# Polyhedral Binary Decision Diagrams for Representing Non-Convex Polyhedra

**Shubhang Kulkarni** (University of Illinois, Urbana-Champaign) and Michael Kruse (Argonne National Labs)

IMPACT 2022

# Polyhedral Model

**Simple Program**

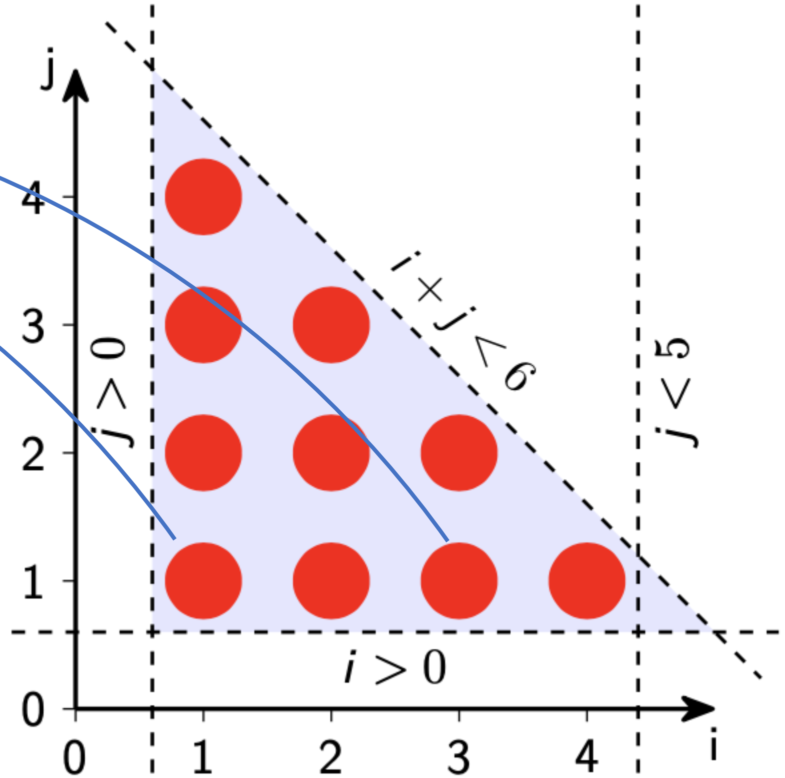
```
for (int i=1; i<5; i++)
  for (int j=1; i+j<6; j++)
    S(i,j);
```

**Statement Domain (Explicit Enumeration)**

$S(1,1)$, $S(1,2)$, $S(1,3)$, $S(1,4)$, $S(2,1)$, $S(2,2)$, $S(2,3)$,
$S(3,1)$, $S(3,2)$, $S(4,1)$

**Statement Domain (Compact Representation)**

$$\{S(i,j) \mid 0 < i,j \wedge i+j < 6\}$$




**Statement Domain Visualization**

# Polyhedral Compilation

Use the polyhedral model for:

- ❖ Program Analysis
  - ❖ Check the legality of a program transform
  - ❖ E.g. Loop interchange
  - ❖ Eventually need to solve a system of linear/affine inequalities
- ❖ Statement Scheduling
  - ❖ Form a "better" schedule according to some objective
  - ❖ E.g. parallelism, data locality, memory access
  - ❖ Eventually need to solve an Integer Linear Program (IntLP)

# Convexity, A Crucial Requirement

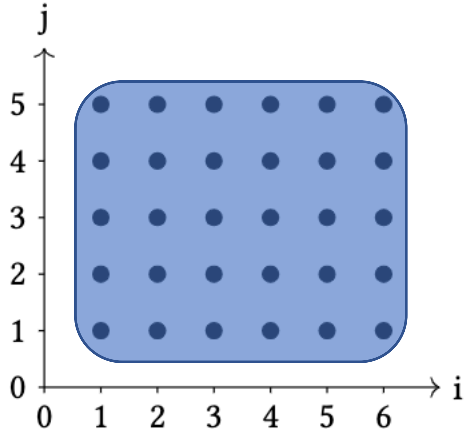
- Statement domain must be convex for compact polyhedral representation
- Does not hold for programs with conditional statements

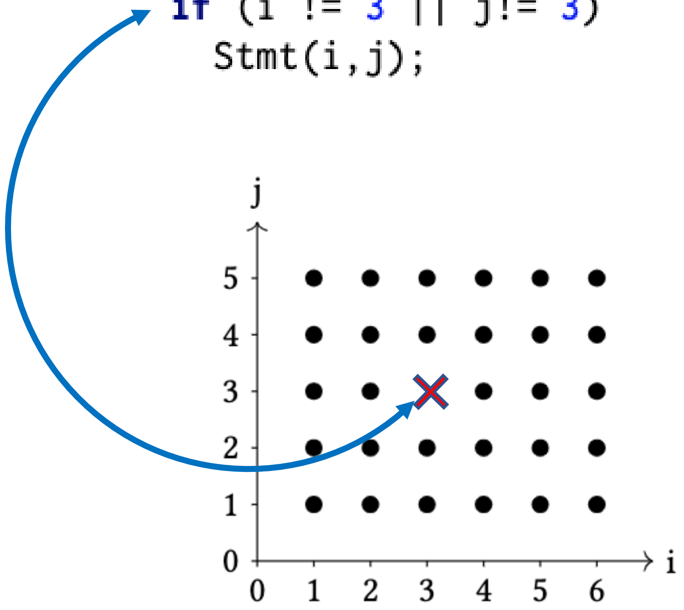
**Programs**

```
for (int i = 1; i <= 6; ++i)
  for (int j = 1; j <= 5; ++j)
     Stmt(i,j);
```

```
for (int i = 1; i <= 6; ++i)
  for (int j = 1; j <= 5; ++j)
    if (i != 3 || j!= 3)
      Stmt(i,j);
```

**Statement Domains**

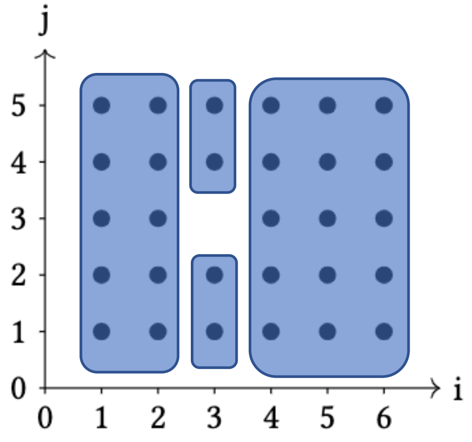


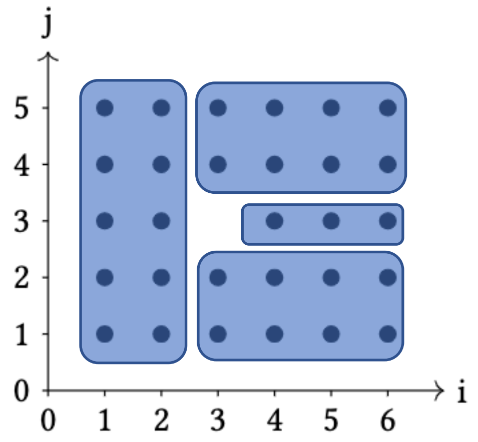



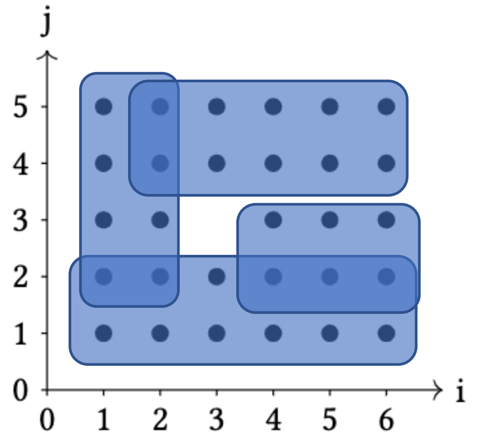
# Union-of-Convex-Polyhedra

- Non-convex vector sets are represented as <u>**union of convex polyhedra**</u>

```
for (int i = 1; i <= 6; ++i)
  for (int j = 1; j <= 5; ++j)
    if (i != 3 || j!= 3)
      Stmt(i,j);
```



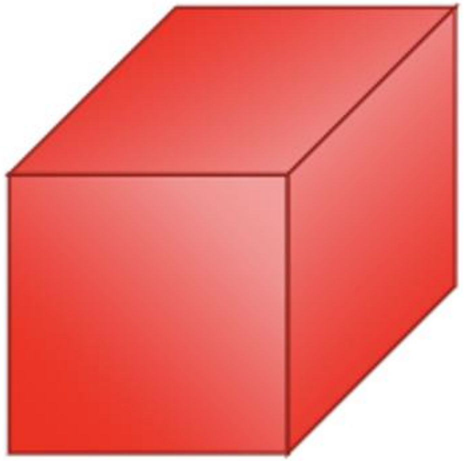



# Motivating Example

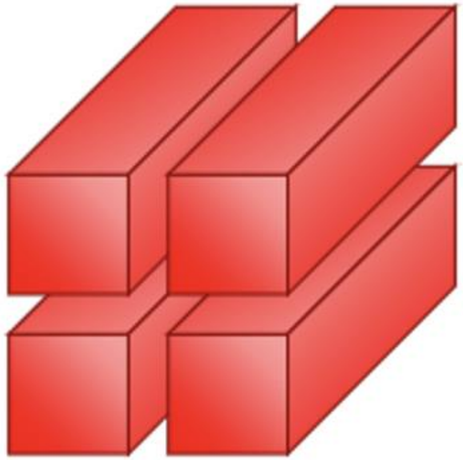
❖Conditionals introduce non-affine constraints (eg non-equality), which leads to splitting of iteration domain

```
for (int i = 0; i < n; i+=1) {
    for (int j = 0; j < n; j+=1) {
        for (int k = 0; k < n; k+=1) {
          Stmt(i, j, k);
        }
    }
}
```

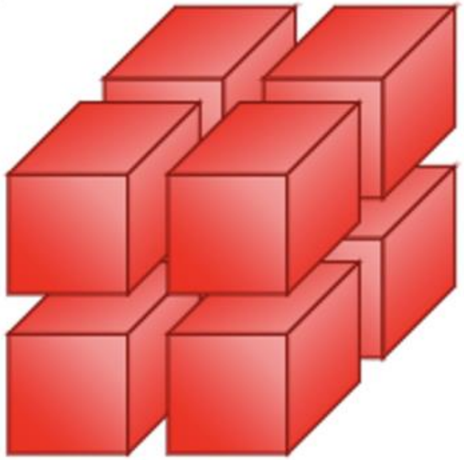


```
for (int i = 0; i < n; i+=1) {
    for (int j = 0; j < n; j+=1) {
        for (int k = 0; k < n; k+=1) {
          if (i == p0)
            continue;
          Stmt(i, j, k);
        }
    }
}
```

```
for (int i = 0; i < n; i+=1) {
    for (int j = 0; j < n; j+=1) {
        for (int k = 0; k < n; k+=1) {
          if (i == p0)
            continue;
          if (j == p1)
            continue;
          Stmt(i, j, k);
        }
    }
}
```



```
for (int i = 0; i < n; i+=1) {
    for (int j = 0; j < n; j+=1) {
        for (int k = 0; k < n; k+=1) {
          if (i == p0)
            continue;
          if (j == p1)
            continue;
          if (k == p2)
            continue;
          Stmt(i, j, k);
        }
    }
}
```



# Polly

- The previous example shows that with $p$ conditional statements, the union-of-convex-polyhedra representation requires $\Omega(2^p)$ polyhedra!
- Infeasible to even write down for polyhedral optimizers even when $p = 20$
- Polly is LLVM’s polyhedral optimizer
- A typical execution of Polly involves
  - Representing programs via polyhedral model
  - Performing several set operations
  - Solving IntLPs

# Integer Set Library (ISL)

- Polly uses Integer Set Library (ISL) to handle polyhedral computations

- ISL uses union-of-convex-polyhedral representation

**Problem:** Polly terminates without program-optimizations when programs have conditional-structure similar to that of motivating example.

**Question:** Is there an alternative representation that avoids this blowup?

**YES!**

# Our Results

- Polyhedral Binary Decision Diagrams (PBDDs) as an alternative representation for non-convex vector sets.
- Proof of concept implementation (Python)
- Case studies comparing scaling behavior of common set operations (needed by Polly) for PBDDs vs ISL’s union-of convex-polyhedra representation

# Our Results – Remarks

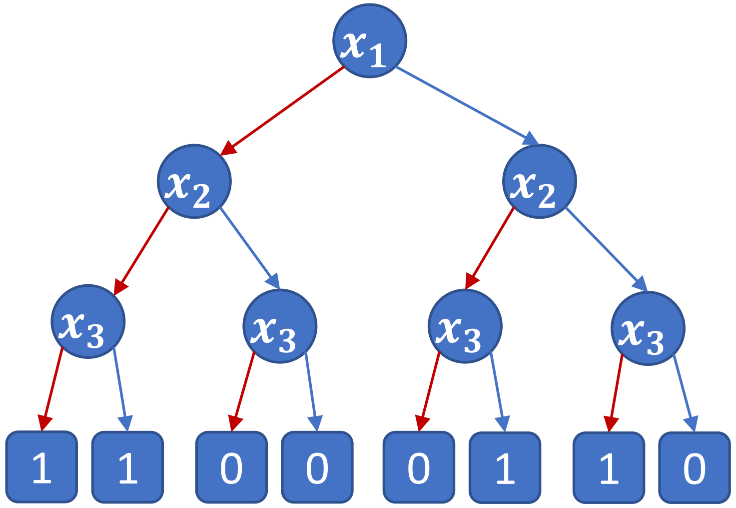
- Polyhedral Binary Decision Diagrams (PBDDs) as an alternative representation for non-convex polyhedral
  - We introduce PBDDs to make representing and performing set operations on non-convex sets feasible
  - Motivating example has a linear size PBDD representation
  - Several set-operations become computationally/algorithmically simple
  - Particularly useful when these operations result in simple sets, but Polly currently terminates prematurely due to representational overhead.
  - **Open Question**: Adapt PBDDs to work with IntLP solvers.

# Binary Decision Diagrams (BDDs)

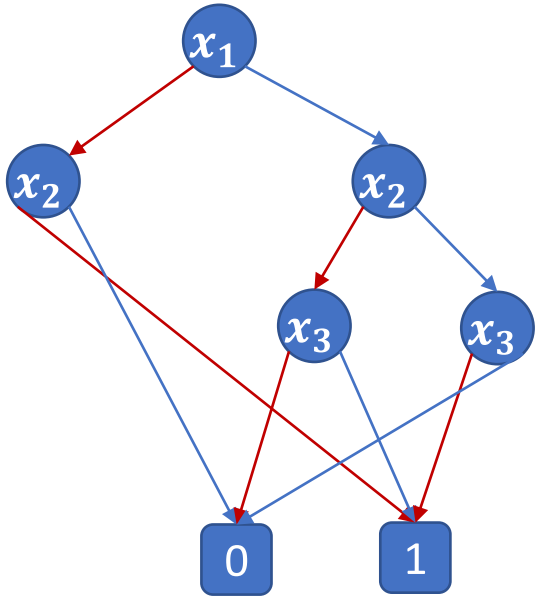
❖ Binary decision diagrams (BDDs) are used to represent Boolean functions

| $x_1$ | $x_2$ | $x_3$ | $f(x_1, x_2, x_3)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Truth Table**




**Binary Decision Tree**




**Binary Decision Diagram**

Function Evaluation Examples:

$f(0,1,0) = 0$

$f(1,1,0) = 1$

# Polyhedral Binary Decision Diagrams (PBDDs)

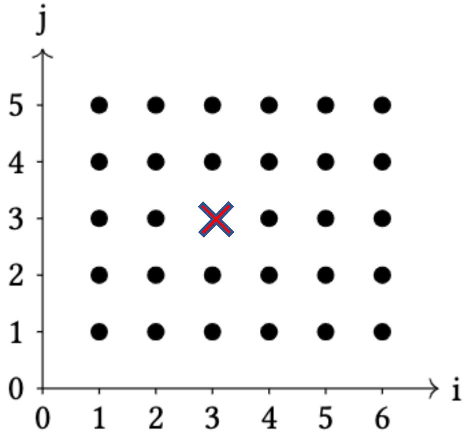
- In PBDDs, we consider the Boolean function being evaluated as the indicator of whether a point is in the polyhedral set.
- Variables correspond to constraints
- Point in iteration space corresponds to indicator vector of which constraints the point satisfies
- Function evaluation is same as in the case of BDDs

# **Polyhedral** Binary Decision Diagrams (**P**BDDs)

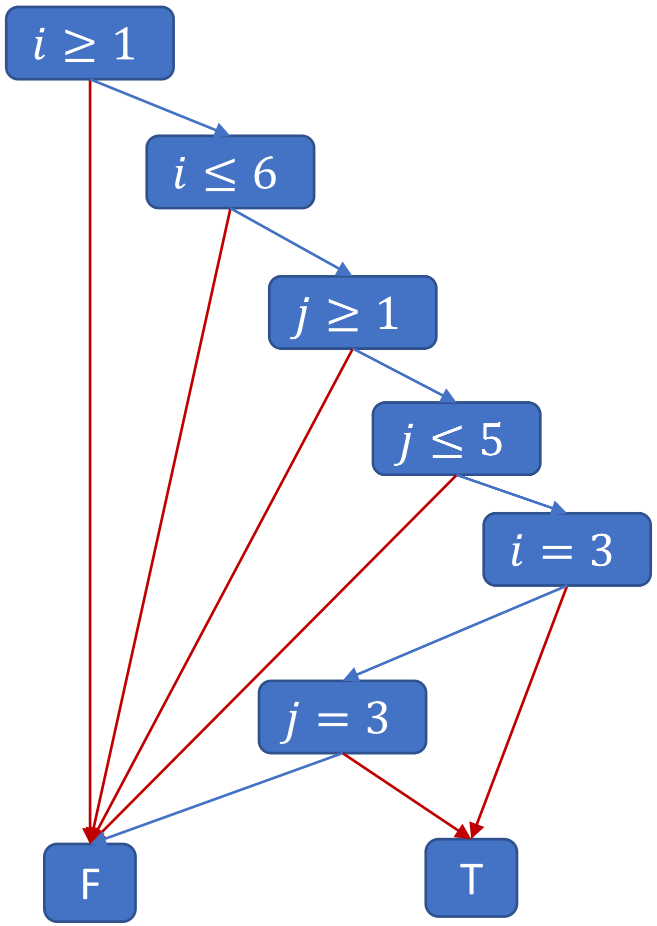
**<u>Program</u>** (recall)

```
for (int i = 1; i <= 6; ++i)
  for (int j = 1; j <= 5; ++j)
    if (i != 3 || j!= 3)
      Stmt(i,j);
```

**<u>Statement Domain</u>** (recall)




**<u>PBDD Representation</u>**




Example Evaluations:

$(0,1) \rightarrow F$

$(2,4) \rightarrow T$

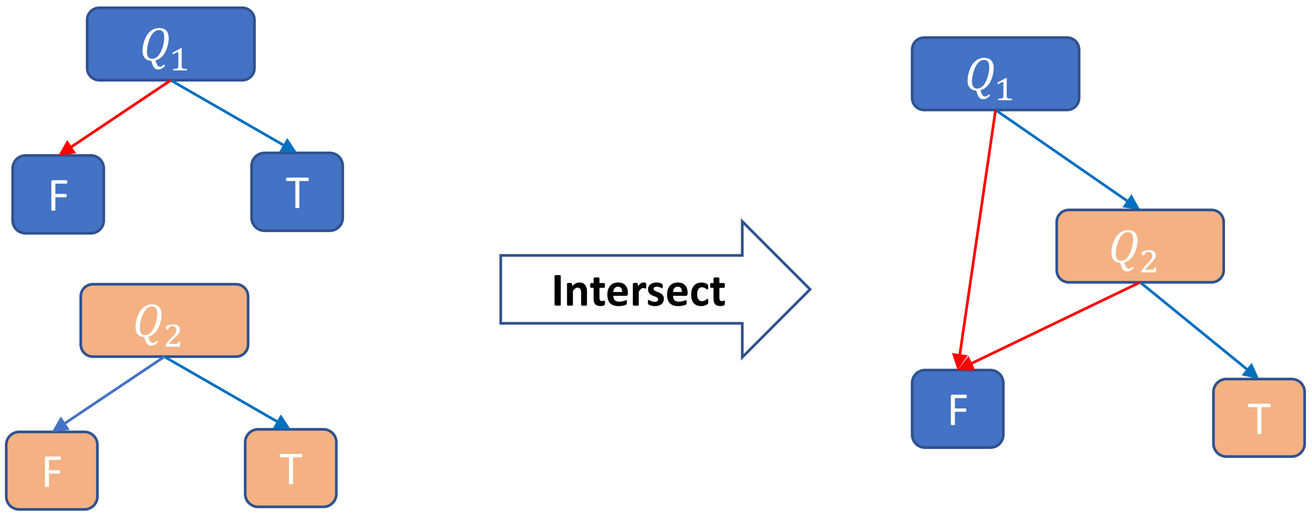
$(3,3) \rightarrow F$

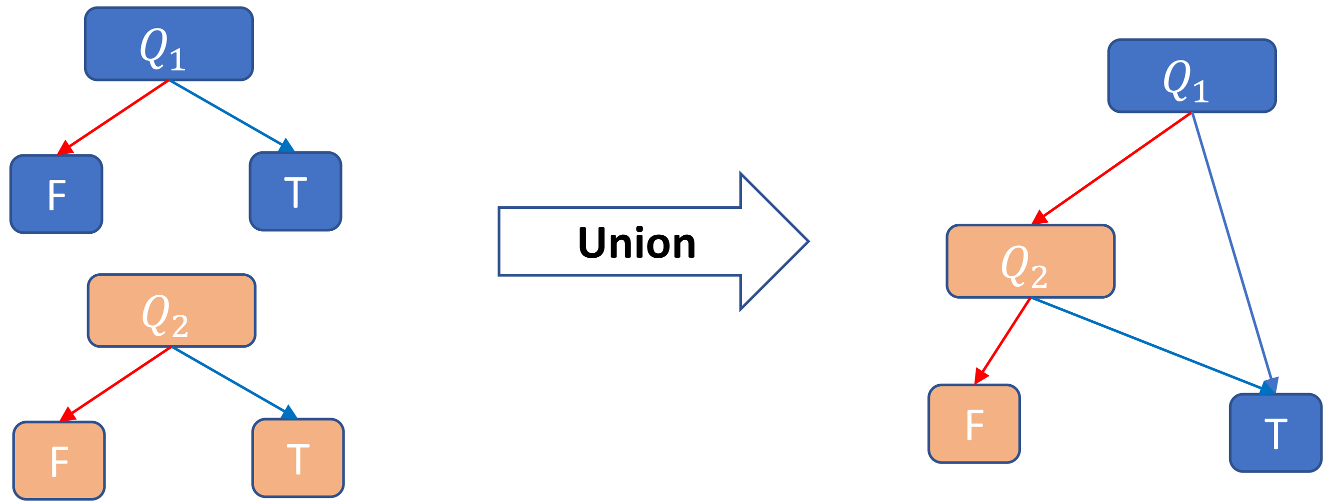
# Distinctions from BDDs and QuASTs

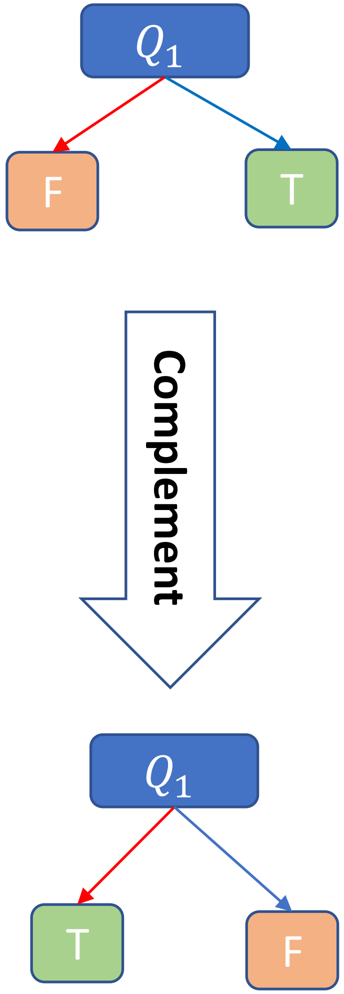
- PBDD vs BDD
  - An assumption of BDDs is that all the input variables are independent
  - For PBDDs that have not been simplified, conflicting constraints can happen
  - We allow non-simplified PBDDs as simplification steps often computationally intensive

- PBDD vs QuAST
  - Quasi-Affine Solution Tree
  - Used to describe the piece-wise defined solution of the lexicographic minimum of a parametric Z-polyhedron.
  - In contrast to PBDD, it is defined by a context-free grammar.
  - QuASTs are always trees instead of DAGs.
  - QuAST is vector-valued whereas PBDD is Boolean-valued (indicator function).

# Simple Set Operations




**Can be implemented to be extremely efficient!!**

# Simple Set Operations

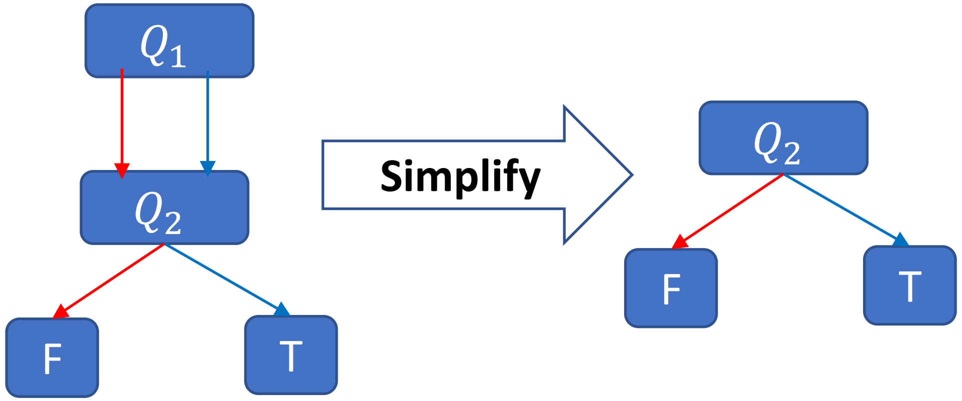
- Main paper also defines and gives algorithms for various set operations:
  - Emptiness check
  - Subset check
  - Subtraction
  - Project Out

- Recursion + Memoization

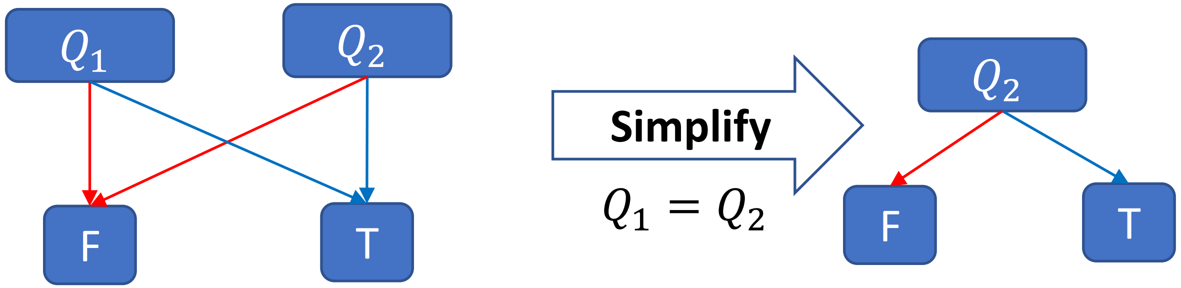
**Note:** Algorithms are recursive as our Python PoC implementation of a PBDD is inherently recursive.

# Simplification Operations

- We use structural simplifications in order to control the sizes of PBDDs
- Conceptually simple but lead to big speedups
- Correctness follows from *Shannon Expansion* (see main paper)
- Experiments show that these are necessary to make representations tractable.



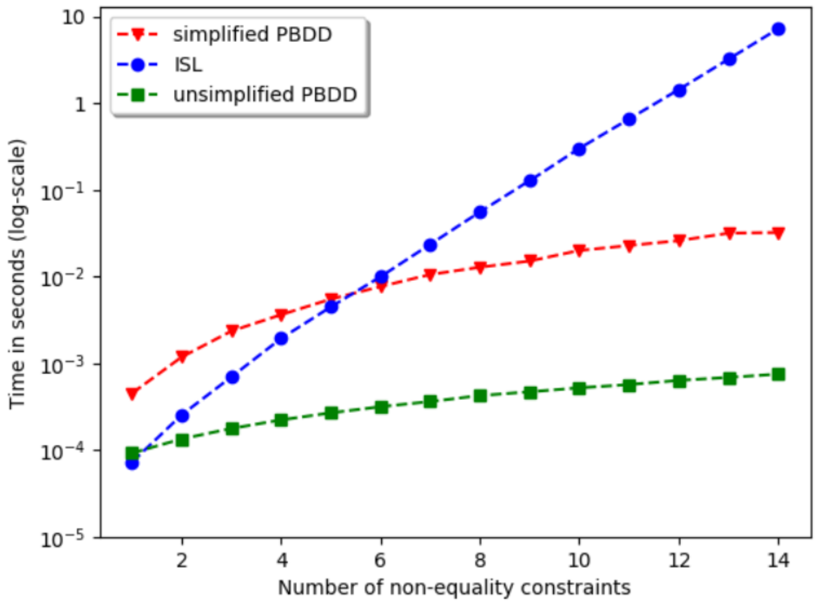

**Redundant Nodes**



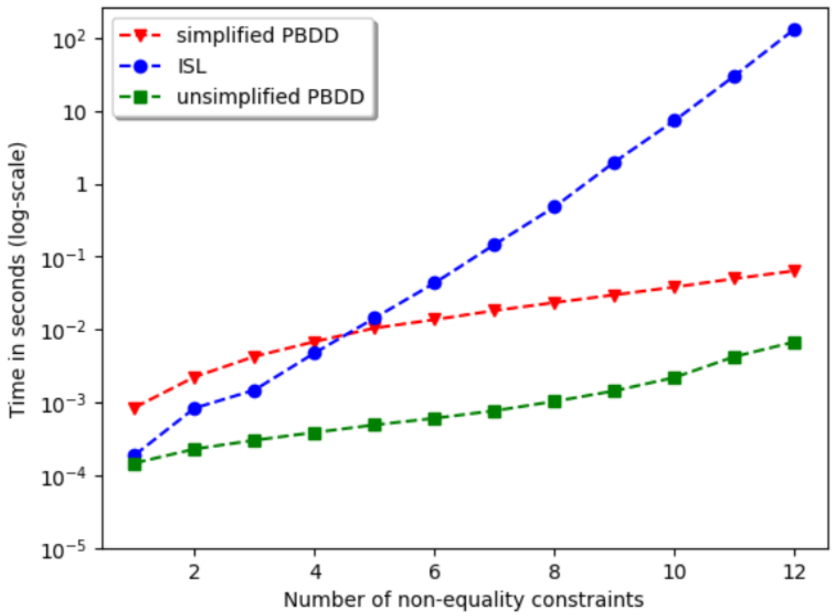

**Isomorphic Nodes**

# Experiments I

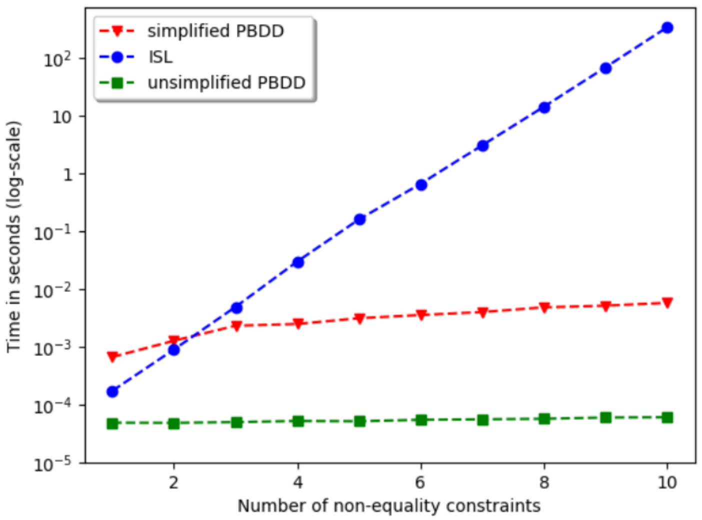
(1) Input: $k$, constraints $c_i$
(2) $P := \texttt{Universe}$
(3) for $i = 1 \ldots k$
  (a) $P = \texttt{intersect}\,(P, \texttt{complement}(c_i))$
(4) Output $P$



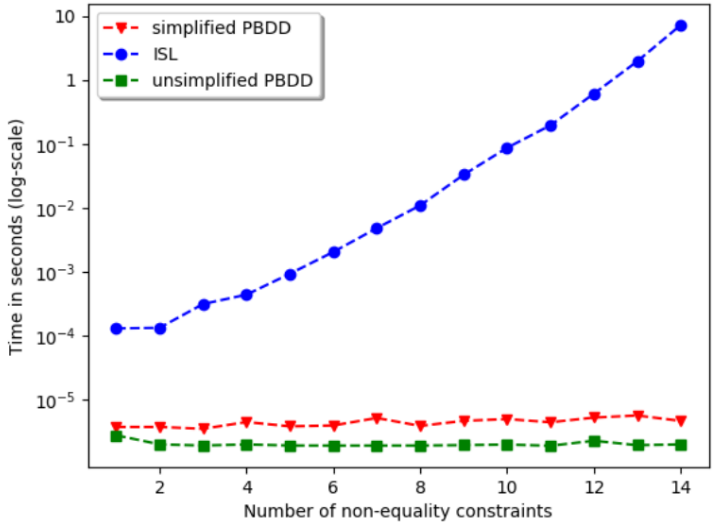

(1) Input: $k$, constraints $c_i$
(2) $P := \texttt{Universe}$
(3) for $i = 1 \ldots k$ :
  (a) $P = \texttt{subtract}\,(P, \texttt{intersect}(c_i, P))$
(4) Output $P$



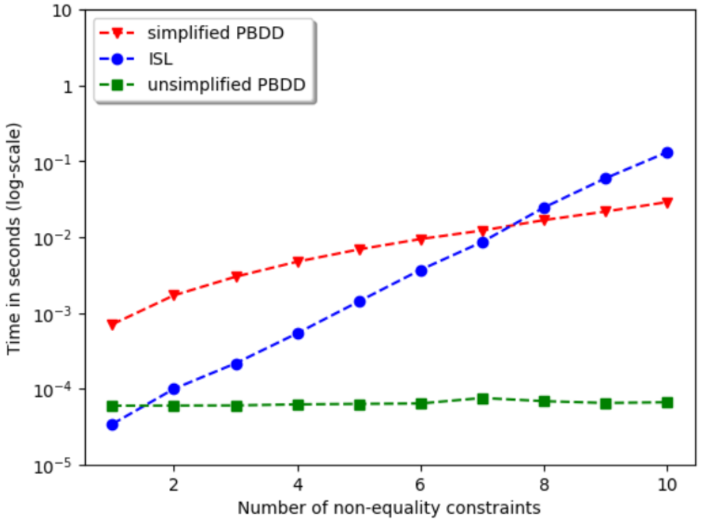
# Experiments II



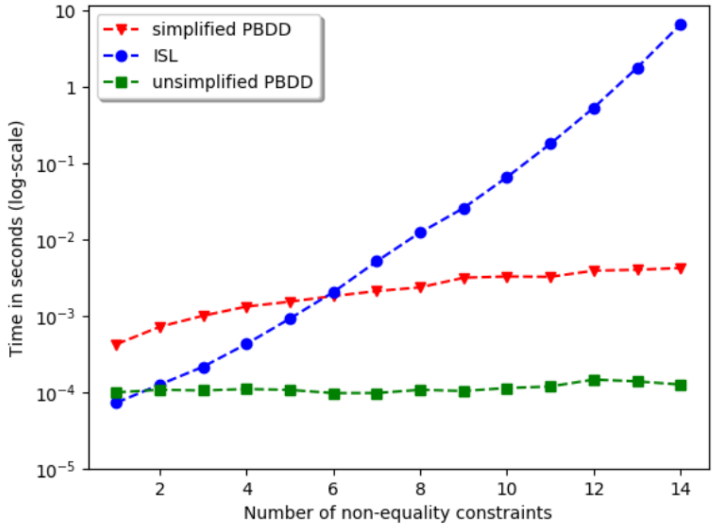

**(a) Non-convex Intersection**




**(c) Non-convex Complement**

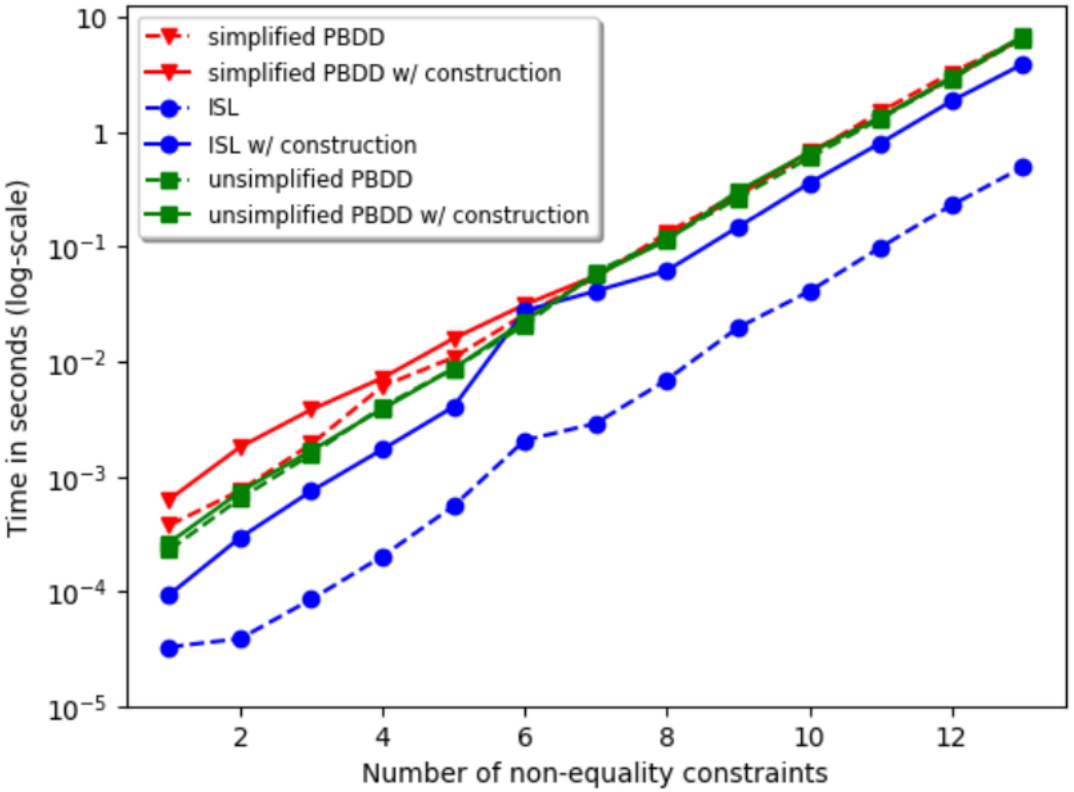



**(b) Non-convex Union**

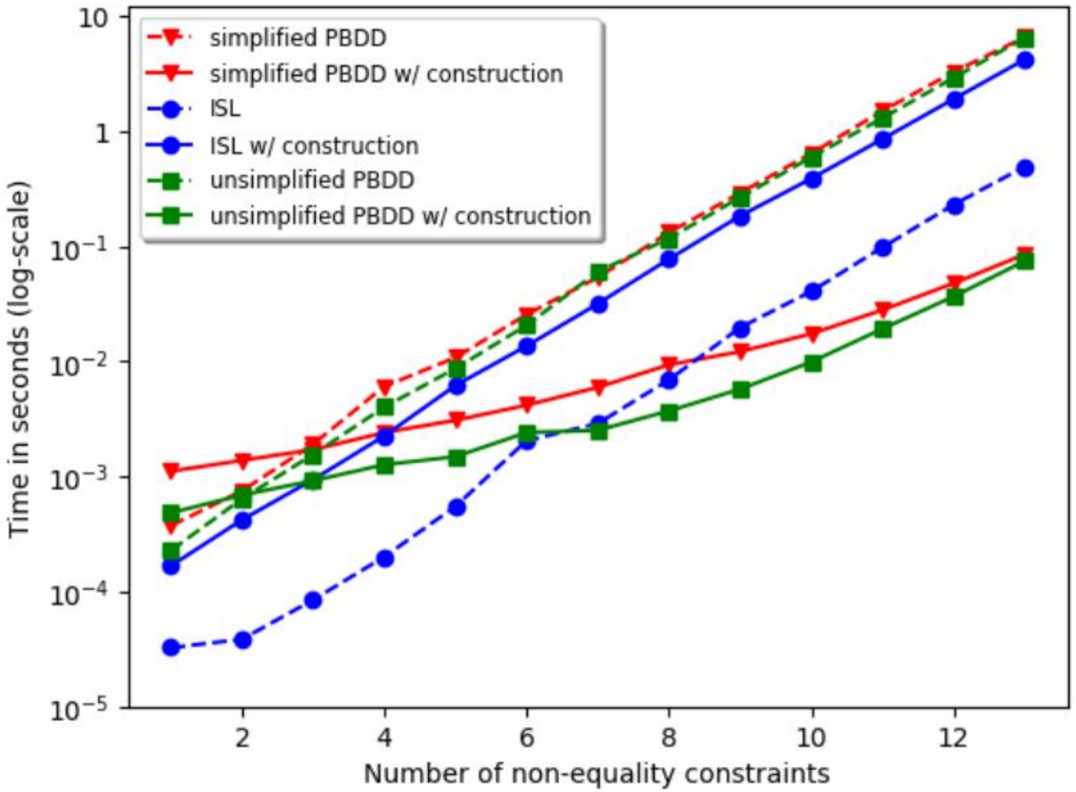



**(d) Non-convex Subtraction**

# Experiments III




**(a) Non-convex project-out**




**(b) Non-convex emptiness check**

# Future Work

- ❖Implementation that does not rely on ISL intermediate data representation
- ❖Parallelism
- ❖Operational Efficiency
- ❖Simplifications
- ❖Decision Orderings
- ❖Approximations
- ❖Typed Roots
- ❖Adapt PBDDs for ILP solvers

**See main paper for discussions on each!**