Boolean Synthesis via Decision Diagrams

Lucas M. Tabajara

based on joint work with Dror Fried, Yi Lin and Moshe Y. Vardi

What is Boolean Synthesis?

Given: specification (Boolean formula)

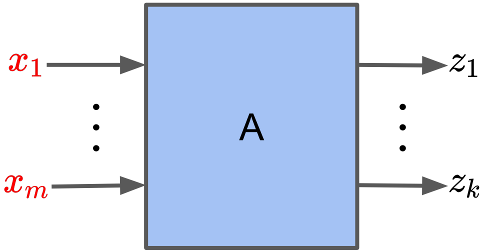
$$f(x_1, \dots, x_m, y_1, \dots, y_n)$$

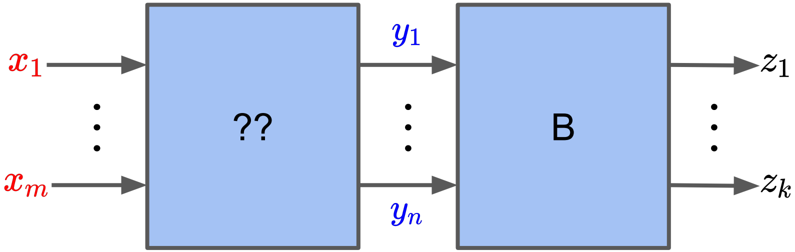
$$x_1, \dots, x_m, y_1, \dots, y_n \in \{0, 1\}$$

Synthesize: implementation (witness functions)

$$y_1 = g_1(x_1, \dots, x_m)$$
$$\dots$$
$$y_n = g_n(x_1, \dots, x_m)$$

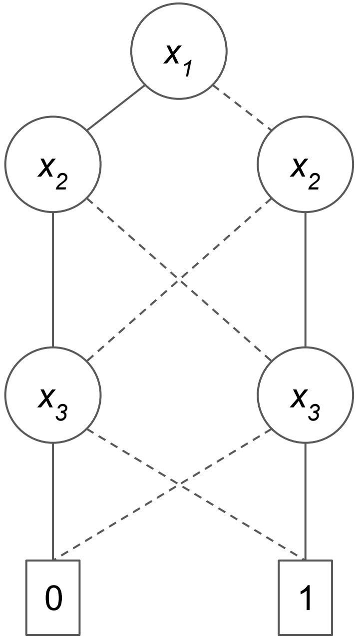
s.t. $\quad f(x_1, \dots, x_m, y_1, \dots, y_n) = 1$





$$\underbrace{A(x_1, \dots, x_n) = B(y_1, \dots, y_n)}_{f(x_1, \dots, x_m, y_1, \dots, y_n)}$$

BDD-Based Boolean Synthesis



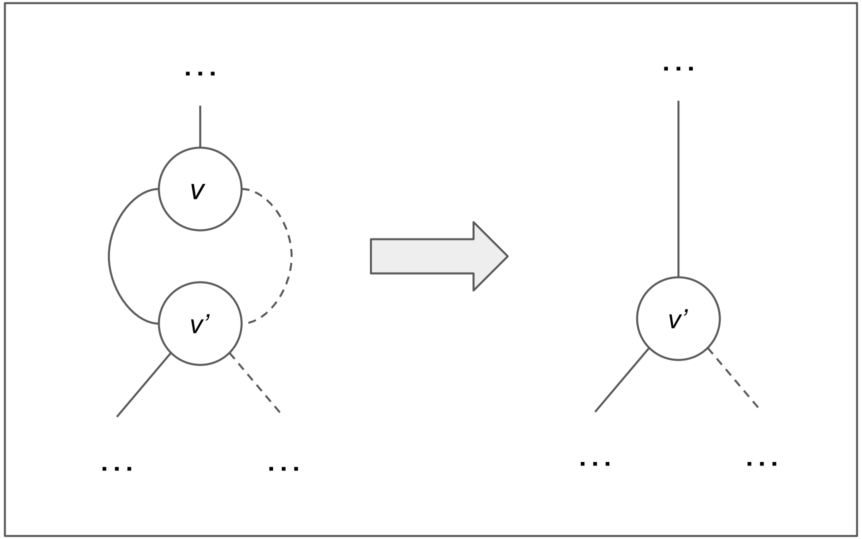
Pros of BDDs:

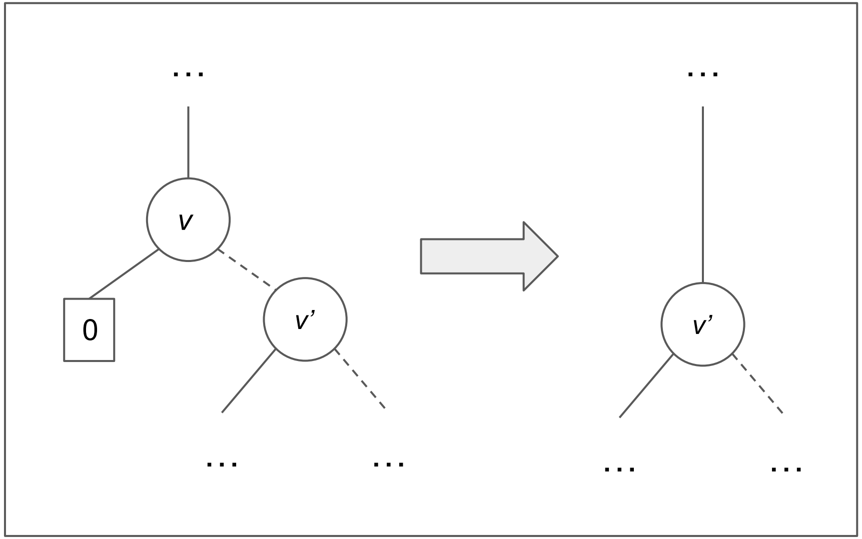
- Canonical and self-minimizing data structure
- Efficient implementations for Boolean operations relevant for synthesis (e.g. existential quantification)
- Widely used in temporal synthesis

Cons of BDDs:

- Performance dependent on a good variable ordering
- Construction of initial BDD can blow up before synthesis even starts

An Alternative to BDDs: Zero-Suppressed Decision Diagrams





BDD reduction rule:

suppress don’t-care nodes

ZDD reduction rule:

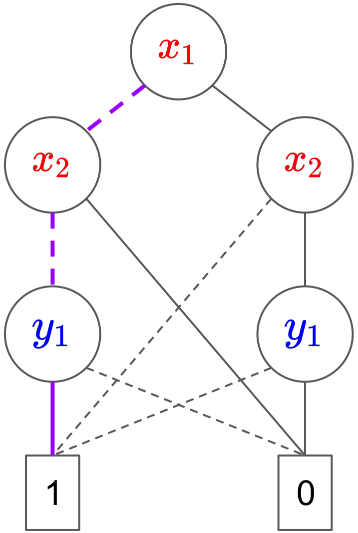
suppress nodes that have to be zero

BDD vs. ZDD Representation

$(x_1 \lor y_1) \land (\neg x_2 \lor \neg y_1)$

BDD: each path to 1 is a satisfying assignment

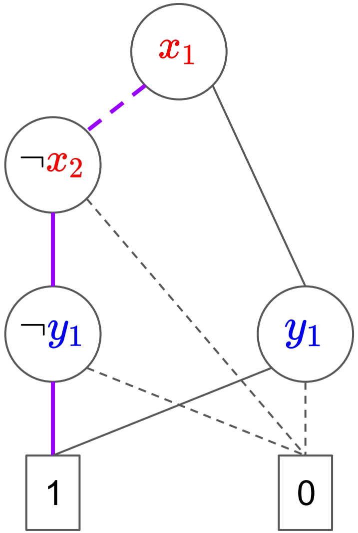
$x_1 \mapsto 0$
$x_2 \mapsto 0$
$y_1 \mapsto 1$



Worst case exponential on size of the formula

ZDD: each path to 1 is a clause

$(\neg x_2 \lor \neg y_1)$



Worst case linear on size of the formula

ZDDs Allow Efficient Representation of Large Clause Spaces

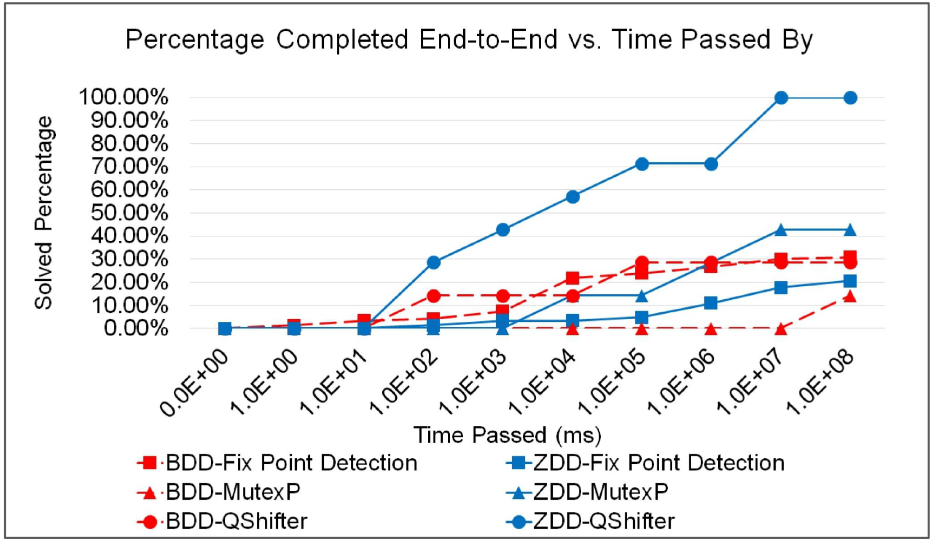
<u>Existential quantification:</u> Implemented by symbolic resolution.

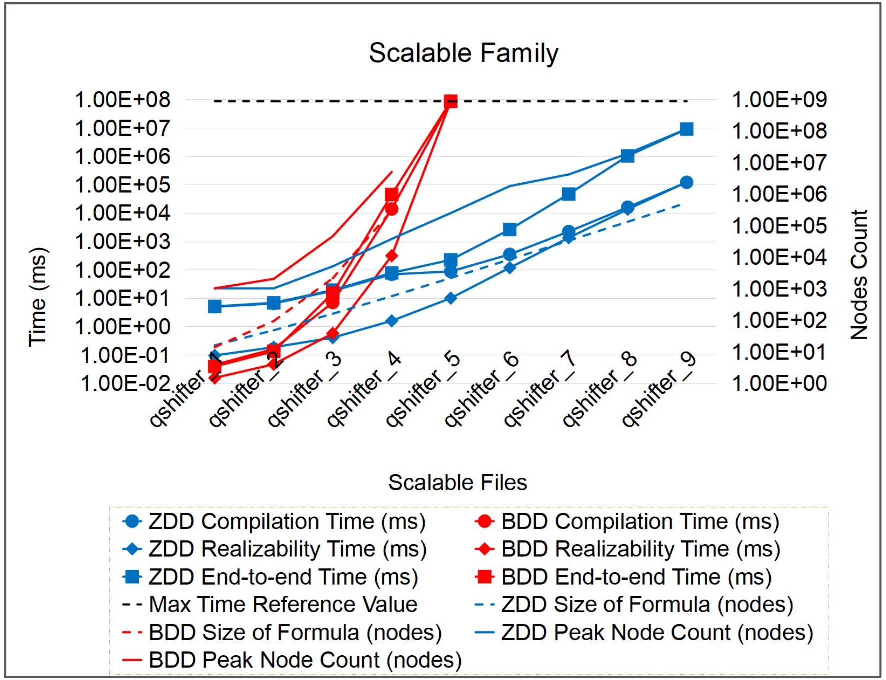
- Existential quantification in CNF can be computed by resolution:

$$\begin{aligned} (x_1 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_1) \wedge (x_1 \vee x_2) & \\ (x_1 \vee y_2 \vee \neg x_2) \wedge (x_1 \vee x_2) & \quad \text{(resolve } y_1\text{)} \\ (x_1 \vee x_2) & \quad \text{(resolve } y_1\text{)} \end{aligned}$$

- In general, resolving a variable can cause **quadratic** increase in the number of clauses.
- But resolution can be performed *symbolically* on the ZDD representation.
- Resulting ZDD can be much more compact, often avoiding quadratic blowup.

BDD-Based Synthesis vs. ZDD-Based Synthesis

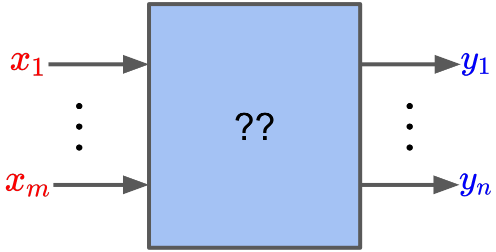




- **ZDDs** have more efficient compilation time
- In synthesis, performance depends on the family
- For some families, **ZDDs** outperform **BDDs** by an exponential factor

Extra Slides

What is Boolean Synthesis?



$$(x_1 \vee y_1 \vee y_2) \wedge (\neg x_2 \vee \neg y_1) \wedge (x_1 \vee x_2)$$

$$x_1, \ldots, x_m, y_1, \ldots, y_n \in \{0, 1\}$$

Given: specification (Boolean formula)

$$f(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

Compute: implementation (witness functions)

$$y_1 = g_1(x_1, \ldots, x_m)$$
$$\ldots$$
$$y_n = g_n(x_1, \ldots, x_m)$$

General Framework for Boolean Synthesis

1. Compile specification into an efficient representation (e.g. Binary Decision Diagram)

2. Compute realizability via existential quantification

$$R(x_1, \ldots, x_m) = \exists y_1, \ldots, y_n . f(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

$$R(x_1, \ldots, x_m) \equiv 1?$$

3. Construct witness functions

$$f(x_1, \ldots, x_m, y)$$

$$\downarrow$$

$$g(x_1, \ldots, x_m) = f(x_1, \ldots, x_m, 1)$$

Realizability and Witness Construction

Phase 1: Realizability

$$R(x_1, \ldots, x_m) = \exists y_1, \ldots, y_n . f(x_1, \ldots, x_m, y_1, \ldots, y_n)$$

$$R(x_1, \ldots, x_m) \equiv 1$$ Fully realizable

Phase 2: Witness Construction

$$f(x_1, \ldots, x_m, y)$$

$$g(x_1, \ldots, x_m) = f(x_1, \ldots, x_m, 1)$$

Default-1 Witness

$$g(x_1, \ldots, x_m) = \neg f(x_1, \ldots, x_m, 0)$$

Default-0 Witness

Synthesis Algorithm

<u>Phase 1: Realizability</u>

<u>Phase 2: Witness Construction</u>

$$f_n = f \quad \Longrightarrow \quad g_n = f_n[y_1 \mapsto g_1]\ldots[y_{n-1} \mapsto g_{n-1}][y_n \mapsto 1]$$

$$f_{n-1} = \exists y_n .\, f \quad \Longrightarrow \quad g_{n-1} = f_{n-1}[y_1 \mapsto g_1]\ldots[y_{n-2} \mapsto g_{n-2}][y_{n-1} \mapsto 1]$$

$$\ldots \qquad \ldots$$

$$f_2 = \exists y_3, \ldots, y_n .\, f \quad \Longrightarrow \quad g_2 = f_2[y_1 \mapsto g_1][y_2 \mapsto 1]$$

$$f_1 = \exists y_2, \ldots, y_n .\, f \quad \Longrightarrow \quad g_1 = f_1[y_1 \mapsto 1]$$

$$f_0 = \exists y_1, \ldots, y_n .\, f \quad \Longrightarrow \quad f_0 \equiv 1?$$

Synthesizing Witnesses

<u>Single output variable:</u>

$$f(x_1, \dots, x_m, y)$$

$$g(x_1, \dots, x_m) = f(x_1, \dots, x_m, 1)$$

Default-1 Witness

$$g(x_1, \dots, x_m) = \neg f(x_1, \dots, x_m, 0)$$

Default-0 Witness

<u>Multiple output variables:</u>

$$g_i(x_1, \dots, x_m) = \exists y_1, \dots, y_{i-1} . f(x_1, \dots, x_m, y_1, \dots, y_{i-1}, 1, \underbrace{g_{i+1}, \dots, g_n}_{\text{Previously-computed witnesses}})$$

ZDD Witness Construction

$$(\neg x_1 \vee \neg y) \wedge (x_1 \vee \neg x_2 \vee y) \wedge (x_2 \vee x_3 \vee \neg y) \wedge (x_1 \vee \neg x_3 \vee y)$$

$$(\neg x_1) \wedge (x_2 \wedge x_3)$$

Default-1 Witness (in Conjunctive Normal Form)

$$(\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$$

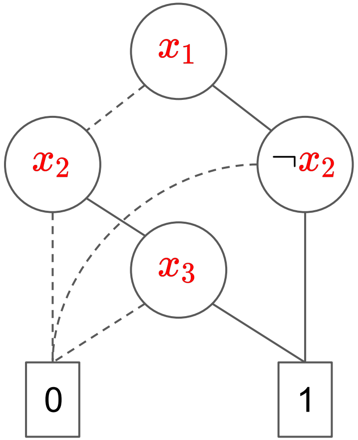
Default-0 Witness (in Disjunctive Normal Form)

Problem: How to perform substitutions? $\exists y_1, \ldots, y_{i-1} . f(x_1, \ldots, x_m, y_1, \ldots, y_{i-1}, 1, \underbrace{g_{i+1}, \ldots, g_n}_{???})$

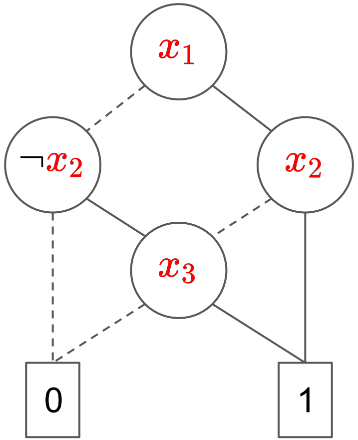
- **CNF witness:** easy to substitute into **positive** literals (formula remains in CNF)
- **DNF witness:** easy to substitute into **negative** literals (formula remains in CNF)

Knuth's Cross Operation

Solution: Convert ZDD of a **CNF** formula into ZDD for an equivalent **DNF** (or vice-versa).



Cross operation on ZDDs (Donald Knuth's *The Art of Computer Programming*, Volume 4, Pre-Fascicle 1B)



$(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)$

CNF witness:
substitute into positive literals

$(x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (\neg x_2 \wedge x_3)$

DNF witness:
substitute into negative literals