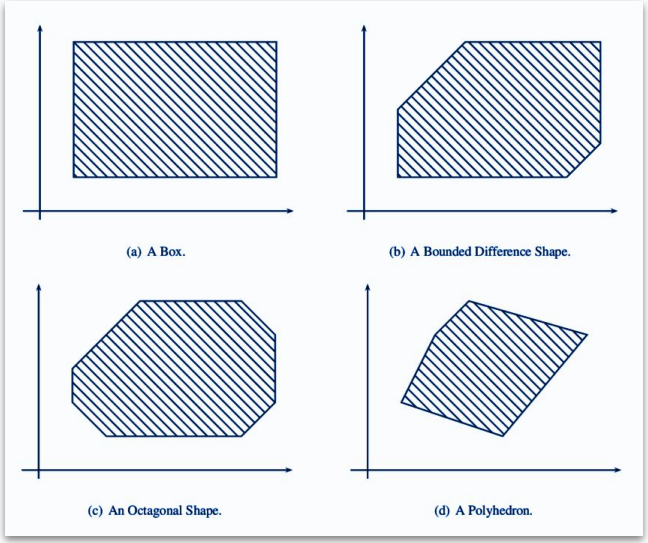
Numerical Abstract Domain

Ashkan Vedadi Gargary

Guideline

1. What is Numerical Abstract Domain?
2. OCTAGON Abstract Domain
3. Polyhedra Abstract Domain
4. Fast Polyhedra Abstract Domain
5. Fast Polyhedra Vs. Octagon



Numerical Abstract Domain

Abstract Domains in Program Analysis

- Mathematical structures → sets of possible program states
- Strike a balance between precision and efficiency
- Capturing relevant properties of **program variables** while abstracting away irrelevant details.
- Proving the **absence of buffer overflow**, **division by zero**, and other properties.

The Octagon Abstract Domain

What is Octagon Abstract domain?

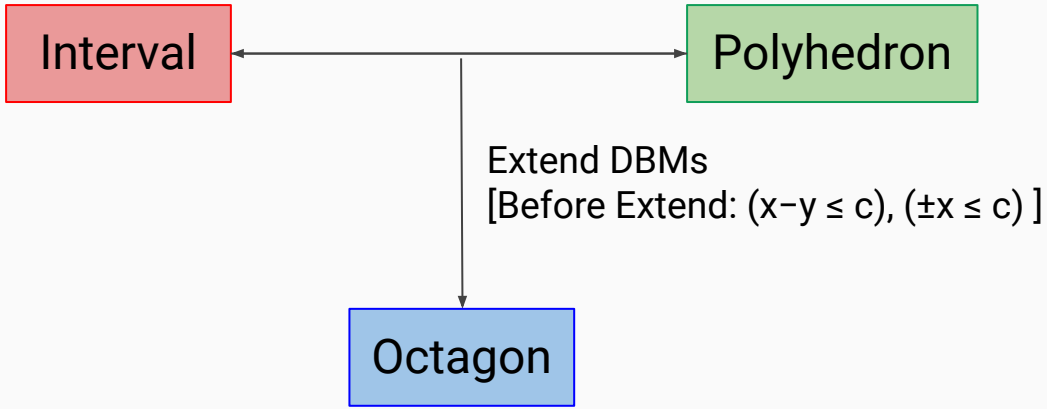
- It is specifically designed for analyzing programs with numerical variables and loops.
- Practical algorithms to represent and manipulate invariants of the form **(± x ± y ≤ c),** where X and y are numerical variables and c is a numeric constant

Previous Work

- It extends the analysis we previously proposed in our **PADO-II** article.

1. **Numerical Abstract Domains**
   - The most used are the lattice of **Intervals** and lattice of **Polyhedra**
   - $\alpha_1 v_1 + \cdots + \alpha_n v_n \leq c$
2. **Difference-Bound Matrices**
   - The model-checking community has developed a practical representation, called Difference-Bound Matrices (DBMs), for constraints of the form **(x−y ≤ c)** and **(±x ≤ c)**, together with many operators, in order to model-check timed automata

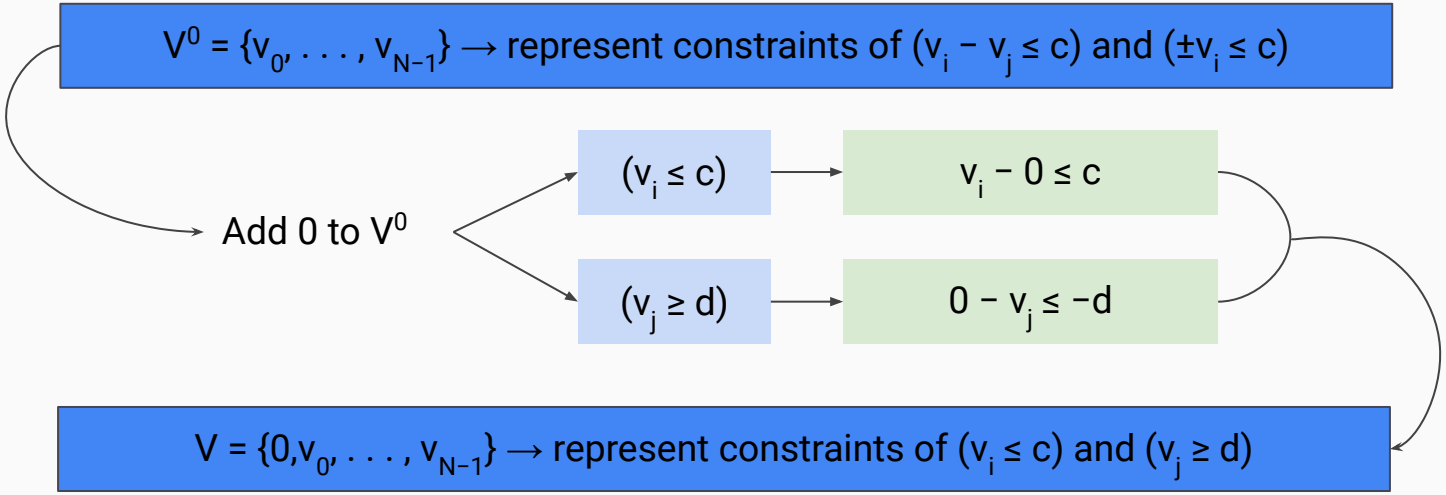
Their GOAL and Motivation



interval constraints (**±x ≤ c**), and sum constraints (**±x ± y ≤ c**)

Extending Difference-Bound Matrices

Representing Intervals



Extending Difference-Bound Matrices

Representing Sums

- $V^+ = \{v_0, \ldots, v_{N-1}\} \rightarrow$ represent constraints of $(\mathbf{v_i - v_j \leq c})$
- we consider that each variable v_i in V^+ comes in two flavors:
  - a positive form v^+_i and a negative form $v^-_i \rightarrow V = \{ v^+_0, v^-_0, \ldots, v^+_{N-1}, v^-_{N-1}\}$

constraint over $\mathcal{V}^+$	constraint(s) over $\mathcal{V}$
$v_i - v_j \leq c \quad (i \neq j)$	$v_i^+ - v_j^+ \leq c, \quad v_j^- - v_i^- \leq c$
$v_i + v_j \leq c \quad (i \neq j)$	$v_i^+ - v_j^- \leq c, \quad v_j^+ - v_i^- \leq c$
$-v_i - v_j \leq c \quad (i \neq j)$	$v_j^- - v_i^+ \leq c, \quad v_i^- - v_j^+ \leq c$
$v_i \leq c$	$v_i^+ - v_i^- \leq 2c$
$v_i \geq c$	$v_i^- - v_i^+ \leq -2$

Operators and Transfer Functions

Describe how to implement the abstract operators and transfer functions needed for static analysis.

- Equality and Inclusion Testing
- Widening
- Narrowing
- Projection
- Union
- Intersection

Lattice Structure

1. Coherent DBMs Lattice
   - Theorem:
     - (M+ $\bot$, $\sqsubseteq$, $\sqcap$, $\sqcup$, $\bot$, $\top$) is a lattice.
     - This lattice is complete if (I, ≤) is complete (I = Z or R, but not Q).
   - Problems:
     - not isomorphic
     - $\sqcup$ is not the most precise upper approximation of the union of two octagons

Lattice Structure

2. **Strongly Closed DBMs Lattice**
   - based on strongly closed DBMs:

$$\top^{\bullet}{}_{ij} \triangleq \begin{cases} 0 & \text{if } i = j, \\ +\infty & \text{elsewhere,} \end{cases}$$

$$\mathbf{m}^+ \sqsubseteq^{\bullet} \mathbf{n}^+ \iff \begin{cases} \text{either} & \mathbf{m}^+ = \bot^{\bullet}, \\ \text{or} & \mathbf{m}^+, \mathbf{n}^+ \neq \bot^{\bullet},\ \mathbf{m}^+ \trianglelefteq \mathbf{n}^+, \end{cases}$$

$$\mathbf{m}^+ \sqcup^{\bullet} \mathbf{n}^+ \triangleq \begin{cases} \mathbf{m}^+ & \text{if } \mathbf{n}^+ = \bot^{\bullet}, \\ \mathbf{n}^+ & \text{if } \mathbf{m}^+ = \bot^{\bullet}, \\ \mathbf{m}^+ \vee \mathbf{n}^+ & \text{elsewhere,} \end{cases}$$

$$\mathbf{m}^+ \sqcap^{\bullet} \mathbf{n}^+ \triangleq \begin{cases} \bot^{\bullet} & \text{if } \bot^{\bullet} \in \{\mathbf{m}^+, \mathbf{n}^+\} \text{ or} \\ & \mathcal{D}^+(\mathbf{m}^+ \wedge \mathbf{n}^+) = \emptyset, \\ (\mathbf{m}^+ \wedge \mathbf{n}^+)^{\bullet} & \text{elsewhere .} \end{cases}$$

   - Create a meaning function γ which is an extension of $\bullet \longrightarrow \mathbf{D^+(\bullet)}$ to $M^{\bullet}_{\bot}$:

$$\gamma(\mathbf{m}^+) \triangleq \begin{cases} \emptyset & \text{if } \mathbf{m}^+ = \bot^{\bullet}, \\ \mathcal{D}^+(\mathbf{m}^+) & \text{elsewhere .} \end{cases}$$

Lattice Structure

2. **Strongly Closed DBMs Lattice**

- Theorem:
  - If $(I, \leq)$ is complete, this lattice is complete and γ is meet-preserving: $\gamma(\Pi^{\bullet} X) = \cap\{\gamma(x) \mid x \in X\}$.
  - $(M^{\bullet}_{\perp}, \sqsubseteq^{\bullet}, \sqcap^{\bullet}, \sqcup^{\bullet}, \bot^{\bullet}, \top^{\bullet})$ is a lattice and γ is one-to-one

Application and Program Analysis

1. Presentation
2. Practical Result
   a. The analysis described above has been implemented in OCaml and used on a small set of rather simple algorithms
   b. Their analyzer was also able to prove that the well-known Bubble sort and Heap sort do not perform out-of-bound error while accessing array elements
3. Precision and Cost
   a. Most abstract operators have a $O(N^3)$ worst case time cost
   b. They are less precise than the costly polyhedron analysis
   c. This domain allows us to manipulate invariants of the form $(\pm x \pm y \leq c)$ with a $O(n^2)$ worst case memory cost per abstract state and a $O(n^3)$ worst case time cost per abstract operation

```
 4   (l0) a ← 0; i ← 1 (l1)
     while i ≤ m do (l2)
 7     if ?
 8       then (l3) a ← a + 1 (l4)
 9       else (l5) a ← a − 1 (l6)
       fi (l7)
       i ← i + 1 (l8)
11   done (l9)
```

$\mathbf{m}_0^+ = \top$
$\mathbf{m}_1^+ = \{i = 1;\ a = 0;\ 1 - i \leq a \leq i - 1\}$

First iteration of the loop
$\mathbf{m}_{2.0}^+ = \{i = 1;\ a = 0;\ 1 - i \leq a \leq i - 1;\ i \leq m\}$
$\mathbf{m}_{3,0}^+ = \mathbf{m}_{5,0}^+ = \mathbf{m}_{2.0}^+$
$\mathbf{m}_{4,0}^+ = \{i = 1;\ a = 1;\ 2 - i \leq a \leq i;\ i \leq m\}$
$\mathbf{m}_{6,0}^+ = \{i = 1;\ a = -1;\ -i \leq a \leq i - 2;\ i \leq m\}$
$\mathbf{m}_{7,0}^+ = \{i = 1;\ a \in [-1, 1];\ -i \leq a \leq i;\ i \leq m\}$
$\mathbf{m}_{8,0}^+ = \{i = 2;\ a \in [-1, 1];\ 1 - i \leq a \leq i - 1;\ i \leq m + 1\}$

Second iteration of the loop
$\mathbf{m}_{2,1}^+ = \mathbf{m}_{3,1}^+ = \mathbf{m}_{5,1}^+ = \mathbf{m}_{2,0}^+ \triangledown (\mathbf{m}_{8,0}^+)_{(i \leq m)}$
$= \{1 \leq i \leq m;\ 1 - i \leq a \leq i - 1\}$
$\mathbf{m}_{4,1}^+ = \{1 \leq i \leq m;\ 2 - i \leq a \leq i\}$
$\mathbf{m}_{6,1}^+ = \{1 \leq i \leq m;\ -i \leq a \leq i - 2\}$
$\mathbf{m}_{7,1}^+ = \{1 \leq i \leq m;\ -i \leq a \leq i\}$
$\mathbf{m}_{8,1}^+ = \{2 \leq i \leq m + 1;\ 1 - i \leq a \leq i - 1\}$

Third iteration of the loop
$\mathbf{m}_{2,2}^+ = \mathbf{m}_{2,1}^+$ *(fixpoint reached)*

$\mathbf{m}_2^+ = \mathbf{m}_{2,1}^+ \qquad \mathbf{m}_8^+ = \mathbf{m}_{8,1}^+$
$\mathbf{m}_9^+ = \{i = m + 1;\ 1 - i \leq a \leq i - 1\}$

Advantages

- Provide some faster approach with good approximation for small codes
- This domain allows us to manipulate invariants of the form ($\pm x \pm y \leq c$) with a $O(n^2)$ worst case memory cost per abstract state and a $O(n^3)$ worst case time cost per abstract operation

Disadvantages

- Works only for small test cases and not able to work in large inputs and program with large variables
- They didn’t provide exact time and cost analysis
- They are less precise than the polyhedron analysis
- Their prototype implementation did not allow us to test our domain on real life programs and they still do not know if it will scale up.

Conclusion

- They presented a new numerical abstract domain that extends DBM
- $(\pm x \pm y \leq c)$ with a $O(n^2)$ worst case memory cost per abstract state and a $O(n^3)$ worst case time cost per abstract operation
- Their approach is fruitful
- Their analyzer was also able to prove that the well-known Bubble sort and Heap sort do not perform out-of-bound error while accessing array elements
- They are less precise than the costly polyhedron analysis

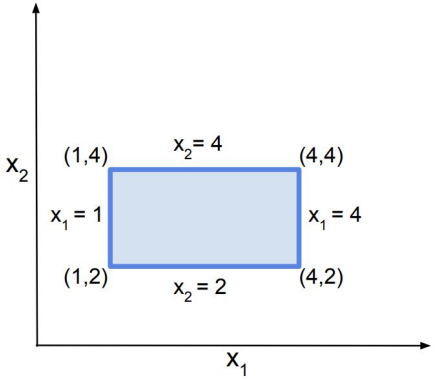
Polyhedra Abstract Domain

What is Polyhedra?

- Prove safety properties in programs
- It is a fully relational numerical domain, i.e., can encode all possible linear constraints between program variables.
- The most precious, but also the most expensive

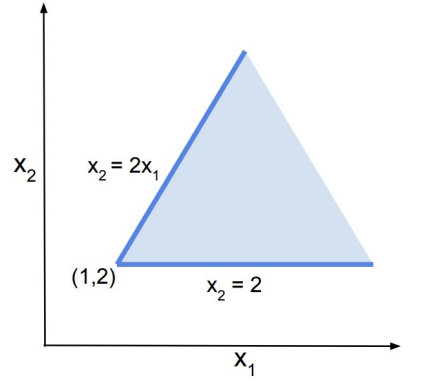
Polyhedron Representation

1) Constraint Representation [Intersection]
2) Generator Representation [Convex Hull]



(a)

$\mathcal{C} = \{-x_1 \leq -1, x_1 \leq 4, -x_2 \leq -2, x_2 \leq 4\}$, or
$\mathcal{G} = \{\mathcal{V} = \{(1,2),(1,4),(4,2),(4,4)\}, \mathcal{R} = \emptyset, \mathcal{Z} = \emptyset\}$.



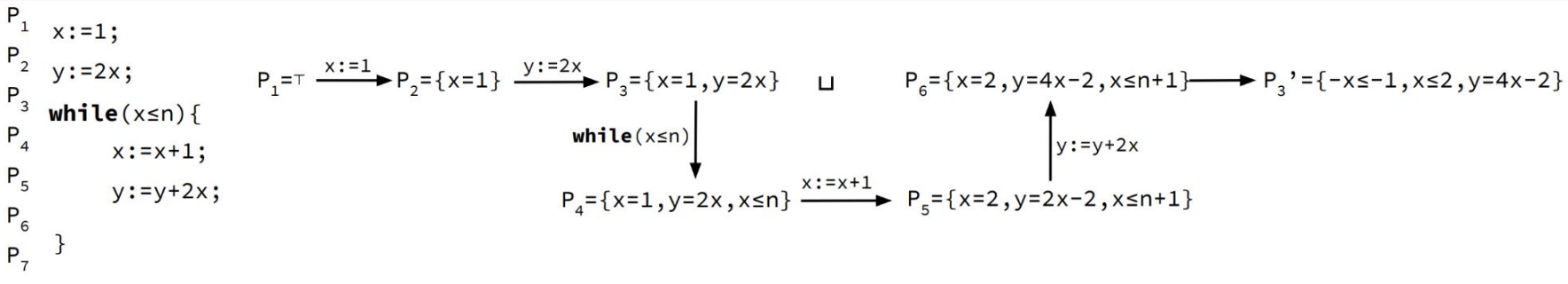
(b)

$\mathcal{C} = \{-x_2 \leq -2, x_2 \leq 2x_1\}$, or
$\mathcal{G} = \{\mathcal{V} = \{(1,2)\}, \mathcal{R} = \{(1,2),(1,0)\}, \mathcal{Z} = \emptyset\}$.

Polyhedron Lattice

- The Polyhedra abstract domain consists of the polyhedra lattice (P, v, t, u, ⊥, >) and a set of operators.
- P is the set of convex closed polyhedra ordered by standard inclusion: ⊑ = ⊆
- The least upper bound (⊔) of two polyhedra P and Q is the convex hull of P and Q, which, in general, is larger than the union P ∪ Q
- The greatest lower bound (⊓) of P and Q is simply the intersection P ∩Q
- Top element ⊤ = $\mathbf{Q^n}$ in the lattice is encoded by C = ∅ or generated by n lines
- The bottom element (⊥) is represented by any unsatisfiable set of constraints in C or with G = ∅.

Polyhedra Operators



- **Inclusion test:** Tests if $P \sqsubseteq Q$ for the given polyhedron P and Q.
- **Equality test:** Tests if two polyhedra P and Q are equal by double inclusion.
- **Join:** Computes $P \sqcup Q$, i.e., the convex hull of P and Q.
- **Meet:** Computes $P \sqcap Q = P \cap Q$.
- **Widening:** As the polyhedra lattice has infinite height, the analysis requires widening to accelerate convergence.
- **Conditional:** Let $\otimes \in \{\leq, =\}$, $1 \leq i \leq n$, and $\alpha \in Q$, the conditional statement $\alpha x_i \otimes \delta$ adds the constraint $(\alpha - a_i)x_i \otimes \delta - a_i x_i$ to the constraint set C.
- **Assignment:** The operator for an assignment $x_i := \delta$ first adds a new variable x'_i to the polyhedron P and then augments C with the constraint $x'_i - \delta = 0$. The variable x_i is then projected out from the constraint set $C \cup \{x'_i - \delta = 0\}$. Finally, the variable x'_i is renamed back to x_i.

Operators and Asymptotic Complexity

- For different operator we use different representation.
- We need conversion between representations → two approaches: lazy and eager.
  - Lazy: when required
  - Eager: after every operation.

Operator	Constraint	Generator	Both
Inclusion ($\sqsubseteq$)	$O(m\,\mathrm{LP}(m,n))$	$O(g\,\mathrm{LP}(g,n))$	$O(ngm)$
Join ($\sqcup$)	$O(nm^{2^{n+1}})$	$O(ng)$	$O(ng)$
Meet ($\sqcap$)	$O(nm)$	$O(ng^{2^{n+1}})$	$O(nm)$
Widening (∇)	$O(m\,\mathrm{LP}(m,n))$	$O(g\,\mathrm{LP}(g,n))$	$O(ngm)$
Conditional	$O(n)$	$O(ng^{2^{n+1}})$	$O(n)$
Assignment	$O(nm^2)$	$O(ng)$	$O(ng)$

Fast Polyhedra Abstract Domain

The Motivation of Fast Polyhedra?

- The famous abstract domain such as Interval and octagon can not able to prove this assertion:
- However, polyhedron can but time and space is very bad and exponential.

```
if(*){y:=2x-1;} else{y:=2x-2;}
assert(y<=2x);
```

What is Fast Polyhedra?

- Speedy Polyhedra domain analysis
- Without lose Precision
- Work with **decomposed polyhedra**

Different Parts of their Approach

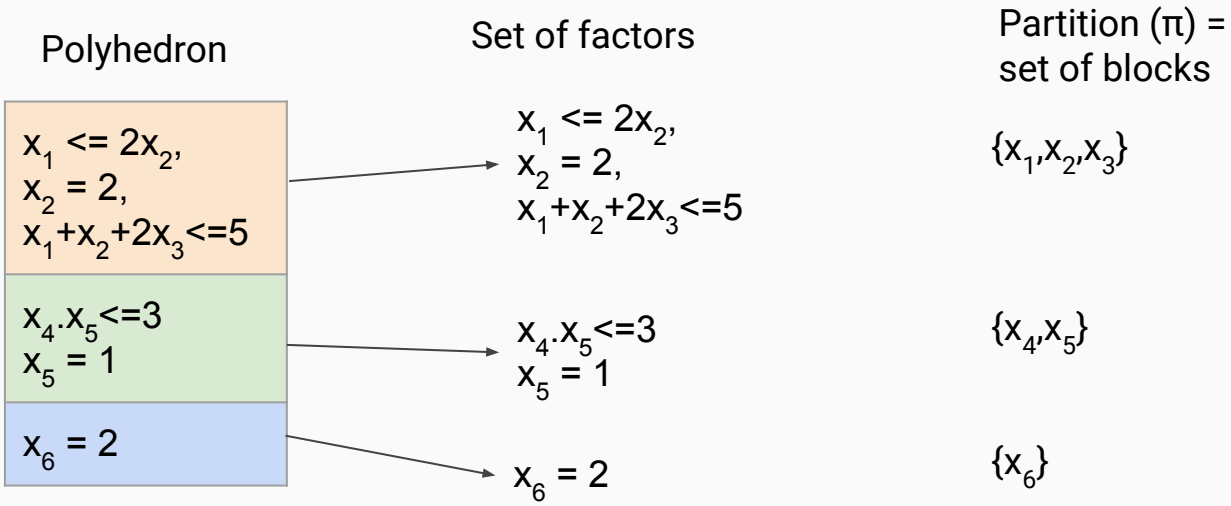
Online Decomposition
Reduction in space and time without losing precision

ELINA
Complete End-to-End implementation and use it for evaluation

Constant Factor Improvement
Via reduce operation count and cache optimization

Evaluation
Effectiveness of our approach showing massive gains in both space and time over state-of-the-art approaches on a large number of benchmarks.

How to Partition?



Fast Polyhedra Operator

- Introduce base Algorithm
- Some optimization for each operator
- An eager approach for the conversion, thus the inputs and the output have both C and G available
- Define operator for best partition and then talk about the permissible partition

Operator: Conditional

Polyhedron [P]	Best Partition (unique)
$x_1 <= 2x_2$, $x_2 = 2$, $x_1+x_2+2x_3<=5$	$\{x_1,x_2,x_3\}$
$x_4.x_5<=3$ $x_5 = 1$	$\{x_4,x_5\}$
$x_6 = 2$	$\{x_6\}$

If ($x_2 >= 2x_5$) →

Polyhedron [O]	Best Partition (unique)
$x_1 <= 2x_2$, $x_2 = 2$, $x_1+x_2+2x_3<=5$ $x_4.x_5<=3$ $x_5 = 1$ $x_2 >= 2x_5$	$\{x_1,x_2,x_3,x_4,x_5\}$
$x_6 = 2$	$\{x_6\}$

Operator: Assignment

Polyhedron [P]	Best Partition (unique)
$x_1 <= 2x_2$, $x_2 = 2$, $x_1+x_2+2x_3<=5$	$\{x_1,x_2,x_3\}$
$x_4.x_5<=3$ $x_5 = 1$	$\{x_4,x_5\}$
$x_6 = 2$	$\{x_6\}$

$x_2 = 2x_4$ →

Polyhedron [O]	Best Partition (unique)
$x_1 <= 4$, $x_1+2x_3<=3$	$\{x_1,x_3\}$
$x_4.x_5<=3$ $x_5 = 1$ $x_2 = 2x_4$	$\{x_2,x_4,x_5\}$
$x_6 = 2$	$\{x_6\}$

Lattice Operators

- **Inclusion(⊑)**: if $P \sqsubseteq Q$ and $P \neq \bot \rightarrow \pi_Q \sqsubseteq \pi_P$
- **Meet (⊓)**: if $P \sqcap Q$ and $Q \neq \bot \rightarrow \pi_O = \pi_Q \sqcap \pi_P$
- **Join:** no general relationship exists between π_O, π_Q, and π_P. It depends on both P and Q.

Lattice Operators: Join

P	Best Partition (unique)
$x_1 - x_2 <= 0$ $x_1 <= 0$	$\{x_1, x_2\}$
$x_3 = 1$	$\{x_3\}$

Q	Best Partition (unique)
$x_1 <= 2,$	$\{x_1\}$
$\emptyset$	$\{x_2\}$
$x_3 = 0$	$\{x_3\}$

$P \sqcup Q$	Best Partition (unique)
$x_1 + 2x_3 <= 2$ $-x_3 <= 0$ $x_3 <= 1$	$\{x_1, x_3\}$
$\emptyset$	$\{x_2\}$

Permissible Partition

Polyhedron	Best Partition (unique)	Permissible Partition	Invalid Partition
$x_1 <= 2x_2$, $x_2 = 2$, $x_1+x_2+2x_3<=5$	$\{x_1,x_2,x_3\}$	$\{x_1,x_2,x_3\}$	$\{x_1,x_2\}$
$x_4.x_5<=3$ $x_5 = 1$	$\{x_4,x_5\}$	$\{x_4,x_5,x_6\}$	$\{x_3,x_4,x_5\}$
$x_6 = 2$	$\{x_6\}$		$\{x_6\}$

Definition: A partition π is permissible for Polyhedron P, if there are no two variables x_i and x_j in different blocks of π related by a constraint in P. They define Operators with Permissible Partition.

Asymptotic Complexity of Operators with Permissible Partitions

Operator	Decomposed
Inclusion ($\sqsubseteq$)	$O(\sum_{i=1}^{r} n_i g_i m_i)$
Join ($\sqcup$)	$O(\sum_{i=1}^{r} n_i g_i m_i + n_{\max} g_{\max})$
Meet ($\sqcap$)	$O(\sum_{i=1}^{r} n_i m_i)$
Widening (∇)	$O(\sum_{i=1}^{r} n_i g_i m_i)$
Conditional	$O(n_{\max})$
Assignment	$O(n_{\max} g_{\max})$

r = number of blocks

Evaluation

- Compared performance of ELINA against NewPolka and PPL
  - http://elina.ethz.ch/
- Time: 4h
- Memory: 12 GB
- More than 1500 benchmarks

Evaluation

Benchmark	Category	LOC	NewPolka		PPL		ELINA		Speedup ELINA vs.	
			time(s)	memory(GB)	time(s)	memory(GB)	time(s)	memory(GB)	NewPolka	PPL
firewire_firedtv	LD	14506	1367	1.7	331	0.9	0.4	0.2	3343	828
net_fddi_skfp	LD	30186	5041	11.2	6142	7.2	9.2	0.9	547	668
mtd_ubi	LD	39334	3633	7	MO	MO	4	0.9	908	>38
usb_core_main0	LD	52152	11084	2.7	4003	1.4	65	2	170	62
tty_synclinkmp	LD	19288	TO	TO	MO	MO	3.4	0.1	>4235	>1186
scsi_advansys	LD	21538	TO	TO	TO	TO	4	0.4	>3600	>3600
staging_vt6656	LD	25340	TO	TO	TO	TO	2	0.4	>7200	>7200
net_ppp	LD	15744	TO	TO	10530	0.15	924	0.3	>16	11.4
p10_l00	CF	592	841	4.2	121	0.9	11	0.8	76	11
p16_l40	CF	1783	MO	MO	MO	MO	11	3	>69	>24
p12_l57	CF	4828	MO	MO	MO	MO	14	0.8	>71	>15
p13_l53	CF	5816	MO	MO	MO	MO	54	2.7	>50	>26
p19_l59	CF	9794	MO	MO	MO	MO	70	1.7	>15	>4
ddv_all	HM	6532	710	1.4	85	0.5	0.05	0.1	12772	1700

Evaluation

Benchmark	Category	LOC	NewPolka		PPL		ELINA		Speedup ELINA vs.	
			time(s)	memory(GB)	time(s)	memory(GB)	time(s)	memory(GB)	NewPolka	PPL
firewire_firedtv	LD	14506	1367	1.7	331	0.9	0.4	0.2	3343	828
net_fddi_skfp	LD	30186	5041	11.2	6142	7.2	9.2	0.9	547	668
mtd_ubi	LD	39334	3633	7	MO	MO	4	0.9	908	>38
usb_core_main0	LD	52152	11084	2.7	4003	1.4	65	2	170	62
tty_synclinkmp	LD	19288	TO	TO	MO	MO	3.4	0.1	>4235	>1186
scsi_advansys	LD	21538	TO	TO	TO	TO	4	0.4	>3600	>3600
staging_vt6656	LD	25340	TO	TO	TO	TO	2	0.4	>7200	>7200
net_ppp	LD	15744	TO	TO	10530	0.15	924	0.3	>16	11.4
p10_l00	CF	592	841	4.2	121	0.9	11	0.8	76	11
p16_l40	CF	1783	MO	MO	MO	MO	11	3	>69	>24
p12_l57	CF	4828	MO	MO	MO	MO	14	0.8	>71	>15
p13_l53	CF	5816	MO	MO	MO	MO	54	2.7	>50	>26
p19_l59	CF	9794	MO	MO	MO	MO	70	1.7	>15	>4
ddv_all	HM	6532	710	1.4	85	0.5	0.05	0.1	12772	1700

Evaluation

Benchmark	Category	LOC	NewPolka		PPL		ELINA		Speedup ELINA vs.	
			time(s)	memory(GB)	time(s)	memory(GB)	time(s)	memory(GB)	NewPolka	PPL
firewire_firedtv	LD	14506	1367	1.7	331	0.9	0.4	0.2	3343	828
net_fddi_skfp	LD	30186	5041	11.2	6142	7.2	9.2	0.9	547	668
mtd_ubi	LD	39334	3633	7	MO	MO	4	0.9	908	>38
usb_core_main0	LD	52152	11084	2.7	4003	1.4	65	2	170	62
tty_synclinkmp	LD	19288	TO	TO	MO	MO	3.4	0.1	>4235	>1186
scsi_advansys	LD	21538	TO	TO	TO	TO	4	0.4	>3600	>3600
staging_vt6656	LD	25340	TO	TO	TO	TO	2	0.4	>7200	>7200
net_ppp	LD	15744	TO	TO	10530	0.15	924	0.3	>16	11.4
p10_l00	CF	592	841	4.2	121	0.9	11	0.8	76	11
p16_l40	CF	1783	MO	MO	MO	MO	11	3	>69	>24
p12_l57	CF	4828	MO	MO	MO	MO	14	0.8	>71	>15
p13_l53	CF	5816	MO	MO	MO	MO	54	2.7	>50	>26
p19_l59	CF	9794	MO	MO	MO	MO	70	1.7	>15	>4
ddv_all	HM	6532	710	1.4	85	0.5	0.05	0.1	12772	1700

Advantages and Disadvantages

- Advantages
  - According to the evaluation, their result was fascinating.
  - Their framework can be used for decomposing other numerical domains, not only Polyhedra.
  - They published their frameworks for us to use

- Disadvantages
  - I would like to know the result what would happen if they increase the memory and time.

Conclusion

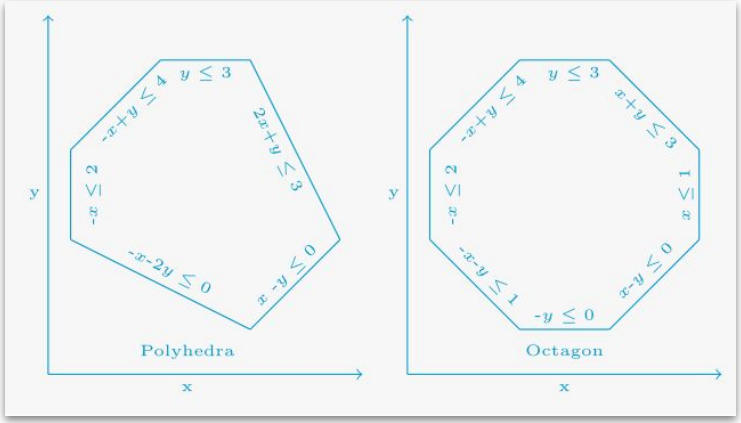
- They decompose a polyhedron to some factors and partition it into
- They define new operators and how they work among different partition
- They got very intriguing result in compare with NewPolka and PPL

Octagon VS. Fast Polyhedra

Compare: Octagon Vs. Fast Polyhedra

- Octagon was not able to prove all cases
- Octagon didn't test on real life programs
- Octagon wouldn't work properly on large scale systems

- Polyhedra abstract domain is precise than octagon
- Polyhedra framework can be used for decomposing other numerical domains, not only Polyhedra
- Polyhedra is more expressive than weakly relational domains such as Octagons



Thank you for your time