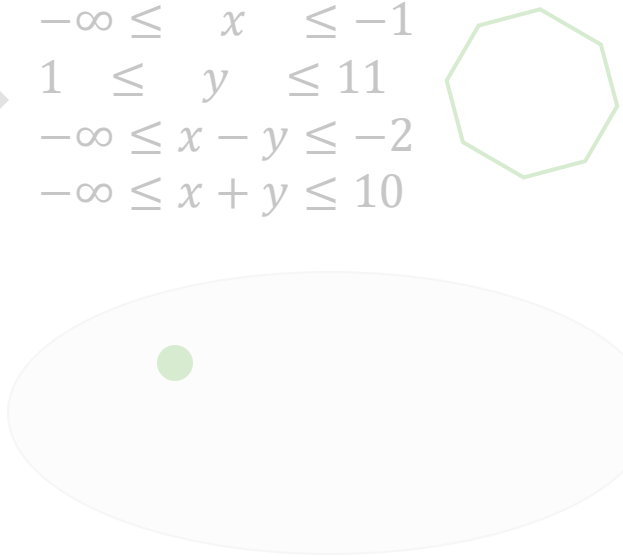
Key Idea 1: Sound Traversable Space of Outputs

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

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

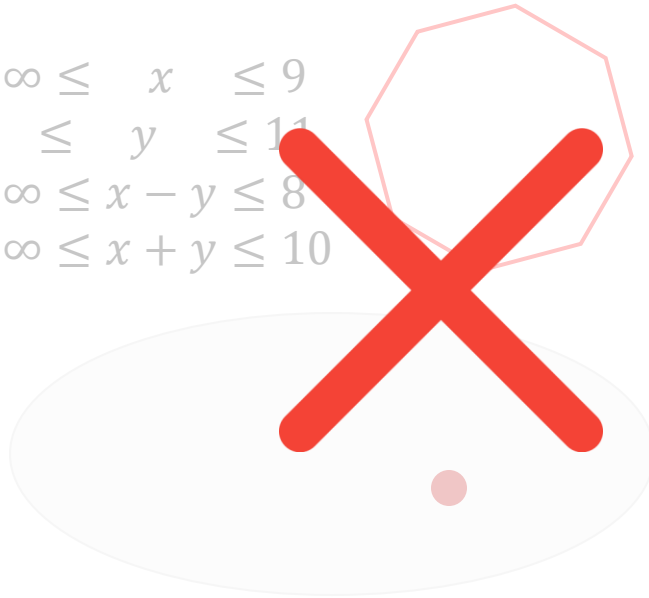
$$(x = x - y)^{\#}$$

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



or

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



Key Idea 1: Sound Traversable Space of Outputs

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

$$
\begin{array}{l}
-\infty \leq \quad x \quad \leq 10 \\
1 \quad \leq \quad y \quad \leq 11 \\
-\infty \leq x - y \\
\leq -1 \\
-\infty \leq x + y \leq 21
\end{array}
\Rightarrow \boxed{(x = x - y)^{\#}} \Rightarrow O'(\theta) = \begin{array}{l}
f_1(\theta) \leq \quad x \quad \leq f_2(\theta) \\
f_3(\theta) \leq \quad y \quad \leq f_4(\theta) \\
f_5(\theta) \leq x - y \leq f_6(\theta) \\
f_7(\theta) \leq x + y \leq f_8(\theta)
\end{array}
$$

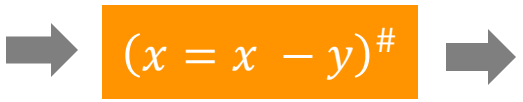
Parameter space $\Theta = \{A\theta \leq b\}$

Tasks can adapt and pick the outputs that suit them!

Key Idea 1: Sound Traversable Space of Outputs

How to capture such a space?

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



$$? \leq x \leq ?$$
$$? \leq y \leq ?$$
$$? \leq x - y \leq \boxed{c}$$
$$? \leq x + y \leq ?$$

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

-2

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

Key Idea 1: Sound Traversable Space of Outputs

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

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

Duality

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.\ \begin{array}{ll} 10\ -x & \geq 0 \\ y\ -1 & \geq 0 \\ -y+11 & \geq 0 \\ y\ -x\ -1 & \geq 0 \\ 21\ -x\ -y & \geq 0 \end{array}$$

Duality

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.\quad \begin{array}{ll} 10\ -x & \geq 0 \leftarrow \boldsymbol{\lambda_1} \\ y\ -1 & \geq 0 \leftarrow \boldsymbol{\lambda_2} \\ -y+11 & \geq 0 \leftarrow \boldsymbol{\lambda_3} \\ y\ -x\ -1 & \geq 0 \leftarrow \boldsymbol{\lambda_4} \\ 21\ -x\ -y & \geq 0 \leftarrow \boldsymbol{\lambda_5} \end{array}$$

Duality

Lagrangian $L(x, y, \lambda) = (x\ -2y) + \boldsymbol{\lambda_1} * (\mathbf{10}\ -\boldsymbol{x}) + \boldsymbol{\lambda_2} * (\boldsymbol{y}\ -\mathbf{1})$ …

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.\quad \begin{array}{ll} 10\ -x & \geq 0 \leftarrow \boldsymbol{\lambda_1} \\ y\ -1 & \geq 0 \leftarrow \boldsymbol{\lambda_2} \\ -y+11 & \geq 0 \leftarrow \boldsymbol{\lambda_3} \\ y\ -x\ -1 & \geq 0 \leftarrow \boldsymbol{\lambda_4} \\ 21\ -x\ -y & \geq 0 \leftarrow \boldsymbol{\lambda_5} \end{array}$$

Duality

Lagrangian $\quad L(x,y,\lambda) = (x\ -2y) + \boldsymbol{\lambda_1} * (\mathbf{10}\ - \boldsymbol{x}) + \boldsymbol{\lambda_2} * (\boldsymbol{y}\ - \mathbf{1})$ …

Lagrangian Dual $\quad g(\lambda) = \max_{\{x,y\}} L(x,y,\lambda)$

Weak Duality: $\quad \forall\ \lambda\ \geq 0, g(\lambda) \geq c^{\#}$

Key Idea 1: Sound Traversable Space of Outputs

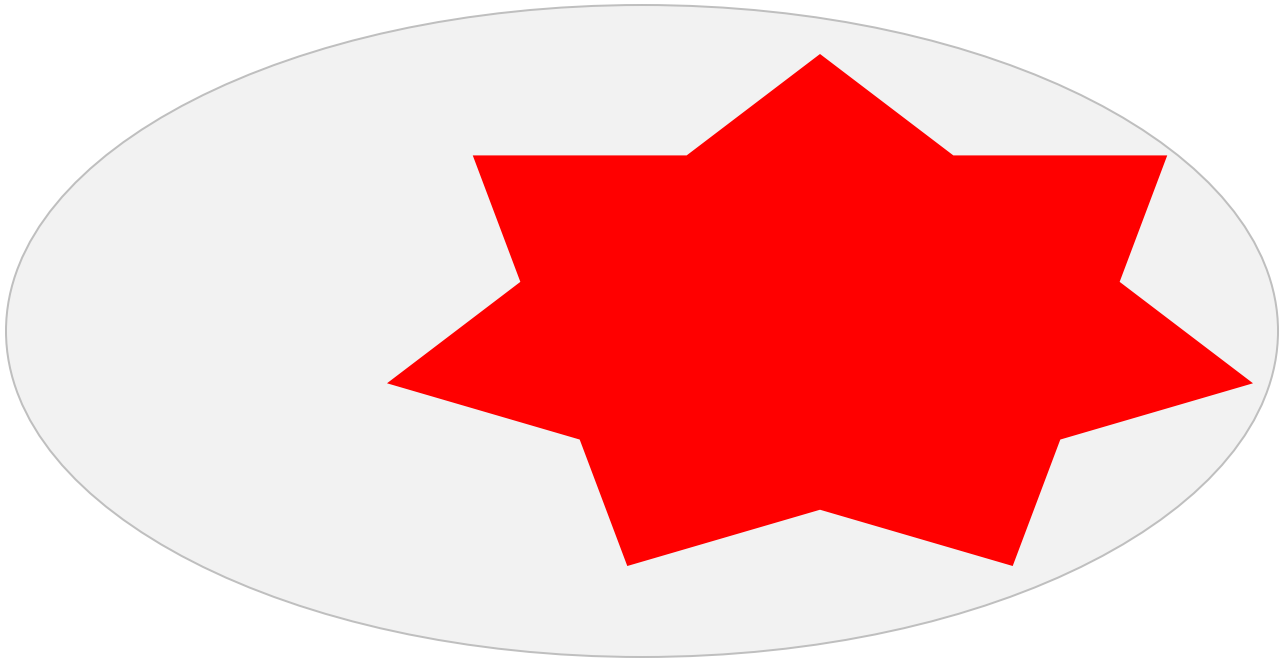
$g(\lambda) = 10\lambda_1 + 11\lambda_3 + 21\lambda_5 - \lambda_2 - \lambda_4$

$\Theta = \lambda_i \geq 0,$

$\lambda_1 + \lambda_4 + \lambda_5 = 1,$

$\lambda_2 + \lambda_4 - \lambda_3 - \lambda_5 = 2$

Complex high-dim equality constrained space. Tough to traverse/project.



Sound ✅

Traversable ❌

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.\ \begin{array}{ll} 10\ -x & \geq 0 \\ y\ -1 & \geq 0 \\ -y+11 & \geq 0 \\ y\ -x\ -1 & \geq 0 \\ 21\ -x\ -y & \geq 0 \end{array}$$

Duality ➡ **Selective Dualization**

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.\quad \begin{array}{ll} 10\ -x & \geq 0 \\ y\ -1 & \geq 0 \\ -y+11 & \geq 0 \\ y\ -x\ -1 & \geq 0 \\ 21\ -x\ -y & \geq 0 \end{array}$$

Duality ➡ **Selective Dualization**

Box Constraints

$$x \in [-\infty, 10],\quad y \in [1, 11]$$

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.\quad \begin{array}{ll} 10\ -x & \geq 0 \\ y\ -1 & \geq 0 \\ -y+11 & \geq 0 \\ y\ -x\ -1 & \geq 0 \\ 21\ -x\ -y & \geq 0 \end{array}$$

Duality ➡ **Selective Dualization**

Relational Constraints

$$\begin{array}{ll} y\ -x\ -1 & \geq 0 \\ 21\ -x\ -y & \geq 0 \end{array}$$

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.$$

$$\begin{array}{lll} 10\ -x & \geq 0 & \\ y\ -1 & \geq 0 & \\ -y+11 & \geq 0 & \\ y\ -x\ -1 & \geq 0 & \leftarrow \boldsymbol{\lambda_1} \\ 21\ -x\ -y & \geq 0 & \leftarrow \boldsymbol{\lambda_2} \end{array}$$

Duality ➡ **Selective Dualization**

Dualize only the relational constraints!

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

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x\ -2y\ \ s.t.$$

$$\begin{array}{lll} 10\ -x & \geq 0 & \\ y\ -1 & \geq 0 & \\ -y+11 & \geq 0 & \\ y\ -x\ -1 & \geq 0 & \leftarrow \boldsymbol{\lambda_1} \\ 21\ -x\ -y & \geq 0 & \leftarrow \boldsymbol{\lambda_2} \end{array}$$

Duality ➡ **Selective Dualization**

Dualize only the relational constraints!

Lagrangian $\quad L(x,y,\lambda) = (x\ -2y) + \boldsymbol{\lambda_1} * (\boldsymbol{y\ -x\ -1}) + \boldsymbol{\lambda_2} * (\boldsymbol{21\ -x\ -y})$

Lagrangian Dual $\quad g(\lambda) = \max_{x\in[-\infty,10],y\in[1,11]} L(x,y,\lambda)$

Key Idea 1: Sound Traversable Space of Outputs

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

$$c^{\#} = \max\ x - 2y \quad s.t. \quad \begin{array}{ll} 10 - x & \geq 0 \\ y - 1 & \geq 0 \\ -y + 11 & \geq 0 \\ y - x - 1 & \geq 0 \leftarrow \boldsymbol{\lambda_1} \\ 21 - x - y & \geq 0 \leftarrow \boldsymbol{\lambda_2} \end{array}$$

Duality ➡ **Selective Dualization**

Dualize only the relational constraints!

Lagrangian $\quad L(x, y, \lambda) = (x - 2y) + \boldsymbol{\lambda_1} * (\boldsymbol{y - x - 1}) + \boldsymbol{\lambda_2} * (\boldsymbol{21 - x - y})$

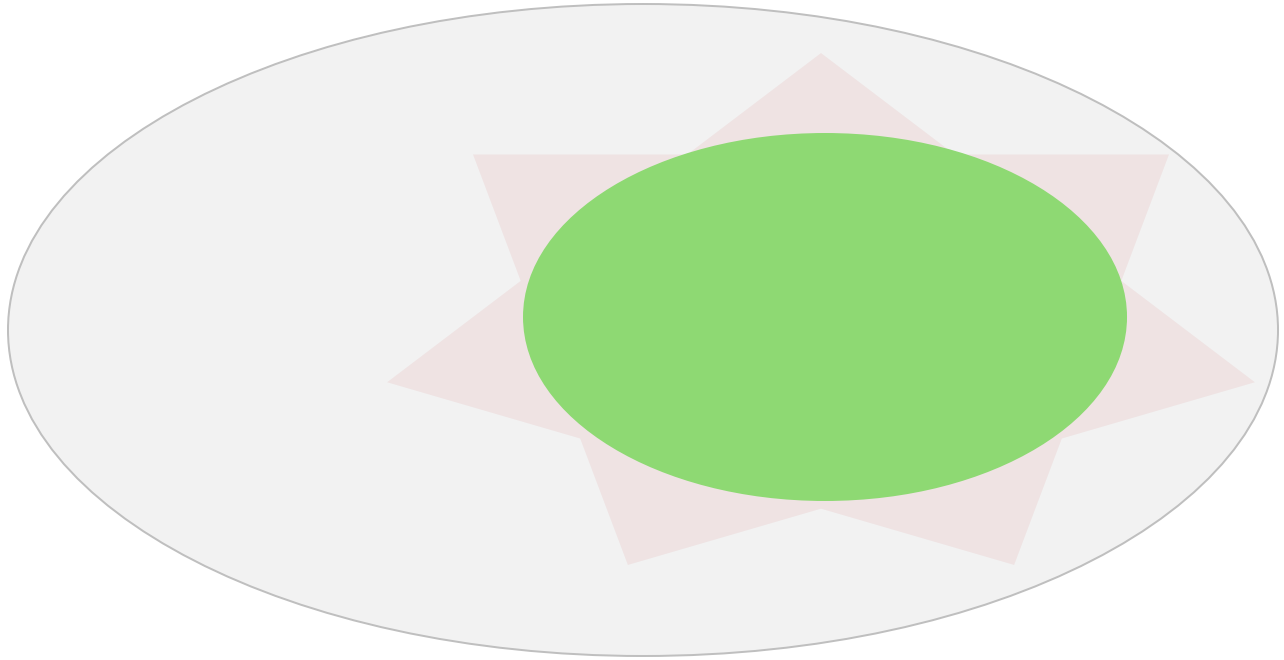
Lagrangian Dual $\quad g(\lambda) = \max_{x \in [-\infty, 10], y \in [1,11]} L(x, y, \lambda)$ ➡ Specialized Bounding Procedure ➡

Key Idea 1: Sound Traversable Space of Outputs

$g(\lambda) = 8 - 10\lambda_1 + 10\lambda_2$

$\Theta = \lambda_i \geq 0,$
$\lambda_1 + \lambda_2 \leq 1$

Lower dim, less and easy constraints!



Sound ✅

Traversable ✅

$g(\lambda) = 10\lambda_1 + 11\lambda_3 + 21\lambda_5 - \lambda_2 - \lambda_4$

$\Theta = \lambda_i \geq 0,$
$\lambda_1 + \lambda_4 + \lambda_5 = 1,$
$\lambda_2 + \lambda_4 - \lambda_3 - \lambda_5 = 2$

Key Idea 1: Sound Traversable Space of Outputs

$$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2 \quad \Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$$

$$\begin{aligned} -\infty \leq\ & x & \leq 10 \\ 1 \leq\ & y & \leq 11 \\ -\infty \leq\ & x - y & \\ \leq\ & -1 & \\ -\infty \leq\ & x + y & \leq 21 \end{aligned}$$

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

$$\begin{aligned} ? \leq\ & x & \leq ? \\ ? \leq\ & y & \leq ? \\ ? \leq\ & x - y & \leq g(\lambda_1, \lambda_2) \\ ? \leq\ & x + y & \leq ? \end{aligned}$$

All Sound!

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

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

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

Interval result at 0!

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

$(\lambda_1, \lambda_2) = (1, 0) \Rightarrow g(\lambda_1, \lambda_2) = -2$ (Most Precise)

Most Precise always in space for linear operators!

Traversing the space of outputs

$$
\begin{array}{l}
-\infty \leq \quad x \quad \leq 10 \\
1 \quad \leq \quad y \quad \leq 11 \\
-\infty \leq x - y \\
\leq -1 \\
-\infty \leq x + y \leq 21
\end{array}
\Rightarrow \boxed{(x = x - y)^{\#}} \Rightarrow O'(\theta) = \begin{array}{l}
f_1(\theta) \leq \quad x \quad \leq f_2(\theta) \\
f_3(\theta) \leq \quad y \quad \leq f_4(\theta) \\
f_5(\theta) \leq x - y \leq f_6(\theta) \\
f_7(\theta) \leq x + y \leq f_8(\theta)
\end{array}
$$

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

- Need a concrete output to proceed (choose some $\theta^* \in \Theta$ to initialize)!
- Ideally, find the most precise in the space of outputs but not trivial.
- For 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.

Traversing the space of outputs

Space is differentiable!

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$ $\quad \Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$

$$
\begin{array}{l}
-\infty \leq \quad x \quad \leq 10 \\
1 \quad \leq \quad y \quad \leq 11 \\
-\infty \leq x - y \\
\leq -1 \\
-\infty \leq x + y \leq 21
\end{array}
\quad \Rightarrow \quad (x = x - y)^{\#} \quad \Rightarrow \quad
\begin{array}{l}
? \leq \quad x \quad \leq ? \\
? \leq \quad y \quad \leq ? \\
? \leq x - y \leq g(\lambda_1, \lambda_2) \\
? \leq x + y \leq ?
\end{array}
$$

- $g(\lambda_1, \lambda_2)$ represents the bound; smaller value => more precise.
- Gradient-descent to move towards precise outputs with gradient steps R. More time -> More Steps -> More Precise
- Naive gradient descent is difficult as we have constraints on parameters.

Key Idea 2: Adaptive Gradient-Guided Search

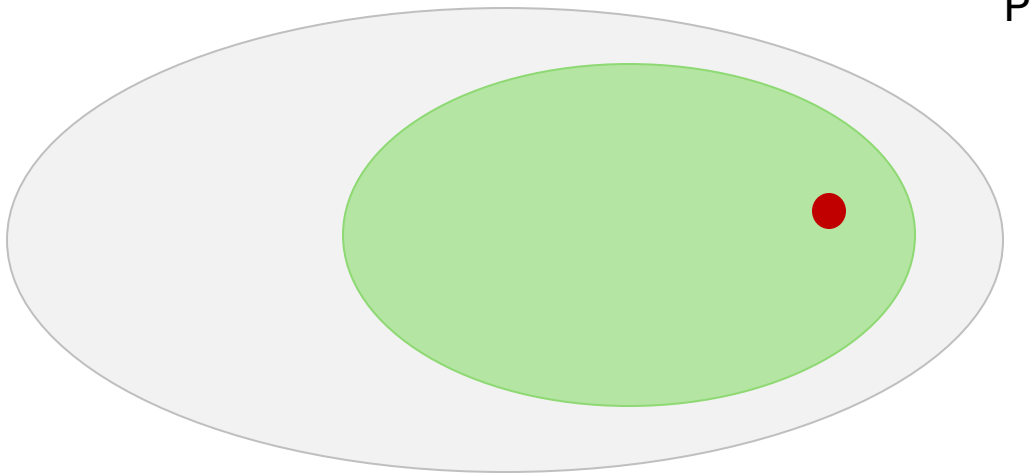
- Use **adaptive gradient descent** to make the bounds precise!
- If inside parameter space Θ, minimize $g(\lambda_1, \lambda_2)$ by descending on it.
- If the search goes outside Θ, we **steer** the search back by minimizing constraint violations.
- Traversal is started at $\theta = 0$ to always be as precise as interval relaxation.

Key Idea 2: Adaptive Gradient-Guided Search

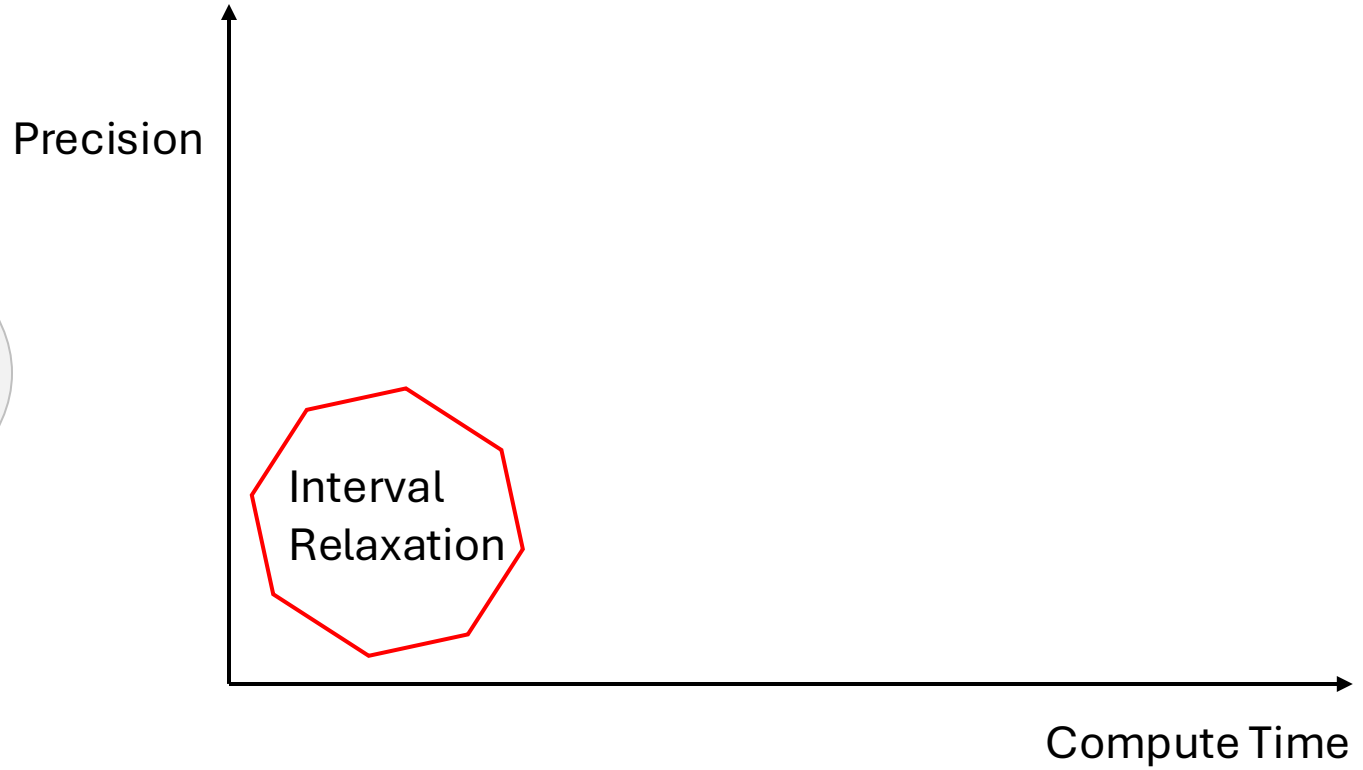
$\Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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



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

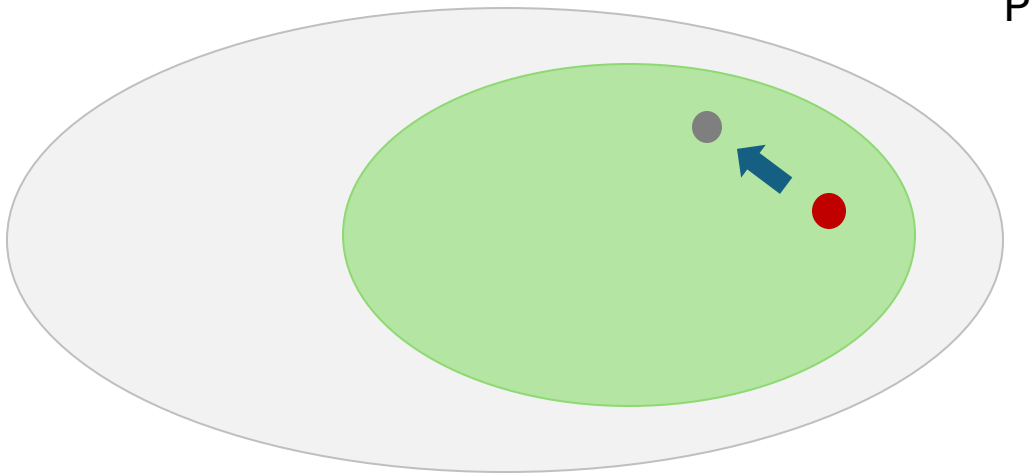


Key Idea 2: Adaptive Gradient-Guided Search

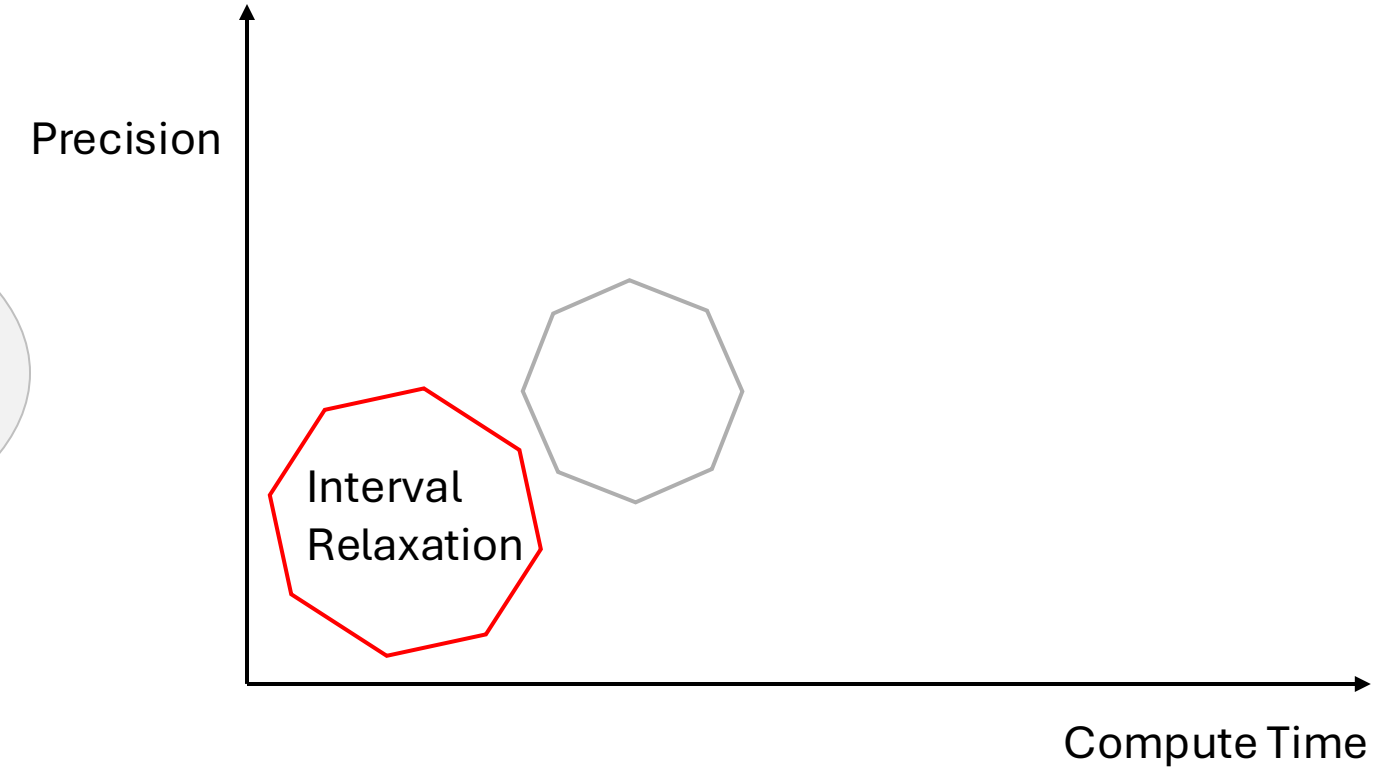
$\Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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



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

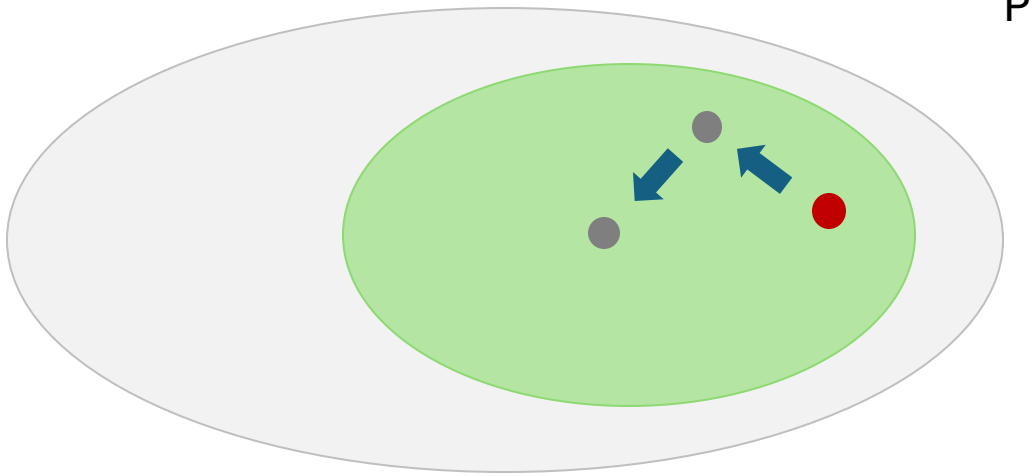


Key Idea 2: Adaptive Gradient-Guided Search

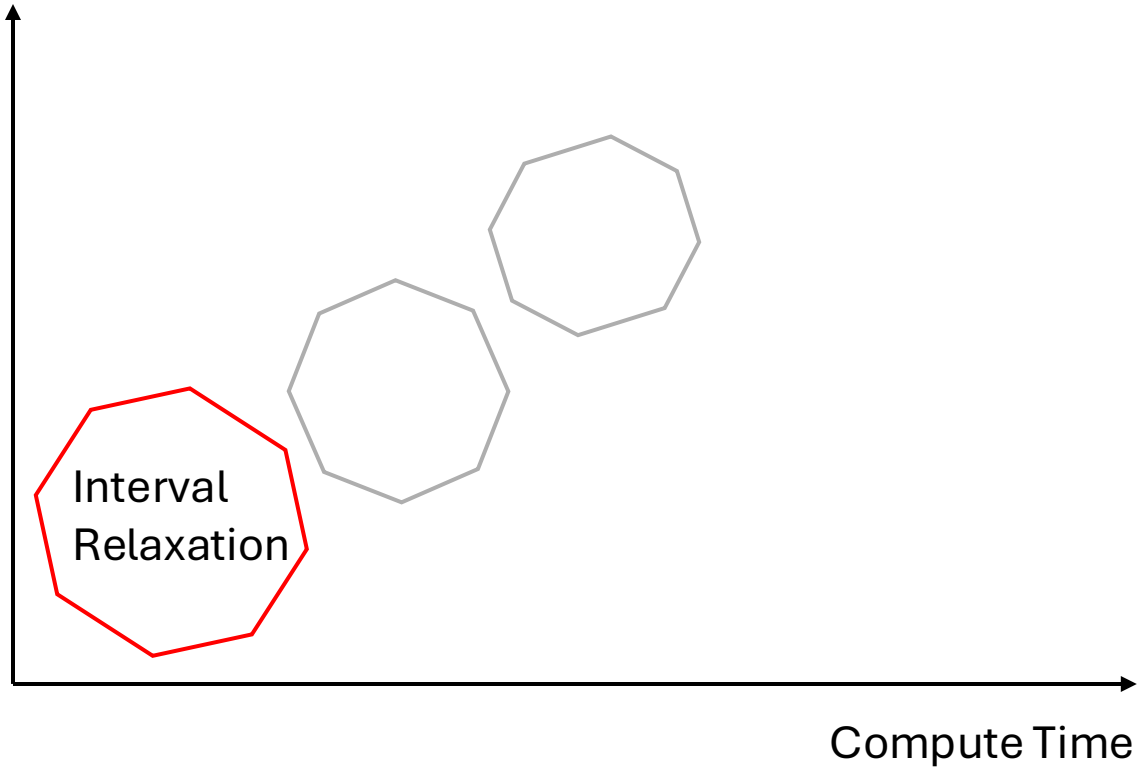
$\Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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



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

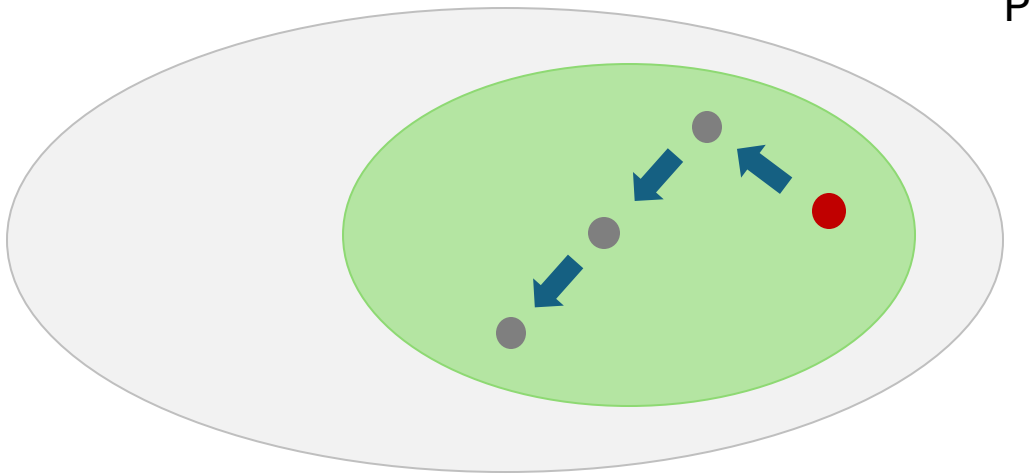


Key Idea 2: Adaptive Gradient-Guided Search

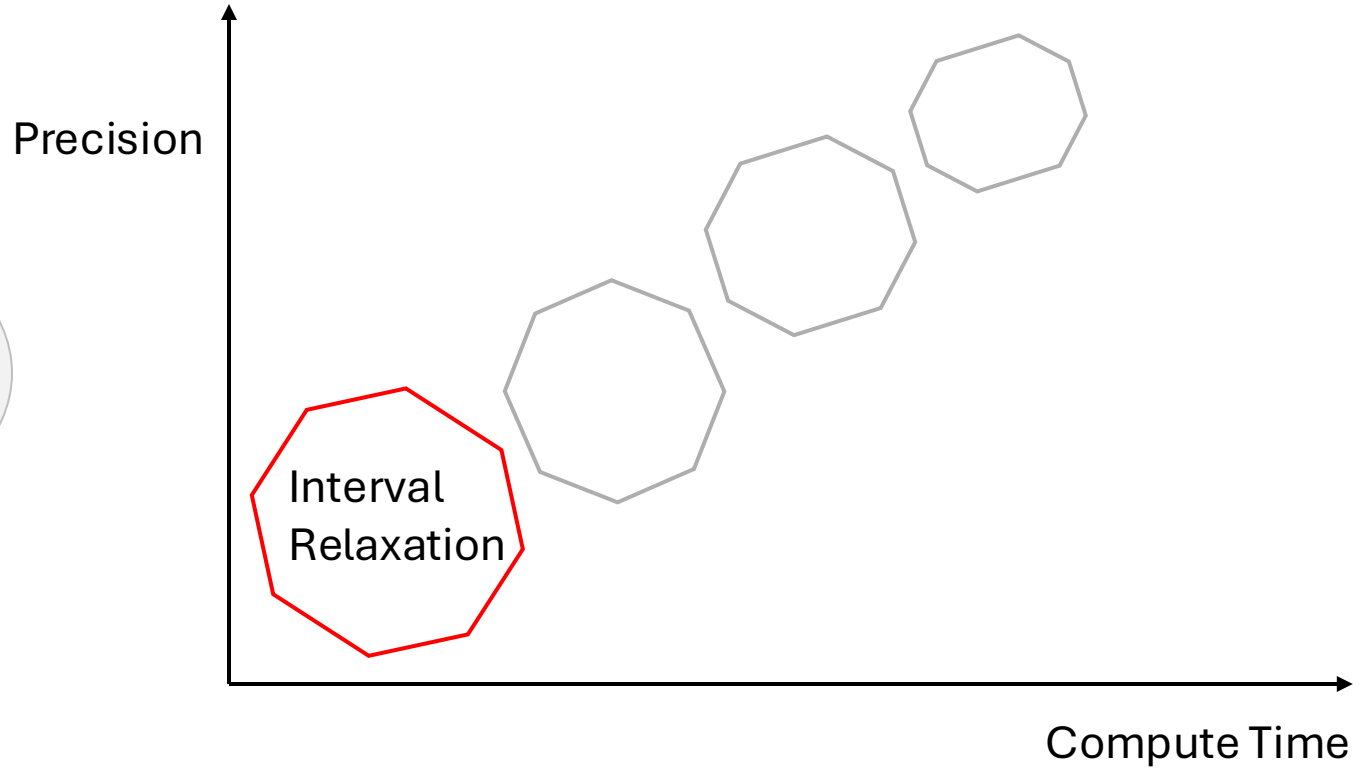
$\Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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



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

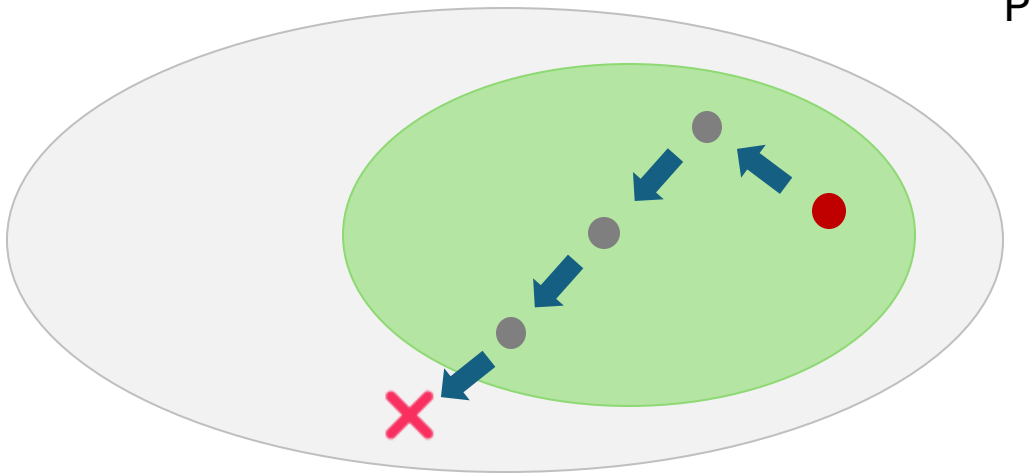


Key Idea 2: Adaptive Gradient-Guided Search

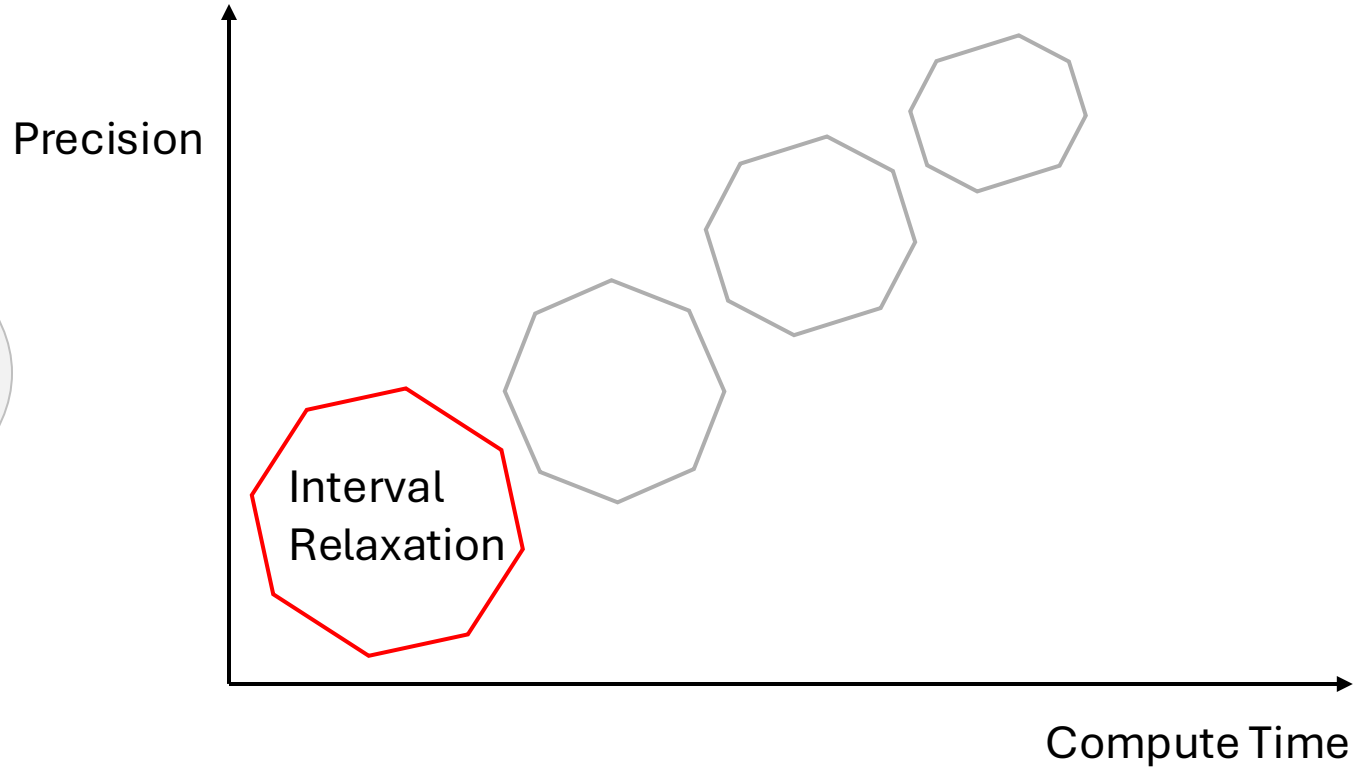
$\Theta = \{\lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}$

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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



$(\lambda_1, \lambda_2) = (1.2, 0)$

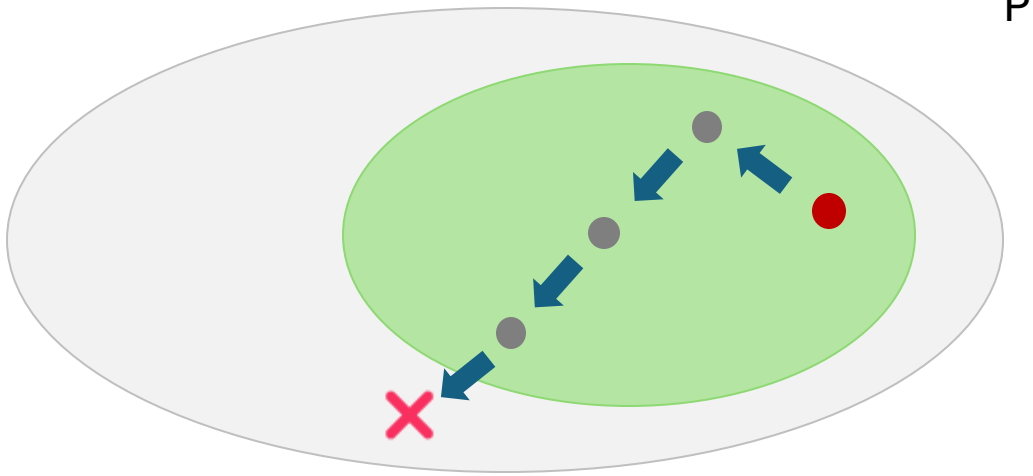


Key Idea 2: Adaptive Gradient-Guided Search

$\Theta = \{\lambda_1, \lambda_2 \geq 0, \boldsymbol{\lambda_1 + \lambda_2 \leq 1}\}$

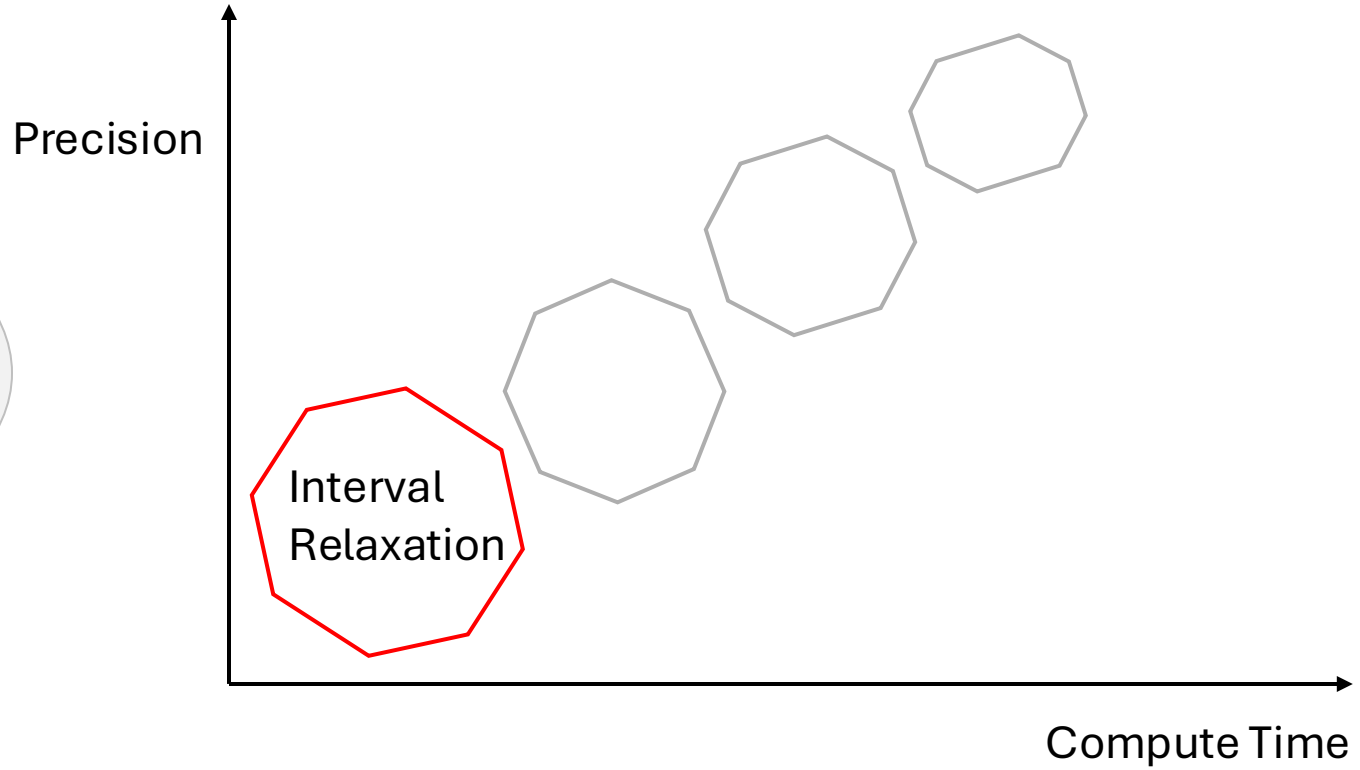
$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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



$(\lambda_1, \lambda_2) = (1.2, 0)$

$\lambda_1 + \lambda_2 \not\leq 1$

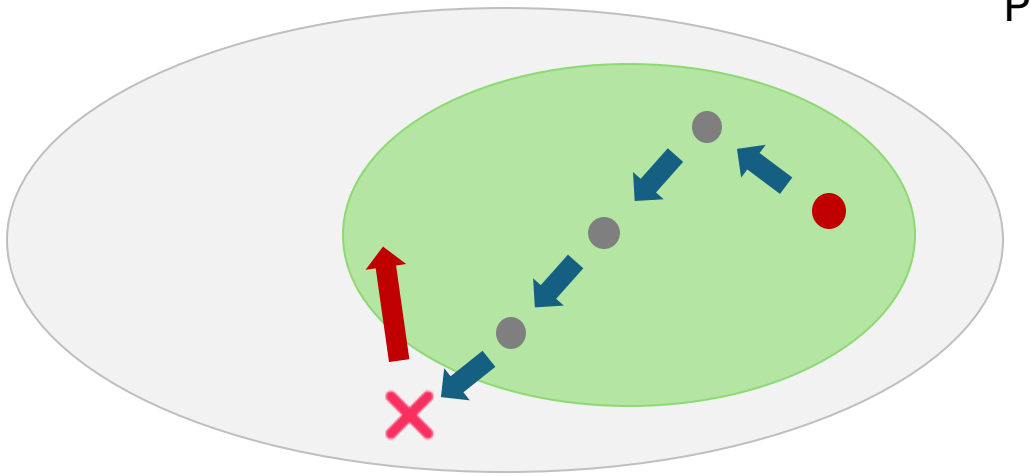


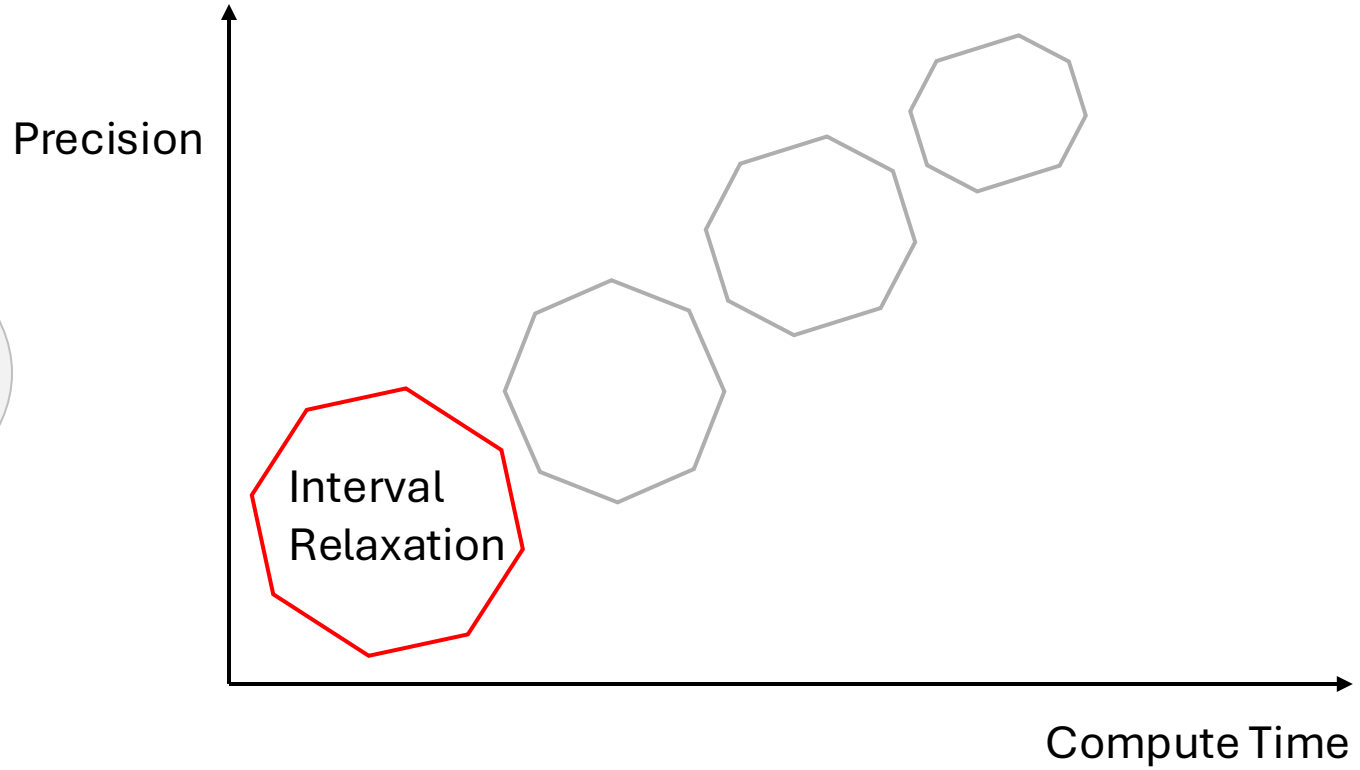
Key Idea 2: Adaptive Gradient-Guided Search

$\Theta = \{\lambda_1, \lambda_2 \geq 0, \boldsymbol{\lambda_1 + \lambda_2 \leq 1}\}$

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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





$(\lambda_1, \lambda_2) = (1.2, 0)$

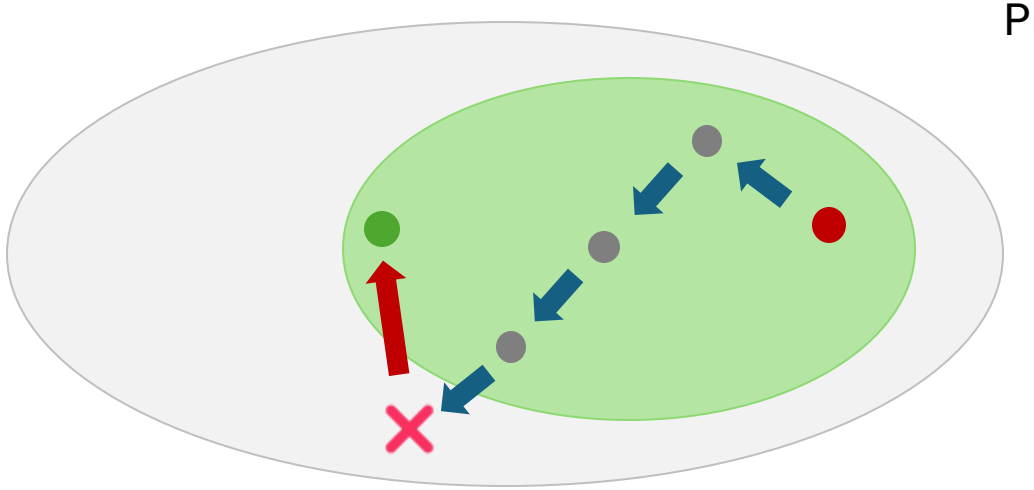
$\lambda_1 + \lambda_2 \not\leq 1$ Descent on $\max(\lambda_1 + \lambda_2 - 1, 0)$ to restore feasibility!

Key Idea 2: Adaptive Gradient-Guided Search

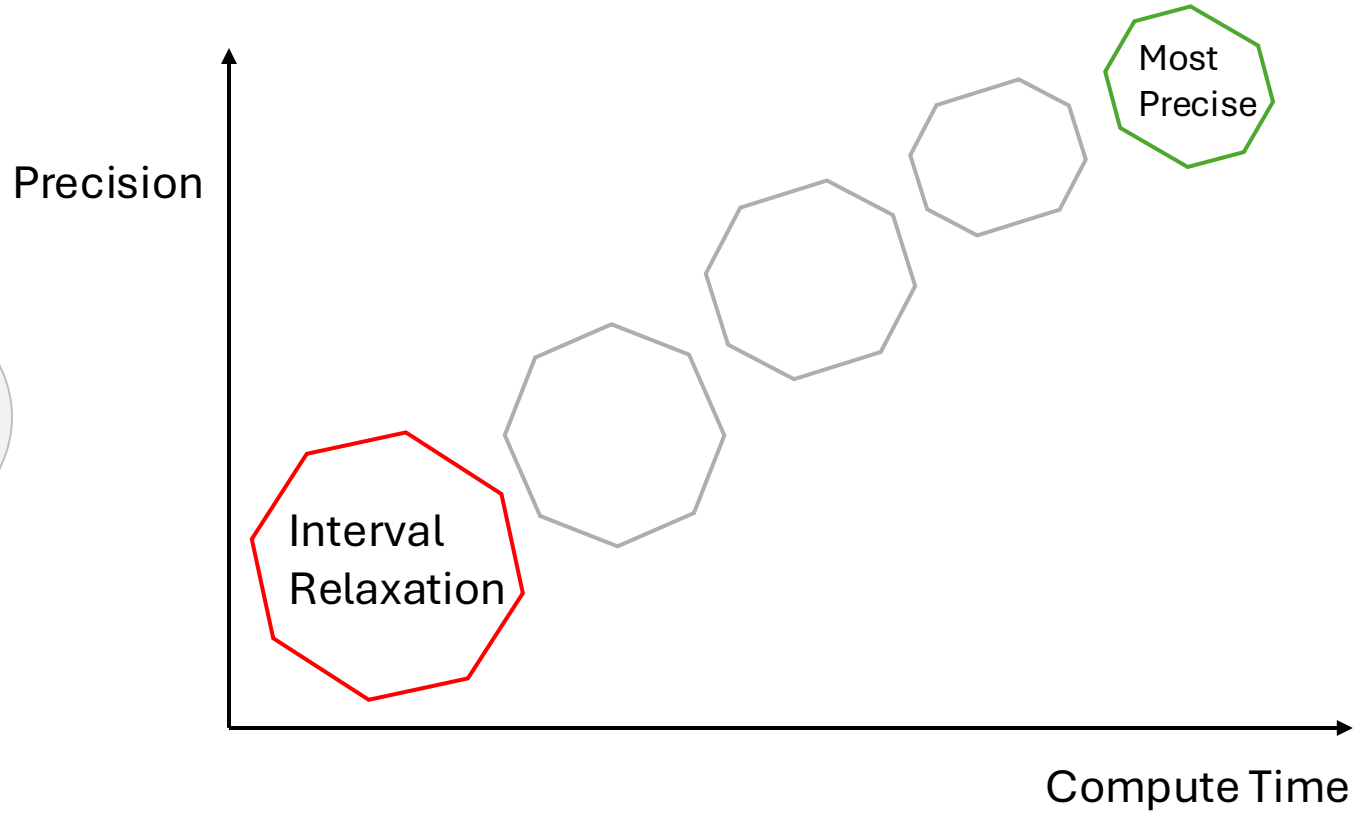
$\Theta = \{\lambda_1, \lambda_2 \geq 0, \boldsymbol{\lambda_1 + \lambda_2 \leq 1}\}$

$g(\lambda_1, \lambda_2) = 8 - 10\lambda_1 + 10\lambda_2$

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



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

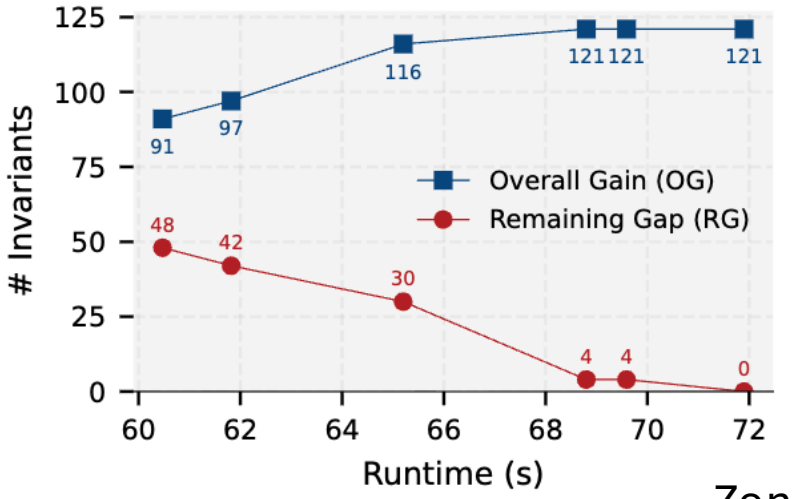


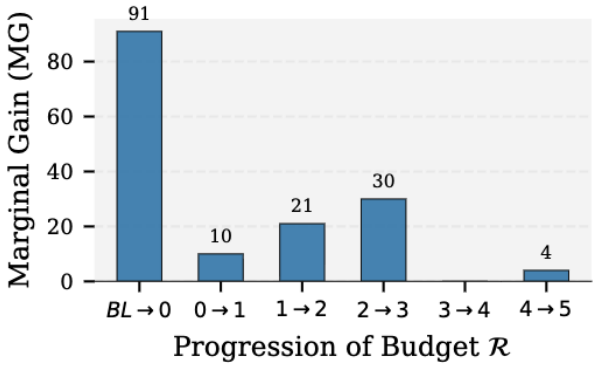
Evolving Abstract Transformers

Per-Domain, Per-Instruction	Works across domains and instructions ✅
Instruction-Wise Reasoning	Supports sequences ✅
Fixed Precision-Efficiency Tradeoff	Adaptable Precision-Efficiency ✅

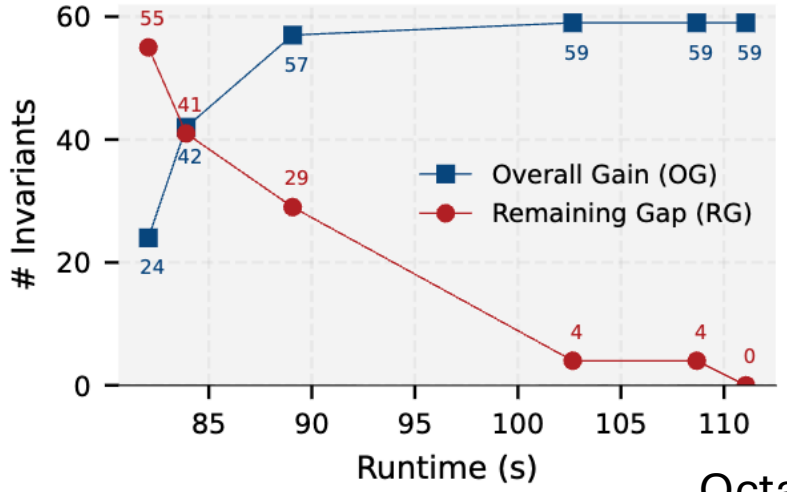
Experimental Evaluation: Linear

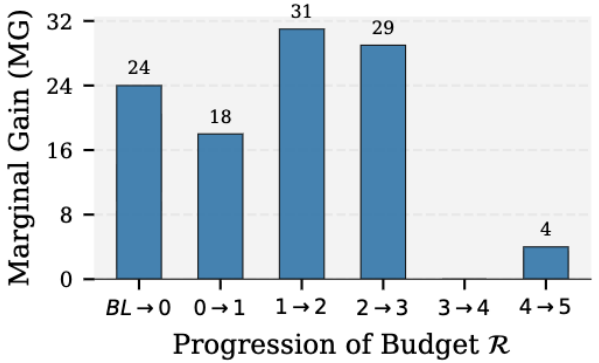
- NLA-Digbench benchmarks: 55 programs. 237 invariants across all program points.
- Analysis run with increasing number of gradient steps R for linear operators.
- **Overall Gain (OG):** # of invariants strengthened over those computed by state-of-the-art library ELINA.
- **Remaining Gap (RG):** # of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





Zones

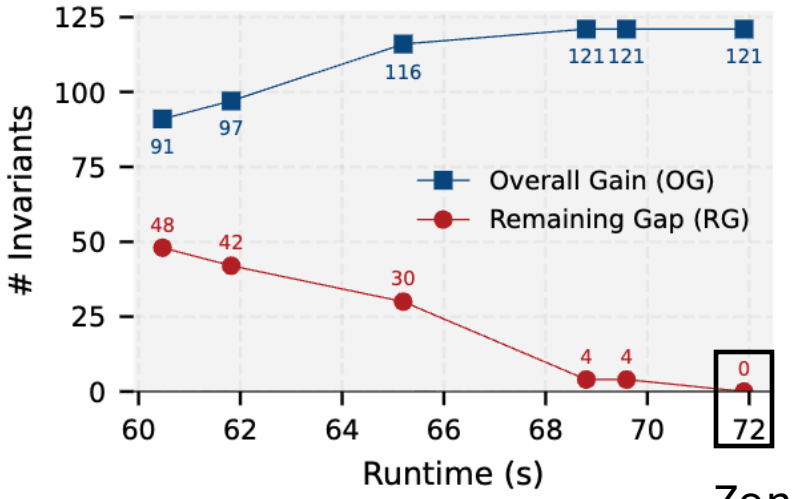


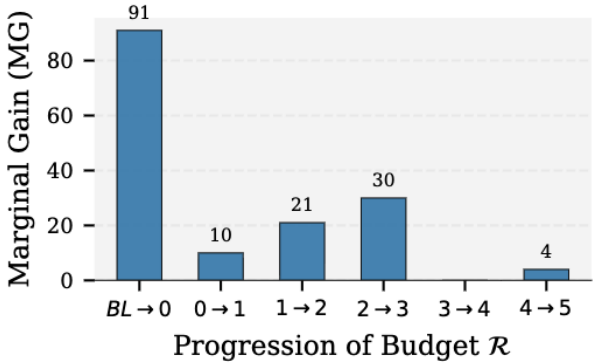


Octagons

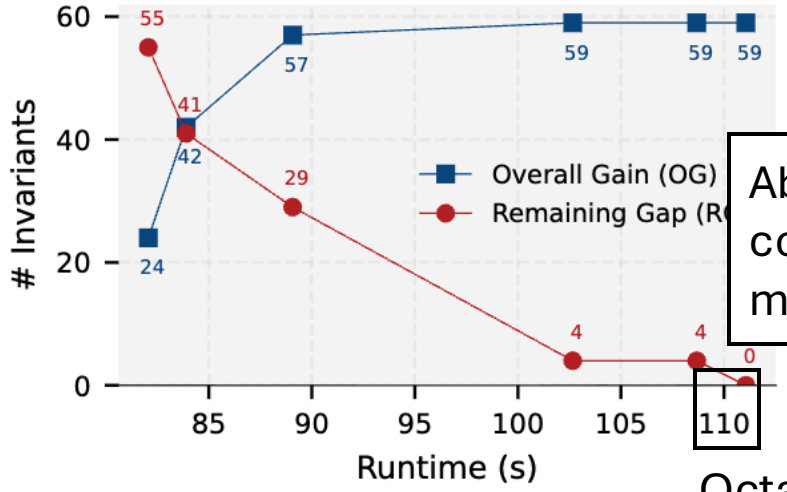
Experimental Evaluation: Linear

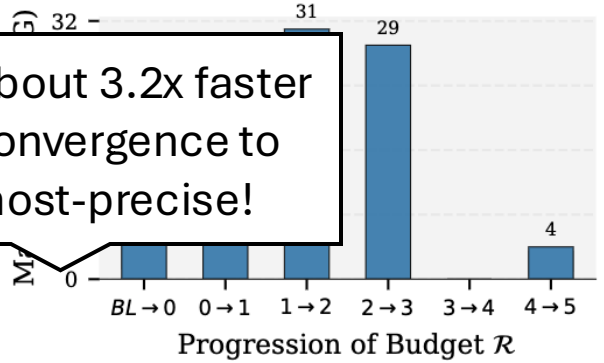
- NLA-Digbench benchmarks: 55 programs. 237 invariants across all program points.
- Analysis run with increasing number of gradient steps R for linear operators.
- **Overall Gain (OG):** # of invariants strengthened over those computed by state-of-the-art library ELINA.
- **Remaining Gap (RG):** # of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





Zones





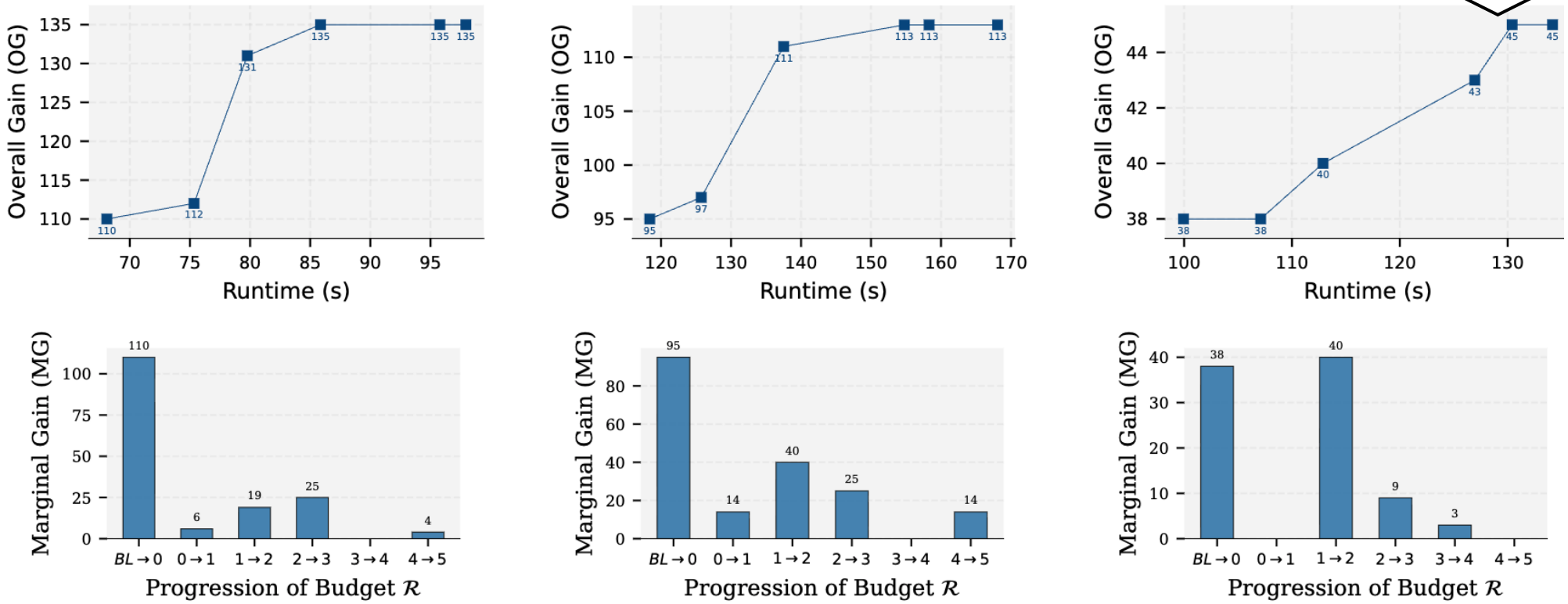
About 3.2x faster convergence to most-precise!

Octagons

Experimental Evaluation: Quadratic

Gains over baseline even for complex domains like Polyhedra!

Analysis run with increasing number of gradient steps R for linear and quadratic operators.



(a) Zones (b) Octagons (c) Polyhedra

Looking Forward

- GPU-driven large-scale numerical abstract interpretation.

- Other analysis tasks like finding an octagon within a target one

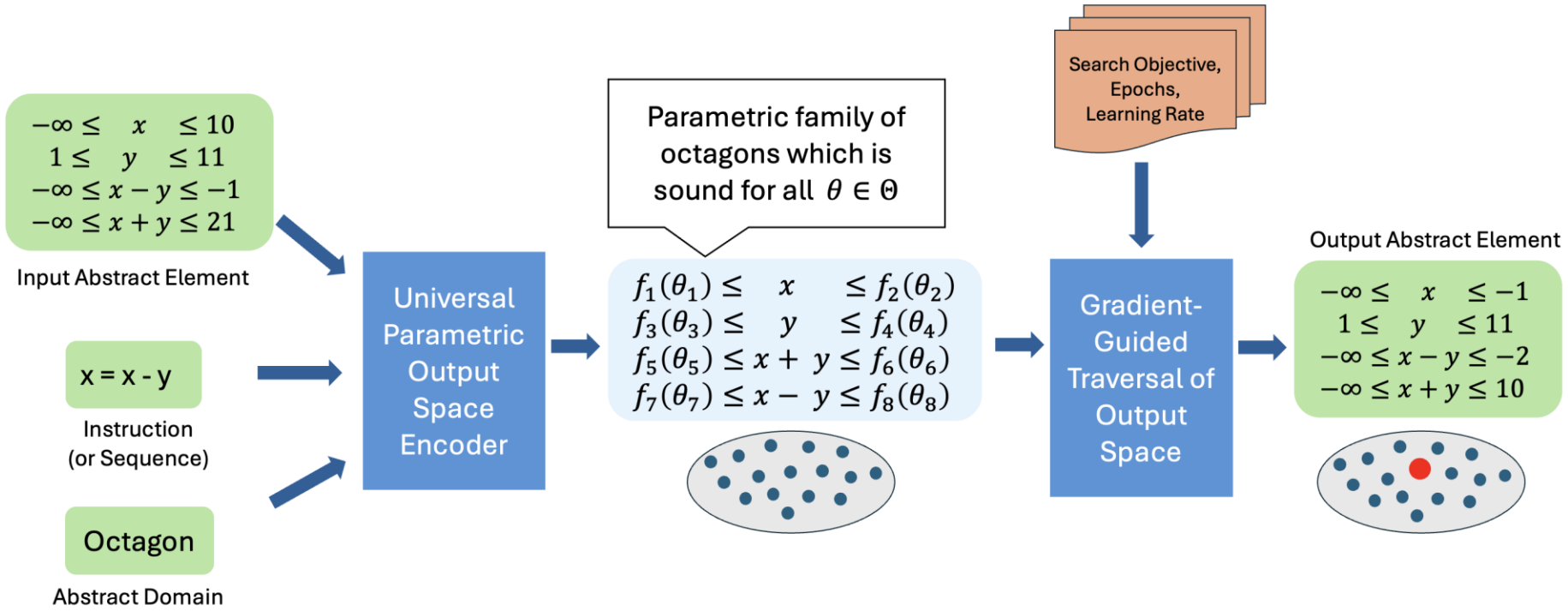
  $x - y \leq g(\lambda_1, \lambda_2) \subseteq x - y \leq 10$ => $\text{Descent on } \max(10 - g(\lambda_1, \lambda_2), 0)$

- Directly use a NN or LLM to learn/predict $\theta \in \Theta$ for various tasks.

- While program verification, if property unproved,
  choices of θ can be refined!

Conclusion

Evolving Abstract Transformers provide a **unified framework** across various domains, instructions, and instruction sequences and work by constructing a **sound space of outputs** and **evolving** within it to allow precision-efficiency adaptability.



Paper:

Code:

Reach out at sgomber2@illinois.edu!