Decision Diagrams for Linear Arithmetic

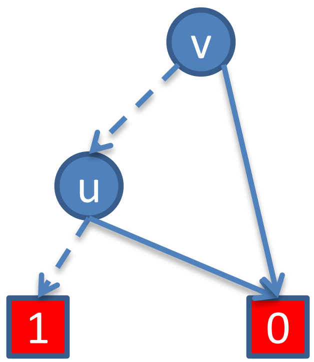
Arie Gurfinkel, SEI
Sagar Chaki, SEI
Ofer Strichman, Technion/SEI

•This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

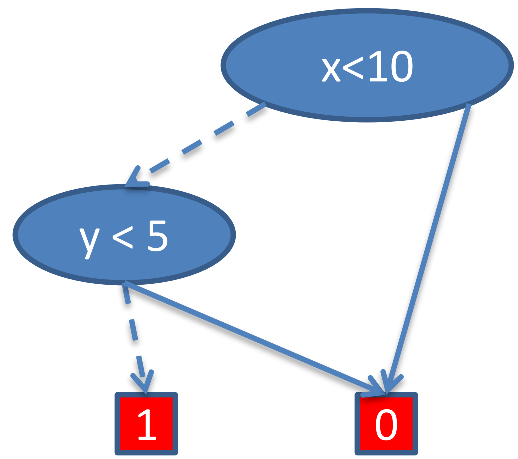
Linear Decision Diagrams (LDDs)

… are Binary Decision Diagrams with nodes labeled by linear inequalities

BDD



LDD



Our contributions:

- implementation on top of CUDD, including
    - support for propositional operations (AND, OR, NOT, ITE)
    - support for projection (i.e., existential quantification, QELIM) of numeric variables
    - dynamic variable ordering (DVO)
- benchmark and experiments

Motivation(1): Predicate + Numeric Abstractions

Predicate and Numeric Abstractions

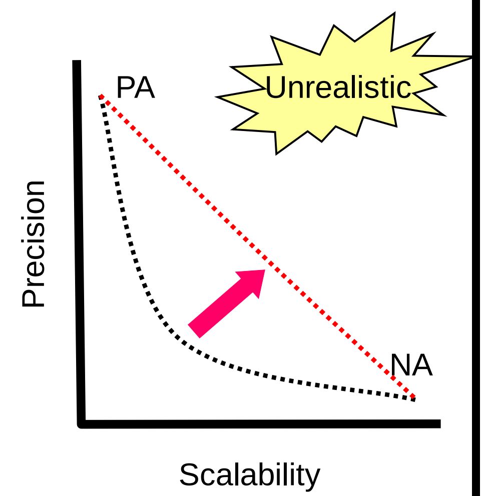
Predicate Abstraction (PA) (e.g., SDV)

- Typical property: no lock is acquired twice
- Program verification reduced to propositional reasoning with model checker
- Works well for control-driven programs
- Works poorly for data-driven programs

Numeric Abstraction (NA) (e.g, ASTREE)

- Typical property: no arithmetic overflow
- Program verification reduced to arithmetic reasoning
- Works well for data-driven programs
- Works poorly for control-driven programs

How to combine PA and NA to get the best of both?



Gurfinkel & Chaki. Combining Predicate and Numeric Abstraction for Software Model Checking. In FMCAD 2008

Motivation (2): Numeric Decision Diagrams

Numeric Decision Diagrams

NDD elements are

$2^P \rightarrow 2^N$

BDDs over Predicate and Numeric Terms

$(p_1 \&\& p_2) \ ||$

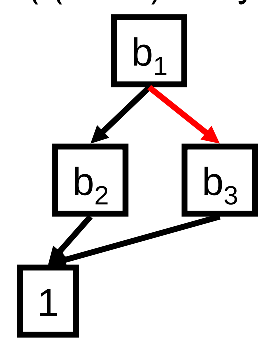
$(x<0 \ \&\& \ y=z)$

normalize

$(x>=0 \ \&\& \ z>0) \ ||$

$(!(x>=0) \ \&\& \ y=z)$

p_1: x>=0, p_2: z>0



1-edges are black, 0-edges are red

edges to 0 node are not shown

b1:x>=0, b2:z>0, b3:y=z

Software Engineering Institute | CarnegieMellon

Combining PA and NA for Soft MC
Gurfinkel and Chaki

27

Gurfinkel & Chaki. Combining Predicate and Numeric Abstraction for Software Model Checking. In FMCAD 2008

Motivation (2): Numeric Decision Diagrams

Numeric Decision Diagrams

Problems with NDDs are:

- No reduction w.r.t. the types of constraints
- All numeric operations are done path-at-a-time (i.e., exponential in the diagram!!!)

Lesson learned: need diagrams for linear arithmetic with efficient (not path-at-a-time) existential quantification

Software Engineering Institute | Carnegie Mellon

Combining PA and NA for Soft MC
Gurfinkel and Chaki

27

Gurfinkel & Chaki. Combining Predicate and Numeric Abstraction for Software Model Checking. In FMCAD 2008

Some Other Applications of LDDs

- Represent and manipulate Boolean formulas over linear arithmetic …
  - to compute predicate abstraction
  - to summarize loop-free code
  - for program analysis with disjunctive abstract domain
  - to combine predicate and numeric abstractions
  - for timed automata verification
  - …
- LDDs are NOT good for SATISFIABILITY checking
  - not a substitute for an SMT solver

Talk Outline

- The basics
  - variable ordering, reduction rules, propositional operations
- Dynamic Variable Ordering
- Quantifier Elimination
  - existential quantification of a single variable
  - heuristics for quantifying multiple variables
- Implementation, Benchmarks, Results
- Conclusion and Future Work

Canonizing Linear Inequalities

coefficient variable constant

Linear Inequality: $\underbrace{a_1\, x_1 + \ldots + a_n\, x_n}_{\text{term}} \leq k$

$x < 10$	$\equiv$	$x \leq 9$
$x + y = 10$	$\equiv$	$x + y \leq 10 \wedge x + y \geq 10$
$x + y \geq 10$	$\equiv$	$-x - y \leq -10$
$-x-y \leq -10$	$\equiv$	$\neg\,(x + y \leq 9)$

LDD Node Ordering

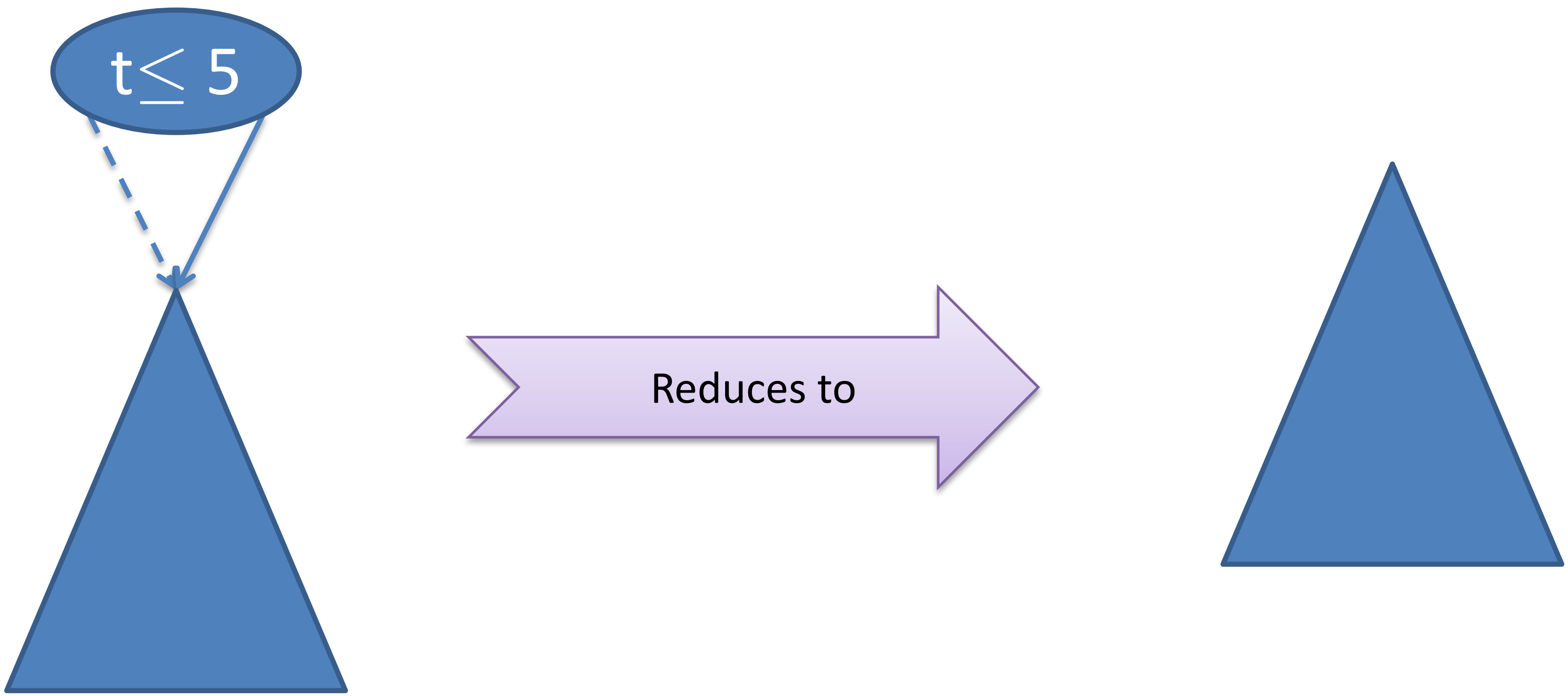
Random: $\{x \leq 0\}$ $\{x \leq 10\}$ $\{y \leq 5\}$ $\{x \leq 5\}$ $\{z \leq 6\}$ $\{y \leq 3\}$

Term-sorted: $\{x \leq 0\}$ $\{x \leq 10\}$ $\{x \leq 5\}$ $\{y \leq 5\}$ $\{y \leq 3\}$ $\{z \leq 6\}$

Ordered: $\{x \leq 0\}$ $\{x \leq 5\}$ $\{x \leq 10\}$ $\{y \leq 3\}$ $\{y \leq 5\}$ $\{z \leq 6\}$

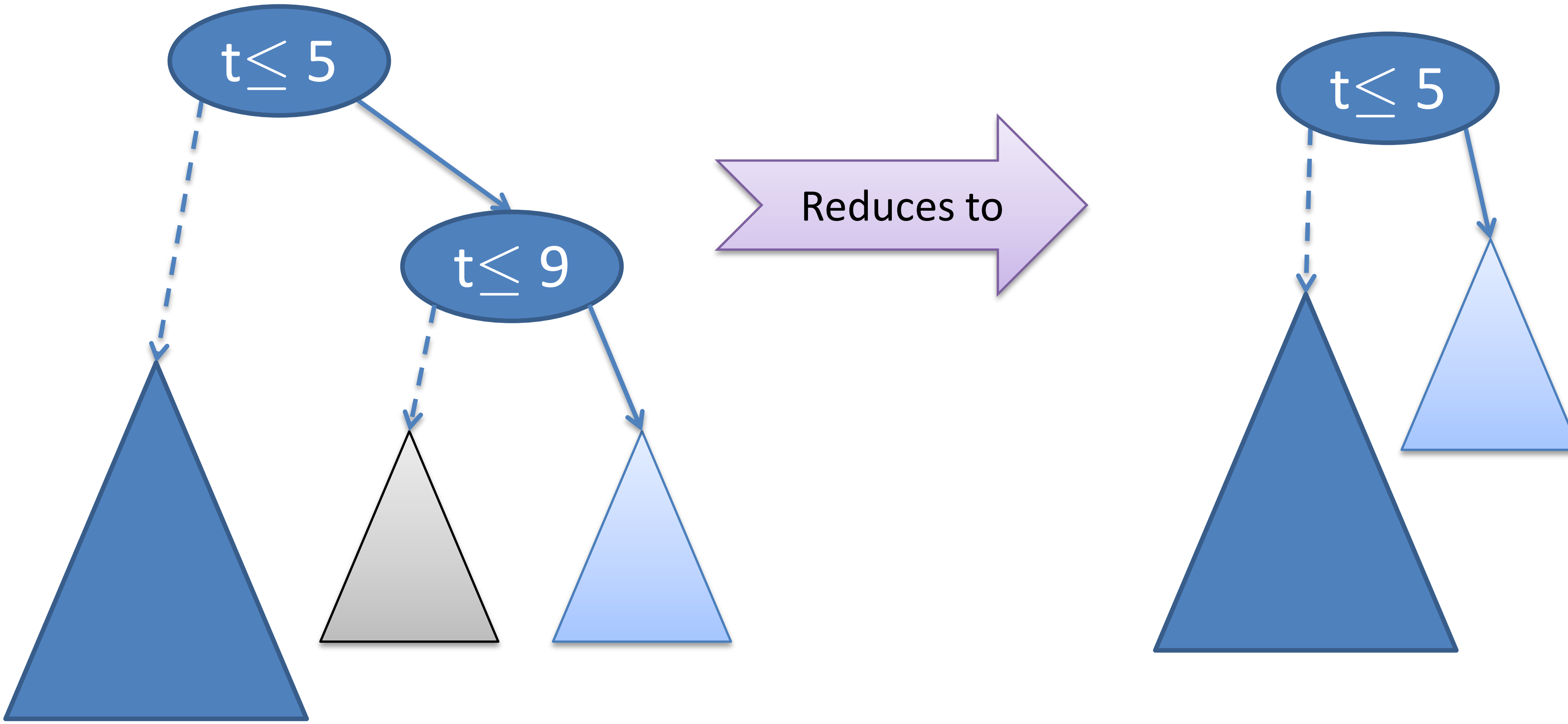
Ordered: $\{y \leq 3\}$ $\{y \leq 5\}$ $\{z \leq 6\}$ $\{x \leq 0\}$ $\{x \leq 5\}$ $\{x \leq 10\}$

Reduction: Different Children

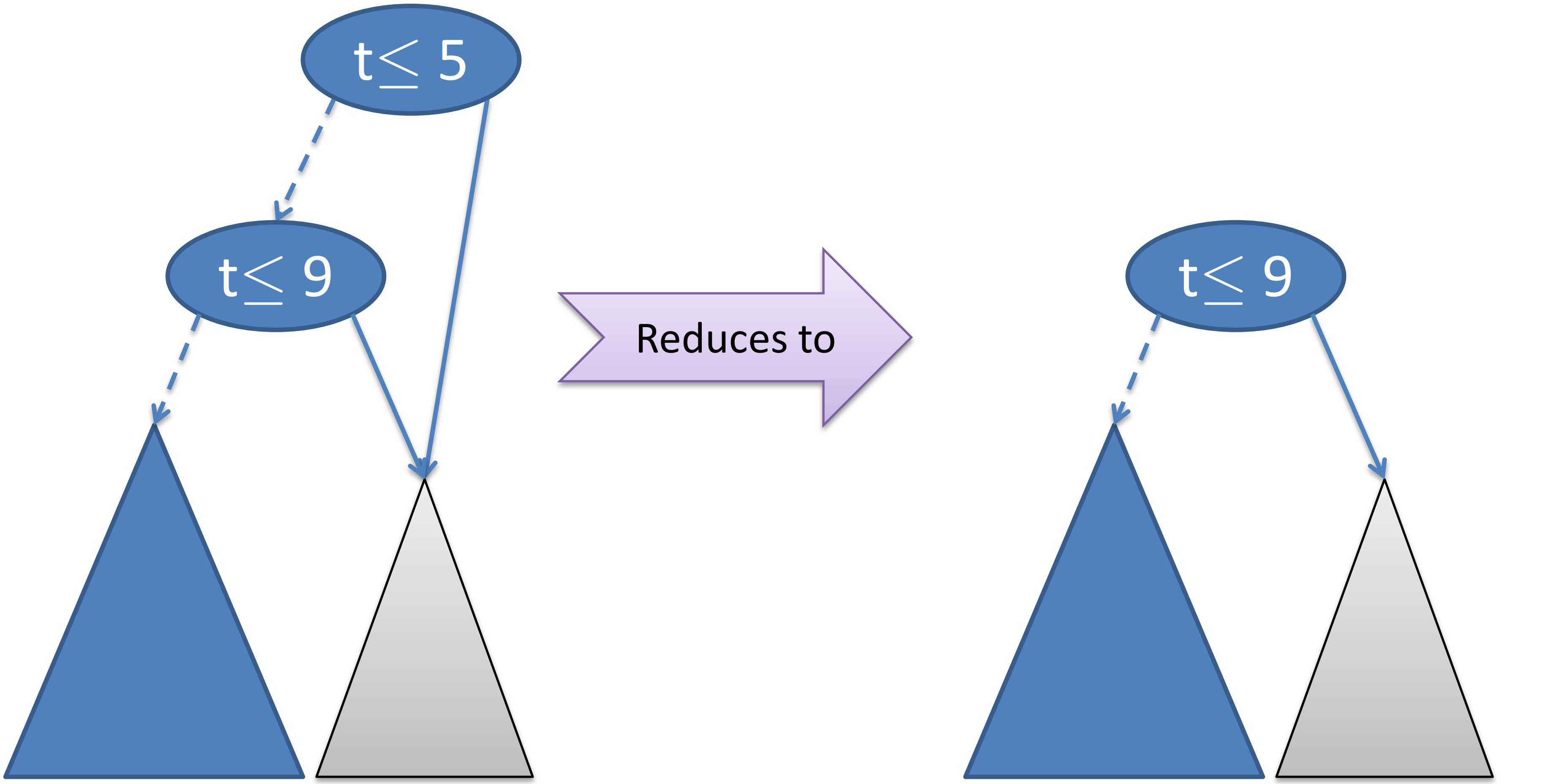


Same as BDD

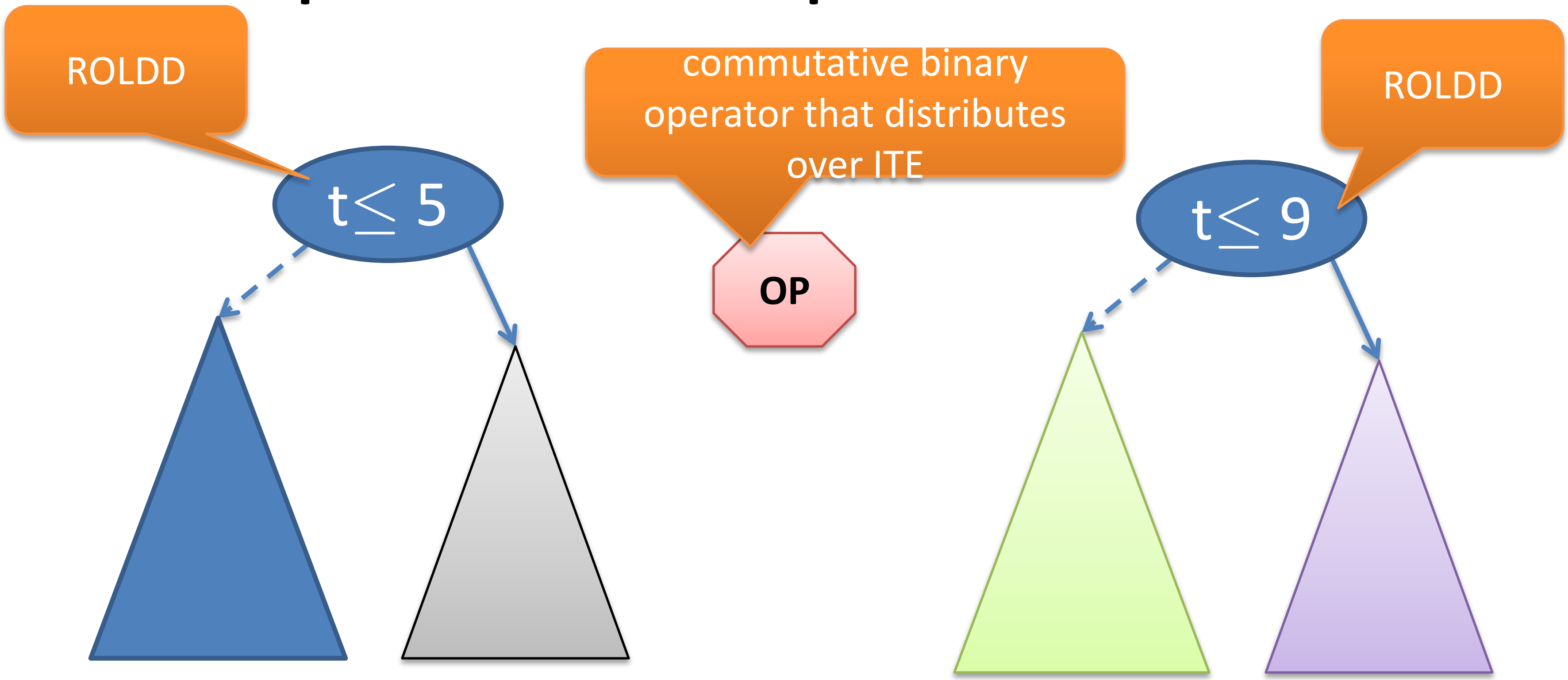
Reduction: Imply High



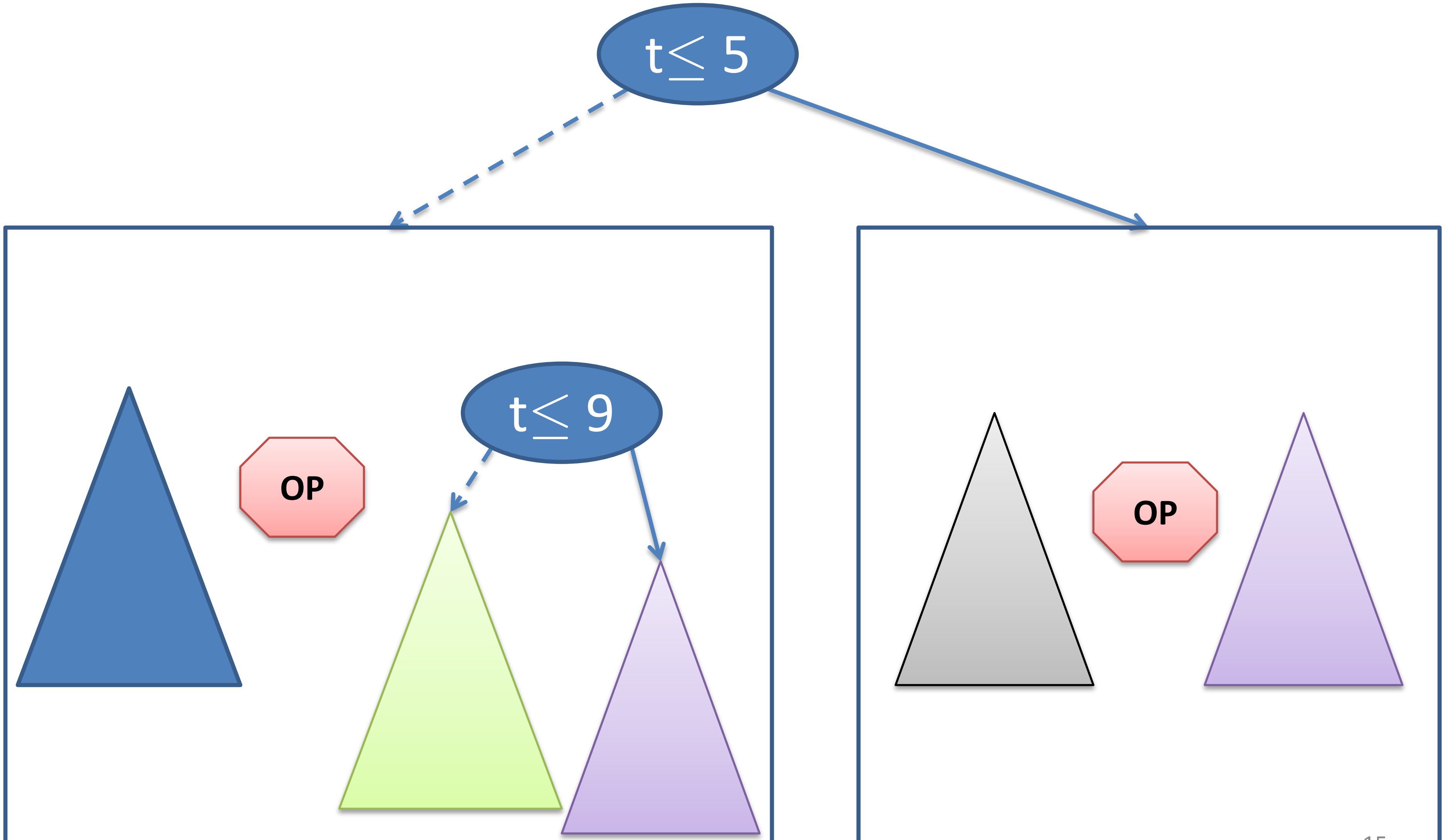
Reduction: Imply Low



Propositional Operations: APPLY

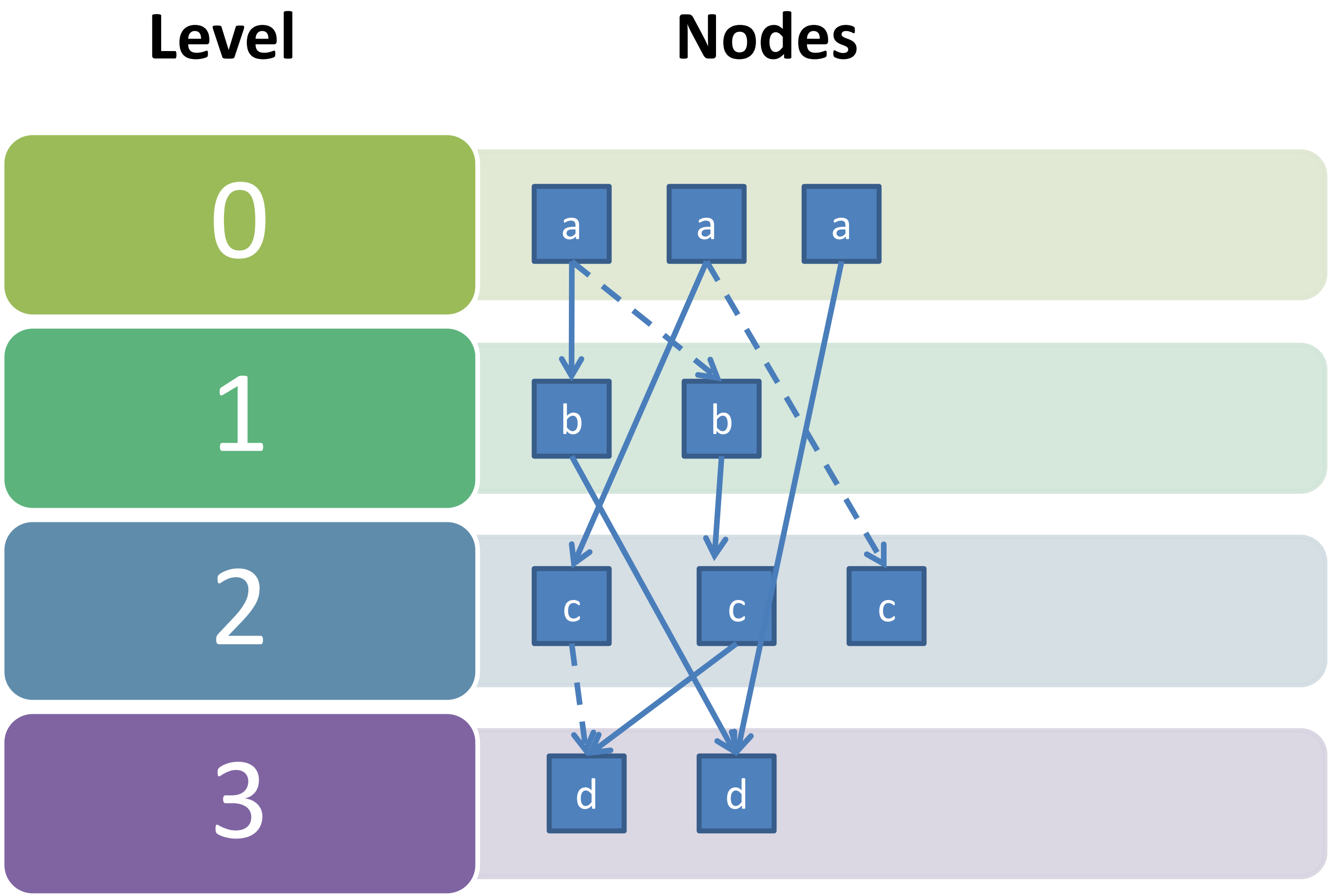


Propositional Operations: APPLY



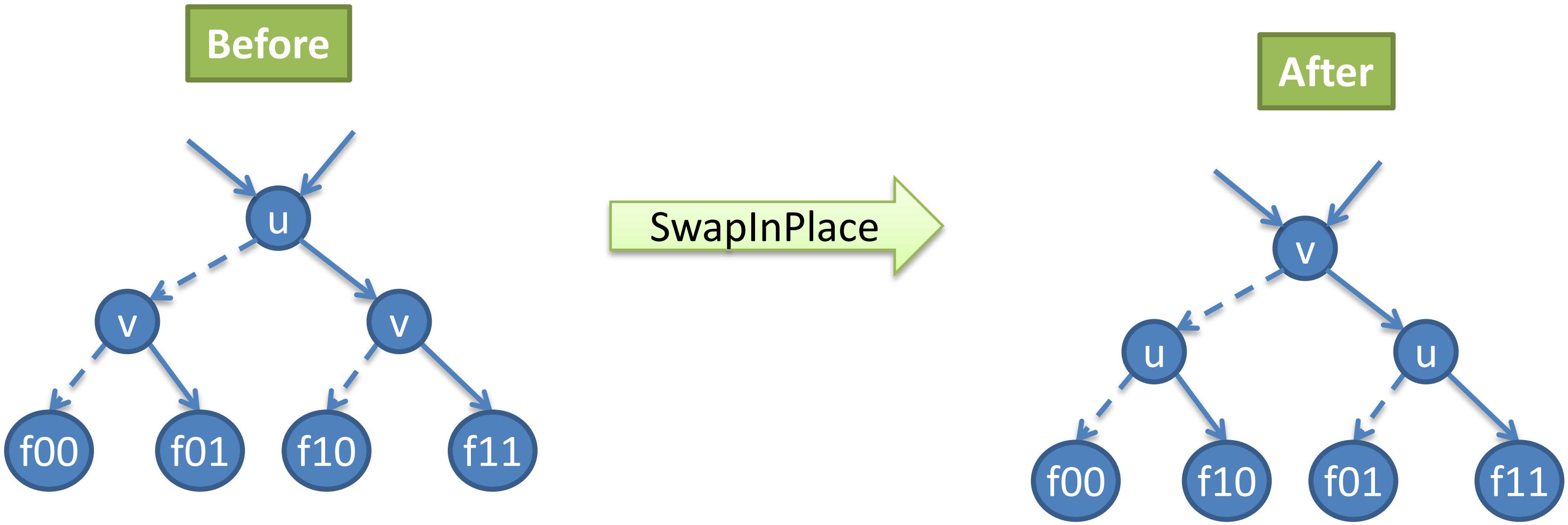
Rudell’s DVO Algorithm for BDDs

Unique Table



*Edges to 0 and 1 are not shown

Example: Changing levels



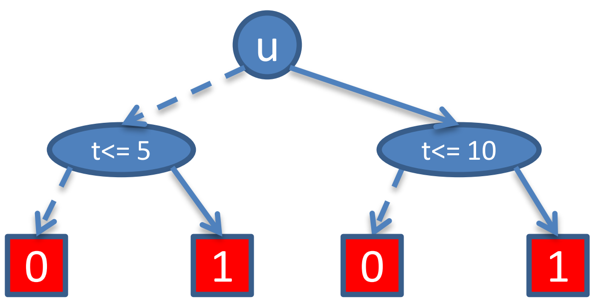
(u, (v, f11, f10), (v, f01, f00)) is overwritten in place by (v, (u, f11, f01), (u, f10, f00))
Trivial new cofactors are reduced, i.e., when f00=f10 or f01=f11
Only the diagram rooted at *u* is changed (both the label and the children are new)

Complexity: linear in the number of nodes labeled with u in the unique table

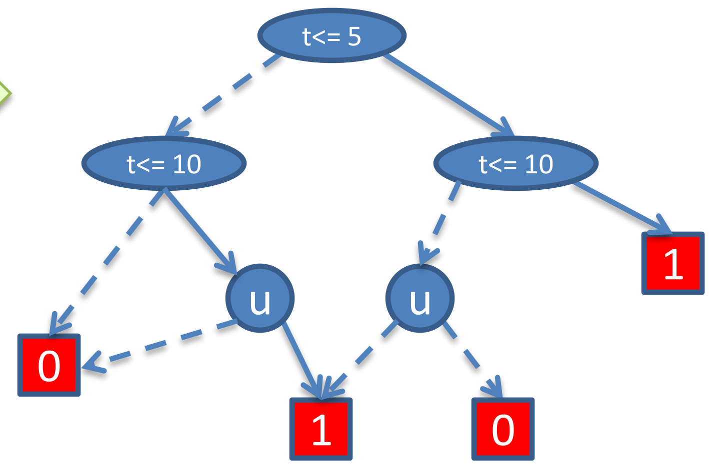
Example: Changing levels in ROLDD

Before Reordering

After Reordering



SwapInPlace



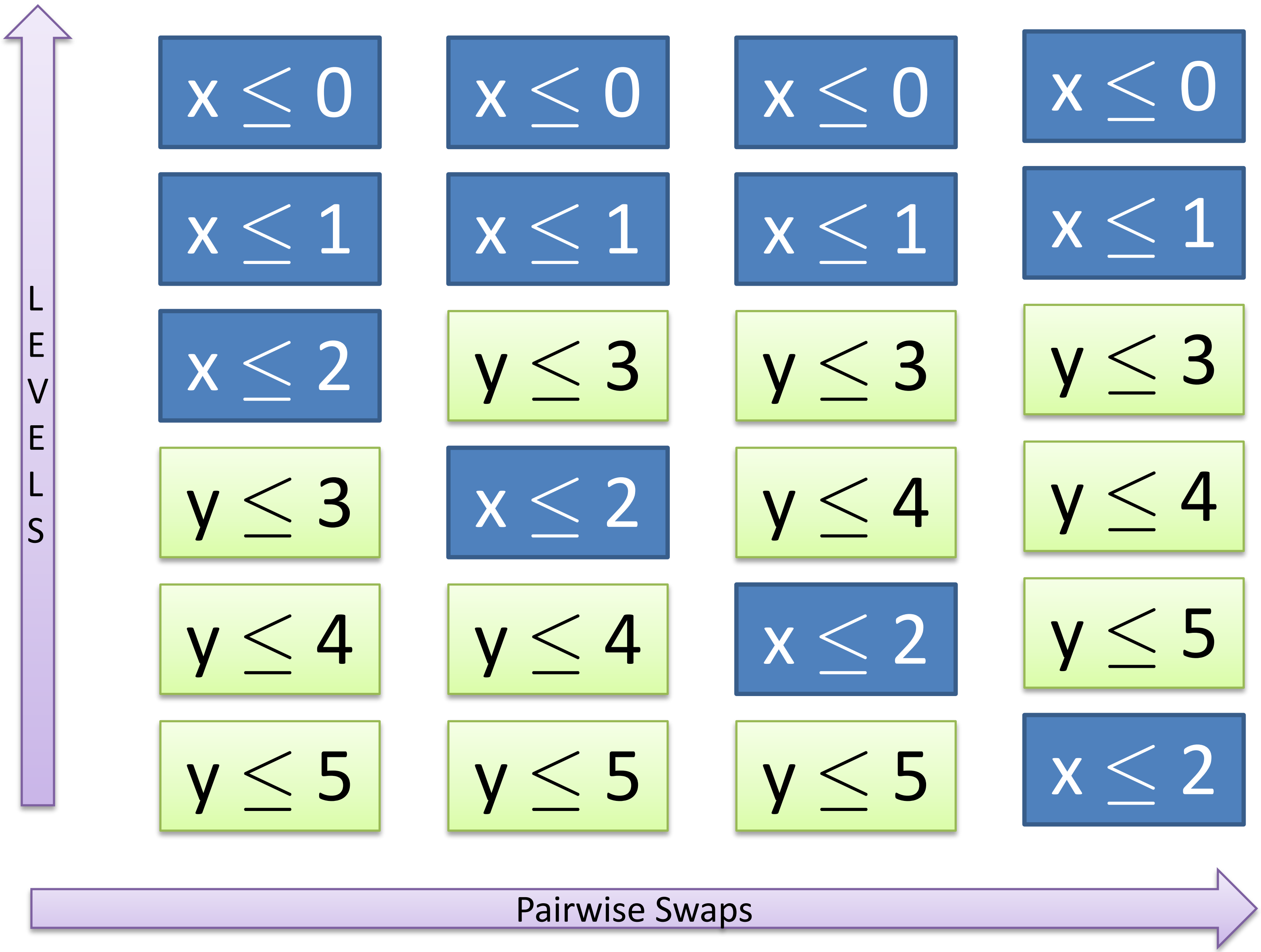
ROLDD with order: u, t<=5, t<=10
(shown as a tree)

New order: t<=5, t<=10,u
Not reduced!

Cannot use BDD reordering for LDD!

Problems extending DVO to LDDs

- Broken ordering constraints
  - **Solution: swap adjacent terms instead of adjacent levels**
- Broken Imply-high and Imply-low rules
  - **Solution: enforce the rules in swapInPlace**
- LDDs are not canonical
  - **Solution: LDDs are "canonical enough": in SwapInPlace the root node is never reduced**


LEVELS
$x \leq 0$
$x \leq 0$
$x \leq 0$
$x \leq 0$
$x \leq 1$
$x \leq 1$
$x \leq 1$
$x \leq 1$
$x \leq 2$
$y \leq 3$
$y \leq 3$
$y \leq 3$
$y \leq 3$
$x \leq 2$
$y \leq 4$
$y \leq 4$
$y \leq 4$
$y \leq 4$
$x \leq 2$
$y \leq 5$
$y \leq 5$
$y \leq 5$
$y \leq 5$
$x \leq 2$
Pairwise Swaps

Two techniques for QELIM

- **Black Box:** use QELIM for conjunctions as a black box. Apply it to all paths of a diagram
  - linear in the number of paths == exponential in the size of the diagram!
  - many examples in the literature. (e.g., we used it in NDDs)
- **White Box:** Extend Fourier-Motzkin QELIM to the DAG of LDD
  - in the best case, same complexity as BDD quantification

Fourier-Motzkin QELIM

FM1(Var *y*, Conjunction φ)
  let *S* be all constraints with *y*
  remove *S* from φ
  add all pairwise resolutions of *S* to φ

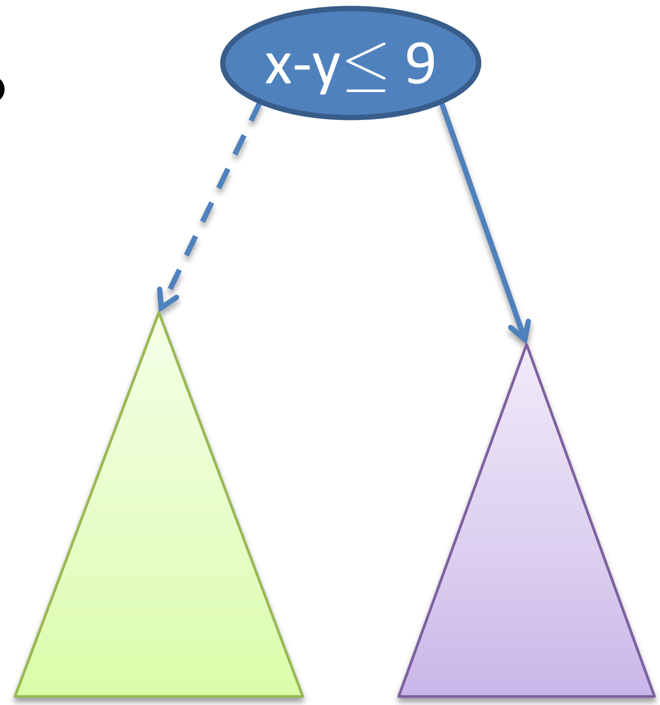
$$\exists\, y\, .\, \boxed{x\text{-}y \leq 5} \;\wedge\; x\text{-}z \geq 8 \;\wedge\; \boxed{y\text{-}z \leq 10}$$

$$x\text{-}z \geq 8 \wedge x\text{-}z \leq 15$$

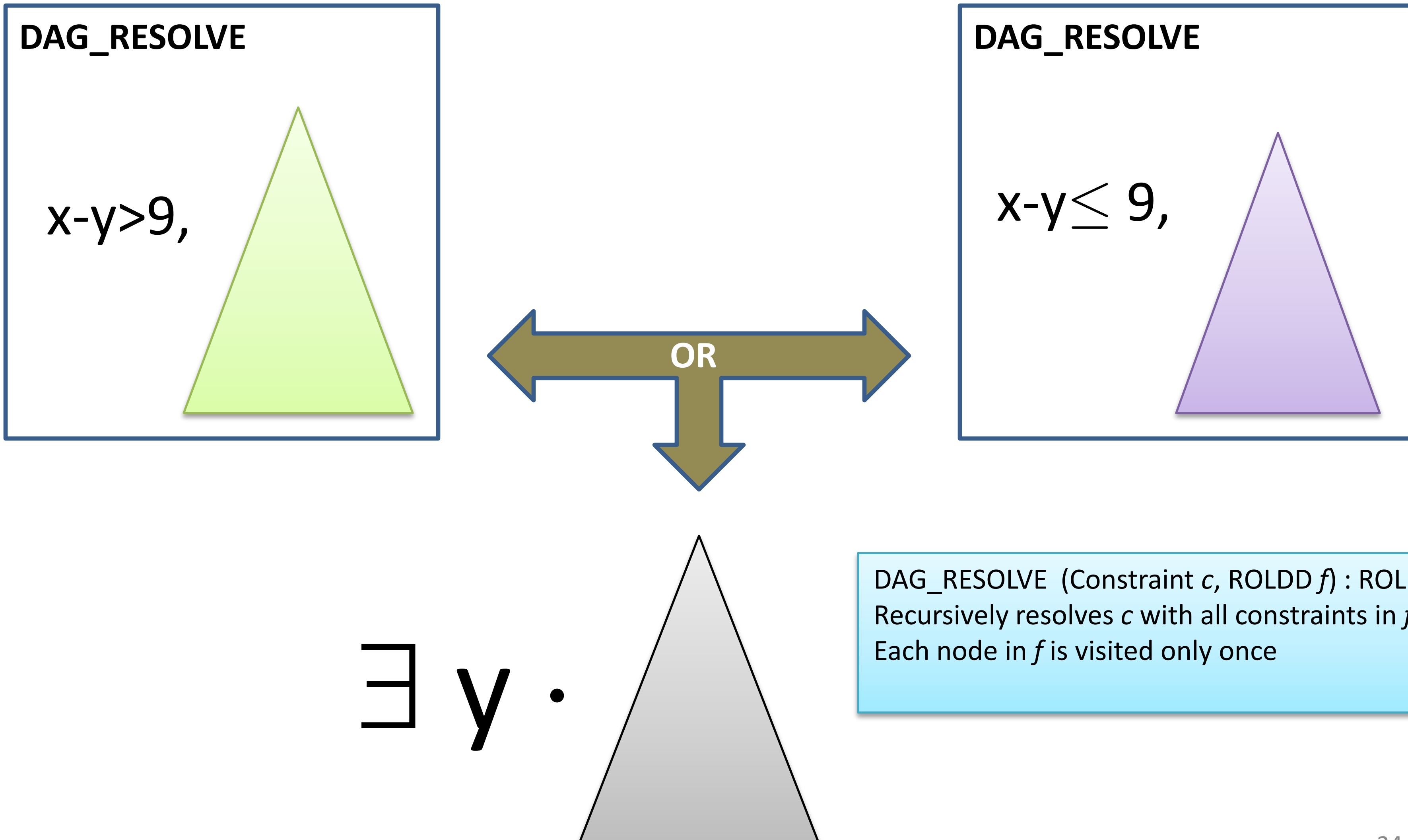
FM2(Var *y*, Formula φ)
  while exists constraint *c* with *y* in φ **do**
    remove *c* from φ
    resolve *c* with remaining constraints in φ
  end while

WB_EXISTS1: Example

$\exists\, y \cdot$



WB_EXISTS1: Example (Cont)



DAG_RESOLVE (Constraint *c*, ROLDD *f*) : ROLDD
Recursively resolves *c* with all constraints in *f*
Each node in *f* is visited only once

Quantifying out multiple variables

```
EXISTS(LDD f, Vars V)
  res = f;
  while (V != empty)
    V' = FIND_DROP_VARS (V, res);
    if (V' != empty)
      res = DC(CONS_OF(V'), res);
      V = V \ V';
      continue;

      u = PICK_VAR (V, res);
      res = WB_EXISTS1(u, res);
      V = V \ {u};
  end while
  return res;
```

EXISTS1 -- any quantification procedure that can eliminate a single variable. In our implementation, it is the optimized WB_EXISTS1 from previous slides

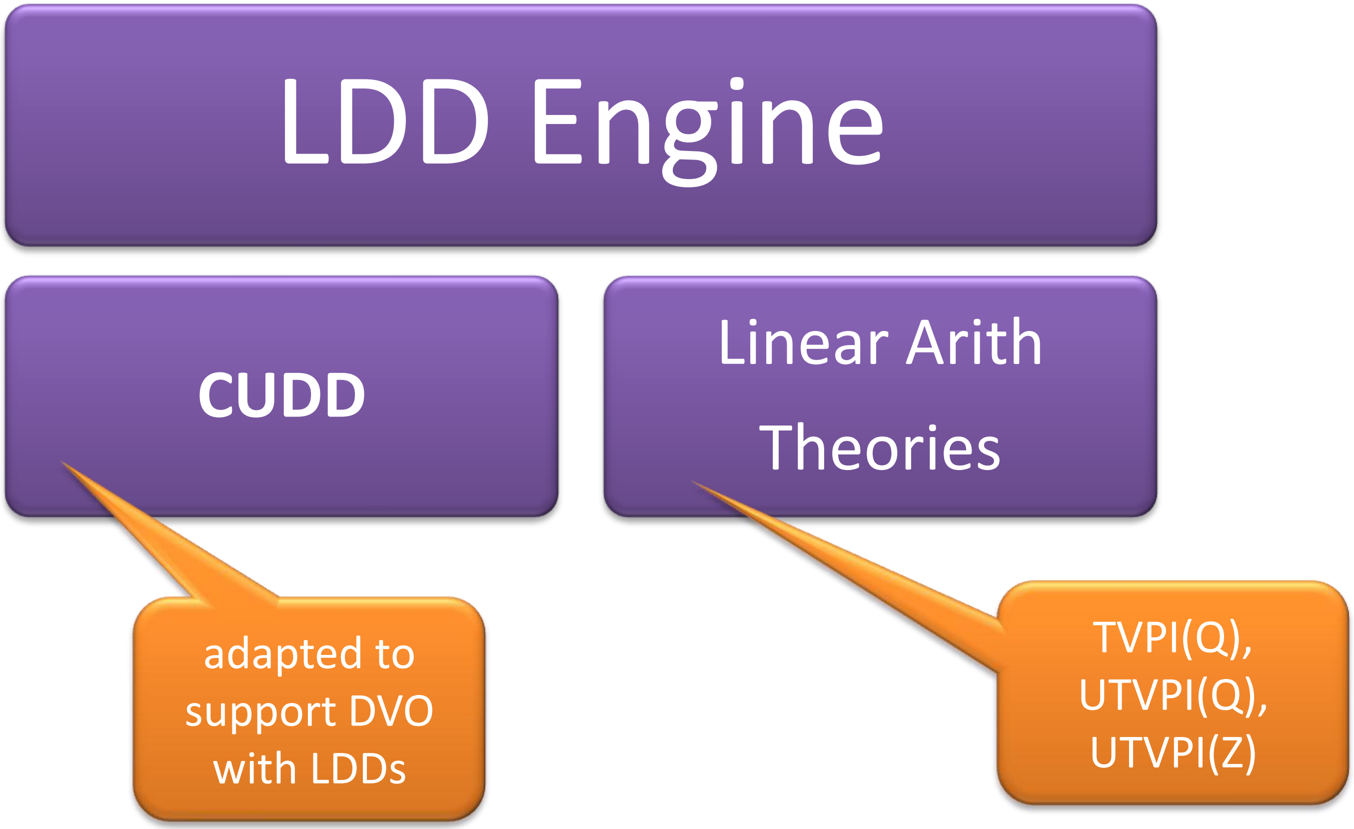
DC short for DROP_CONS

FIND_DROP_VARS(V, res) – finds all variables in V that have trivial resolutions on all 1-paths of res

PICK_VAR (V, res) -- picks a variable from V to be quantified out next

In our implementation, FIND_DROP_VARS and PICK_VAR are based on looking at the set of all constraints that are in support of res.

The Implementation



Benchmark: Image Computation

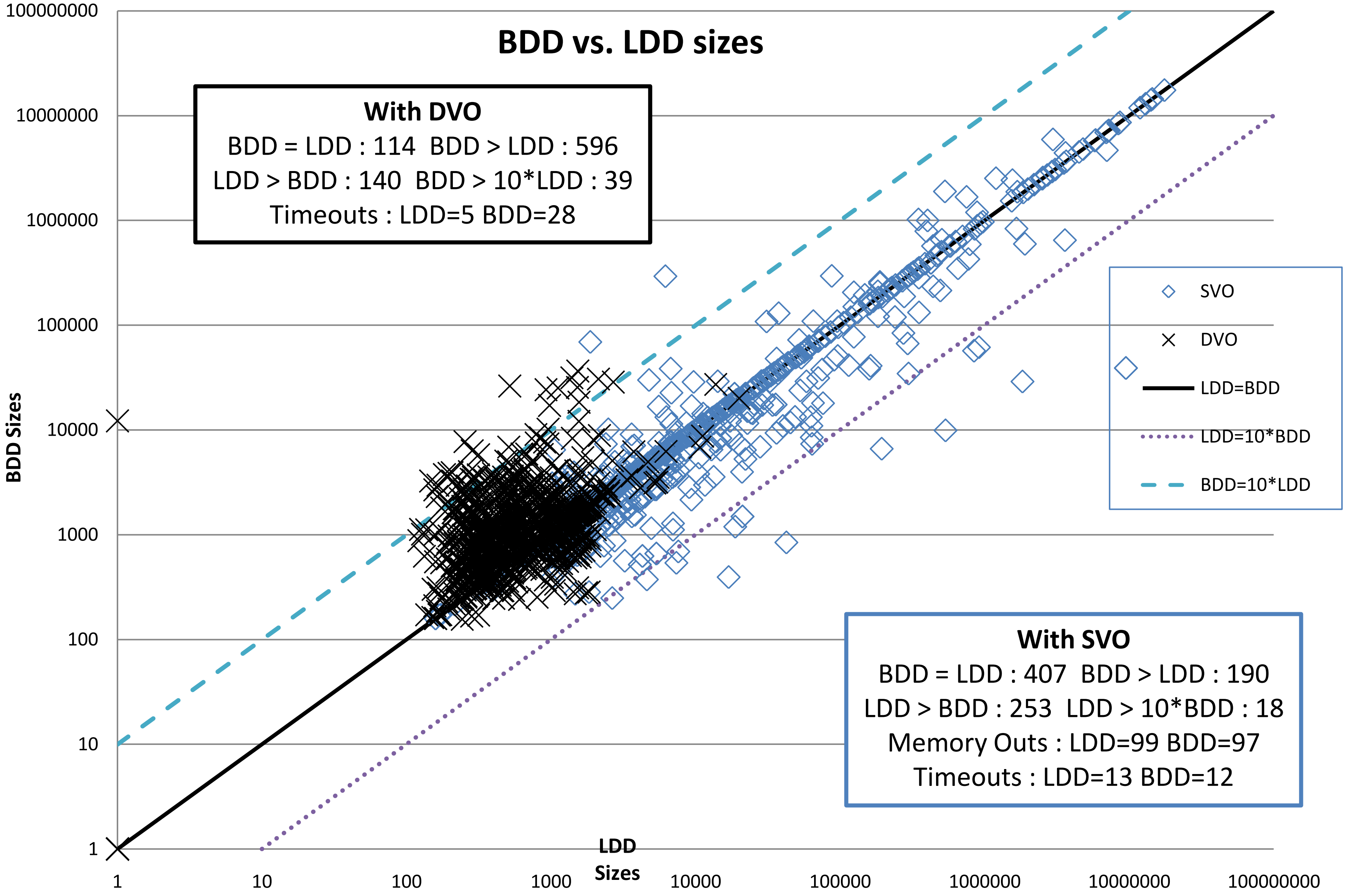
Test case: $\exists V \cdot R(V, V')$

transition relation of a loop-free program fragment

- Each test case is constructed
  - from open source software: CUDD, mplayer, bzip2,...
  - extracted using LLVM into SSA with optimizations, aggressive loop-unrolling, and inlining
  - approximated using UTVPI constraints
- Stats: 850 test cases

  4KB – 700KB (in SMT-LIB format),

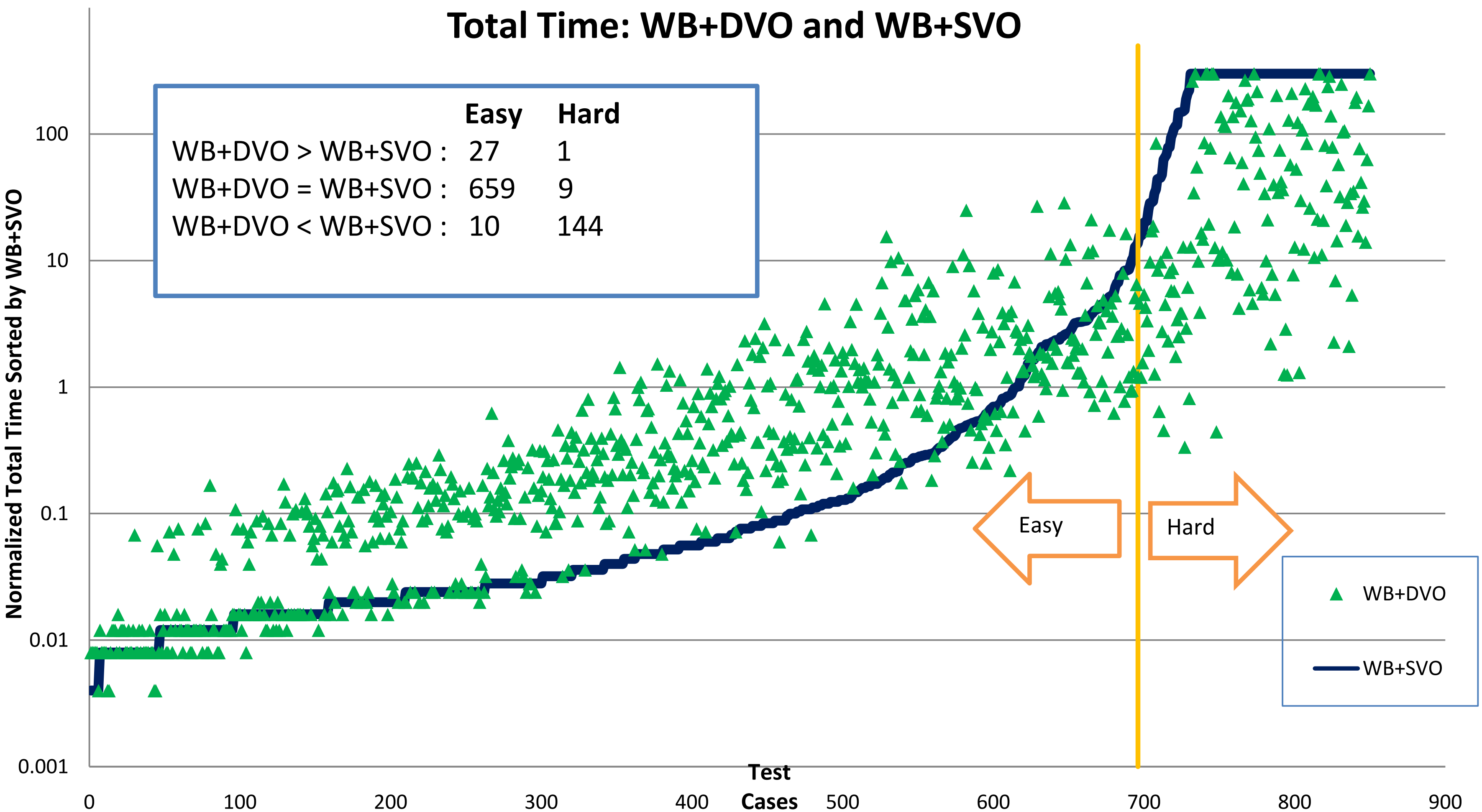
  30 – 7,956 variables



BDD vs. LDD sizes
With DVO
BDD = LDD : 114 BDD > LDD : 596
LDD > BDD : 140 BDD > 10*LDD : 39
Timeouts : LDD=5 BDD=28
With SVO
BDD = LDD : 407 BDD > LDD : 190
LDD > BDD : 253 LDD > 10*BDD : 18
Memory Outs : LDD=99 BDD=97
Timeouts : LDD=13 BDD=12
SVO
DVO
LDD=BDD
LDD=10*BDD
BDD=10*LDD
BDD Sizes
LDD Sizes
100000000
10000000
1000000
100000
10000
1000
100
10
1

Overall Results for QELIM

	Hard (154 cases)				Easy (696 cases)		
Alg.	***Total (sec)***	***QE (sec)***	***TO***	***MO***	***Total (sec)***	***QE (sec)***	***TO***
BB	--	--	141	0	--	--	670
WB+SVO	38,739	36,511	21	99	395	80	0
WB+DVO	10,953	3,329	9	0	784	219	0


Total Time: WB+DVO and WB+SVO
Easy Hard
WB+DVO > WB+SVO : 27 1
WB+DVO = WB+SVO : 659 9
WB+DVO < WB+SVO : 10 144
Normalized Total Time Sorted by WB+SVO
100
10
1
0.1
0.01
0.001
0
100
200
300
400
500
600
700
800
900
Test Cases
Easy
Hard
WB+DVO
WB+SVO

Predicate Abstraction with LDDs

$$\exists V \cdot R(V) \wedge \bigwedge_i (b_i \leftrightarrow p_i(V))$$

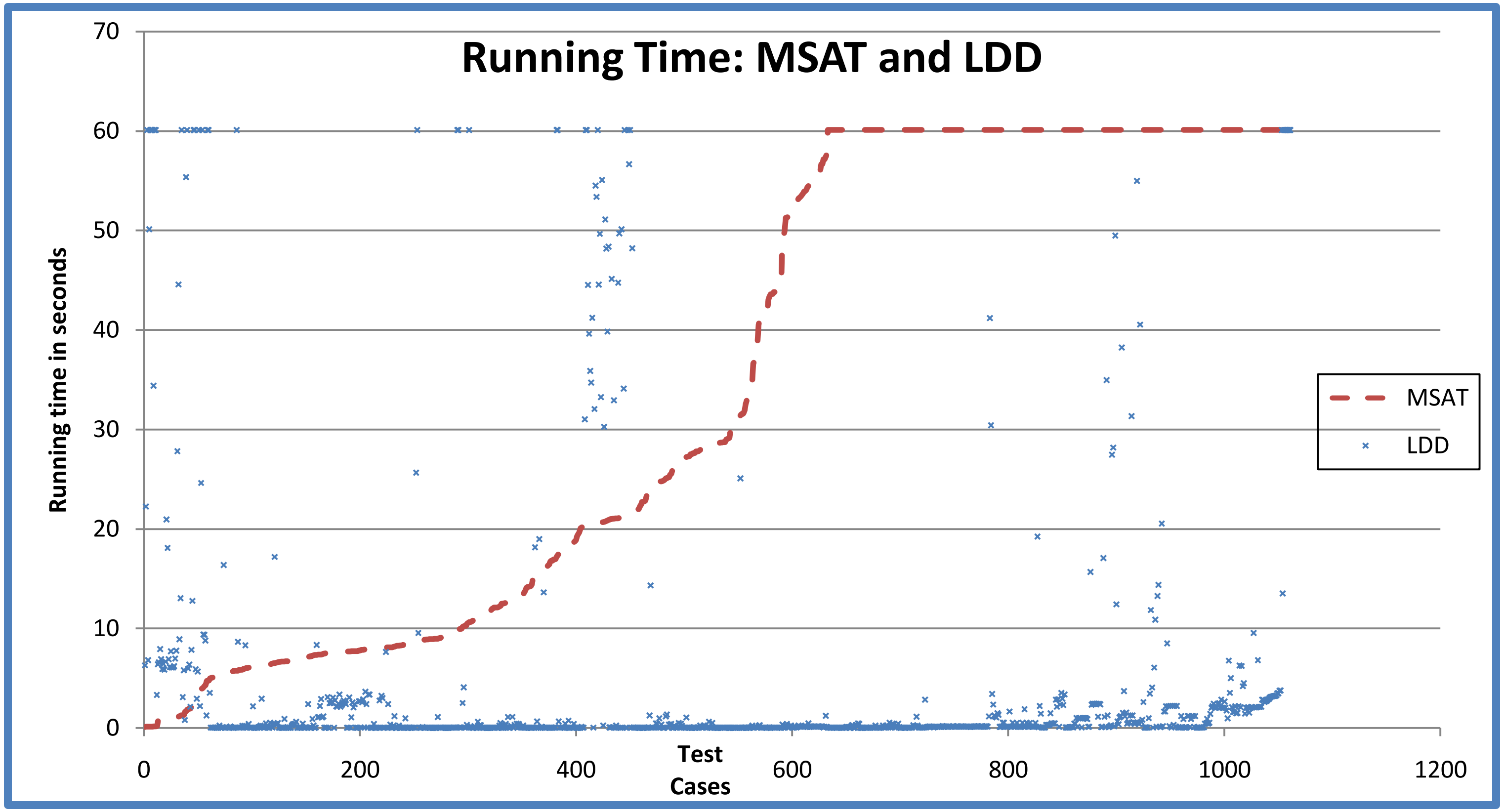
transition relation of a loop-free program fragment

Boolean variable

predicate

Predicate Abstraction with LDDs

$$\exists V \cdot R(V) \wedge \bigwedge_i (b_i \leftrightarrow p_i(V))$$



Related Work

Decision Diagrams (over linear constraints)

- Strehl. *Interval Diagram Techniques…* 1999
- Moller et al. *Difference Decision Diagrams.* 1999
- Larsen et al. *Clock Difference Diagrams.* 1999

Quantifier Elimination in Large Boolean Formulas

- Clarke et al. *SATABS: A SAT-Based PA for ANSI-C*. 2005
- Lahiri et al. *SMT Techniques for Fast PA.* 2006
- Cavada et al. *Computing PA by Integrating BDDs and SMT Solver. 2007*
- D. Monniaux. *A QELIM Algorithm for Linear Real Arith. 2008*

Future Work

- Predicate Abstractions with LDDs
- An LDD-based Abstract Domain
  - first step is a disjoint-box domain for variable range analysis
  - designing a widening is the main challenge
- Public release of the library
  - send email to arie@cmu.edu for more info

THE END