CS249, Spring 2023, **Due: 11:59pm, Sunday 06/11, 2023**
Paper Reading
Name: Ashkan Vedadi Gargary **Student ID: 862352576**

---

1 Papers

The Octagon Abstract Domain [2007] and Fast Polyhedra Abstract Domain [2017]

2 Introduction

Abstract Domains are mathematical structures used in program analysis to represent sets of possible program state or behaviors. Numerical Abstract Domains capture relevant properties of program variables with abstracting away the irrelevant details and they are very useful for detecting program error and very important for proving the absence of buffer overflows or division by zero or other numerical properties. On our course, we focus on Interval Abstract Domain. The Interval Abstract Domain support few variables. But it is not so much precise and at first in Octagon Abstract Domain, they have introduce a wider abstract domain which is more precise than Interval. In second paper, they talk about a Polyhedra Abstract Domain and a new algorithm that make it Fast Polyhedra Abstract Domain.

3 Numerical Abstract Domain

Here I want to introduce what is abstract domain and specially Numerical Abstract Domain. Numerical Abstract Domain serves as a foundational concept within the field of program analysis, offering a powerful framework for examining the numerical aspects of computer programs. It facilitates the exploration and understanding of program behavior by abstracting variable and expression values into a more manageable and workable form.

The primary objective of Numerical Abstract Domain is to capture the potential range of values that variables may assume during program execution. By employing abstract mathematical structures such as intervals, polyhedra, or octagons, this technique allows for the representation and approximation of sets of possible values. These abstract representations enable effective analysis and reasoning regarding program properties, including safety, termination, and optimization.

For our numerical abstract domain, program variables are mapped onto domains, which establish a lattice structure. The most used are the lattice of **Intervals** and lattice of **Polyhedra**. The interval is very efficient, linear memory and time cost, but not very precise. Polyhedron is much more precise but has a huge memory cost.

In summary, Numerical Abstract Domain constitutes an essential technique within program analysis, enabling the abstraction and approximation of numerical values within a program. They are very useful for detecting program error and critical program properties and very important for proving the absence of buffer overflows or division by zero or other numerical properties.

4 Interval

It's a out of papers sections. In our lecture, we had learn what is Interval Abstract Domains. It provides a simple and effective way to find the ranges of possible values that variables can take during program

execution. In the Interval Abstract Domain, program variables are abstracted to intervals, which represent ranges of possible values. It means, it can only consider $x \geq a$ and $x \leq b$ (an interval $[a, b]$ represents all values between a and b, including a and b for variable x).

5 Octagon

The first mentioned paper, the authors introduce Octagon Abstract Domain. In our Interval Abstract Domain we only can represent interval for variables. What if we have $x + y \leq 5$ or other related things? The key advantage of using the Octagon Abstract Domain is that it allows for precise reasoning about the relationships between variables.

As same as it's name, it would be like octagon shape. They provide a practical algorithms to represent and manipulate invariant of the form $(\pm x \pm y \leq c)$, where x and y are numerical variables and c is a numeric constant.

5.1 Difference-Bound Matrices

Before starting to get in to the details of how octagon works, let's talk about Difference-Bound Matrices. There are many satisfiability algorithms for set of constraints involving only two variables per constraint have been proposed in order to solve constraint logical programming problems. The simple case of constraint forms is $x - y < c$ and $x < c$. In constraint logical programming we only care about satisfiability but we need manipulate them and apply operators too. Therefore, the model-checking community has developed a practical representation, called Difference-Bound Matrices (DBMs), for constraints of the form $(x - y \leq c)$ and $(\pm x \leq c)$, together with many operators, in order to model-check timed automata.

Difference-Bound Matrices (DBMs) are a powerful abstract domain used in program analysis and verification to reason about the timing and ordering of events in real-time systems. They provide a compact and efficient representation for modeling temporal constraints among variables.

Difference-Bound Matrices are essentially matrices that describe the differences between variable values over time. Each matrix entry represents the maximum allowable time difference between two variables. The diagonal entries correspond to upper bounds on individual variables. By manipulating and analyzing the DBM, it becomes possible to reason about the feasibility of program execution and the satisfaction of timing constraints.

5.2 Motivation

According to the previous details that I provide for numerical abstract domain, we know that the interval abstract domain is very efficient, linear memory and time cost, but **not very precise**. On the other side, polyhedra abstract domain is much more precise but has a **huge memory cost**. Therefore, they try to introduce a abstract domain between these two abstract domain and name it Octagon.

In fact, they use the idea of Difference-Bound Matrices and introduce a way to extend it in order to propose a way to show invariant of the form $(\pm x \pm y \leq c)$, where x and y are numerical variables and c is a numeric constant (Figure 1) .

5.3 Extending Difference Bound Matrices

Here, in this subsection, I want to introduce the way they extend the DBMs to represent Intervals.

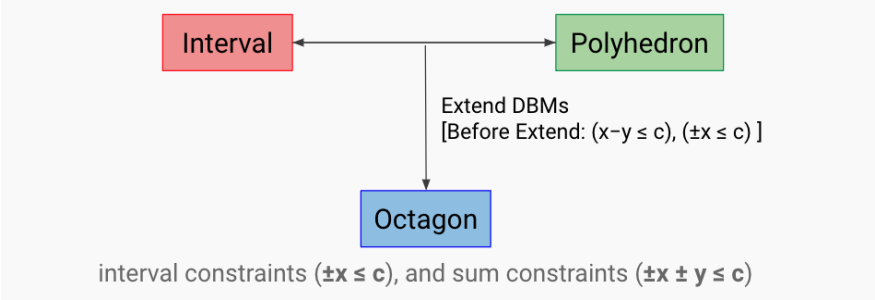

Figure 1: The Motivation

5.3.1 Representing Intervals

By little change in the set of variables we are able to represent our new constraints. If we have a set of Variables $V^0 = \{v_0, v_1, ..., v_{n-1}\}$ and we know that with DBMs we can represent $v_i - v_j \leq c$ and $\pm v_i \leq c$. By adding new constant as a variable to our variable set without any problem we have $V = \{0, v_0, v_1, ..., v_{n-1}\}$.

Now we are able to define $v_i \leq c$ and $v_j \geq d$ with this modification. We need to rewrite them as $v_i - 0 \leq c$ and $0 - v_j \geq -d$ (Figure 2).

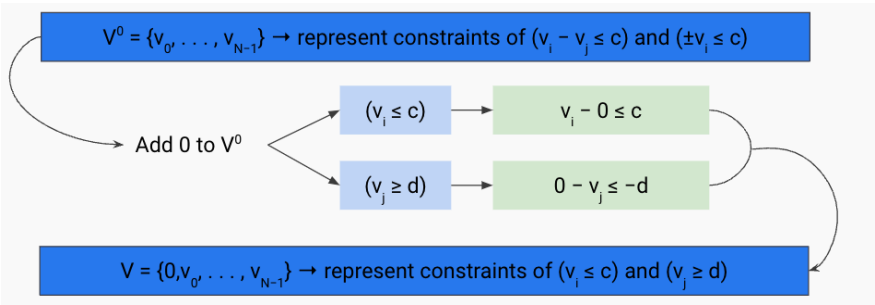

Figure 2: How to Represent Interval with Extended DBMs

5.3.2 Representing Sums

As same as what they have done for the previous representation, they introduce a way to rewrite the variable set. Consider $V^+ = \{v_0, v_1, ..., v_{n-1}\}$ as our program variable set. We know that with DBMs we can represent $v_i - v_j \leq c$. Now, we consider that each variable v_i in V^+ come in two flavours:

- A positive form v_i^+ of v_i
- A negative form v_i^- of v_i

With this we have a new variable set which is $V = \{v_0^+, v_0^-, ..., v_{n-1}^+, v_{n-1}^-\}$ and now we can our new constraint over our set easily which you can see in Figure 3.

5.4 Operators and Transfer Functions

As same as Interval Abstract Analysis, the Octagon Abstract Domain supports various operations that facilitate program analysis. These operations include **intersection**, **union**, **widening**, and **narrowing**. Intersection and union operations allow for combining or splitting octagons, respectively, while widening and

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

Figure 3: Translation between extended constraints over $V^+ and potential constraints over V$

narrowing operations provide a way to refine the octagon representation over iterations of the analysis. Also, they provide information about Equality and Inclusion Testing and Projection.

5.5 Lattice Structure

They provide two different lattice structure. The first one has two problems. First, it is not isomorphic to a sub-lattice of $P(V^+ \to I)$ as two different DBMs can have the same V^+ Domain. Second, the least upper bound operator is not the most precise approximation of the union of two octagon. And to overcome these problems, they describe a Strongly Closed DBMs Lattice. They modify each item attribute of the first complete lattice to force them to be strongly closed.

5.6 Summary and Result

The Octagon Abstract Domain is a unique numerical abstract domain that utilizes octagon-shaped representations to analyze the relationships between program variables. This domain offers a precise and efficient approach to examining programs with linear constraints, making it valuable for various program analysis and verification tasks.

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.

5.6.1 Advantages

- The Octagon Abstract 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
- It's able to provide a precise representation for more complex relational program variables than Interval Abstract Domains.

5.6.2 Disadvantages

- It works only for small test cases and not able to work in large inputs and program with large variables
- They are less precise than the polyhedron analysis
- Their prototype implementation did not allow them to test their domain on real life programs and they still do not know if it will scale up.
- It's not very good with scalability

6 Polyhedra

Another famous abstract domain for finding the relation of program variables is polyhedra abstract domain. In the Polyhedra Abstract Domain, variables and constraints are represented using polyhedra, which are multidimensional geometric shapes bounded by a set of linear inequalities. In other words, It is a fully relational numerical domain and can encode all possible linear constraints between program variables. It is the most precious abstract domain, but also the most costly approach.

6.1 Polyhedra Representation

We have two way to represent polyhedra that they describe in their paper.

- **Constraint Representation (C)**: It's a Intersection of some finite number of closed half space and a finite number of sub spaces.
- **Generator Representation (G)**: It's the convex hull of some finite sets (vertices, rayse, and lines).

6.2 Operators and Asymptotic Complexity

As same as Octagon this abstraction either provide information about lattice and operators. For different Operators and different representations we have different complexity. For some of them, Constraint Representation works better and for the others Generator Representation works better (Figure 4).

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	$O(nm^2)$	$O(ng)$	$O(ng)$

Figure 4: Asymptotic time complexity of Polyhedra operators with different representations

Therefore, we need conversion between representations and we have two approach for it. But, overall this cost is exponential.

- **Lazy:** This approach computes the conversion only when required to amortize the cost over many operations.
- **Eager:** This approach computes the conversion after every operation. The point of eager conversion is that they will save both representaion during the all steps.

7 Fast Polyhedra

7.1 Motivation

The polyhedra that we define previously is the most precise one. The famous abstract domain such as Interval and octagon can not able to prove some assertion ($y \leq 2x$). And the only one that able to handle them is Polyhedra. But, it's very costly specially when the number of program variables increase and take

exponential time. So, the authors of the Fast Polyhedra Abstract Domain try to make a new approach and algorithm that make polyhedra faster. They do this by **decomposed polyhedra** and do the polyhedron for each partition.

Fast Polyhedra Abstract Domain is as same as regular one, but because of decomposing and partitioning, can be faster than before (very impressively).

7.2 How to Partition?

First point that we need demonstrate that is the way to partitioning our polyhedra. Let x be the set of n variables. For a given polyhedron, x can be partitioned into subsets x_k we call blocks such that constraints only exists between variables in the same block. You can see an example of it in Figure 5.

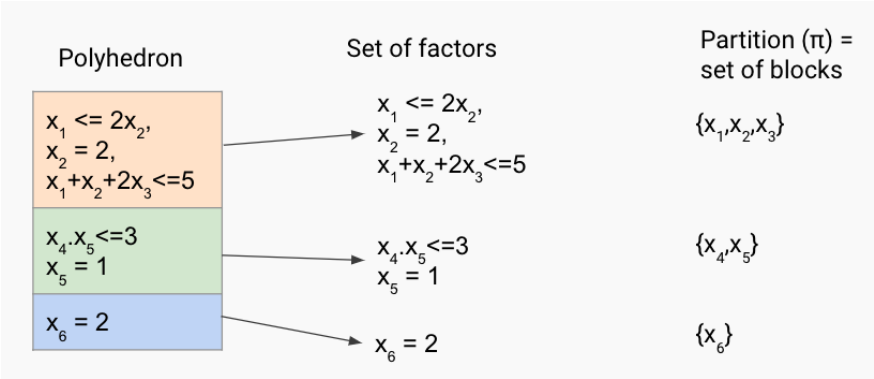

Figure 5: Example of Partitioning

7.3 Fast Polyhedra Opeator

They demonstrate the modified version of each needed operator that the regular polyhedra had for its lattice. They use eager approach for the conversion, this the inputs and the outputs have both C and G available. These operations include intersection, union, widening, and narrowing.

For each operator, they first describe the base algorithm that followed by their adaptation of that algorithm to use partitions. They provide algorithms for different operator and puts them in their paper. I provide some example of some operators (Condition (Figure 6) and Assignment(Figure 7)).

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

Figure 6: Example of Conditional Operator

Also, they define Inclusion and Meet operators.

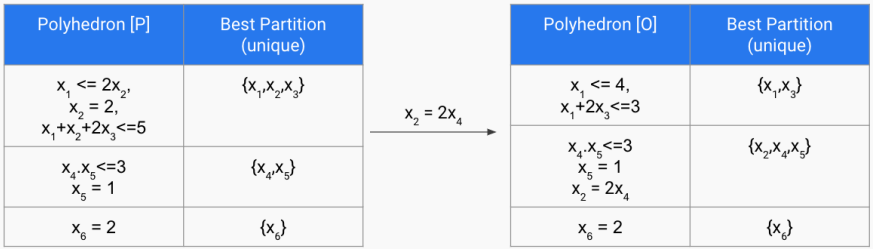

Figure 7: Example of Assignment Operator

- **Inclusion**($\sqsubseteq$): if $P \sqsubseteq Q$ and $P \neq \bot$, then $\pi_Q \sqsubseteq \pi_P$
- **Meet**($\sqcap$): if $P \sqcap Q$ and $Q \neq \bot$, then $\pi_O = \pi_Q \sqcap \pi_P$

But for join there is no general relationship exist between P, Q, and R. and It depends on both P and Q. Join was the most challenging part for them but in the evaluation part you can see how it works properly.

And here is the complexity of operator for decomposed abstract domain mention in Figure 8.

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	$O(n_{\max} g_{\max})$

Figure 8: Asymptotic time complexity of Polyhedra operators with decomposition

7.4 Evaluation

In evaluation part they show how their Implementation (ELINA) and their fascinating result. ELINA uses improved algorithms, online decomposition as well as state of the art performance optimizations from linear algebra such as vectorization, locality of reference, scalar replacement etc. to significantly improve the performance of static analysis with the numerical domains.

For most of the case old approach (PPL and NewPolka) had Time Limit or Memory Limit but ELINA responser very fast and impressive.

7.5 Summary

Fast Polyhedra is a faster version of polyhedra that is very good for large test cases and work properly by looking at the evaluation result. It keeps the precision and works with all assertion.

7.5.1 Advantages

- According to the evaluation, their result was fascinating and it works for many test cases.

- Their framework can be used for decomposing other numerical domains, not only Polyhedra.
- They published their framework (ELINA) for us to use.

7.5.2 Disadvantages

- Complexity of Widening and Narrowing according to the their definition can be too large.

8 Octagon vs. Fast Polyhedra

Here I want to compare Octagon and Fast Polyhedra.

- Octagon was not able to handle all assertions such as $x \leq 2y$.
- Octagon didn't provide tests for real life programs and it wouldn't work properly on large scale systems
- Polyhedra is more precise than Octagon
- Polyhedra is more expressive than weakly relational domains such as Octagons