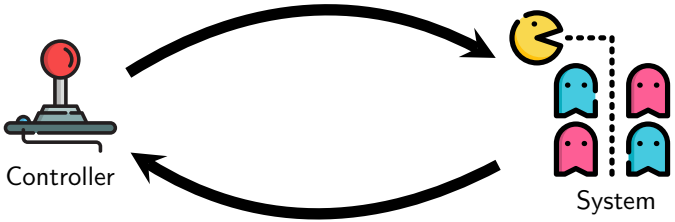
# Explaining Control Policies through Predicate Decision Diagrams

Debraj Chakraborty[1] Clemens Dubslaff[2] Sudeep Kanav[1]
Jan Křetínský[1,3] Christoph Weinhuber[4]

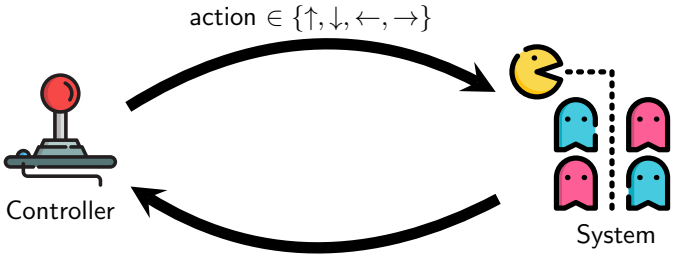
1-Masaryk University, Czech Republic
2-Eindhoven University of Technology, The Netherlands
3-Technical University of Munich, Germany
4-University of Oxford, United Kingdom

Hybrid Systems: Computation and Control
Irvine, California, May 2025

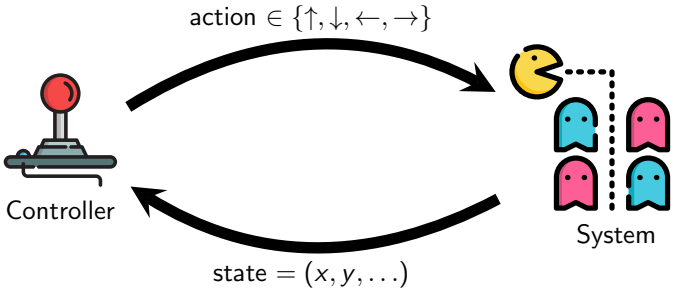
# Controller



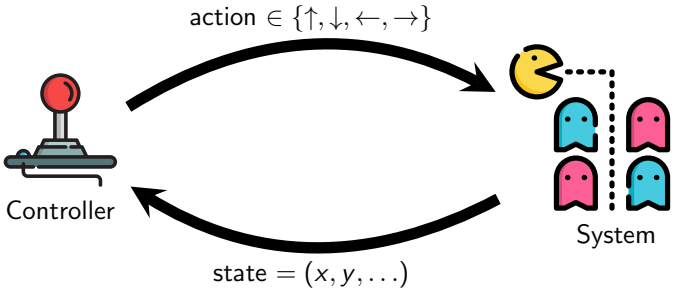
# Controller


action $\in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$
Controller
System

# Controller


action $\in \{\uparrow, \downarrow, \leftarrow, \rightarrow\}$
Controller
System
state $= (x, y, \ldots)$

# Controller




- Control Policy : States $\rightarrow$ Actions

# Control Policy

- Control Policy : States $\rightarrow$ Actions

# Control Policy

- Control Policy : States $\rightarrow$ Actions
- Hard to construct manually

# Control Policy

- Control Policy : States $\rightarrow$ Actions
- Hard to construct manually
- Automated synthesis creates huge lookup tables

```
⋮

57048 0 1 2 3 4 5 6 7 8
57049 0 1 3 4 6 7
57050 0 1 3 4 6 7
57054 2 5
57055 1 2 4 5
57056 1 2 4 5
57057 0 1 2 3 4 5
57058 0 1 2 3 4 5
57059 0 1 2 3 4 5
57063 1 2 4 5
57064 1 2 4 5
57065 0 1 2 3 4 5

⋮
```

# Control Policy

- Control Policy : States $\rightarrow$ Actions
- Hard to construct manually
- Automated synthesis creates huge lookup tables
- Lacks explainability

⋮

```
57048 0 1 2 3 4 5 6 7 8
57049 0 1 3 4 6 7
57050 0 1 3 4 6 7
57054 2 5
57055 1 2 4 5
57056 1 2 4 5
57057 0 1 2 3 4 5
57058 0 1 2 3 4 5
57059 0 1 2 3 4 5
57063 1 2 4 5
57064 1 2 4 5
57065 0 1 2 3 4 5
```

⋮

# Explainable Control Policy

# Explainable Control Policy

- States are defined by multiple state variables

# Explainable Control Policy

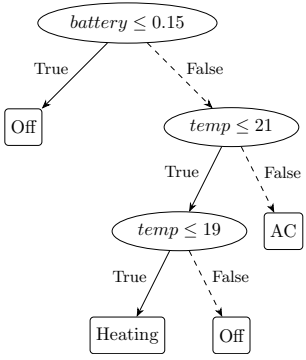
- States are defined by multiple state variables
- Predicates: comparison between variables and values
  - **Axis-aligned**: $x \geq 2$
  - **Linear**: $x + y < 7$
  - **Categorical**: colour $=$ red

# Explainable Control Policy

- States are defined by multiple state variables
- Predicates: comparison between variables and values
  - **Axis-aligned**: $x \geq 2$
  - **Linear**: $x + y < 7$
  - **Categorical**: colour $=$ red
- Decision Trees: Explainability through predicates

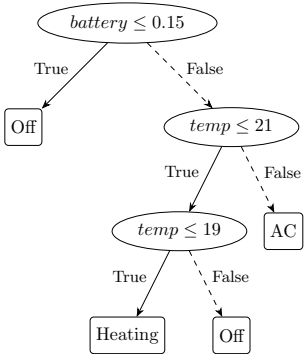
# Decistion Trees (DT)

- Binary tree
- Internal nodes = predicates
- Branches = True/False outcome of that predicate
- Leaf nodes = actions



# Decistion Trees (DT)

- Binary tree
- Internal nodes = predicates
- Branches = True/False outcome of that predicate
- Leaf nodes = actions

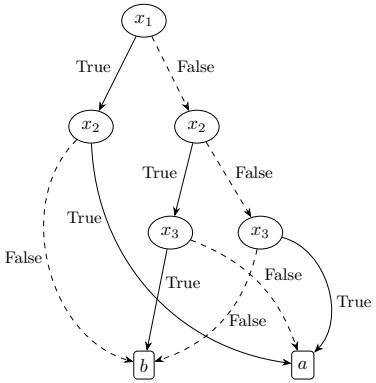



✓ Easy to visualize and interpret

✗ Often redundant due to repeated subtrees

# Binary Decision Diagrams (BDD)

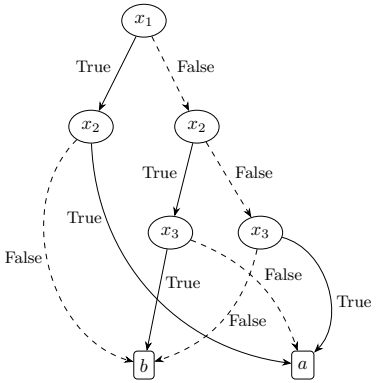
- Directed acyclic graph
- Internal nodes = Boolean variables
- Multi-terminal BDD: Leaf nodes = actions
- Reduced BDD: No redundant nodes
- Ordered BDD: Fixed variable order



# Binary Decision Diagrams (BDD)

- Directed acyclic graph
- Internal nodes = Boolean variables
- Multi-terminal BDD: Leaf nodes = actions
- Reduced BDD: No redundant nodes
- Ordered BDD: Fixed variable order

✓ Compact representation



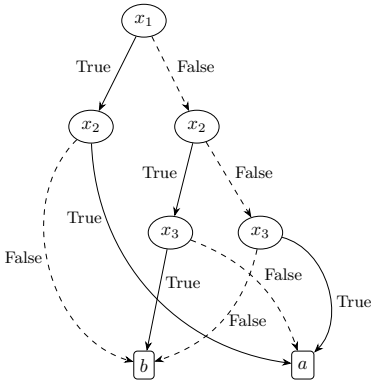
# Binary Decision Diagrams (BDD)

- Directed acyclic graph
- Internal nodes = Boolean variables
- Multi-terminal BDD: Leaf nodes = actions
- Reduced BDD: No redundant nodes
- Ordered BDD: Fixed variable order

✓ Compact representation

✗ Non-Boolean variables require bit-blasting

✗ Direct human interpretation is difficult



# Explainable Control Policy

**Decision Trees**

- ✓ Easy to visualize and interpret
- ✗ Not suitable for symbolic operations
- ✗ Often redundant due to repeated subtrees

**Binary Decision Diagrams**

- ✗ Direct human interpretation is difficult
- ✓ Supports BDD-based algorithms
- ✗ Non-Boolean variables require bit-blasting

# Explainable Control Policy

**Decision Trees**

- ✓ Easy to visualize and interpret
- ✗ Not suitable for symbolic operations
- ✗ Often redundant due to repeated subtrees

**Binary Decision Diagrams**

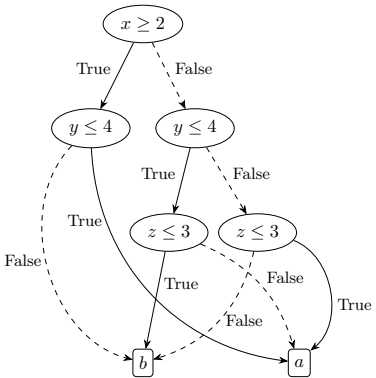
- ✗ Direct human interpretation is difficult
- ✓ Supports BDD-based algorithms
- ✗ Non-Boolean variables require bit-blasting

🎯 **Our goal** : Find a representation that is:

- ✓ Uses predicates over variables like DTs $\rightsquigarrow$ explainable
- ✓ Supports BDD-based algorithms
- ✓ Avoids bit-blasting; stays at semantic predicate level

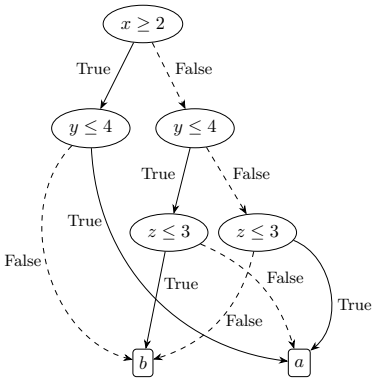
# Predicate Decision Diagrams (PDD)

- Extends BDDs with predicates



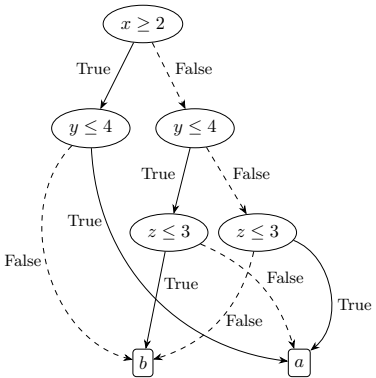
# Predicate Decision Diagrams (PDD)

- Extends BDDs with predicates
- Internal nodes = predicates (similar to DTs)



# Predicate Decision Diagrams (PDD)

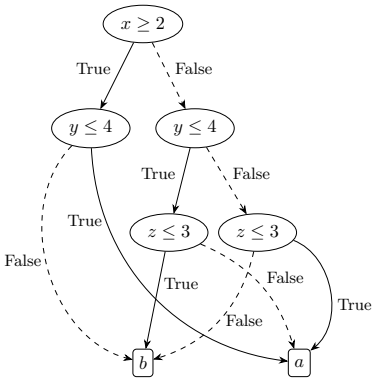
- Extends BDDs with predicates
- Internal nodes = predicates (similar to DTs)
- Leaf nodes = actions



# Predicate Decision Diagrams (PDD)

- Extends BDDs with predicates
- Internal nodes = predicates (similar to DTs)
- Leaf nodes = actions

- ✓ Merges isomorphic subgraphs, reducing redundancy
- ✓ Compact representation (similar to BDDs)
- ✓ Avoids bit-blasting: operate directly on predicates over state variables

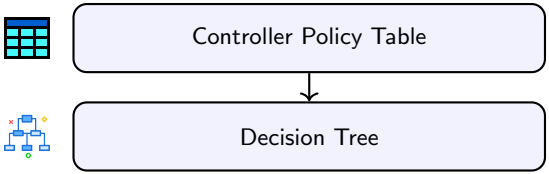


# How to Synthesize a PDD?

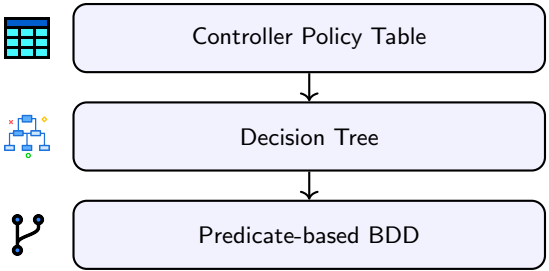
# How to Synthesize a PDD?

Controller Policy Table

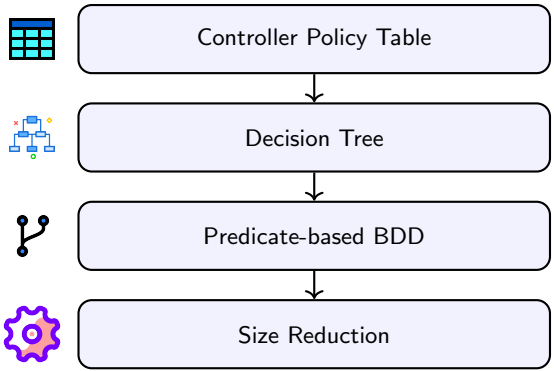
# How to Synthesize a PDD?



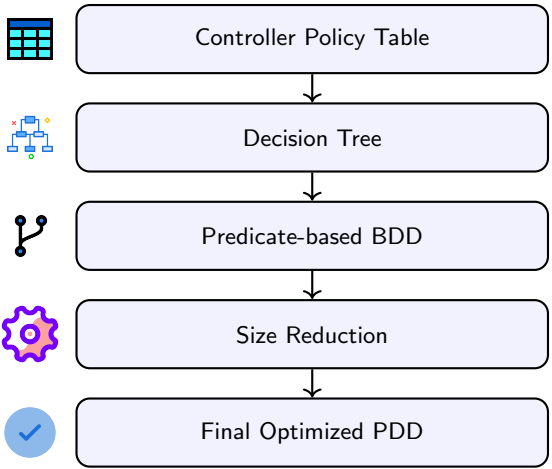
# How to Synthesize a PDD?



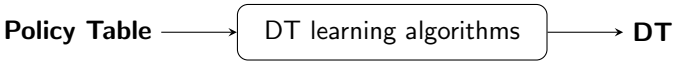
# How to Synthesize a PDD?



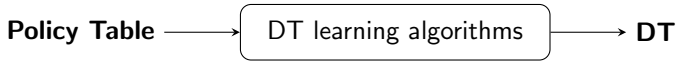
# How to Synthesize a PDD?



# Phase 1 : Learning a DT

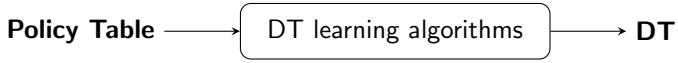


# Phase 1 : Learning a DT

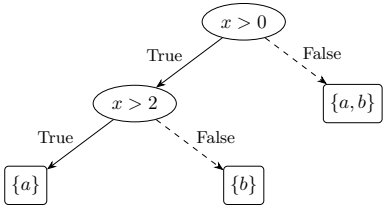



| $x$ | $y$ | actions |
|---|---|---|
| 0 | 0 | $\{a, b\}$ |
| 1 | 0 | $\{b\}$ |
| 2 | 0 | $\{b\}$ |
| 3 | 1 | $\{a\}$ |

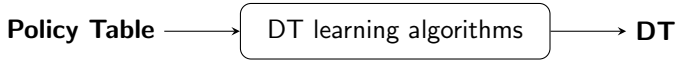
# Phase 1 : Learning a DT



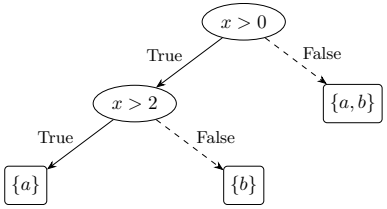

| $x$ | $y$ | actions |
|---|---|---|
| 0 | 0 | $\{a, b\}$ |
| 1 | 0 | $\{b\}$ |
| 2 | 0 | $\{b\}$ |
| 3 | 1 | $\{a\}$ |



# Phase 1 : Learning a DT



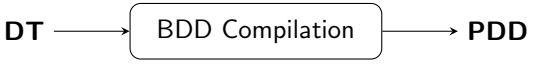

| $x$ | $y$ | actions |
|---|---|---|
| 0 | 0 | $\{a, b\}$ |
| 1 | 0 | $\{b\}$ |
| 2 | 0 | $\{b\}$ |
| 3 | 1 | $\{a\}$ |

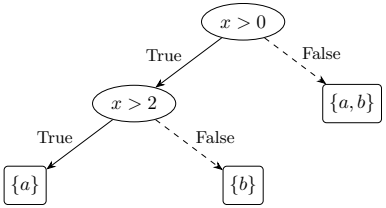



- Learn DT from the policy
- No early stopping : completely capture the policy

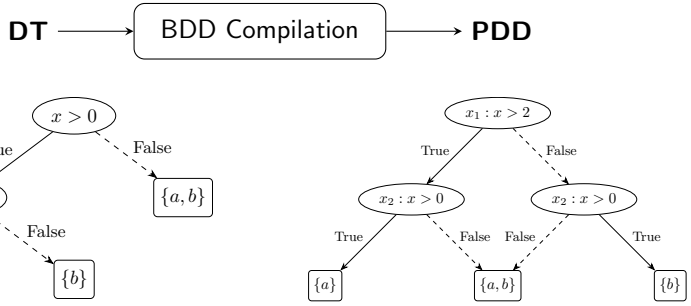
# Phase 2 : Compiling to a BDD with Predicate Labels

# Phase 2 : Compiling to a BDD with Predicate Labels



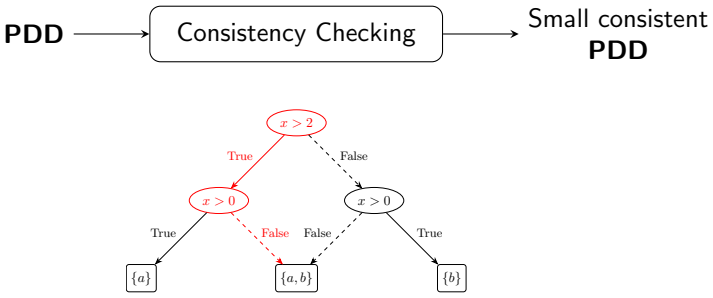
# Phase 2 : Compiling to a BDD with Predicate Labels




- Each predicate corresponds to a Boolean variable
- Reduce and merge to get canonical ROBDD

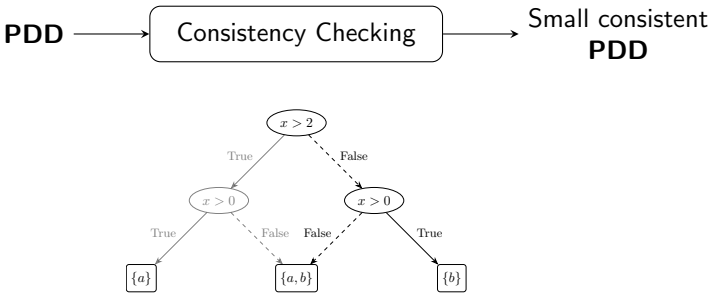
# Phase 3 : Size Reduction by Consistency Checking

# Phase 3 : Size Reduction by Consistency Checking



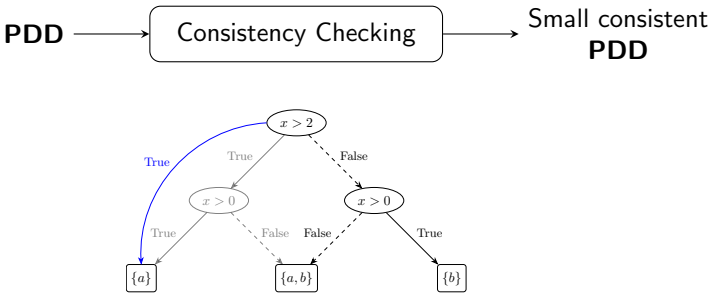

- Inconsistent combination of predicates : $(x > 2) \wedge \neg(x > 0)$

# Phase 3 : Size Reduction by Consistency Checking



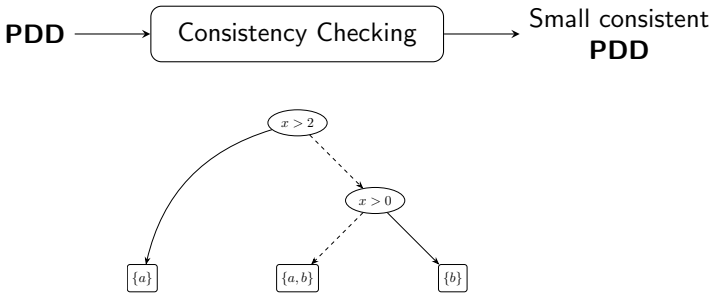

- Inconsistent combination of predicates : $(x > 2) \wedge \neg(x > 0)$
- SMT-based simplification by finding and removing inconsistencies

# Phase 3 : Size Reduction by Consistency Checking




- Inconsistent combination of predicates : $(x > 2) \wedge \neg(x > 0)$
- SMT-based simplification by finding and removing inconsistencies

# Phase 3 : Size Reduction by Consistency Checking




- Inconsistent combination of predicates : $(x > 2) \wedge \neg(x > 0)$
- SMT-based simplification by finding and removing inconsistencies

# Phase 3 (Continued) : Further Size Reduction

# Phase 3 (Continued) : Further Size Reduction

- **Variable reordering**
  - Goal: Find a good predicate order to reduce size.
  - Uses *sifting* algorithm[1].

[1] Rudell, R. "Dynamic variable ordering for ordered binary decision diagrams", ICCAD'93.

# Phase 3 (Continued) : Further Size Reduction

- **Variable reordering**
  - Goal: Find a good predicate order to reduce size.
  - Uses *sifting* algorithm[1].
- **Care-set reduction**
  - Unreachable state $\rightsquigarrow$ action does not matter
  - Compresses the diagram by only preserving outputs for relevant states.
  - Based on *restrict* operator[2].

[1] Rudell, R. "Dynamic variable ordering for ordered binary decision diagrams", ICCAD'93.
[2] Coudert, & Madre. "A unified framework for the formal verification of sequential circuits", ICCAD'90.

# Experiments

**Benchmarks**:

- Cyber-physical systems :
  - Benchmarks from SCOTS [3] and UppAal [4]
- Markov Decision Processes:
  - Quantitative Verification Benchmark Set (`https://qcomp.org`)
  - Policy extracted using Storm[5]

Implemented in python: dtControl + BuDDy

[3] Rungger, M, and Zamani M. "SCOTS: A tool for the synthesis of symbolic controllers." HSCC'16.
[4] David, A, et al. "Uppaal stratego." TACAS'15.
[5] Dehnert, C, et al.. "A storm is coming: A modern probabilistic model checker." CAV'17.

# Results: Size Comparison

Table: Selected results: PDD vs. DT and bit-blasted BDD sizes

| Controller | States | BDD | DT | PDD | Comments |
|---|---|---|---|---|---|
| `10rooms` | 26,244 | 1102 | 8648 | **344** | PDD much smaller |
| `helicopter` | 280,539 | 3348 | 3169 | **3158** | PDD has comparable size to DTs |
| `cartpole` | 271 | 197 | **126** | **126** | DT = PDD |
| `blocksworld.5` | 1,124 | 4043 | **617** | 796 | PDD more compact than the BDD |
| `pnueli-zuck.5` | 303,427 | **59,217** | 85,685 | 72,192 | PDD more compact than the DT |

- PDDs are on average 77% smaller than the bit-blasted BDDs
- PDDs are on average 16% smaller than the DTs

# Ablation Study

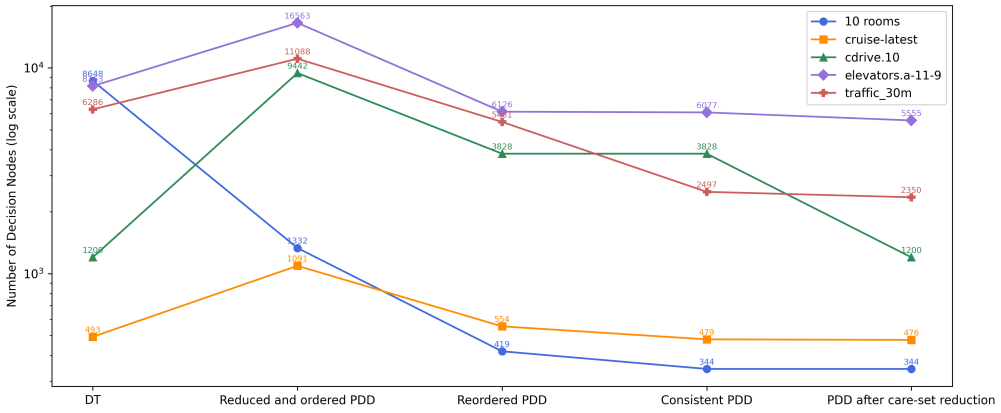



Figure: Effect of different steps in the pipeline in selected benchmarks

# Conclusion

**Predicate Decision Diagrams**

- Uses predicates over variables like DTs $\rightsquigarrow$ explainable
- Supports BDD-based algorithms
- Avoids bit-blasting; stays at semantic predicate level

# Conclusion

**Predicate Decision Diagrams**

- Uses predicates over variables like DTs $\rightsquigarrow$ explainable
- Supports BDD-based algorithms
- Avoids bit-blasting; stays at semantic predicate level

**Synthesis Pipeline**

1. Mining predicates using DT learning
2. Compiling to PDDs
3. Size reduction and optimization

# Conclusion

**Predicate Decision Diagrams**

- Uses predicates over variables like DTs $\rightsquigarrow$ explainable
- Supports BDD-based algorithms
- Avoids bit-blasting; stays at semantic predicate level

**Synthesis Pipeline**

1. Mining predicates using DT learning
2. Compiling to PDDs
3. Size reduction and optimization

Thank You