Knowledge Compilation for Boolean Functional Synthesis

Supratik Chakraborty

Indian Institute of Technology Bombay

Joint work with S. Akshay, Jatin Arora, Ajith John, S. Krishna, Divya Raghunathan, Shetal Shah

Boolean Relations and Functions

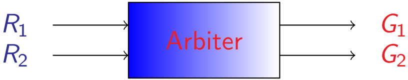
- Boolean functions: fundamental building blocks in computing.

Boolean Relations and Functions

- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs

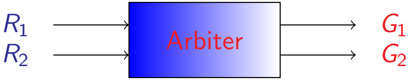
Boolean Relations and Functions

- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



Boolean Relations and Functions

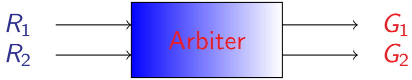
- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



  - $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2))$

Boolean Relations and Functions

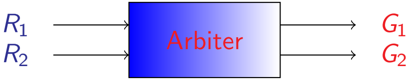
- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



  - $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2)) \wedge \neg(G_1 \wedge G_2)$

Boolean Relations and Functions

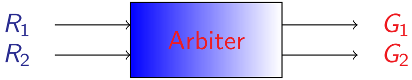
- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



- $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2)) \wedge \neg(G_1 \wedge G_2) \wedge (G_1 \rightarrow R_1)$

Boolean Relations and Functions

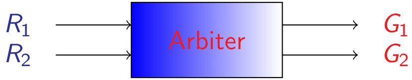
- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



- $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2)) \wedge \neg(G_1 \wedge G_2) \wedge (G_1 \rightarrow R_1)$
  $\wedge (G_2 \rightarrow R_2)$

Boolean Relations and Functions

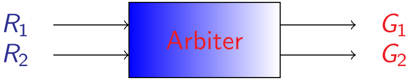
- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



    - $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2)) \wedge \neg(G_1 \wedge G_2) \wedge (G_1 \rightarrow R_1) \wedge (G_2 \rightarrow R_2)$
  - Doesn't specify how to obtain G_1, G_2 as functions of R_1, R_2.

Boolean Relations and Functions

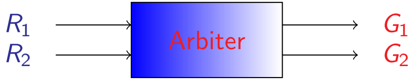
- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



  - $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2)) \wedge \neg(G_1 \wedge G_2) \wedge (G_1 \rightarrow R_1) \wedge (G_2 \rightarrow R_2)$
  - Doesn't specify how to obtain G_1, G_2 as functions of R_1, R_2.
- But we need them in **functional form**
  - Outputs as functions of inputs

Boolean Relations and Functions

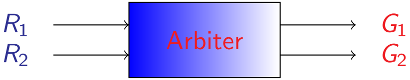
- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



    - $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2)) \wedge \neg(G_1 \wedge G_2) \wedge (G_1 \rightarrow R_1) \wedge (G_2 \rightarrow R_2)$
  - Doesn't specify how to obtain G_1, G_2 as functions of R_1, R_2.
- But we need them in **functional form**
  - Outputs as functions of inputs
  - Multiple solutions:
    - $G_1 = (R_1 \wedge \neg R_2)$, $G_2 = R_2$
    - $G_1 = R_1$, $G_2 = (\neg R_1 \wedge R_2)$

Boolean Relations and Functions

- Boolean functions: fundamental building blocks in computing.
- Often easy to specify **relationally**;
  - Relation between inputs and outputs
  - E.g. (Simplified) Arbiter



    - $((R_1 \vee R_2) \rightarrow (G_1 \vee G_2)) \wedge \neg(G_1 \wedge G_2) \wedge (G_1 \rightarrow R_1) \wedge (G_2 \rightarrow R_2)$
  - Doesn't specify how to obtain G_1, G_2 as functions of R_1, R_2.
- But we need them in **functional form**
  - Outputs as functions of inputs
  - Multiple solutions:
    - $G_1 = (R_1 \wedge \neg R_2)$, $G_2 = R_2$
    - $G_1 = R_1$, $G_2 = (\neg R_1 \wedge R_2)$

Boolean Functional Synthesis

Synthesizing Boolean functions from a relational specification.

Boolean Functional Synthesis (BFnS)

Formal definition

Given Boolean relation $\varphi(x_1, .., x_n, y_1, .., y_m)$

- x_i *input* variables (vector X)
- y_j *output* variables (vector Y)

Boolean Functional Synthesis (BFnS)

Formal definition

Given Boolean relation $\varphi(x_1, .., x_n, y_1, .., y_m)$

- x_i *input* variables (vector X)
- y_j *output* variables (vector Y)

Synthesize Boolean functions $F_j(X)$ for each y_j s.t.

$$\forall X(\ \exists y_1 \ldots y_m\ \varphi(X, y_1 \ldots y_m)\ \Leftrightarrow\ \varphi(X, F_1(X), \ldots F_m(X))\ )$$

Boolean Functional Synthesis (BFnS)

Formal definition

Given Boolean relation $\varphi(x_1, .., x_n, y_1, .., y_m)$

- x_i *input* variables (vector X)
- y_j *output* variables (vector Y)

Synthesize Boolean functions $F_j(X)$ for each y_j s.t.

$$\forall X(\ \exists y_1 \dots y_m \ \varphi(X, y_1 \dots y_m) \ \Leftrightarrow \ \varphi(X, F_1(X), \dots F_m(X)) \ )$$

$F_j(X)$ is also called a *Skolem function* for y_j in φ.

Boolean Functional Synthesis (BFnS)

Formal definition

Given Boolean relation $\varphi(x_1, .., x_n, y_1, .., y_m)$

- x_i *input* variables (vector X)
- y_j *output* variables (vector Y)

Synthesize Boolean functions $F_j(X)$ for each y_j s.t.

$$\forall X (\ \exists y_1 \ldots y_m \ \varphi(X, y_1 \ldots y_m) \ \Leftrightarrow \ \varphi(X, F_1(X), \ldots F_m(X))\ )$$

$F_j(X)$ is also called a *Skolem function* for y_j in φ.

- Uninteresting if $|X|$ is "small" (say, constant)
  - Tabulate with $2^{|X|}$ calls to $\mathrm{SAT}(\varphi(X, Y))$

Boolean Functional Synthesis (BFnS)

Formal definition

Given Boolean relation $\varphi(x_1, .., x_n, y_1, .., y_m)$

- x_i *input* variables (vector X)
- y_j *output* variables (vector Y)

Synthesize Boolean functions $F_j(X)$ for each y_j s.t.

$$\forall X(\ \exists y_1 \ldots y_m\ \varphi(X, y_1 \ldots y_m) \ \Leftrightarrow\ \varphi(X, F_1(X), \ldots F_m(X))\ )$$

$F_j(X)$ is also called a *Skolem function* for y_j in φ.

- Uninteresting if $|X|$ is "small" (say, constant)
  - Tabulate with $2^{|X|}$ calls to $\mathrm{SAT}(\varphi(X, Y))$
- What if $\forall X \exists Y\ \varphi(X, Y) = 0$ ("unrealizable" specification) ?

Boolean Functional Synthesis (BFnS)

Formal definition

Given Boolean relation $\varphi(x_1, .., x_n, y_1, .., y_m)$

- x_i *input* variables (vector X)
- y_j *output* variables (vector Y)

Synthesize Boolean functions $F_j(X)$ for each y_j s.t.

$$\forall X(\ \exists y_1 \ldots y_m\ \varphi(X, y_1 \ldots y_m) \ \Leftrightarrow\ \varphi(X, F_1(X), \ldots F_m(X))\ )$$

$F_j(X)$ is also called a *Skolem function* for y_j in φ.

- Uninteresting if $|X|$ is "small" (say, constant)
  - Tabulate with $2^{|X|}$ calls to $\mathrm{SAT}(\varphi(X, Y))$
- What if $\forall X \exists Y\ \varphi(X, Y) = 0$ ("unrealizable" specification) ?
  - Interesting as long as $\exists X \exists Y\ \varphi(X, Y) = 1$

Boolean Functional Synthesis (BFnS)

Formal definition

Given Boolean relation $\varphi(x_1, .., x_n, y_1, .., y_m)$

- x_i *input* variables (vector X)
- y_j *output* variables (vector Y)

Synthesize Boolean functions $F_j(X)$ for each y_j s.t.

$$\forall X(\ \exists y_1 \ldots y_m\ \varphi(X, y_1 \ldots y_m) \ \Leftrightarrow\ \varphi(X, F_1(X), \ldots F_m(X))\ )$$

$F_j(X)$ is also called a *Skolem function* for y_j in φ.

- Uninteresting if $|X|$ is "small" (say, constant)
  - Tabulate with $2^{|X|}$ calls to $\mathrm{SAT}(\varphi(X, Y))$
- What if $\forall X \exists Y\ \varphi(X, Y) = 0$ ("unrealizable" specification) ?
  - Interesting as long as $\exists X \exists Y\ \varphi(X, Y) = 1$
  - $F(X)$ must give right value of Y for all X s.t. $\exists Y \varphi(X, Y) = 1$
    - $F(X)$ inconsequential for other X

A challenging example: Bounded Integer Factorization

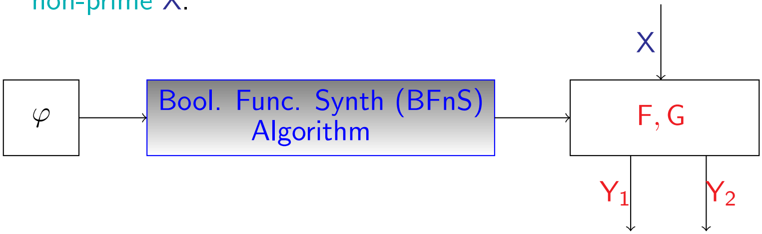
- n-bit integers Y_1, Y_2; $2n$ bit integer X

A challenging example: Bounded Integer Factorization

- n-bit integers Y_1, Y_2; $2n$ bit integer X
- Relational specification $\varphi(X, Y_1, Y_2)$
  - $(X = Y_1 \times_{[n]} Y_2) \wedge (Y_1 \neq 1_{[n]}) \wedge (Y_2 \neq 1_{[n]})$

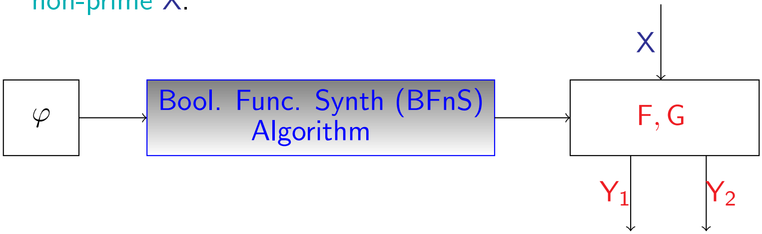
A challenging example: Bounded Integer Factorization

- n-bit integers Y_1, Y_2; $2n$ bit integer X
- Relational specification $\varphi(X, Y_1, Y_2)$
  - $(X = Y_1 \times_{[n]} Y_2) \wedge (Y_1 \neq 1_{[n]}) \wedge (Y_2 \neq 1_{[n]})$
- Synthesize $F(X), G(X)$ s.t. $\varphi(X, F(X), G(X)) = 1$ for all non-prime X.



A challenging example: Bounded Integer Factorization

- n-bit integers Y_1, Y_2; $2n$ bit integer X
- Relational specification $\varphi(X, Y_1, Y_2)$
  - $(X = Y_1 \times_{[n]} Y_2) \wedge (Y_1 \neq 1_{[n]}) \wedge (Y_2 \neq 1_{[n]})$
- Synthesize $F(X), G(X)$ s.t. $\varphi(X, F(X), G(X)) = 1$ for all non-prime X.



- For every non-prime X, finds non-trivial factors
- From prime X, values of $F(X)$ and $G(X)$ inconsequential.
  - $\exists Y_1, Y_2\ \varphi(X, Y_1, Y_2) = 0$ for such X.

Applications of Boolean Functional Synthesis

1. Cryptanalysis: Interesting but hard for synthesis!
2. Disjunctive decomposition of symbolic transition relations [Trivedi et al'02]
3. Quantifier elimination, of course!
   - $\exists Y\ \varphi(X, Y) \equiv \varphi(X, F(X))$
4. Certifying QBF-SAT solvers
   - Nice survey of applications by Shukla et al'19
5. Reactive controller synthesis
   - Synthesizing moves to stay within winning region
6. Program synthesis
   - Combinatorial sketching [Solar-Lezama et al'06, Srivastava et al'13]
   - Complete functional synthesis [Kuncak et al'10]
7. Repair/partial synthesis of circuits [Fujita et al'13]

Existing Approaches

1. Closely related to most general Boolean unifiers
   - Boole'1847, Lowenheim'1908, Macii'98
2. Extract Sk. functions from proof of validity of $\forall X \exists Y \varphi(X, Y)$
   - Bendetti'05, Jussilla et al'07, Balabanov et al'12, Heule et al'14
3. Using templates: Solar-Lezama et al'06, Srivastava et al'13
4. Self-substitution + function composition: Jiang'09, Trivedi'03
5. Synthesis from special normal form representation of specification
   - From ROBDDs: Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
   - **From SynNNF:** Akshay et al'09
6. Incremental determinization: Rabe et al'17,'18
7. Quantifier instantiation techniques in SMT solvers
   - Barrett et al'15, Bierre et al'17
8. Input/output component separation: C. et al'18
9. Guess/learn Skolem function candidate + check + repair
   - John et al'15, Akshay et al'17,'18,'20, Golia et al'20

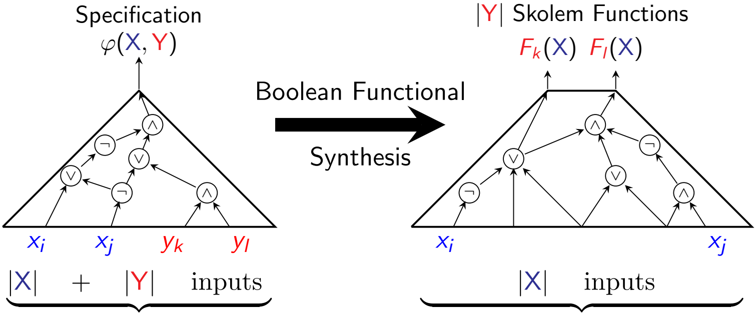
Representation of Specification & Skolem Functions

Representation of Specification & Skolem Functions

- Boolean circuit (DAG)
  - $\wedge$, $\vee$ and $\neg$ -labeled internal nodes, variable-labeled leaves

Representation of Specification & Skolem Functions

- Boolean circuit (DAG)
  - $\wedge$, $\vee$ and $\neg$ -labeled internal nodes, variable-labeled leaves
- Specification $\varphi(X, Y)$: $(|X| + |Y|)$-input, 1-output circuit
  - Other forms (ROBDD/CNF/DNF ...) efficiently converted
- Desired Sk. fn. vector $F(X)$: $|X|$-input, $|Y|$-output circuit
  - No additional restrictions (ROBDD/CNF/DNF ...)



How Hard is Boolean Functional Synthesis?

How Hard is Boolean Functional Synthesis?

BFnS is *NP*-hard

- Not surprising!

How Hard is Boolean Functional Synthesis?

BFnS is *NP*-hard

- Not surprising!

Can we always synthesize compact Skolem functions (perhaps spending exponential time)?

How Hard is Boolean Functional Synthesis?

BFnS is *NP*-hard

- Not surprising!

Can we always synthesize compact Skolem functions (perhaps spending exponential time)?

- Lower bound results in circuit-size typically refer to monotone circuits [Razbarov 1985; Alon and Boppana 1987]

How Hard is Boolean Functional Synthesis?

BFnS is *NP*-hard

- Not surprising!

Can we always synthesize compact Skolem functions (perhaps spending exponential time)?

- Lower bound results in circuit-size typically refer to monotone circuits [Razbarov 1985; Alon and Boppana 1987]
  - Monotone: Input $0 \rightarrow 1$ can't cause output $1 \rightarrow 0$
  - Skolem functions need not be monotone (reason for hope?)

How Hard is Boolean Functional Synthesis?

BFnS is *NP*-hard

- Not surprising!

Can we always synthesize compact Skolem functions (perhaps spending exponential time)?

- Lower bound results in circuit-size typically refer to monotone circuits [Razbarov 1985; Alon and Boppana 1987]
  - Monotone: Input $0 \rightarrow 1$ can't cause output $1 \rightarrow 0$
  - Skolem functions need not be monotone (reason for hope?)
- Bad news: [CAV2018]
  - Unless $\Pi_2^P = \Sigma_2^P$, there exist $\varphi(X, Y)$ for which Skolem function sizes are super-polynomial in $|\varphi|$.

How Hard is Boolean Functional Synthesis?

BFnS is *NP*-hard

- Not surprising!

Can we always synthesize compact Skolem functions (perhaps spending exponential time)?

- Lower bound results in circuit-size typically refer to monotone circuits [Razbarov 1985; Alon and Boppana 1987]
  - Monotone: Input $0 \rightarrow 1$ can't cause output $1 \rightarrow 0$
  - Skolem functions need not be monotone (reason for hope?)
- Bad news: [CAV2018]
  - Unless $\Pi_2^P = \Sigma_2^P$, there exist $\varphi(X, Y)$ for which Skolem function sizes are super-polynomial in $|\varphi|$.
  - Unless non-uniform exponential-time hypothesis fails, there exist $\varphi(X, Y)$ for which Skolem function sizes are exponential in $|\varphi|$.

How Hard is Boolean Functional Synthesis?

BFnS is *NP*-hard

- Not surprising!

Can we always synthesize compact Skolem functions (perhaps spending exponential time)?

- Lower bound results in circuit-size typically refer to monotone circuits [Razbarov 1985; Alon and Boppana 1987]
  - Monotone: Input $0 \to 1$ can't cause output $1 \to 0$
  - Skolem functions need not be monotone (reason for hope?)
- Bad news: [CAV2018]
  - Unless $\Pi_2^P = \Sigma_2^P$, there exist $\varphi(X, Y)$ for which Skolem function sizes are super-polynomial in $|\varphi|$.
  - Unless non-uniform exponential-time hypothesis fails, there exist $\varphi(X, Y)$ for which Skolem function sizes are exponential in $|\varphi|$.

Efficient algorithms for Boolean functional synthesis unlikely

Some good news [CAV2018, FMCAD2019]

If φ is represented in special normal form, synthesis solvable in polynomial (in $|\varphi|$) time and space.

Some good news [CAV2018, FMCAD2019]

If φ is represented in special normal form, synthesis solvable in polynomial (in $|\varphi|$) time and space.

- Synthesis Negation Normal Form (SynNNF)
  - Subsumes well-known forms like ROBDD, DNNF, ...

Some good news [CAV2018, FMCAD2019]

If φ is represented in special normal form, synthesis solvable in polynomial (in $|\varphi|$) time and space.

- Synthesis Negation Normal Form (SynNNF)
  - Subsumes well-known forms like ROBDD, DNNF, ...
- Caveat: (Conditional) Lower bounds imply compilation to SynNNF inefficient in general.

Some good news [CAV2018, FMCAD2019]

If φ is represented in special normal form, synthesis solvable in polynomial (in $|\varphi|$) time and space.

- Synthesis Negation Normal Form (SynNNF)
  - Subsumes well-known forms like ROBDD, DNNF, ...
- Caveat: (Conditional) Lower bounds imply compilation to SynNNF inefficient in general.
- Silver Lining: Experimental evidence shows (refined) SynNNF common in practice

Some good news [CAV2018, FMCAD2019]

If φ is represented in special normal form, synthesis solvable in polynomial (in $|\varphi|$) time and space.

- Synthesis Negation Normal Form (SynNNF)
  - Subsumes well-known forms like ROBDD, DNNF, ...
- Caveat: (Conditional) Lower bounds imply compilation to SynNNF inefficient in general.
- Silver Lining: Experimental evidence shows (refined) SynNNF common in practice
- Compilation to ROBDD (using any variable order), FDD or DNNF already yields SynNNF
  - Mature compilation tools exist for these normal forms

Some good news [CAV2018, FMCAD2019]

If φ is represented in special normal form, synthesis solvable in polynomial (in $|\varphi|$) time and space.

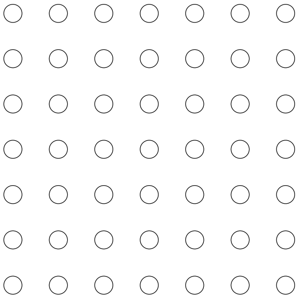
- Synthesis Negation Normal Form (SynNNF)
  - Subsumes well-known forms like ROBDD, DNNF, ...
- Caveat: (Conditional) Lower bounds imply compilation to SynNNF inefficient in general.
- Silver Lining: Experimental evidence shows (refined) SynNNF common in practice
- Compilation to ROBDD (using any variable order), FDD or DNNF already yields SynNNF
  - Mature compilation tools exist for these normal forms
  - Efficient synthesis doesn't require full capabilities of these stronger normal forms.

Easy Case of Synthesis ($|Y| = 1$)

Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$

Easy Case of Synthesis ($|Y| = 1$)

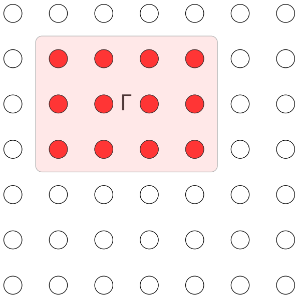
Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$



— Set of all valuations of X.

Easy Case of Synthesis ($|Y| = 1$)

Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$

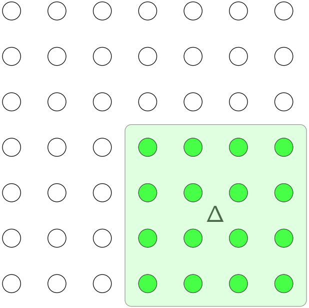


— Can't set y to 1 to satisfy φ: $\Gamma(X) \triangleq \neg\varphi(X, y)[y \mapsto 1]$

E.g. If $\varphi \equiv (x_1 \vee y) \wedge (x_1 \vee x_2 \vee \neg y)$, then
$\Gamma(X) = \neg((x_1 \vee 1) \wedge (x_1 \vee x_2 \vee 0)) = \neg(x_1 \vee x_2) = \neg x_1 \wedge \neg x_2$

Easy Case of Synthesis ($|Y| = 1$)

Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$

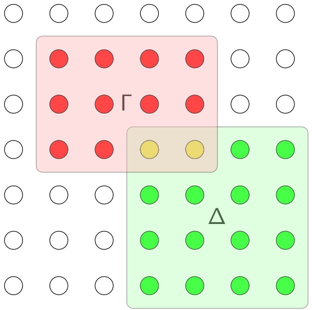


— Can't set y to 0 to satisfy φ: $\Delta(X) \triangleq \neg\varphi(X, y)[y \mapsto 0]$

E.g. If $\varphi \equiv (x_1 \vee y) \wedge (x_1 \vee x_2 \vee \neg y)$, then
$\Delta(X) = \neg((x_1 \vee 0) \wedge (x_1 \vee x_2 \vee 1)) = \neg x_1$

Easy Case of Synthesis ($|Y| = 1$)

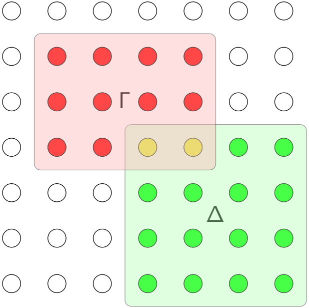
Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$



— Can't set y to 1 to satisfy φ: $\Gamma(X) \triangleq \neg\varphi(X, y)[y \mapsto 1]$
— Can't set y to 0 to satisfy φ: $\Delta(X) \triangleq \neg\varphi(X, y)[y \mapsto 0]$

Easy Case of Synthesis ($|Y| = 1$)

Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$



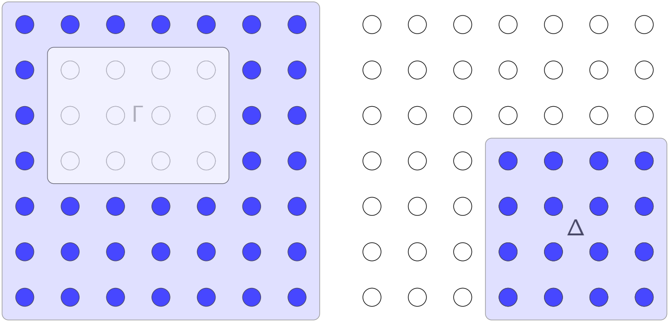
Lemma [Trivedi'03, Jiang'09,Fried et al'16]

Every Skolem function for y in φ must

- Evaluate to 1 in $(\Delta \setminus \Gamma)$ and to 0 in $(\Gamma \setminus \Delta)$
- Be an **interpolant** of $(\Delta \setminus \Gamma)$ and $(\Gamma \setminus \Delta)$

Easy Case of Synthesis ($|Y| = 1$)

Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$

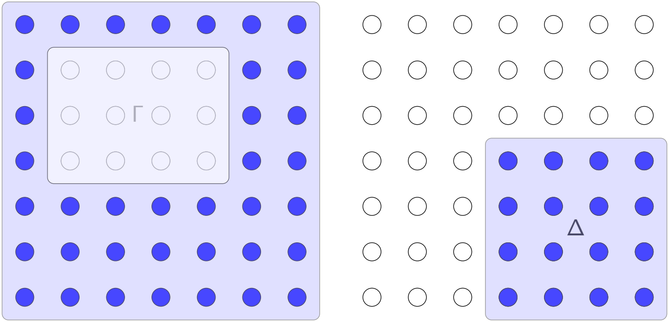


— Specific interpolants of $(\Delta \setminus \Gamma)$ & $(\Gamma \setminus \Delta)$

- $\neg\Gamma \triangleq \varphi(X, y)[y \mapsto 1] \equiv \varphi(X, 1)$
- $\Delta \triangleq \neg\varphi(X, y)[y \mapsto 0] \equiv \neg\varphi(X, 0)$.

Easy Case of Synthesis ($|Y| = 1$)

Find $F(X)$ such that $\exists y\ \varphi(X, y) \equiv \varphi(X, F(X))$



— Specific interpolants of $(\Delta \setminus \Gamma)$ & $(\Gamma \setminus \Delta)$

- $\neg\Gamma \triangleq \varphi(X, y)[y \mapsto 1] \equiv \varphi(X, 1)$: Easy solution for 1 output var
- $\Delta \triangleq \neg\varphi(X, y)[y \mapsto 0] \equiv \neg\varphi(X, 0)$.

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

- **Yes!!!**
  - Informally, synthesize $F_2(X)$ s.t. **"we can always find y_1 to satisfy $\varphi(X, y_1, F_2(X))$"**

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

- **Yes!!!**
  - Informally, synthesize $F_2(X)$ s.t. **"we can always find** y_1 **to satisfy** $\varphi(X, y_1, F_2(X))$**"**
  - Formally, $\varphi_1(X, y_2) \equiv \exists y_1\, \varphi(X, y_1, y_2)$ is the new spec

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

- **Yes!!!**
  - Informally, synthesize $F_2(X)$ s.t. **"we can always find** y_1 **to satisfy** $\varphi(X, y_1, F_2(X))$**"**
  - Formally, $\varphi_1(X, y_2) \equiv \exists y_1\, \varphi(X, y_1, y_2)$ is the new spec
- Synthesize $F_2(X)$ from $\varphi_1(X, y_2)$
  - Example: $\varphi_1(X, y_2) \quad \equiv \quad (x_1 \vee x_2 \vee \neg y_2)$

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

- **Yes!!!**
  - Informally, synthesize $F_2(X)$ s.t. **"we can always find** y_1 **to satisfy** $\varphi(X, y_1, F_2(X))$**"**
  - Formally, $\varphi_1(X, y_2) \equiv \exists y_1\, \varphi(X, y_1, y_2)$ is the new spec
- Synthesize $F_2(X)$ from $\varphi_1(X, y_2)$
  - Example: $\varphi_1(X, y_2) \equiv (x_1 \vee x_2 \vee \neg y_2)$
    $F_2(X) \equiv \varphi_1(X, 1) \equiv (x_1 \vee x_2)$

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

- **Yes!!!**
  - Informally, synthesize $F_2(X)$ s.t. **"we can always find** y_1 **to satisfy** $\varphi(X, y_1, F_2(X))$**"**
  - Formally, $\varphi_1(X, y_2) \equiv \exists y_1\, \varphi(X, y_1, y_2)$ is the new spec
- Synthesize $F_2(X)$ from $\varphi_1(X, y_2)$
  - Example: $\varphi_1(X, y_2) \quad \equiv \quad (x_1 \vee x_2 \vee \neg y_2)$
    $F_2(X) \equiv \varphi_1(X, 1) \equiv (x_1 \vee x_2)$
- Synthesize $F_1(X)$ from $\varphi(X, y_1, F_2(X))$

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

- **Yes!!!**
  - Informally, synthesize $F_2(X)$ s.t. **"we can always find** y_1 **to satisfy** $\varphi(X, y_1, F_2(X))$**"**
  - Formally, $\varphi_1(X, y_2) \equiv \exists y_1\, \varphi(X, y_1, y_2)$ is the new spec
- Synthesize $F_2(X)$ from $\varphi_1(X, y_2)$
  - Example: $\varphi_1(X, y_2) \equiv (x_1 \vee x_2 \vee \neg y_2)$
    $F_2(X) \equiv \varphi_1(X, 1) \equiv (x_1 \vee x_2)$
- Synthesize $F_1(X)$ from $\varphi(X, y_1, F_2(X))$
  - $\varphi(X, y_1, (x_1 \vee x_2)) \equiv (x_1 \vee x_2) \oplus y_1$

What Happens If $|Y| > 1$?

Suppose $\varphi(X, Y) \equiv (x_1 \vee x_2 \vee y_1) \wedge (y_1 \oplus y_2)$

- $F_2(X)$ **must be** $\neg F_1(X)$ in all solutions
  - $F_2(X)$ and $F_1(X)$ cannot be synthesized independently.

Can we reduce to a synthesis problem on 1 output?

- **Yes!!!**
  - Informally, synthesize $F_2(X)$ s.t. **"we can always find** y_1 **to satisfy** $\varphi(X, y_1, F_2(X))$**"**
  - Formally, $\varphi_1(X, y_2) \equiv \exists y_1\, \varphi(X, y_1, y_2)$ is the new spec
- Synthesize $F_2(X)$ from $\varphi_1(X, y_2)$
  - Example: $\varphi_1(X, y_2) \equiv (x_1 \vee x_2 \vee \neg y_2)$
    $F_2(X) \equiv \varphi_1(X, 1) \equiv (x_1 \vee x_2)$
- Synthesize $F_1(X)$ from $\varphi(X, y_1, F_2(X))$
  - $\varphi(X, y_1, (x_1 \vee x_2)) \equiv (x_1 \vee x_2) \oplus y_1$
    $F_1(X) \equiv \varphi(X, 1, (x_1 \vee x_2)) \equiv \neg(x_1 \vee x_2)$

Generalizing to Arbitrarily Many Outputs (Y)

Fix a linear ordering of outputs: $y_1 \prec y_2 \prec \cdots \prec y_m$

Generalizing to Arbitrarily Many Outputs (Y)

Fix a linear ordering of outputs: $y_1 \prec y_2 \prec \cdots \prec y_m$

- Synthesize $F_m(X)$ from $\exists \boxed{y_1 \ldots y_{m-1}} \varphi(X, \boxed{y_1, \ldots y_{m-1}}, y_m)$

Generalizing to Arbitrarily Many Outputs (Y)

Fix a linear ordering of outputs: $y_1 \prec y_2 \prec \cdots \prec y_m$

- Synthesize $F_m(X)$ from $\exists \boxed{y_1 \ldots y_{m-1}} \varphi(X, \boxed{y_1, \ldots y_{m-1}}, y_m)$
- Synthesize $F_{m-1}(X)$ from $\exists \boxed{y_1 \ldots y_{m-2}} \varphi(X, \boxed{y_1, \ldots y_{m-2}}, y_{m-1}, \boxed{\boxed{F_m(X)}})$

Generalizing to Arbitrarily Many Outputs (Y)

Fix a linear ordering of outputs: $y_1 \prec y_2 \prec \cdots \prec y_m$

- Synthesize $F_m(X)$ from $\exists \boxed{y_1 \ldots y_{m-1}} \varphi(X, \boxed{y_1, \ldots y_{m-1}}, y_m)$
- Synthesize $F_{m-1}(X)$ from

  $\exists \boxed{y_1 \ldots y_{m-2}} \varphi(X, \boxed{y_1, \ldots y_{m-2}}, y_{m-1}, \boxed{\boxed{F_m(X)}})$
- $\vdots$
- Synthesize $F_1(X)$ from $\varphi(X, y_1, \boxed{\boxed{F_2(X), \ldots F_m(X)}})$

Generalizing to Arbitrarily Many Outputs (Y)

Fix a linear ordering of outputs: $y_1 \prec y_2 \prec \cdots \prec y_m$

- Synthesize $F_m(X)$ from $\exists \boxed{y_1 \ldots y_{m-1}} \varphi(X, \boxed{y_1, \ldots y_{m-1}}, y_m)$
- Synthesize $F_{m-1}(X)$ from
  $\exists \boxed{y_1 \ldots y_{m-2}} \varphi(X, \boxed{y_1, \ldots y_{m-2}}, y_{m-1}, \boxed{\boxed{F_m(X)}})$
- $\vdots$
- Synthesize $F_1(X)$ from $\varphi(X, y_1, \boxed{\boxed{F_2(X), \ldots F_m(X)}})$

Generalizing to Arbitrarily Many Outputs (Y)

Fix a linear ordering of outputs: $y_1 \prec y_2 \prec \cdots \prec y_m$

- Synthesize $F_m(X)$ from $\exists \boxed{y_1 \ldots y_{m-1}} \varphi(X, \boxed{y_1, \ldots y_{m-1}}, y_m)$
- Synthesize $F_{m-1}(X)$ from

  $\exists \boxed{y_1 \ldots y_{m-2}} \varphi(X, \boxed{y_1, \ldots y_{m-2}}, y_{m-1}, \boxed{\boxed{F_m(X)}})$
- $\vdots$
- Synthesize $F_1(X)$ from $\varphi(X, y_1, \boxed{\boxed{F_2(X), \ldots F_m(X)}})$

Centrality of quantifying y_i's & composing $F_j(X)$'s in **given order**

Generalizing to Arbitrarily Many Outputs (Y)

Fix a linear ordering of outputs: $y_1 \prec y_2 \prec \cdots \prec y_m$

- Synthesize $F_m(X)$ from $\exists y_1 \ldots y_{m-1}\, \varphi(X, y_1, \ldots y_{m-1}, y_m)$
- Synthesize $F_{m-1}(X)$ from $\exists y_1 \ldots y_{m-2}\, \varphi(X, y_1, \ldots y_{m-2}, y_{m-1}, F_m(X))$
- $\vdots$
- Synthesize $F_1(X)$ from $\varphi(X, y_1, F_2(X), \ldots F_m(X))$

Centrality of quantifying y_i's & composing $F_j(X)$'s in **given order**

How do we compute

$\exists y_1, \ldots y_i\, \varphi(X, y_1, \ldots y_i, F_{i+1}(X), \ldots F_m(X))$ **efficiently?**

Synthesis from ROBDDs

- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs

Synthesis from ROBDDs

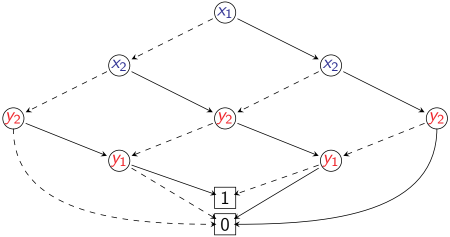
- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs
- Key idea: Input (X) first ordering of variables

Synthesis from ROBDDs

- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs
- Key idea: Input (X) first ordering of variables
  - Allows easy quantification of y_i's and composition of $F_j(X)$'s in **BDD order of y_i's**

Synthesis from ROBDDs

- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs
- Key idea: Input (X) first ordering of variables
  - Allows easy quantification of y_i's and composition of $F_j(X)$'s in **BDD order of y_i's**

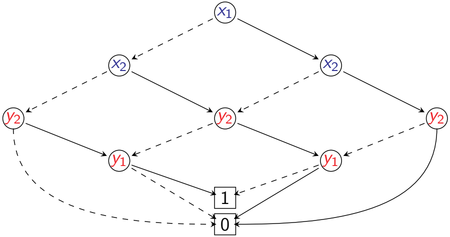


Synthesis from ROBDDs

- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs
- Key idea: Input (X) first ordering of variables
  - Allows easy quantification of y_i's and composition of $F_j(X)$'s in **BDD order of y_i's**

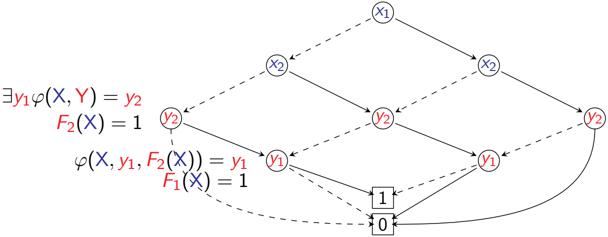
$\exists y_1 \varphi(X, Y) = y_2$

$F_2(X) = 1$



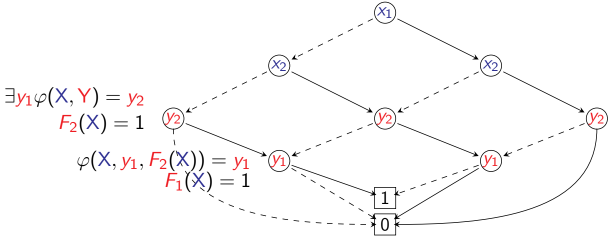
Synthesis from ROBDDs

- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs
- Key idea: Input (X) first ordering of variables
  - Allows easy quantification of y_i's and composition of $F_j(X)$'s in **BDD order of y_i's**



Synthesis from ROBDDs

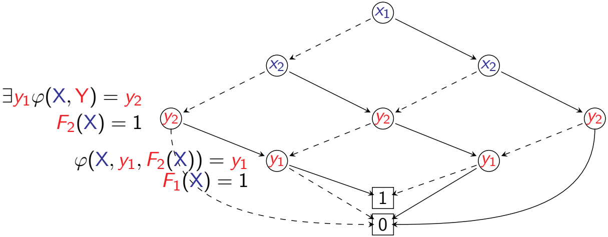
- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs
- Key idea: Input (X) first ordering of variables
  - Allows easy quantification of y_i's and composition of $F_j(X)$'s in **BDD order of y_i's**



- Significant optimizations in Fried et al'16, Tabajara et al'17

Synthesis from ROBDDs

- Tronci'98, Kukula et al'00, Kuncak et al'10, Fried et al'16, Tabajara et al'17
- Spec $\varphi(X, Y)$ as ROBDD, Skolem functions as ROBDDs
- Key idea: Input (X) first ordering of variables
  - Allows easy quantification of y_i's and composition of $F_j(X)$'s in **BDD order of y_i's**



- Significant optimizations in Fried et al'16, Tabajara et al'17
- Spec ROBDD can be exp. larger with input-first ordering
  - $\varphi(X, Y) \equiv \bigwedge_{i=1}^{n} (x_i \Leftrightarrow y_i)$
  - Size $\Omega(2^n)$ with input-first ordering, $\Theta(n)$ with interleaved input-output ordering,

Going beyond Input-First Ordered ROBDDs

- ROBDDs have much more structure than we need.

Going beyond Input-First Ordered ROBDDs

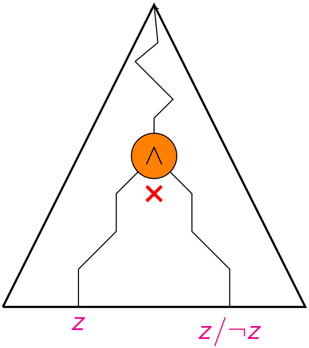
- ROBDDs have much more structure than we need.
- What if we're fine with Skolem functions as circuits, not as ROBDDs?
  - Can we avoid exponential blow-ups?

Going beyond Input-First Ordered ROBDDs

- ROBDDs have much more structure than we need.
- What if we're fine with Skolem functions as circuits, not as ROBDDs?
    - Can we avoid exponential blow-ups?
- What if the best variable order for the specification is not input-first?
    - Can we synthesize efficiently for such specs?

Decomposable NNF (DNNF) and weak DNNF are better!

$\varphi(X, Y)$ in DNNF except on X

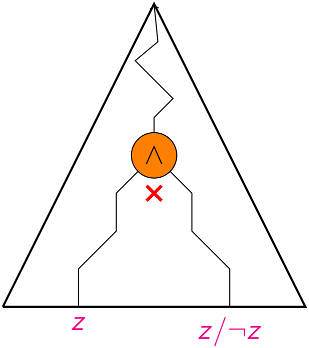


z is y_j

Disallowed paths

Decomposable NNF (DNNF) and weak DNNF are better!

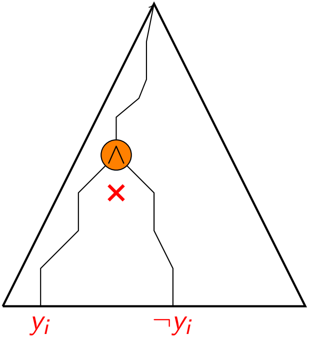
$\varphi(X, Y)$ in DNNF except on X



z is y_j

Disallowed paths

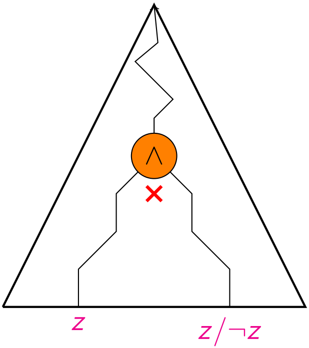
$\varphi(X, Y)$ in wDNNF except on X



(Dis)allowed paths

Decomposable NNF (DNNF) and weak DNNF are better!

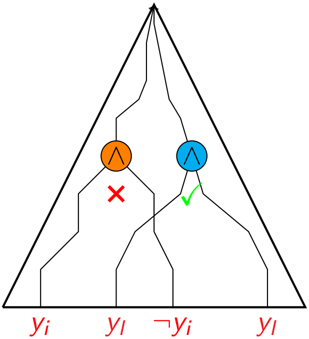
$\varphi(X, Y)$ in DNNF except on X



z is y_j

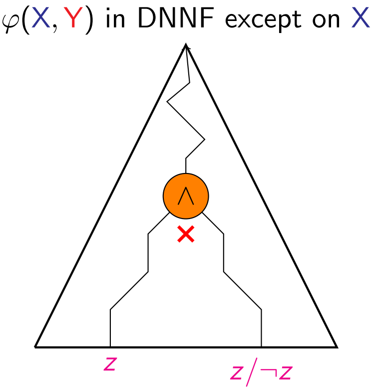
Disallowed paths

$\varphi(X, Y)$ in wDNNF except on X



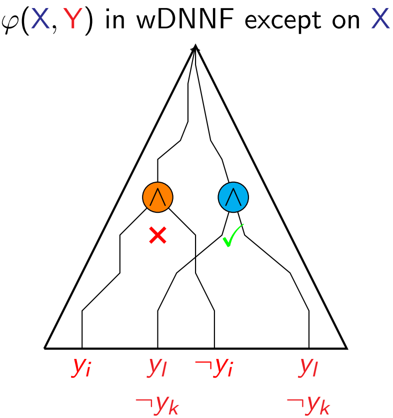
(Dis)allowed paths

Decomposable NNF (DNNF) and weak DNNF are better!



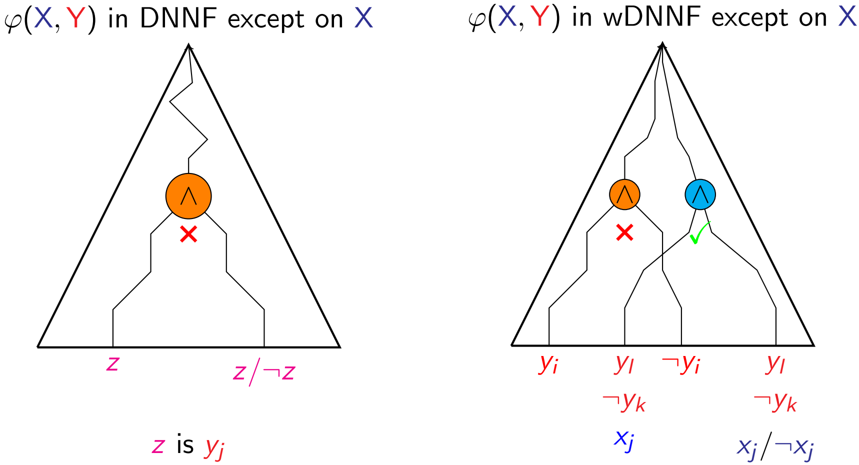
z is y_j

Disallowed paths



(Dis)allowed paths

Decomposable NNF (DNNF) and weak DNNF are better!

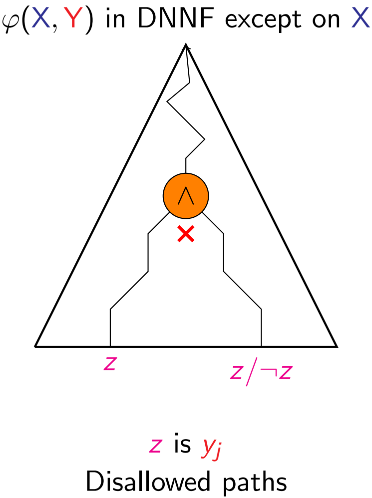


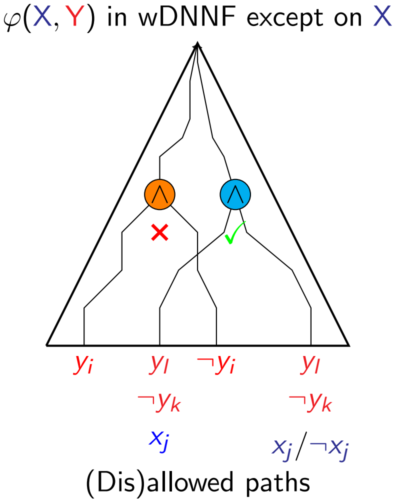
z is y_j

Disallowed paths

(Dis)allowed paths

Decomposable NNF (DNNF) and weak DNNF are better!





Allows quantification of y_i's and composition of F_j's in arbitrary order

Current Talk

Is there a weaker (than wDNNF) representation form of φ that guarantees poly-time (in $|\varphi|$) synthesis?

Current Talk

Is there a weaker (than wDNNF) representation form of φ that guarantees poly-time (in $|\varphi|$) synthesis?

- **YES: Synthesis Negation Normal Form (SynNNF)**
- Subsumes and exponentially more succinct than BDD/DNNF/wDNNF/...

Current Talk

Is there a weaker (than wDNNF) representation form of φ that guarantees poly-time (in $|\varphi|$) synthesis?

- **YES: Synthesis Negation Normal Form (SynNNF)**
- Subsumes and exponentially more succinct than BDD/DNNF/wDNNF/...

Can we synthesize Skolem functions from a "simplified" specification?

Current Talk

Is there a weaker (than wDNNF) representation form of φ that guarantees poly-time (in $|\varphi|$) synthesis?

- **YES: Synthesis Negation Normal Form (SynNNF)**
- Subsumes and exponentially more succinct than BDD/DNNF/wDNNF/...

Can we synthesize Skolem functions from a "simplified" specification?

- **YES:** Folklore wisdom
- **Formalized as refinement w.r.t. synthesis**

Current Talk

Is there a weaker (than wDNNF) representation form of φ that guarantees poly-time (in $|\varphi|$) synthesis?

- **YES: Synthesis Negation Normal Form (SynNNF)**
- Subsumes and exponentially more succinct than BDD/DNNF/wDNNF/...

Can we synthesize Skolem functions from a "simplified" specification?

- **YES:** Folklore wisdom
- **Formalized as refinement w.r.t. synthesis**

Can we algorithmically compile φ to a refined SynNNF spec $\widetilde{\varphi}$?

Current Talk

Is there a weaker (than wDNNF) representation form of φ that guarantees poly-time (in $|\varphi|$) synthesis?

- **YES: Synthesis Negation Normal Form (SynNNF)**
- Subsumes and exponentially more succinct than BDD/DNNF/wDNNF/...

Can we synthesize Skolem functions from a "simplified" specification?

- **YES:** Folklore wisdom
- **Formalized as refinement w.r.t. synthesis**

Can we algorithmically compile φ to a refined SynNNF spec $\widetilde{\varphi}$?

- **YES:** Super-polynomial time in worst-case
- **Practical performance promising!**

Classical Knowledge Compilation

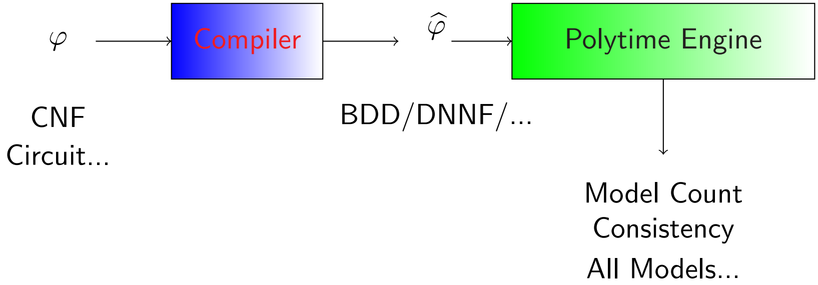
Wikipedia

... a family of approaches for addressing the intractability of a number of artificial intelligence problems. A propositional model is compiled in an off-line phase in order to support some queries in polytime.

Classical Knowledge Compilation

Wikipedia

... a family of approaches for addressing the intractability of a number of artificial intelligence problems. A propositional model is compiled in an off-line phase in order to support some queries in polytime.



Knowledge Compilation for Synthesis

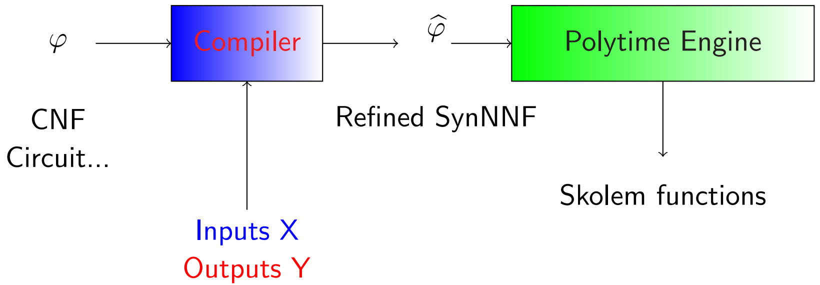
Our Definition

... a family of approaches for addressing the intractability of synthesis problems. A propositional model is compiled in an off-line phase in order to support some queries in polytime.

Knowledge Compilation for Synthesis

Our Definition

… a family of approaches for addressing the intractability of synthesis problems. A propositional model is compiled in an off-line phase in order to support some queries in polytime.

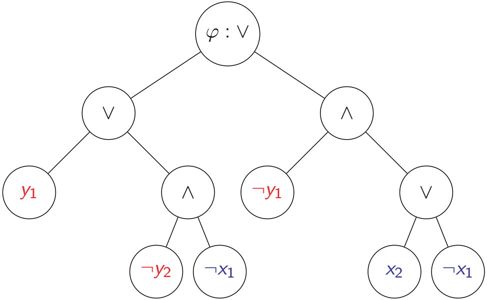


Towards a normal form for efficient synthesis

- Represent $\varphi(x_1, .., x_n, y_1, .., y_m)$ as NNF DAG
  - Boolean circuit, $\wedge$ and $\vee$ at internal nodes, $\neg$ only at leaves

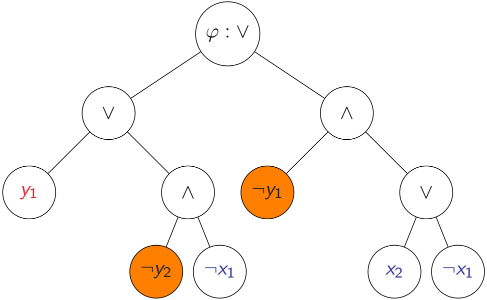
Towards a normal form for efficient synthesis

- Represent $\varphi(x_1, .., x_n, y_1, .., y_m)$ as NNF DAG
  - Boolean circuit, $\wedge$ and $\vee$ at internal nodes, $\neg$ only at leaves



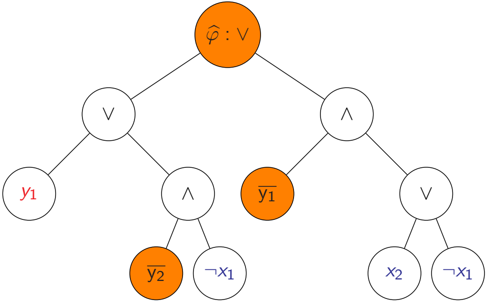
Towards a normal form for efficient synthesis

- Represent $\varphi(x_1, .., x_n, y_1, .., y_m)$ as NNF DAG
  - Boolean circuit, $\wedge$ and $\vee$ at internal nodes, $\neg$ only at leaves



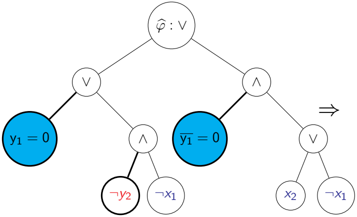
Towards a normal form for efficient synthesis

- Represent $\varphi(x_1, .., x_n, y_1, .., y_m)$ as NNF DAG
  - Boolean circuit, $\wedge$ and $\vee$ at internal nodes, $\neg$ only at leaves

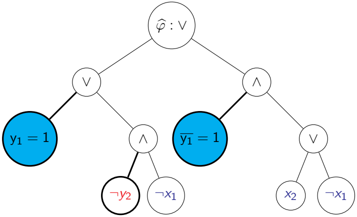


- **Positive form of specification:**
  $\widehat{\varphi}(x_1, \ldots x_n,\ y_1, \ldots y_m,\ \overline{y_1}, \ldots \overline{y_m})$
  - Monotone w.r.t all y_i and $\overline{y_i}$

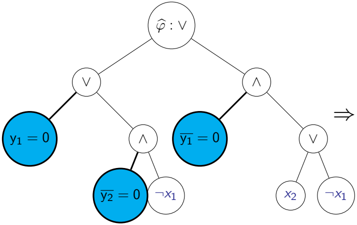
Simple properties of $\widehat{\varphi}$



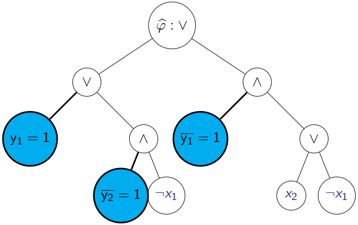
$\Rightarrow \quad \exists y_1 \varphi \quad \Rightarrow$



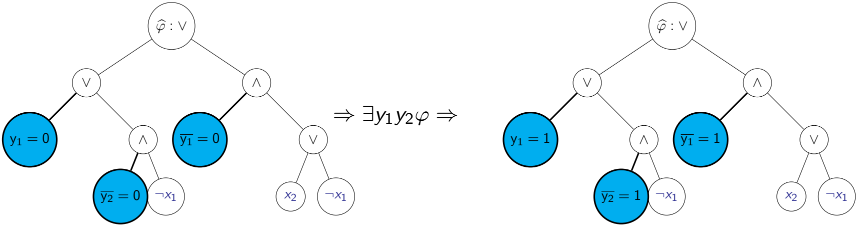
Simple properties of $\widehat{\varphi}$



$\Rightarrow \exists y_1 y_2 \varphi \Rightarrow$



Simple properties of $\widehat{\varphi}$



- $\widehat{\varphi}(x_1 \ldots x_n, \overbrace{0..0}^{i}, y_{i+1} \ldots y_m, \overbrace{0..0}^{i}, \neg y_{i+1} \ldots \neg y_m) \Rightarrow \exists y_1 \ldots y_i \; \varphi(\ldots)$
- $\widehat{\varphi}(x_1 \ldots x_n, \overbrace{1..1}^{i}, y_{i+1} \ldots y_m, \overbrace{1..1}^{i}, \neg y_{i+1} \ldots \neg y_m) \Leftarrow \exists y_1 \ldots y_i \; \varphi(\ldots)$

Iterated reducts of $\widehat{\varphi}$

Given

- Positive form of spec $\widehat{\varphi}(x_1, \ldots x_n, y_1, \ldots y_m, \overline{y_1}, \ldots \overline{y_m})$
- Linear order of outputs $y_1 \prec \cdots \prec y_m$

Iterated reducts of $\widehat{\varphi}$

Given

- Positive form of spec $\widehat{\varphi}(x_1, \ldots x_n, y_1, \ldots y_m, \overline{y_1}, \ldots \overline{y_m})$
- Linear order of outputs $y_1 \prec \cdots \prec y_m$

Define $[\widehat{\varphi}]_i$ as

$$\widehat{\varphi}(x_1, \ldots x_n, \overbrace{1..1}^{i-1}, \ \mathrm{y_i}, \ y_{i+1} \ldots y_m, \overbrace{1..1}^{i-1}, \ \overline{\mathrm{y_i}}, \ \neg y_{i+1} \ldots \neg y_m).$$

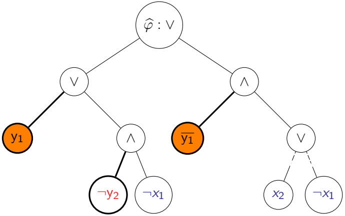
Iterated reducts of $\widehat{\varphi}$

Given

- Positive form of spec $\widehat{\varphi}(x_1, \ldots x_n, y_1, \ldots y_m, \overline{y_1}, \ldots \overline{y_m})$
- Linear order of outputs $y_1 \prec \cdots \prec y_m$

Define $[\widehat{\varphi}]_i$ as

$$\widehat{\varphi}(x_1, \ldots x_n, \overbrace{1..1}^{i-1}, \ \mathrm{y_i}, \ y_{i+1} \ldots y_m, \overbrace{1..1}^{i-1}, \ \overline{\mathrm{y_i}}, \ \neg y_{i+1} \ldots \neg y_m).$$



$[\widehat{\varphi}]_1(x_1, x_2, y_1, y_2, \overline{y_1})$

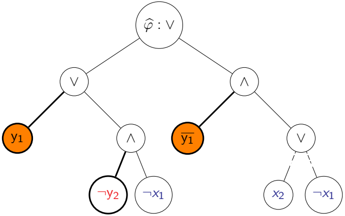
Iterated reducts of $\widehat{\varphi}$

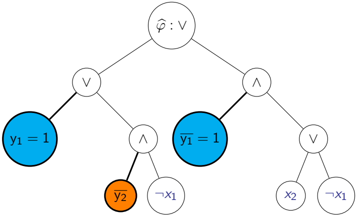
Given

- Positive form of spec $\widehat{\varphi}(x_1, \dots x_n, y_1, \dots y_m, \overline{y_1}, \dots \overline{y_m})$
- Linear order of outputs $y_1 \prec \cdots \prec y_m$

Define $[\widehat{\varphi}]_i$ as

$$\widehat{\varphi}(x_1, \dots x_n, \overbrace{1..1}^{i-1}, \ \mathrm{y_i}, \ y_{i+1} \dots y_m, \overbrace{1..1}^{i-1}, \ \overline{\mathrm{y_i}}, \ \neg y_{i+1} \dots \neg y_m).$$





$[\widehat{\varphi}]_1(x_1, x_2, y_1, y_2, \overline{y_1})$

$[\widehat{\varphi}]_2(x_1, x_2, y_2, \overline{y_2})$

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \;\Rightarrow\; \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1}$

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \;\Rightarrow\; \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \quad \Leftrightarrow \; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \Rightarrow \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \;\Rightarrow\; \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \;\not\Leftarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \Rightarrow \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \not\Leftarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?
- Exactly when
  - $[\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1} = 1$
  - $\exists y_1 \varphi(X, Y) \Leftrightarrow \varphi\,|_{y_1=1} \lor \varphi\,|_{y_1=0} = 0$

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \;\Rightarrow\; \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \;\not\Leftarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?
- Exactly when
  - $[\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1} \;=\; 1$
  - $\exists y_1 \varphi(X, Y) \;\Leftrightarrow\; \varphi\,|_{y_1=1} \;\vee\; \varphi\,|_{y_1=0} \;=\; 0$
    - $\varphi\,|_{y_1=1} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=0} \;=\; 0$
    - $\varphi\,|_{y_1=0} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=1} \;=\; 0$

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \Rightarrow \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \not\Leftarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?
- Exactly when
  - $[\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1} = 1$
  - $\exists y_1 \varphi(X, Y) \Leftrightarrow \varphi\,|_{y_1=1} \vee \varphi\,|_{y_1=0} = 0$
    - $\varphi\,|_{y_1=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=0} = 0$
    - $\varphi\,|_{y_1=0} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=1} = 0$
    - (By monotoniciy of $\widehat{\varphi}$ w.r.t y_1 and $\overline{y_1}$) $[\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=0} = 0$

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \;\Rightarrow\; \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \;\not\Leftarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?
- Exactly when
  - $[\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1} \;=\; 1$
  - $\exists y_1 \varphi(X, Y) \;\Leftrightarrow\; \varphi\,|_{y_1=1} \;\vee\; \varphi\,|_{y_1=0} \;=\; 0$
    - $\varphi\,|_{y_1=1} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=0} \;=\; 0$
    - $\varphi\,|_{y_1=0} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=1} \;=\; 0$
    - (By monotoniciy of $\widehat{\varphi}$ w.r.t y_1 and $\overline{y_1}$) $[\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=0} \;=\; 0$
- In other words, when $[\widehat{\varphi}]_1$ "behaves like" $y_1 \wedge \overline{y_1}$.

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \Rightarrow \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \not\Leftarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?
- Exactly when
  - $[\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1} = 1$
  - $\exists y_1 \varphi(X, Y) \Leftrightarrow \varphi\,|_{y_1=1} \vee \varphi\,|_{y_1=0} = 0$
    - $\varphi\,|_{y_1=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=0} = 0$
    - $\varphi\,|_{y_1=0} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=1} = 0$
    - (By monotoniciy of $\widehat{\varphi}$ w.r.t y_1 and $\overline{y_1}$) $[\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=0} = 0$
- In other words, when $[\widehat{\varphi}]_1$ "behaves like" $y_1 \wedge \overline{y_1}$.

Insight

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \Rightarrow \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \not\Leftarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?
- Exactly when
  - $[\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1} = 1$
  - $\exists y_1 \varphi(X, Y) \Leftrightarrow \varphi\,|_{y_1=1} \vee \varphi\,|_{y_1=0} = 0$
    - $\varphi\,|_{y_1=1} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=0} = 0$
    - $\varphi\,|_{y_1=0} \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=1} = 0$
    - (By monotoniciy of $\widehat{\varphi}$ w.r.t y_1 and $\overline{y_1}$) $[\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=0} = 0$
- In other words, when $[\widehat{\varphi}]_1$ "behaves like" $y_1 \wedge \overline{y_1}$.

Insight

- $\exists y_1 \varphi(X, Y) \Leftrightarrow [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$ for all $X, y_2, \ldots y_m$ iff $\neg\exists X, y_2, \ldots y_m \left([\widehat{\varphi}]_1 \Leftrightarrow y_1 \wedge \overline{y_1}\right)$.

Iterated reducts and existential quantification

Already seen $\exists y_1 \varphi(X, Y) \;\Rightarrow\; \widehat{\varphi}\,|_{y_1=1,\overline{y_1}=1} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$
Under what conditions is the implication strict?

- When do we have $\exists y_1 \varphi(X, Y) \;\not\Leftarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$?
- Exactly when
  - $[\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1} \;=\; 1$
  - $\exists y_1 \varphi(X, Y) \;\Leftrightarrow\; \varphi\,|_{y_1=1} \;\vee\; \varphi\,|_{y_1=0} \;=\; 0$
    - $\varphi\,|_{y_1=1} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=0} \;=\; 0$
    - $\varphi\,|_{y_1=0} \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=1} \;=\; 0$
    - (By monotoniciy of $\widehat{\varphi}$ w.r.t y_1 and $\overline{y_1}$) $[\widehat{\varphi}]_1\,|_{y_1=0,\overline{y_1}=0} \;=\; 0$
- In other words, when $[\widehat{\varphi}]_1$ "behaves like" $y_1 \wedge \overline{y_1}$.

Insight

- $\exists y_1 \varphi(X, Y) \;\Leftrightarrow\; [\widehat{\varphi}]_1\,|_{y_1=1,\overline{y_1}=1}$ for all $X, y_2, \ldots y_m$ iff $\neg\exists X, y_2, \ldots y_m \left([\widehat{\varphi}]_1 \;\Leftrightarrow\; y_1 \wedge \overline{y_1}\right)$.
- Inductively, $\exists y_1, \ldots y_i \varphi(X, Y) \;\Leftrightarrow\; [\widehat{\varphi}]_i\,|_{y_i=1,\overline{y_i}=1}$ iff $\neg\exists X, y_{i+1}, \ldots y_m \left([\widehat{\varphi}]_i \;\Leftrightarrow\; y_i \wedge \overline{y_i}\right)$.

SynNNF: A negation normal form for efficient synthesis

An NNF specification $\varphi(X, Y)$ is in

SynNNF w.r.t. a linear order $y_1 \prec y_2 \prec \cdots \prec y_m$ iff

$\neg\exists x_1, \ldots x_n, y_{i+1} \ldots y_m \quad (\; [\widehat{\varphi}]_i \Leftrightarrow \; \mathrm{y_i} \wedge \overline{\mathrm{y_i}} \;)$ for all $i \in \{1, \ldots n\}$

SynNNF: A negation normal form for efficient synthesis

An NNF specification $\varphi(X, Y)$ is in

SynNNF w.r.t. a linear order $y_1 \prec y_2 \prec \cdots \prec y_m$ iff

$\neg\exists x_1, \ldots x_n, y_{i+1} \ldots y_m \quad (\; [\widehat{\varphi}]_i \Leftrightarrow \; \mathrm{y_i} \wedge \overline{\mathrm{y_i}} \;)$ for all $i \in \{1, \ldots n\}$

Can $[\widehat{\varphi}]_i$ be made to "behave like" $\mathrm{y_i} \wedge \overline{\mathrm{y_i}}$ for any i?

If yes, φ is not in SynNNF; else it is in SynNNF

SynNNF: A negation normal form for efficient synthesis

An NNF specification $\varphi(\mathrm{X}, \mathrm{Y})$ is in

SynNNF w.r.t. a linear order $y_1 \prec y_2 \prec \cdots \prec y_m$ iff

$\neg\exists x_1, \ldots x_n, y_{i+1} \ldots y_m \quad ([\widehat{\varphi}]_i \Leftrightarrow \mathrm{y_i} \wedge \overline{\mathrm{y_i}})$ for all $i \in \{1, \ldots n\}$

Can $[\widehat{\varphi}]_i$ be made to "behave like" $\mathrm{y_i} \wedge \overline{\mathrm{y_i}}$ for any i?

If yes, φ is not in SynNNF; else it is in SynNNF

Skolem fn for y_i (in terms of $y_{i+1}, \ldots y_m, \mathrm{X}$)

- $\exists y_1, \ldots y_{i-1}\ \varphi(\mathrm{X}, y_1, \ldots y_{i-1}, 1, y_{i+1}, \ldots y_m)$

SynNNF: A negation normal form for efficient synthesis

An NNF specification $\varphi(X, Y)$ is in

SynNNF w.r.t. a linear order $y_1 \prec y_2 \prec \cdots \prec y_m$ iff

$\neg\exists x_1, \ldots x_n, y_{i+1} \ldots y_m \;\; (\, [\widehat{\varphi}]_i \Leftrightarrow \; \mathrm{y_i} \wedge \overline{\mathrm{y_i}} \;)$ for all $i \in \{1, \ldots n\}$

Can $[\widehat{\varphi}]_i$ be made to "behave like" $\mathrm{y_i} \wedge \overline{\mathrm{y_i}}$ for any i?

If yes, φ is not in SynNNF; else it is in SynNNF

Skolem fn for y_i (in terms of $y_{i+1}, \ldots y_m, X$)

- $\exists y_1, \ldots y_{i-1} \; \varphi(X, y_1, \ldots y_{i-1}, 1, y_{i+1}, \ldots y_m)$
- Equivalently, $[\widehat{\varphi}]_i \,|_{y_i=1, \overline{y_i}=0}$, if φ in SynNNF

SynNNF: A negation normal form for efficient synthesis

An NNF specification $\varphi(\mathrm{X}, \mathrm{Y})$ is in
SynNNF w.r.t. a linear order $y_1 \prec y_2 \prec \cdots \prec y_m$ iff
$\neg\exists x_1, \ldots x_n, y_{i+1} \ldots y_m \quad ([\widehat{\varphi}]_i \Leftrightarrow \mathrm{y_i} \wedge \overline{\mathrm{y_i}})$ for all $i \in \{1, \ldots n\}$

Can $[\widehat{\varphi}]_i$ be made to "behave like" $\mathrm{y_i} \wedge \overline{\mathrm{y_i}}$ for any i?

If yes, φ is not in SynNNF; else it is in SynNNF

Skolem fn for y_i (in terms of $y_{i+1}, \ldots y_m, \mathrm{X}$)

- $\exists y_1, \ldots y_{i-1}\ \varphi(\mathrm{X}, y_1, \ldots y_{i-1}, 1, y_{i+1}, \ldots y_m)$
- Equivalently, $[\widehat{\varphi}]_i\ |_{y_i=1, \overline{y_i}=0}$, if φ in SynNNF

Observations:

- Not purely structural restriction on representation of φ

SynNNF: A negation normal form for efficient synthesis

An NNF specification $\varphi(X, Y)$ is in

SynNNF w.r.t. a linear order $y_1 \prec y_2 \prec \cdots \prec y_m$ iff

$\neg\exists x_1, \ldots x_n, y_{i+1} \ldots y_m \;\; ([\widehat{\varphi}]_i \Leftrightarrow \; \mathrm{y_i} \wedge \overline{\mathrm{y_i}} \;)$ for all $i \in \{1, \ldots n\}$

Can $[\widehat{\varphi}]_i$ be made to "behave like" $\mathrm{y_i} \wedge \overline{\mathrm{y_i}}$ for any i?

If yes, φ is not in SynNNF; else it is in SynNNF

Skolem fn for y_i (in terms of $y_{i+1}, \ldots y_m, X$)

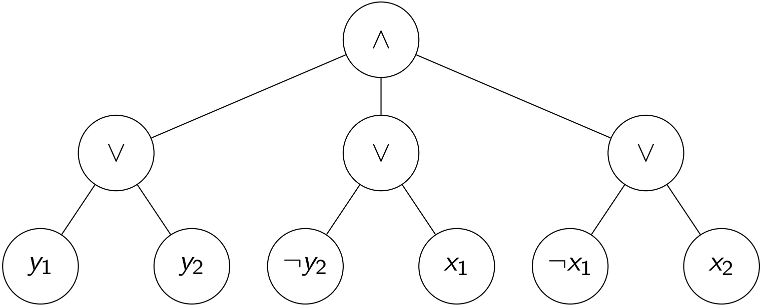
- $\exists y_1, \ldots y_{i-1} \; \varphi(X, y_1, \ldots y_{i-1}, 1, y_{i+1}, \ldots y_m)$
- Equivalently, $[\widehat{\varphi}]_i \,|_{y_i=1, \overline{y_i}=0}$, if φ in SynNNF

Observations:

- Not purely structural restriction on representation of φ
- Reminiscent of Deterministic DNNF (dDNNF)
    - For every $\vee$ node representing $\varphi_1 \vee \varphi_2$, require $\varphi_1 \wedge \varphi_2 = \bot$.

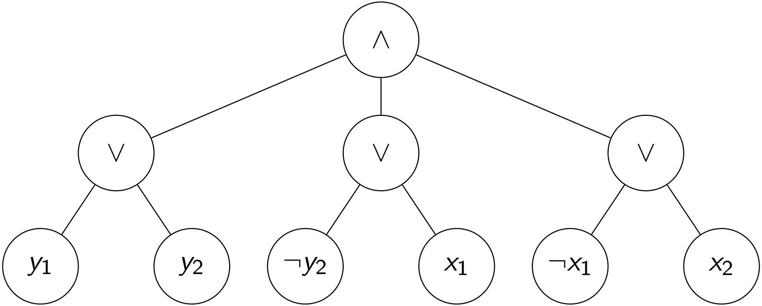
SynNNF: An Example

$$\varphi(x_1, x_2, y_1, y_2) = (y_1 \vee y_2) \wedge (\neg y_2 \vee x_1) \wedge (\neg x_1 \vee x_2)$$



SynNNF: An Example

$$\varphi(x_1, x_2, y_1, y_2) = (y_1 \vee y_2) \wedge (\neg y_2 \vee x_1) \wedge (\neg x_1 \vee x_2)$$

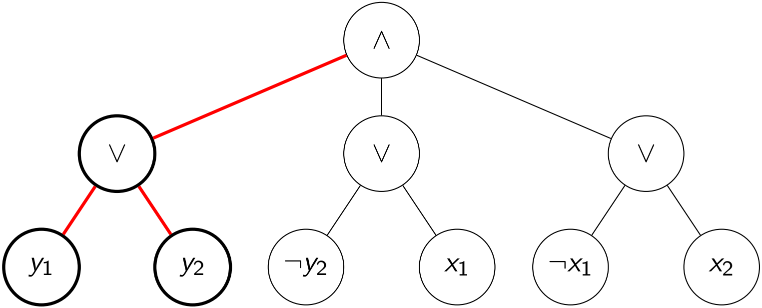


Representation of φ **not** in DNNF/wDNNF

SynNNF: An Example

$\varphi(x_1, x_2, y_1, y_2) = (y_1 \vee y_2) \wedge (\neg y_2 \vee x_1) \wedge (\neg x_1 \vee x_2)$

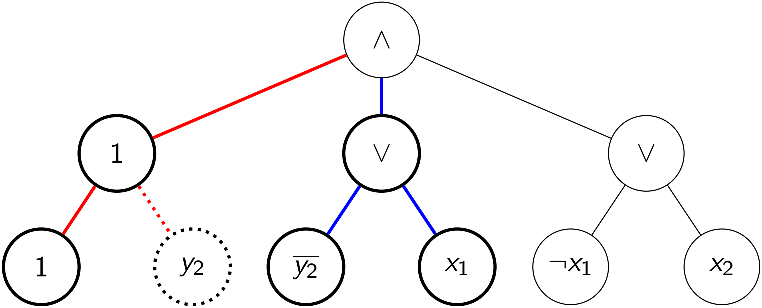
Output sequence: $\underline{y_1} \prec y_2$



SynNNF: An Example

$\varphi(x_1, x_2, y_1, y_2) = (y_1 \vee y_2) \wedge (\neg y_2 \vee x_1) \wedge (\neg x_1 \vee x_2)$

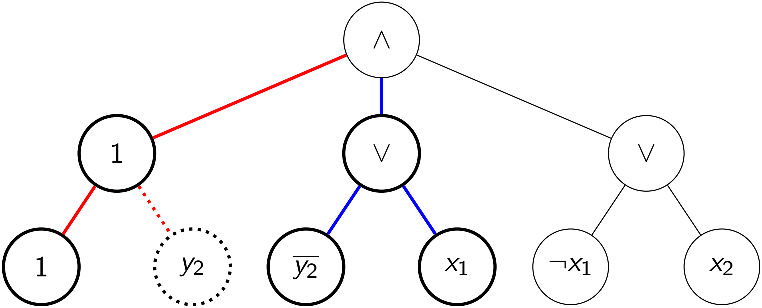
Output sequence: $y_1 \prec \underline{y_2}$



SynNNF: An Example

$\varphi(x_1, x_2, y_1, y_2) = (y_1 \vee y_2) \wedge (\neg y_2 \vee x_1) \wedge (\neg x_1 \vee x_2)$

Output sequence: $y_1 \prec \underline{y_2}$

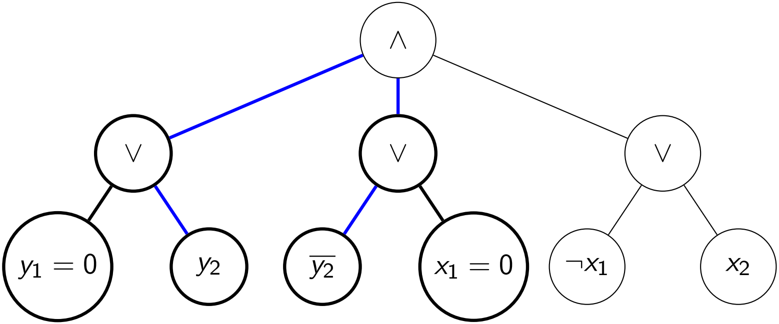


Representation of φ in SynNNF **w.r.t** $y_1 \prec y_2$

Non-SynNNF: An Example

$\varphi(x_1, x_2, y_1, y_2) = (y_1 \vee y_2) \wedge (\neg y_2 \vee x_1) \wedge (\neg x_1 \vee x_2)$

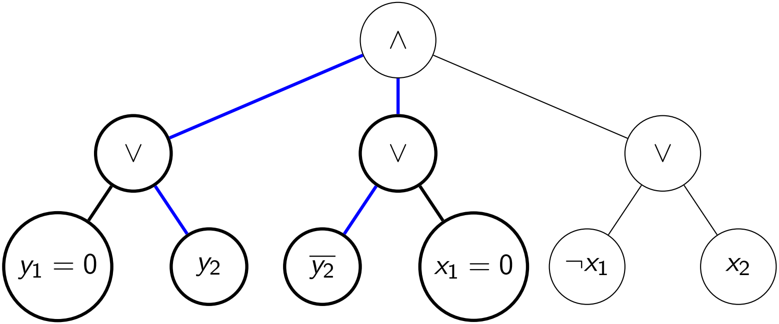
Output sequence: $\underline{y_2} \prec y_1$



Non-SynNNF: An Example

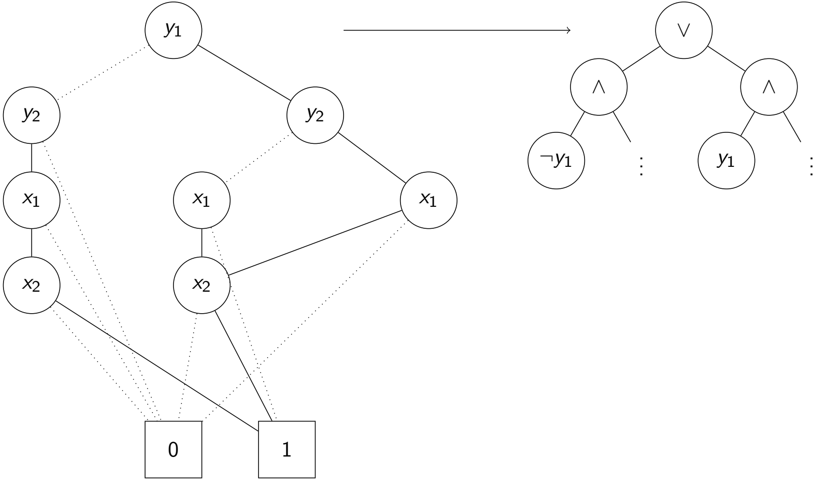
$\varphi(x_1, x_2, y_1, y_2) = (y_1 \vee y_2) \wedge (\neg y_2 \vee x_1) \wedge (\neg x_1 \vee x_2)$

Output sequence: $\underline{y_2} \prec y_1$

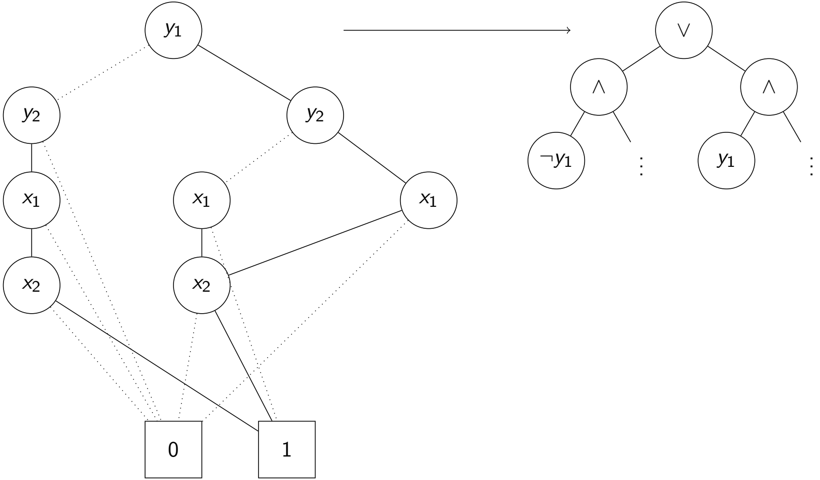


Representation of φ not in SynNNF **w.r.t** $y_2 \prec y_1$

BDD and SynNNF

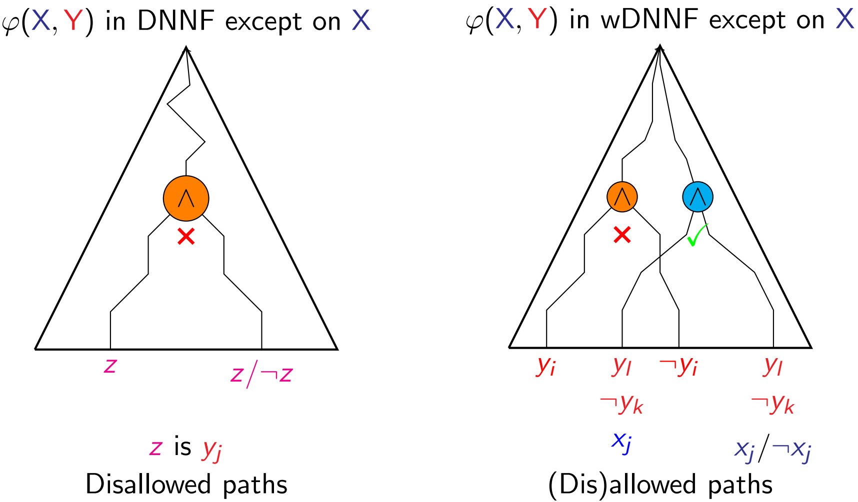


BDD and SynNNF

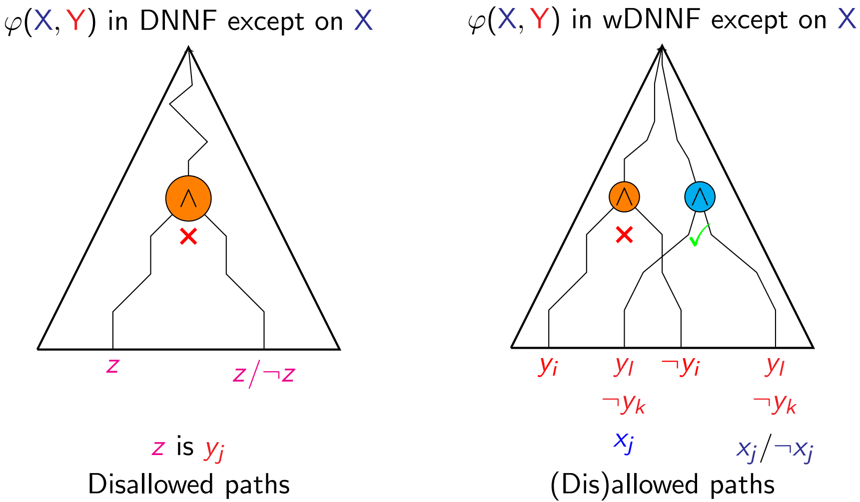


BDD → SynNNF in linear time for any output order ≺ and any BDD variable order.

DNFF, wDNNF and SynNNF



DNNF, wDNNF and SynNNF



A specification in DNNF or wDNNF is already in SynNNF for any output order $\prec$.

SynNNF: Some observations

- Every propositional formula representable in SynNNF for every ordering of outputs
  - DNF always SynNNF for any output order

SynNNF: Some observations

- Every propositional formula representable in SynNNF for every ordering of outputs
  - DNF always SynNNF for any output order
- A formula may have multiple SynNNF representations
  - DNF, BDD, DNNF ...

SynNNF: Some observations

- Every propositional formula representable in SynNNF for every ordering of outputs
  - DNF always SynNNF for any output order
- A formula may have multiple SynNNF representations
  - DNF, BDD, DNNF ...
- A given representation may be SynNNF for one order of outputs and not in SynNNF for another order.

SynNNF: Some observations

- Every propositional formula representable in SynNNF for every ordering of outputs
  - DNF always SynNNF for any output order
- A formula may have multiple SynNNF representations
  - DNF, BDD, DNNF ...
- A given representation may be SynNNF for one order of outputs and not in SynNNF for another order.
- Given an output order $\prec$ and an NNF specification φ, checking if φ is in SynNNF w.r.t. $\prec$ is coNP-complete.

SynNNF: Some observations

- Every propositional formula representable in SynNNF for every ordering of outputs
  - DNF always SynNNF for any output order
- A formula may have multiple SynNNF representations
  - DNF, BDD, DNNF ...
- A given representation may be SynNNF for one order of outputs and not in SynNNF for another order.
- Given an output order $\prec$ and an NNF specification φ, checking if φ is in SynNNF w.r.t. $\prec$ is coNP-complete.
- Given φ, checking if φ is in SynNNF w.r.t. any (unspecificed) order $\prec$ is in Σ_2^P.

SynNNF is relatively succinct

There exist polynomial sized SynNNF specifications that only admit

SynNNF is relatively succinct

There exist polynomial sized SynNNF specifications that only admit

- Exponential-sized ROBDD/FBDD representations

SynNNF is relatively succinct

There exist polynomial sized SynNNF specifications that only admit

- Exponential-sized ROBDD/FBDD representations
- Exponential-sized DNNF representations

SynNNF is relatively succinct

There exist polynomial sized SynNNF specifications that only admit

- Exponential-sized ROBDD/FBDD representations
- Exponential-sized DNNF representations
- Super-polynomial sized dDNNF representations, unless P = VNP

SynNNF is relatively succinct

There exist polynomial sized SynNNF specifications that only admit

- Exponential-sized ROBDD/FBDD representations
- Exponential-sized DNNF representations
- Super-polynomial sized dDNNF representations, unless P = VNP

There exist poly-sized NNF representations that only admit super-polynomial sized SynNNF representations

- Unless the polynomial hierarchy collapses

SynNNF is relatively succinct

There exist polynomial sized SynNNF specifications that only admit

- Exponential-sized ROBDD/FBDD representations
- Exponential-sized DNNF representations
- Super-polynomial sized dDNNF representations, unless P $=$ VNP

There exist poly-sized NNF representations that only admit super-polynomial sized SynNNF representations

- Unless the polynomial hierarchy collapses

NNF $\sqsubset$ SynNNF $\sqsubset$ DNNF $\sqsubset$ dDNNF $\sqsubset$ BDD

Operations with SynNNF

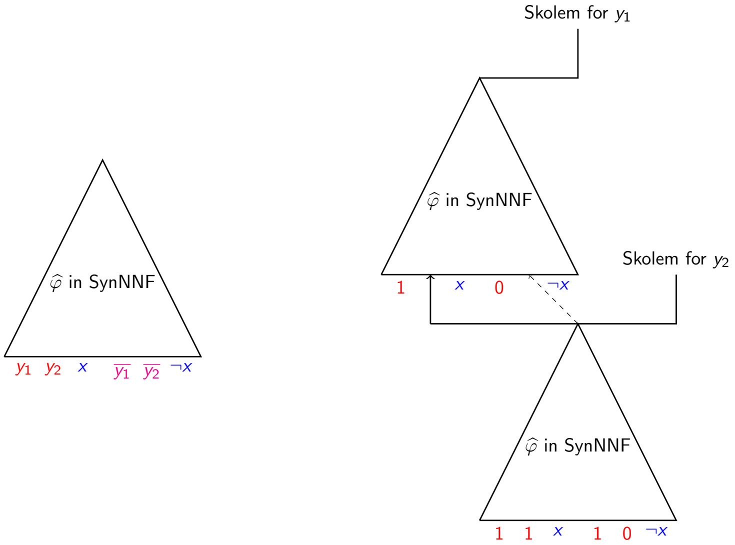
Given $\varphi_1(X, Y)$ and $\varphi_2(X, Y)$ in SynNNF w.r.t. the same ordering of Y

Operations with SynNNF

Given $\varphi_1(X, Y)$ and $\varphi_2(X, Y)$ in SynNNF w.r.t. the same ordering of Y

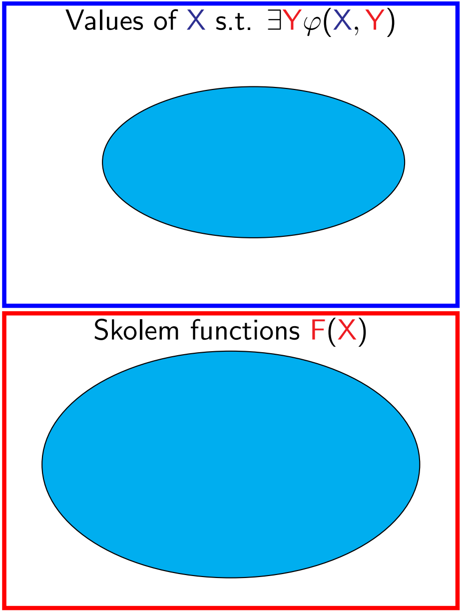
- Computing $\varphi_1 \wedge \varphi_2$ in SynNNF in poly-time not possible unless P = NP
- Computing $\varphi_1 \vee \varphi_2$ in SynNNF in same ordering of Y takes constant time
- Existentially quantifying $y_1, \ldots y_m$ takes linear time.
    - Quantifying subset of Y not possible in linear time in general.

How does SynNNF help Skolem function synthesis?



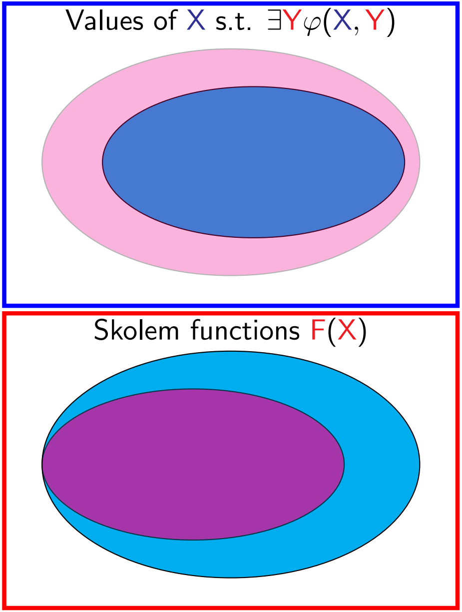
Synthesis: $m \times |\varphi|$ circuit size, $\mathcal{O}(m^2)$ additional wires.

Refinement w.r.t. synthesis



Given spec: $\varphi(X, Y)$

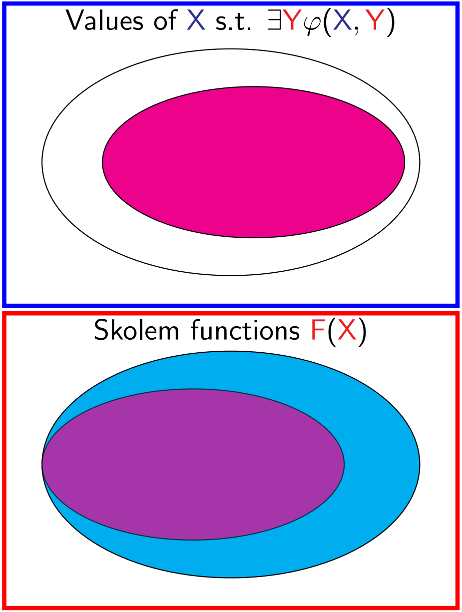
Refinement w.r.t. synthesis



Given spec: $\varphi(X, Y)$

Refined spec: $\widetilde{\varphi}(X, Y)$

$\widetilde{\varphi} \preceq_{syn} \varphi$



Given spec: $\varphi(X, Y)$

Strongly
Refined spec: $\widetilde{\varphi}(X, Y)$
$\widetilde{\varphi} \preceq^*_{syn} \varphi$

Refinement w.r.t. synthesis

Lemma

If $\widetilde{\varphi}(X, Y) \preceq_{syn} \varphi(X, Y)$, every Skolem function vector for Y in $\widetilde{\varphi}$ is also a Skolem function vector for Y in φ.

Refinement w.r.t. synthesis

Lemma

If $\widetilde{\varphi}(X, Y) \preceq_{syn} \varphi(X, Y)$, every Skolem function vector for Y in $\widetilde{\varphi}$ is also a Skolem function vector for Y in φ.

Example: $(y_2 \wedge y_1) \preceq_{syn}$
$((\neg y_1 \vee y_2 \vee x_1) \wedge (y_1 \vee \neg y_2) \wedge (y_1 \vee \neg x_1) \wedge (y_2 \vee x_2))$

Properties of refinement

1. Both $\preceq_{syn}$ and $\preceq^{*}_{syn}$ are reflexive and transitive relations on Boolean relational specifications.

Properties of refinement

1. Both $\preceq_{syn}$ and $\preceq^*_{syn}$ are reflexive and transitive relations on Boolean relational specifications.
2. If $\bigwedge_{x_j \in \mathsf{X}} \left(F|_{x_j=0} \Leftrightarrow F|_{x_j=1}\right)$ and $\pi \models F(\mathsf{Y}, \mathsf{X})$, then $\mathsf{form}(\pi{\downarrow}\mathsf{Y}) \preceq^*_{syn} F$.

Properties of refinement

1. Both $\preceq_{syn}$ and $\preceq^*_{syn}$ are reflexive and transitive relations on Boolean relational specifications.
2. If $\bigwedge_{x_j \in \mathsf{X}} \left(F|_{x_j=0} \Leftrightarrow F|_{x_j=1}\right)$ and $\pi \models F(\mathsf{Y}, \mathsf{X})$, then $\mathsf{form}(\pi{\downarrow}\mathsf{Y}) \preceq^*_{syn} F$.
3. If $\bigwedge_{y_i \in \mathsf{Y}} (F|_{y_i=0} \Leftrightarrow F|_{y_i=1})$, then $1 \preceq_{syn} F$ and $F|_{\mathsf{Y}=\mathsf{a}} \preceq^*_{syn} F$, where a is any vector in $\{0,1\}^m$.

Properties of refinement

1. Both $\preceq_{syn}$ and $\preceq^*_{syn}$ are reflexive and transitive relations on Boolean relational specifications.
2. If $\bigwedge_{x_j \in \mathrm{X}} \left(F|_{x_j=0} \Leftrightarrow F|_{x_j=1}\right)$ and $\pi \models F(\mathrm{Y},\mathrm{X})$, then $\mathrm{form}(\pi{\downarrow}\mathrm{Y}) \preceq^*_{syn} F$.
3. If $\bigwedge_{y_i \in \mathrm{Y}} (F|_{y_i=0} \Leftrightarrow F|_{y_i=1})$, then $1 \preceq_{syn} F$ and $F|_{\mathrm{Y}=\mathrm{a}} \preceq^*_{syn} F$, where a is any vector in $\{0,1\}^m$.
4. If F is positive (resp. negative) unate in $y_i \in \mathrm{Y}$, then $y_i \wedge F|_{y_i=1}$ (resp. $\neg y_i \wedge F|_{y_i=0}$) $\preceq^*_{syn} F$. pause
5.
    1. Let $\widetilde{F}_1 \preceq^*_{syn} F_1$ and $\widetilde{F}_2 \preceq^*_{syn} F_2$. Then $(\widetilde{F}_1 \vee \widetilde{F}_2) \preceq^*_{syn} (F_1 \vee F_2)$.

Properties of refinement

1. Both $\preceq_{syn}$ and $\preceq^*_{syn}$ are reflexive and transitive relations on Boolean relational specifications.
2. If $\bigwedge_{x_j \in \mathsf{X}} \left(F|_{x_j=0} \Leftrightarrow F|_{x_j=1}\right)$ and $\pi \models F(\mathsf{Y}, \mathsf{X})$, then $\mathsf{form}(\pi{\downarrow}\mathsf{Y}) \preceq^*_{syn} F$.
3. If $\bigwedge_{y_i \in \mathsf{Y}} (F|_{y_i=0} \Leftrightarrow F|_{y_i=1})$, then $1 \preceq_{syn} F$ and $F|_{\mathsf{Y}=\mathsf{a}} \preceq^*_{syn} F$, where a is any vector in $\{0,1\}^m$.
4. If F is positive (resp. negative) unate in $y_i \in \mathsf{Y}$, then $y_i \wedge F|_{y_i=1}$ (resp. $\neg y_i \wedge F|_{y_i=0}$) $\preceq^*_{syn} F$. pause
5.
   1. Let $\widetilde{F_1} \preceq^*_{syn} F_1$ and $\widetilde{F_2} \preceq^*_{syn} F_2$. Then $(\widetilde{F_1} \vee \widetilde{F_2}) \preceq^*_{syn} (F_1 \vee F_2)$.
   2. Let $\widetilde{F_1} \preceq_{syn} F_1$ and $\widetilde{F_2} \preceq_{syn} F_2$. If the output supports of F_1 and F_2, and similarly of $\widetilde{F_1}$ and $\widetilde{F_2}$, are disjoint, then $(\widetilde{F_1} \wedge \widetilde{F_2}) \preceq_{syn} (F_1 \wedge F_2)$. If, in addition, $\widetilde{F_1} \preceq^*_{syn} F_1$ and $\widetilde{F_2} \preceq^*_{syn} F_2$, then $(\widetilde{F_1} \wedge \widetilde{F_2}) \preceq^*_{syn} (F_1 \wedge F_2)$.

Putting it all together

Tool C2Syn:

- Input: φ in CNF (or AIG)
- Output: Refined $\widetilde{\varphi}$ in SynNNF

- Branches only on output variables
- Aggressively tries to refine whenever possible
- Details in our FMCAD 2019 paper

Experimental Results

Comparison of run-time with

- CADET [Rabe et al 2016]
- BFSS [Akshay et al 2018]
- BDD [BDD pipeline of BFSS]

Benchmarks: QBFEVAL 2018 and Factorization (408 total)

Benchmarks (Total)	Compiled By C2Syn			BDD compilation	Total in SynNNF
	Stage I	Stage II	Total		
QBFEVAL (402)	103	83	186	153	**283**
FA.QD (6)	0	6	6	6	**6**

Table: Compilation into SynNNF

Bench mark	C2Syn vs CADET		C2Syn vs BFSS		C2Syn \ (CADET ∪ BFSS)
	C2Syn\ CADET	CADET\ C2Syn	C2Syn\ BFSS	BFSS\ C2Syn	
QBFEVAL	78	105	83	78	**75**
FA.QD	2	0	3	0	**2**

Table: Comparison Results of C2Syn

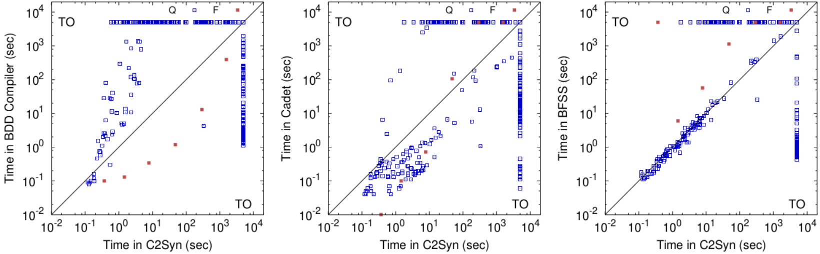


Fig. 1: Time comparisons: C2Syn vs BDD^{BFSS}, CADET, BFSS

Conclusion

- SynNNF: A new normal form for polynomial-time synthesis
- Refinement w.r.t. synthesis useful in practice
  - Formalization of folklore
  - CNF $\rightarrow$ Refined SynNNF much more efficient than CNF $\rightarrow$ SynNNF
- Experimental results with preliminary implementation show promise
- It appears that SynNNF can be further weakened to achieve poly-time synthesis
  - Ongoing work