Automated Synthesis: Towards the Holy Grail of AI

Kuldeep S. Meel[1], Supratik Chakraborty[2], S Akshay[2], Priyanka Golia[1,3], Subhajit Roy[3]

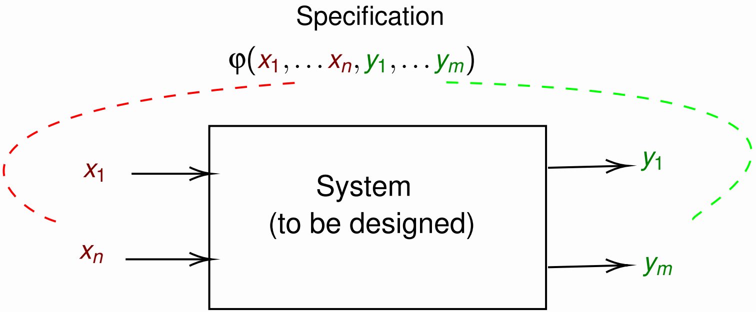




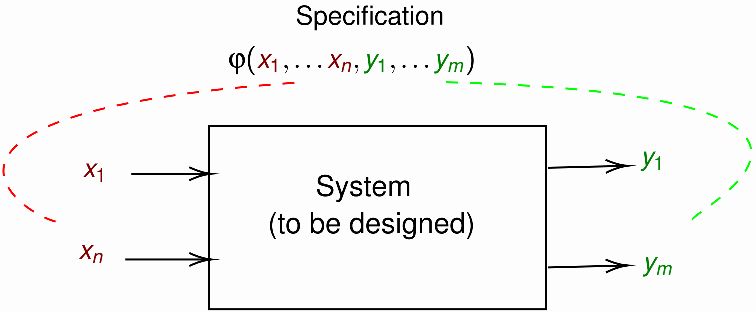
[1]National University of Singapore
[2]Indian Institute of Technology Bombay
[3]Indian Institute of Technology Kanpur

AAAI-2022

Synthesis: A Generic View

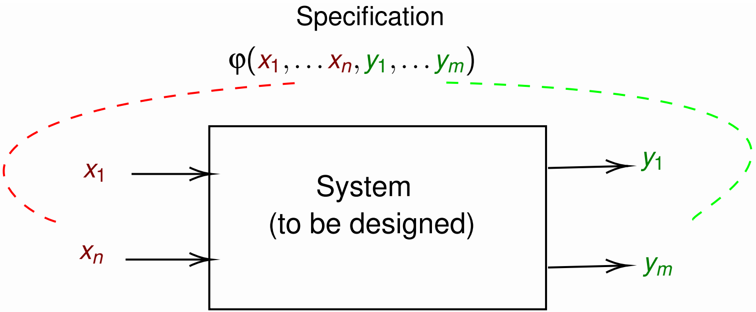


Synthesis: A Generic View



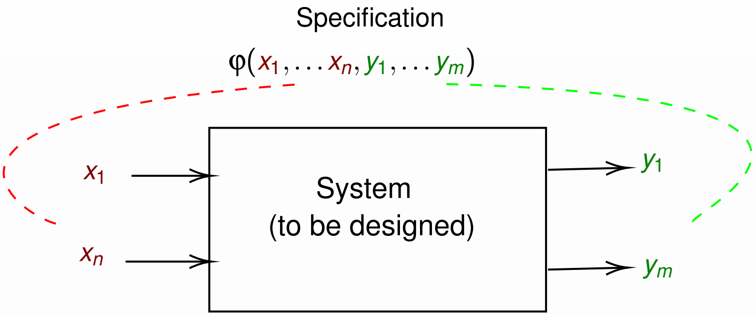
- Goal: Automatically synthesize system s.t. it satisfies $\varphi(x_1,..,x_n,y_1,..,y_m)$
  - x_i *input* variables (vector **X**)
  - y_k *output* variables (vector **Y**)

Synthesis: A Generic View



- Goal: Automatically synthesize system s.t. it satisfies $\varphi(x_1, .., x_n, y_1, .., y_m)$ **whenever possible**.
  - x_i *input* variables (vector **X**)
  - y_k *output* variables (vector **Y**)

Synthesis: A Generic View



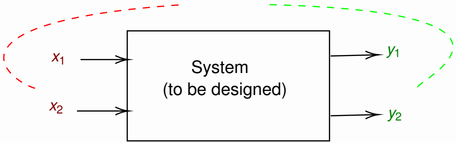
- Goal: Automatically synthesize system s.t. it satisfies $\varphi(x_1, .., x_n, y_1, .., y_m)$ **whenever possible**.
  - x_i *input* variables (vector $\mathbf{X}$)
  - y_k *output* variables (vector $\mathbf{Y}$)
- Need $\mathbf{Y}$ as functions $\mathbf{F}$ of
  - "History" of $\mathbf{X}$ and $\mathbf{Y}$, "State" of system, ...

such that $\varphi(\mathbf{X}, \mathbf{F})$ is satisfied.

Example 1: Application to Reactive Synthesis

Temporal Spec

$$\mathbf{G}(x_1 \Rightarrow \mathbf{F}y_1) \wedge \mathbf{G}(x_2 \Rightarrow \mathbf{F}y_2) \wedge \mathbf{G}(\neg y_1 \vee \neg y_2)$$

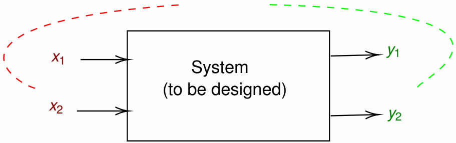


- Synthesize **Y** as function of
  - State (summarizing “history” of **X** and **Y**)

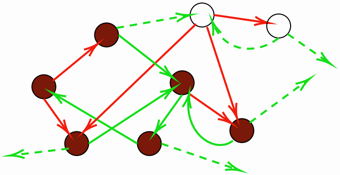
Example 1: Application to Reactive Synthesis

Temporal Spec

$$\mathbf{G}(x_1 \Rightarrow \mathbf{F}y_1) \wedge \mathbf{G}(x_2 \Rightarrow \mathbf{F}y_2) \wedge \mathbf{G}(\neg y_1 \vee \neg y_2)$$



- Synthesize **Y** as function of
  - State (summarizing “history” of **X** and **Y**)

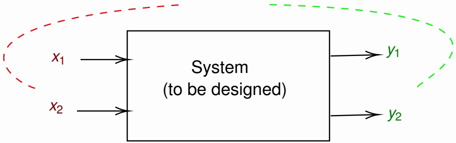


Winning Region

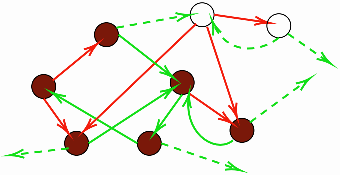
Example 1: Application to Reactive Synthesis

Temporal Spec

$\mathbf{G}(x_1 \Rightarrow \mathbf{F}y_1) \wedge \mathbf{G}(x_2 \Rightarrow \mathbf{F}y_2) \wedge \mathbf{G}(\neg y_1 \vee \neg y_2)$



- Synthesize **Y** as function of
  - State (summarizing "history" of **X** and **Y**)



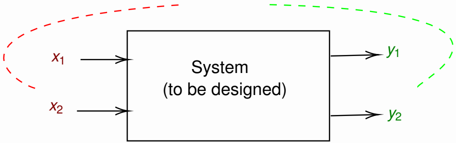
Winning Region

- Synthesize *winning strategy* to stay within *winning region*

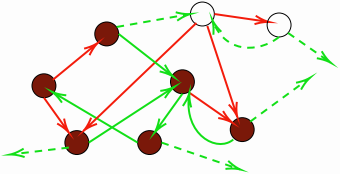
Example 1: Application to Reactive Synthesis

Temporal Spec

$\mathbf{G}(x_1 \Rightarrow \mathbf{F}y_1) \wedge \mathbf{G}(x_2 \Rightarrow \mathbf{F}y_2) \wedge \mathbf{G}(\neg y_1 \vee \neg y_2)$



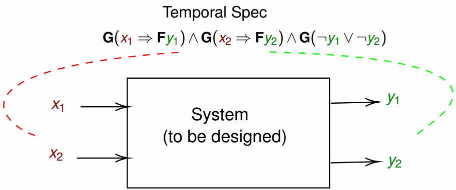
- Synthesize **Y** as function of
  - State (summarizing "history" of **X** and **Y**)



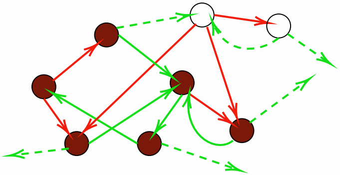
Winning Region

- Synthesize *winning strategy* to stay within *winning region*
  - WinRgn(NxtSt(state, **Y**)) = 1

Example 1: Application to Reactive Synthesis



- Synthesize **Y** as function of
  - State (summarizing "history" of **X** and **Y**)



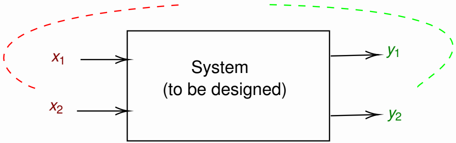
Winning Region

- Synthesize *winning strategy* to stay within *winning region*
  - WinRgn(NxtSt(state, **Y**)) = 1
    - No temporal operators

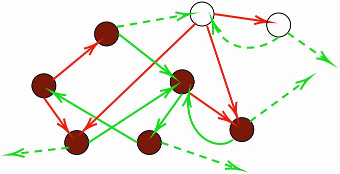
Example 1: Application to Reactive Synthesis

Temporal Spec

$$\mathbf{G}(x_1 \Rightarrow \mathbf{F}y_1) \wedge \mathbf{G}(x_2 \Rightarrow \mathbf{F}y_2) \wedge \mathbf{G}(\neg y_1 \vee \neg y_2)$$



- Synthesize **Y** as function of
  - State (summarizing "history" of **X** and **Y**)



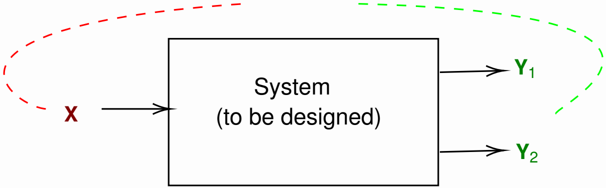
Winning Region

- Synthesize *winning strategy* to stay within *winning region*
  - WinRgn(NxtSt(state, **Y**)) = 1
    - No temporal operators
  - Not always satisfiable

Example 2: Application to Cryptanalysis

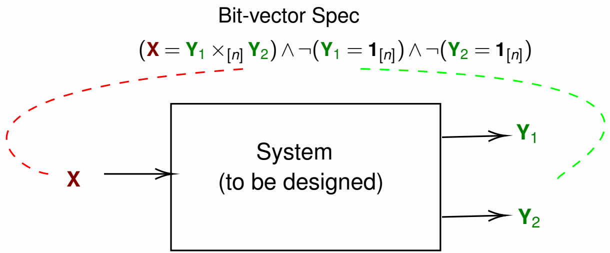
Bit-vector Spec

$$(\mathbf{X} = \mathbf{Y}_1 \times_{[n]} \mathbf{Y}_2) \wedge \neg(\mathbf{Y}_1 = \mathbf{1}_{[n]}) \wedge \neg(\mathbf{Y}_2 = \mathbf{1}_{[n]})$$



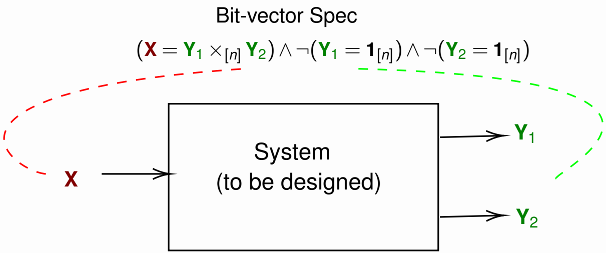
- Synthesize $\mathbf{Y}_1, \mathbf{Y}_2$ as functions of $\mathbf{X}$

Example 2: Application to Cryptanalysis



- Synthesize $\mathbf{Y}_1, \mathbf{Y}_2$ as functions of $\mathbf{X}$
  - Spec has no temporal operators

Example 2: Application to Cryptanalysis

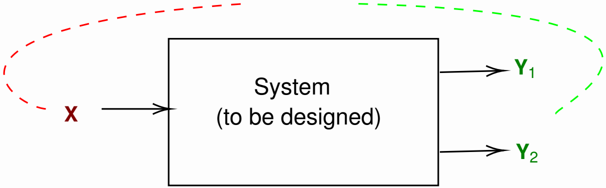


- Synthesize $\mathbf{Y}_1, \mathbf{Y}_2$ as functions of $\mathbf{X}$
  - Spec has no temporal operators
  - $\mathbf{Y}_1, \mathbf{Y}_2$ must be non-trivial factors of $\mathbf{X}$

Example 2: Application to Cryptanalysis

Bit-vector Spec

$$(\mathbf{X} = \mathbf{Y}_1 \times_{[n]} \mathbf{Y}_2) \wedge \neg(\mathbf{Y}_1 = \mathbf{1}_{[n]}) \wedge \neg(\mathbf{Y}_2 = \mathbf{1}_{[n]})$$

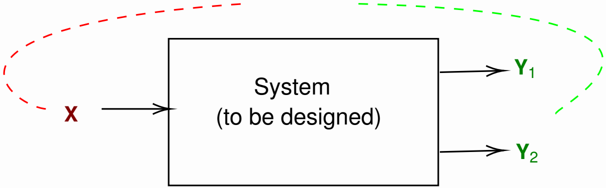


- Synthesize $\mathbf{Y}_1, \mathbf{Y}_2$ as functions of $\mathbf{X}$
  - Spec has no temporal operators
  - $\mathbf{Y}_1, \mathbf{Y}_2$ must be non-trivial factors of $\mathbf{X}$
  - Not always satisfiable (if $\mathbf{X}$ is prime)

Example 2: Application to Cryptanalysis

Bit-vector Spec

$$(\mathbf{X} = \mathbf{Y}_1 \times_{[n]} \mathbf{Y}_2) \wedge \neg(\mathbf{Y}_1 = \mathbf{1}_{[n]}) \wedge \neg(\mathbf{Y}_2 = \mathbf{1}_{[n]})$$



- Synthesize $\mathbf{Y}_1, \mathbf{Y}_2$ as functions of $\mathbf{X}$
  - Spec has no temporal operators
  - $\mathbf{Y}_1, \mathbf{Y}_2$ must be non-trivial factors of $\mathbf{X}$
  - Not always satisfiable (if $\mathbf{X}$ is prime)
  - Efficient solution would break crypto systems

Formal definition

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

- x_i *input* variables (vector $\mathbf{X}$)
- y_j *output* variables (vector $\mathbf{Y}$)

Boolean Functional Synthesis

Formal definition

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

- x_i *input* variables (vector $\mathbf{X}$)
- y_j *output* variables (vector $\mathbf{Y}$)

Synthesize Boolean functions $F_j(\mathbf{X})$ for each y_j s.t.

$$\forall \mathbf{X} \left(\exists y_1 \ldots y_m \, \varphi(\mathbf{X}, y_1 \ldots y_m) \Leftrightarrow \varphi(\mathbf{Y}, F_1(\mathbf{X}), \ldots F_m(\mathbf{X})) \right)$$

Boolean Functional Synthesis

Formal definition

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

- x_i *input* variables (vector $\mathbf{X}$)
- y_j *output* variables (vector $\mathbf{Y}$)

Synthesize Boolean functions $F_j(\mathbf{X})$ for each y_j s.t.

$$\forall \mathbf{X} \left(\exists y_1 \ldots y_m \, \varphi(\mathbf{X}, y_1 \ldots y_m) \Leftrightarrow \varphi(\mathbf{Y}, F_1(\mathbf{X}), \ldots F_m(\mathbf{X})) \right)$$

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

More Applications of Boolean Functional Synthesis

1. Disjunctive decomposition of symbolic transition relations [Trivedi et al'02]
2. Quantifier elimination, of course!
   - $\exists \mathbf{Y}\ \varphi(\mathbf{X}, \mathbf{Y}) \equiv \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X}))$
3. Certifying QBF-SAT solvers
   - Nice survey of applications by Shukla et al'19
4. Program synthesis
   - Combinatorial sketching [Solar-Lezama et al'06, Srivastava et al'13]
   - Complete functional synthesis [Kuncak et al'10]
5. Repair/partial synthesis of circuits [Fujita et al'13]

How Hard (or Easy) is Boolean function synthesis?

- Boolean circuit: DAG with AND-, OR-, NOT-labeled nodes
- Input: $\varphi(\mathbf{X}, \mathbf{Y})$ as $(|\mathbf{X}| + |\mathbf{Y}|)$-input, 1-output circuit
- Output: Sk. func. vector $\mathbf{F}(\mathbf{X})$: $|\mathbf{X}|$-input, $|\mathbf{Y}|$-output circuit

- Boolean function synthesis is *NP*-hard
  - Unlikely, we will get a poly-time algorithm

- What about size of Skolem functions?
  - Does there always exist compact Skolem functions, although synthesizing may take exponential time?
- Lower bound results in circuit-size refer to monotone circuits [Razbarov 1985; Alon and Boppana 1987]
  - Monotone circuit
    - Output can't change $1 \to 0$ due to an input changing $0 \to 1$.
  - Skolem functions need not be monotone
  - Different argument for lower bounds on Skolem circuits

Some Good and Bad News

Bad news: [CAV2018]

- Unless $\Pi_2^P = \Sigma_2^P$, there exist relational specs φ for which Skolem function sizes must be super-polynomial in $|\varphi|$.
- Unless non-uniform exponential-time hypothesis fails, there exist relational specs φ for which Skolem function sizes must be exponential in $|F|$.

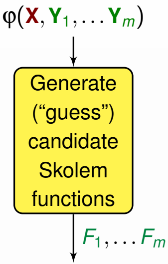
Efficient algorithms for Boolean functional synthesis unlikely

Good news: [CAV2018,FMCAD2019]

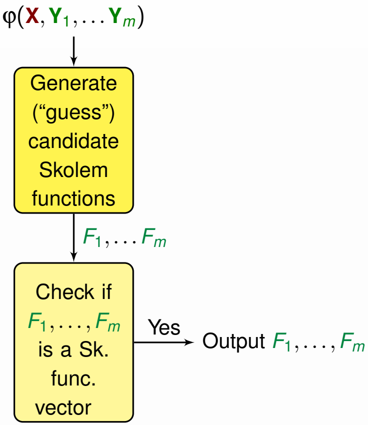
- If φ is represented in special normal form, synthesis solvable in polynomial (in $|\varphi|$) time and space.
  - Synthesis Negation Normal Form (SynNNF)
    - Talk in "Beyond Satisfiability" workshop on Mar 23
  - Reasonably common in practice

Experiments: Guess-check-repair algorithms work well in practice

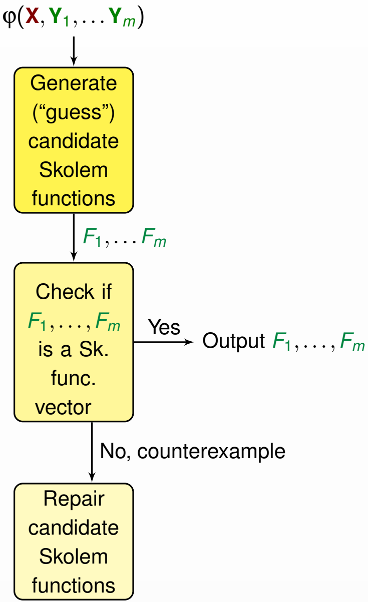
Overview of Guess-Check-Repair Paradigm



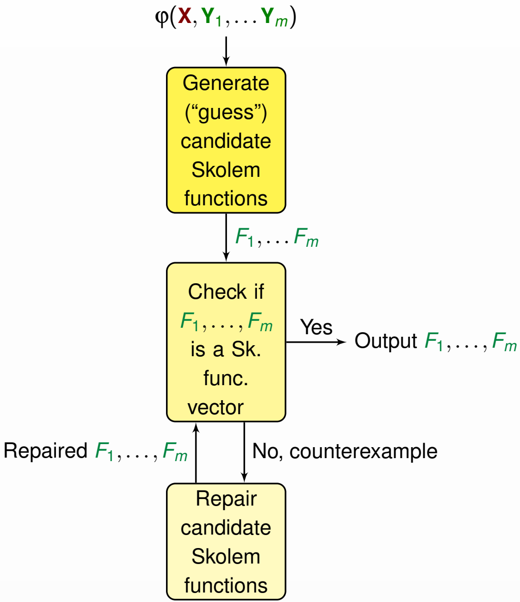
Overview of Guess-Check-Repair Paradigm



Overview of Guess-Check-Repair Paradigm



Overview of Guess-Check-Repair Paradigm

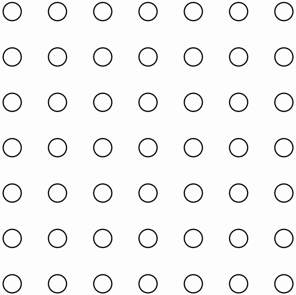


"Guess"-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

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

“Guess”-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

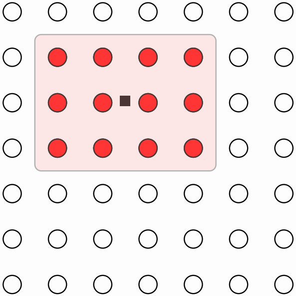
Find $\mathbf{F}(\mathbf{X})$ such that $\exists \mathbf{y}\ \varphi(\mathbf{X}, \mathbf{y}) \equiv \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X}))$



— Set of all valuations of $\mathbf{X}$.

“Guess”-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

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

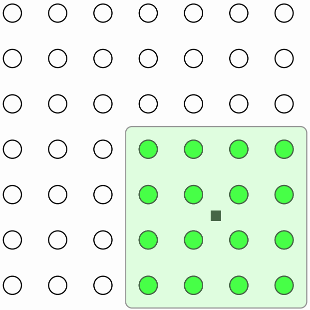


— Can't set $\mathbf{y}$ to 1 to satisfy φ: $\Gamma(\mathbf{X}) \triangleq \neg\varphi(\mathbf{X},\mathbf{y})[\mathbf{y}1]$

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

"Guess"-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

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

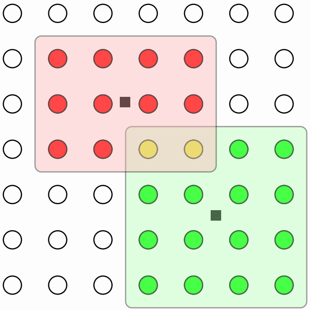


— Can't set $\mathbf{y}$ to 0 to satisfy φ: $\Delta(\mathbf{X}) \triangleq \neg\varphi(\mathbf{X}, \mathbf{y})[\mathbf{y}0]$

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

"Guess"-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

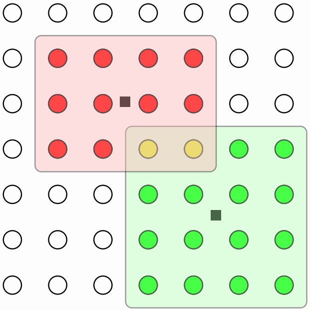
Find $\mathbf{F}(\mathbf{X})$ such that $\exists \mathbf{y}\ \varphi(\mathbf{X}, \mathbf{y}) \equiv \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X}))$



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

“Guess”-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

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



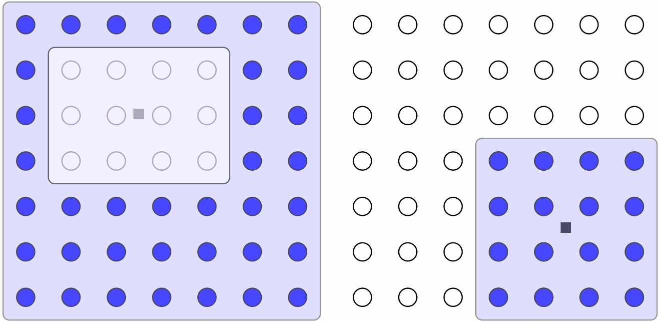
Lemma [Trivedi’03, Jiang’09,Fried et al’16]

Every Skolem function for $\mathbf{y}$ 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 $(\Gamma \setminus \Delta)$

“Guess”-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

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

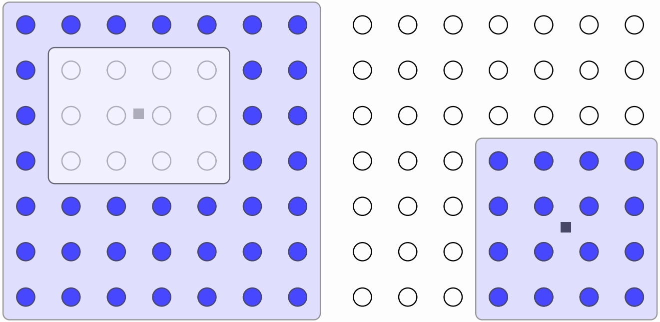


— Specific interpolants of $(\Delta \setminus \Gamma)$ & $(\Gamma \setminus \Delta)$

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

“Guess”-ing candidate Skolem functions ($|\mathbf{Y}| = 1$)

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



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

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

"Guess"-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

“Guess”-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general

“Guess”-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general
- E.g. $\varphi(\mathbf{X}, y_1, y_2) \equiv (x_1 \vee x_2 \vee y_1 \vee y_2) \wedge (y_1 \oplus y_2)$

“Guess”-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general
- E.g. $\varphi(\mathbf{X}, y_1, y_2) \equiv (x_1 \vee x_2 \vee y_1 \vee y_2) \wedge (y_1 \oplus y_2)$
  - y_2 **must be** $\neg y_1$

“Guess”-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general
- E.g. $\varphi(\mathbf{X}, y_1, y_2) \equiv (x_1 \vee x_2 \vee y_1 \vee y_2) \wedge (y_1 \oplus y_2)$
  - y_2 **must be** $\neg y_1$
- For what values of $\mathbf{X}$ can we not set y_1 to 1 (or 0)?

“Guess”-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general
- E.g. $\varphi(\mathbf{X}, y_1, y_2) \equiv (x_1 \vee x_2 \vee y_1 \vee y_2) \wedge (y_1 \oplus y_2)$
  - y_2 **must be** $\neg y_1$
- For what values of $\mathbf{X}$ can we not set y_1 to 1 (or 0)?
  - $\Gamma^{y_1}(\mathbf{X}) = \neg \exists y_2\, \varphi(\mathbf{X}, 1, y_2) = 0$

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general
- E.g. $\varphi(\mathbf{X}, y_1, y_2) \equiv (x_1 \vee x_2 \vee y_1 \vee y_2) \wedge (y_1 \oplus y_2)$
  - y_2 **must be** $\neg y_1$
- For what values of $\mathbf{X}$ can we not set y_1 to 1 (or 0)?
  - $\Gamma^{y_1}(\mathbf{X}) = \neg\exists y_2\ \varphi(\mathbf{X}, 1, y_2) = 0$
  - $\Delta^{y_1}(\mathbf{X}) = \neg\exists y_2\ \varphi(\mathbf{X}, 0, y_2) = 0$

“Guess”-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general
- E.g. $\varphi(\mathbf{X}, y_1, y_2) \equiv (x_1 \vee x_2 \vee y_1 \vee y_2) \wedge (y_1 \oplus y_2)$
  - y_2 **must be** $\neg y_1$
- For what values of $\mathbf{X}$ can we not set y_1 to 1 (or 0)?
  - $\Gamma^{y_1}(\mathbf{X}) = \neg\exists y_2\, \varphi(\mathbf{X}, 1, y_2) = 0$
  - $\Delta^{y_1}(\mathbf{X}) = \neg\exists y_2\, \varphi(\mathbf{X}, 0, y_2) = 0$
- From $\Gamma^{y_1}(\mathbf{X})$ and $\Delta^{y_1}(\mathbf{X})$, find Skolem function $F_1(\mathbf{X})$ for y_1
  - E.g. $F_1(\mathbf{X}) = \neg\Gamma^{y_1}(\mathbf{X}) = 1$

“Guess”-ing Game: ($|\mathbf{Y}| = 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, y_2)$

- Skolem function for y_2 depends on that for y_1 in general
- E.g. $\varphi(\mathbf{X}, y_1, y_2) \equiv (x_1 \vee x_2 \vee y_1 \vee y_2) \wedge (y_1 \oplus y_2)$
  - y_2 **must be** $\neg y_1$
- For what values of $\mathbf{X}$ can we not set y_1 to 1 (or 0)?
  - $\Gamma^{y_1}(\mathbf{X}) = \neg\exists y_2\ \varphi(\mathbf{X}, 1, y_2) = 0$
  - $\Delta^{y_1}(\mathbf{X}) = \neg\exists y_2\ \varphi(\mathbf{X}, 0, y_2) = 0$
- From $\Gamma^{y_1}(\mathbf{X})$ and $\Delta^{y_1}(\mathbf{X})$, find Skolem function $F_1(\mathbf{X})$ for y_1
  - E.g. $F_1(\mathbf{X}) = \neg\Gamma^{y_1}(\mathbf{X}) = 1$
- To find Skolem function for y_2, consider y_2 as sole output in $\varphi(\mathbf{X}, F_1(\mathbf{X}), y_2)$
  - E.g. $\varphi(\mathbf{X}, 1, y_2) = \neg y_2$
  - $\Gamma^{y_2}(\mathbf{X}) = \neg\varphi(\mathbf{X}, 1, 1) = 1$; $\Delta^{y_2}(\mathbf{X}) = \neg\varphi(\mathbf{X}, 1, 0) = 0$
  - $F_2(\mathbf{X}) = \neg\Gamma^{y_2}(\mathbf{X}) = 0$

"Guess"-ing Game: ($|\mathbf{Y}| > 2$)

Suppose relational spec is $\varphi(\mathbf{X}, y_1, \boxed{\mathbf{Y}_{2..m}})$

- Skolem function for $\boxed{\mathbf{Y}_{2..m}}$ depends on that for y_1 in general
- For what values of $\mathbf{X}$ can we not set y_1 to 1 (or 0)?
  - $\Gamma^{y_1}(\mathbf{X}) = \neg\exists\, \mathbf{Y}_{2..m}\ \varphi(\mathbf{X}, 1, \boxed{\mathbf{Y}_{2..m}})$
  - $\Delta^{y_1}(\mathbf{X}) = \neg\exists\, \mathbf{Y}_{2..m}\ \varphi(\mathbf{X}, 0, \boxed{\mathbf{Y}_{2..m}})$
- From $\Gamma^{y_1}(\mathbf{X})$ and $\Delta^{y_1}(\mathbf{X})$, find Skolem function $F_1(\mathbf{X})$ for y_1
- To find Skolem function for y_2, consider y_2 as sole output in $\varphi(\mathbf{X}, F_1(\mathbf{X}), y_2, \boxed{\mathbf{Y}_{3..m}})$

Drawbacks of approach:

- Existential quant elimination over long sequences of outputs expensive
- Nested compositions lead to blowup of representation

Can we work around these drawbacks?

A Useful Observation

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

A Useful Observation

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

Express

- y_m as $G_m(\mathbf{X}, y_1, \ldots y_{m-1})$

A Useful Observation

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

Express

- y_m as $G_m(\mathbf{X}, y_1, \ldots y_{m-1})$
- y_{m-1} as $G_{m-1}(\mathbf{X}, y_1, \ldots y_{m-2})$
- $\vdots$
- y_1 as $G_1(\mathbf{X})$

A Useful Observation

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

Express

- y_m as $G_m(\mathbf{X}, y_1, \ldots y_{m-1})$
- y_{m-1} as $G_{m-1}(\mathbf{X}, y_1, \ldots y_{m-2})$
- $\vdots$
- y_1 as $G_1(\mathbf{X})$

A $|\mathbf{X}|$-input, $|\mathbf{Y}|$-output circuit computing the desired Skolem function vector $(F_1, \ldots F_m)$ can be constructed with

- #gates $\leq \sum_{i=1}^{m}$ #gates(G_i) $+2m$
- #wires $\leq \sum_{i=1}^{m}$ #wires(G_i) $+\frac{m(m-1)}{2}$

A Useful Observation

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

Express

- y_m as $G_m(\mathbf{X}, y_1, \ldots y_{m-1})$
- y_{m-1} as $G_{m-1}(\mathbf{X}, y_1, \ldots y_{m-2})$
- $\vdots$
- y_1 as $G_1(\mathbf{X})$

A $|\mathbf{X}|$-input, $|\mathbf{Y}|$-output circuit computing the desired Skolem function vector $(F_1, \ldots F_m)$ can be constructed with

- #gates $\leq \sum_{i=1}^{m}$ #gates(G_i) $+2m$
- #wires $\leq \sum_{i=1}^{m}$ #wires(G_i) $+\frac{m(m-1)}{2}$

Sufficient to compute the G_i functions

“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m \; \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$

“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m) \quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$

“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

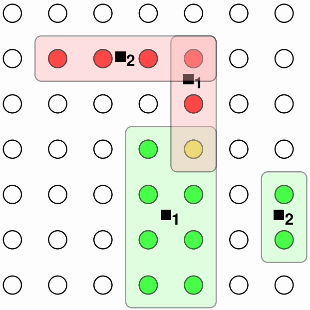
$$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m) \quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$$
$$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots) \quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$$

"Guess"-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$ $\quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$

$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots)$ $\quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$

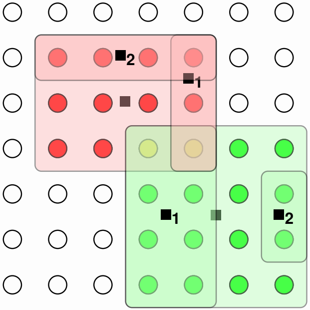


“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$ $\quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$

$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots)$ $\quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$



Lemma

If $\Gamma^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \wedge \varphi_2)(\mathbf{X}, 1, \ldots)$, then $\Gamma_1^{y_1} \vee \Gamma_2^{y_1} \Rightarrow \Gamma^{y_1}$

If $\Delta^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \wedge \varphi_2)(\mathbf{X}, 0, \ldots)$, then $\Delta_1^{y_1} \vee \Delta_2^{y_1} \Rightarrow \Delta^{y_1}$

“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \;\vee\; \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

"Guess"-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \vee \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$ $\quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$

$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots)$ $\quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$

Lemma

If $\Gamma^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 1, \ldots)$, then $\Gamma_1^{y_1} \wedge \Gamma_2^{y_1} \Leftrightarrow \Gamma^{y_1}$

If $\Delta^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 0, \ldots)$, then $\Delta_1^{y_1} \wedge \Delta_2^{y_1} \Leftrightarrow \Delta^{y_1}$

“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \vee \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$ $\quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$

$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots)$ $\quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$

Lemma

If $\Gamma^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 1, \ldots)$, then $\Gamma_1^{y_1} \wedge \Gamma_2^{y_1} \Leftrightarrow \Gamma^{y_1}$

If $\Delta^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 0, \ldots)$, then $\Delta_1^{y_1} \wedge \Delta_2^{y_1} \Leftrightarrow \Delta^{y_1}$

What if calculating $\Gamma_1^{y_i}$ or $\Delta_1^{y_i}$ hard?

"Guess"-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \vee \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$ $\quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$

$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots)$ $\quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$

Lemma

If $\Gamma^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 1, \ldots)$, then $\Gamma_1^{y_1} \wedge \Gamma_2^{y_1} \Leftrightarrow \Gamma^{y_1}$

If $\Delta^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 0, \ldots)$, then $\Delta_1^{y_1} \wedge \Delta_2^{y_1} \Leftrightarrow \Delta^{y_1}$

What if calculating $\Gamma_1^{y_i}$ or $\Delta_1^{y_i}$ hard?

- Long sequences of quantification are of concern!

"Guess"-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \lor \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m \; \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$ $\quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m \; \varphi_1(\mathbf{X}, 0, \ldots)$

$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \; \varphi_2(\mathbf{X}, 1, \ldots)$ $\quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$

Lemma

If $\Gamma^{y_1} \triangleq \neg\exists y_2 \ldots y_m \; (\varphi_1 \lor \varphi_2)(\mathbf{X}, 1, \ldots)$, then $\Gamma_1^{y_1} \land \Gamma_2^{y_1} \Leftrightarrow \Gamma^{y_1}$

If $\Delta^{y_1} \triangleq \neg\exists y_2 \ldots y_m \; (\varphi_1 \lor \varphi_2)(\mathbf{X}, 0, \ldots)$, then $\Delta_1^{y_1} \land \Delta_2^{y_1} \Leftrightarrow \Delta^{y_1}$

What if calculating $\Gamma_1^{y_i}$ or $\Delta_1^{y_i}$ hard?

- Long sequences of quantification are of concern!
- Using under-approximations of $\Gamma_1^{y_i}$ and $\Delta_1^{y_i}$ yields under-approximations of Γ^{y_i} and Δ^{y_i}

“Guess”-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \vee \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m) \quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$$
$$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots) \quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$$

Lemma

If $\Gamma^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 1, \ldots)$, then $\Gamma_1^{y_1} \wedge \Gamma_2^{y_1} \Leftrightarrow \Gamma^{y_1}$
If $\Delta^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 0, \ldots)$, then $\Delta_1^{y_1} \wedge \Delta_2^{y_1} \Leftrightarrow \Delta^{y_1}$

What if calculating $\Gamma_1^{y_i}$ or $\Delta_1^{y_i}$ hard?

- Long sequences of quantification are of concern!
- Using under-approximations of $\Gamma_1^{y_i}$ and $\Delta_1^{y_i}$ yields under-approximations of Γ^{y_i} and Δ^{y_i}
  - Not so for over-approximations!
    - $\Gamma_1{}^{y_i} \vee (\wedge)\ \Gamma_2{}^{y_i} \Rightarrow (\Leftrightarrow)\Gamma^{y_i}$
    - $\Delta_1{}^{y_i} \vee (\wedge)\ \Delta_2{}^{y_i} \Rightarrow (\Leftrightarrow)\Delta^{y_i}$

"Guess"-ing Compositionally: A High-level View

Suppose $\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \vee \varphi_2(\mathbf{X}, Y)$, where $Y = y_1, \ldots y_m$

$\Gamma_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 1, y_2 \ldots y_m)$ $\quad \Delta_1^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_1(\mathbf{X}, 0, \ldots)$

$\Gamma_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ \varphi_2(\mathbf{X}, 1, \ldots)$ $\quad \Delta_2^{y_1} \triangleq \neg\exists y_2 \ldots y_m \varphi_2(\mathbf{X}, 0, \ldots)$

Lemma

If $\Gamma^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 1, \ldots)$, then $\Gamma_1^{y_1} \wedge \Gamma_2^{y_1} \Leftrightarrow \Gamma^{y_1}$

If $\Delta^{y_1} \triangleq \neg\exists y_2 \ldots y_m\ (\varphi_1 \vee \varphi_2)(\mathbf{X}, 0, \ldots)$, then $\Delta_1^{y_1} \wedge \Delta_2^{y_1} \Leftrightarrow \Delta^{y_1}$

What if calculating $\Gamma_1^{y_i}$ or $\Delta_1^{y_i}$ hard?

- Long sequences of quantification are of concern!
- Using under-approximations of $\Gamma_1^{y_i}$ and $\Delta_1^{y_i}$ yields under-approximations of Γ^{y_i} and Δ^{y_i}
  - Not so for over-approximations!
    - $\Gamma_1{}^{y_i} \vee (\wedge)\ \Gamma_2{}^{y_i} \Rightarrow (\Leftrightarrow)\Gamma^{y_i}$
    - $\Delta_1{}^{y_i} \vee (\wedge)\ \Delta_2{}^{y_i} \Rightarrow (\Leftrightarrow)\Delta^{y_i}$
- Fortunately, non-trivial under-approx of Γ^{y_i} and Δ^{y_i} not hard to obtain

“Guess”-ing with under-approximations of Γ, Δ

- Suppose $\gamma_1^{y_i} \Rightarrow \Gamma_1^{y_i}$; $\delta_1^{y_i} \Rightarrow \Delta_1^{y_i}$

“Guess”-ing with under-approximations of Γ, Δ

- Suppose $\gamma_1^{y_i} \Rightarrow \Gamma_1^{y_i}$; $\delta_1^{y_i} \Rightarrow \Delta_1^{y_i}$
- $\varphi \equiv \varphi_1 \wedge \varphi_2$
  - $\gamma_1^{y_i} \vee \gamma_1^{y_i} \;\Rightarrow\; \Gamma_1^{y_i} \vee \Gamma_1^{y_i} \;\Rightarrow\; \Gamma^{y_i}$

"Guess"-ing with under-approximations of Γ, Δ

- Suppose $\gamma_1^{y_i} \Rightarrow \Gamma_1^{y_i}$; $\delta_1^{y_i} \Rightarrow \Delta_1^{y_i}$
- $\varphi \equiv \varphi_1 \wedge \varphi_2$
  - $\gamma_1^{y_i} \vee \gamma_1^{y_i} \Rightarrow \Gamma_1^{y_i} \vee \Gamma_1^{y_i} \Rightarrow \Gamma^{y_i}$
- $\varphi \equiv \varphi_1 \vee \varphi_2$
  - $\gamma_1^{y_i} \wedge \gamma_1^{y_i} \Rightarrow \Gamma_1^{y_i} \wedge \Gamma_1^{y_i} \Leftrightarrow \Gamma^{y_i}$
- Similarly for Δ^{y_i}

“Check”-ing correctness of candidate Skolem func. vector

Given candidate Skolem functions $F_1, \ldots F_m$,

$$\text{Is } \forall \mathbf{X}\left(\exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \Leftrightarrow \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X})\right) ?$$

“Check”-ing correctness of candidate Skolem func. vector

Given candidate Skolem functions $F_1, \ldots F_m$,

$$\text{Is } \forall \mathbf{X} \big(\, \exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \;\Leftrightarrow\; \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X})) \, \big) \text{ ?}$$

Can we avoid using a QBF solver?

“Check”-ing correctness of candidate Skolem func. vector

Given candidate Skolem functions $F_1, \ldots F_m$,

$$\text{Is } \forall \mathbf{X} \left(\exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \ \Leftrightarrow \ \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X}) \right) \ ?$$

Can we avoid using a QBF solver?

Yes, we can! [FMCAD15]

- Propositional error formula $\varepsilon(\mathbf{X}, \mathbf{Y}, \mathbf{Y}')$:

$$\left(\varphi(\mathbf{X}, \mathbf{Y}') \wedge \bigwedge_{j=1}^{m} (\mathbf{Y}_j \Leftrightarrow F_j) \wedge \neg\varphi(\mathbf{X}, \mathbf{Y})\right)$$

“Check”-ing correctness of candidate Skolem func. vector

Given candidate Skolem functions $F_1, \ldots F_m$,

$$\text{Is } \forall \mathbf{X} \big(\exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \Leftrightarrow \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X}) \big) \text{ ?}$$

Can we avoid using a QBF solver?

Yes, we can! [FMCAD15]

- Propositional error formula $\varepsilon(\mathbf{X}, \mathbf{Y}, \mathbf{Y}')$:
$$\big(\varphi(\mathbf{X}, \mathbf{Y}') \wedge \bigwedge_{j=1}^{m} (\mathbf{Y}_j \Leftrightarrow F_j) \wedge \neg\varphi(\mathbf{X}, \mathbf{Y})\big)$$
- ε unsatisfiable iff $F_1, \ldots F_m$ is correct Skolem function vector

"Check"-ing correctness of candidate Skolem func. vector

Given candidate Skolem functions $F_1, \ldots F_m$,

$$\text{Is } \forall \mathbf{X} \left(\exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \Leftrightarrow \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X})\right) ?$$

Can we avoid using a QBF solver?

Yes, we can! [FMCAD15]

- Propositional error formula $\varepsilon(\mathbf{X}, \mathbf{Y}, \mathbf{Y}')$:

$$\left(\varphi(\mathbf{X}, \mathbf{Y}') \wedge \bigwedge_{j=1}^{m} (\mathbf{Y}_j \Leftrightarrow F_j) \wedge \neg\varphi(\mathbf{X}, \mathbf{Y})\right)$$

- ε unsatisfiable iff $F_1, \ldots F_m$ is correct Skolem function vector
  - Say, σ = satisfying assignment of ε

"Check"-ing correctness of candidate Skolem func. vector

Given candidate Skolem functions $F_1, \ldots F_m$,

$$\text{Is } \forall \mathbf{X} \Big(\exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \Leftrightarrow \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X})\Big) \text{ ?}$$

Can we avoid using a QBF solver?

Yes, we can! [FMCAD15]

- Propositional error formula $\varepsilon(\mathbf{X}, \mathbf{Y}, \mathbf{Y}')$:
$$\left(\varphi(\mathbf{X}, \mathbf{Y}') \wedge \bigwedge_{j=1}^{m} (\mathbf{Y}_j \Leftrightarrow F_j) \wedge \neg\varphi(\mathbf{X}, \mathbf{Y})\right)$$
- ε unsatisfiable iff $F_1, \ldots F_m$ is correct Skolem function vector
  - Say, σ = satisfying assignment of ε
  - On input $\sigma(\mathbf{X})$, $\mathbf{F}$ evaluates to $\sigma(\mathbf{Y})$, where
    - $\varphi(\sigma(\mathbf{X}), \sigma(\mathbf{Y})) = 0$
    - $\varphi(\sigma(\mathbf{X}), \sigma(\mathbf{Y}')) = 1$

"Check"-ing correctness of candidate Skolem func. vector

Given candidate Skolem functions $F_1, \ldots F_m$,

$$\text{Is } \forall \mathbf{X} \left(\exists \mathbf{Y} \varphi(\mathbf{X}, \mathbf{Y}) \Leftrightarrow \varphi(\mathbf{X}, \mathbf{F}(\mathbf{X}) \right) ?$$

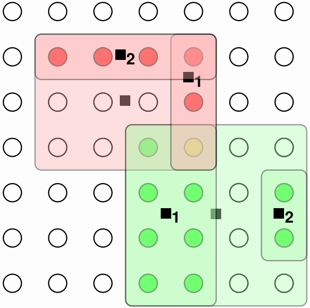
Can we avoid using a QBF solver?

Yes, we can! [FMCAD15]

- Propositional error formula $\varepsilon(\mathbf{X}, \mathbf{Y}, \mathbf{Y}')$:
$$\left(\varphi(\mathbf{X}, \mathbf{Y}') \wedge \bigwedge_{j=1}^{m} (\mathbf{Y}_j \Leftrightarrow F_j) \wedge \neg\varphi(\mathbf{X}, \mathbf{Y})\right)$$
- ε unsatisfiable iff $F_1, \ldots F_m$ is correct Skolem function vector
  - Say, σ = satisfying assignment of ε
  - On input $\sigma(\mathbf{X})$, $\mathbf{F}$ evaluates to $\sigma(\mathbf{Y})$, where
    - $\varphi(\sigma(\mathbf{X}), \sigma(\mathbf{Y})) = 0$
    - $\varphi(\sigma(\mathbf{X}), \sigma(\mathbf{Y}')) = 1$
  - σ is counterexample to the claim that $F_1, \ldots F_m$ is a correct Skolem function vector

Repairing candidate Skolem functions: A High-level View

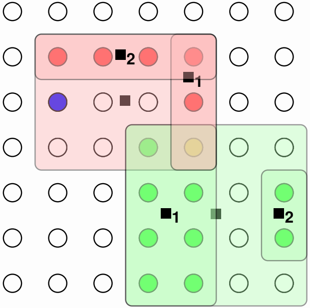
$$\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \ \wedge \ \varphi_2(\mathbf{X}, Y)$$



Repairing candidate Skolem functions: A High-level View

$\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$

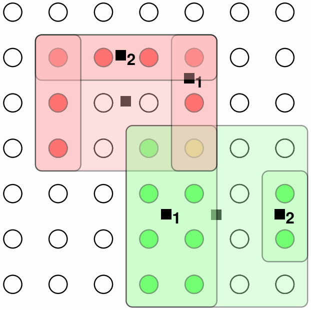
Counterexample



Repairing candidate Skolem functions: A High-level View

$\varphi(\mathbf{X}, Y) \equiv \varphi_1(\mathbf{X}, Y) \wedge \varphi_2(\mathbf{X}, Y)$

Expansion around CEX

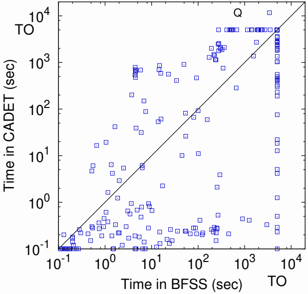


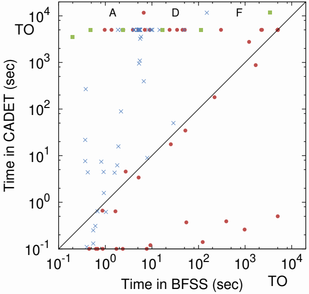
- Always work with under-approximations of Γ and Δ
- Since "proposed" Skolem function is $\neg\Gamma$, intermediate approximations of Skolem functions are over-approximations (abstractions)

Comparison with other tools

BFSS vis-a-vis *CADET* [Rabe & Seshia'16]

[Comparisons with other tools in paper]

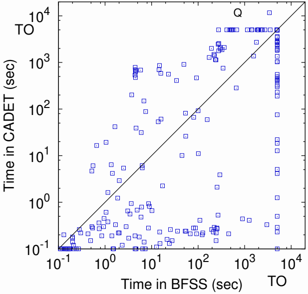


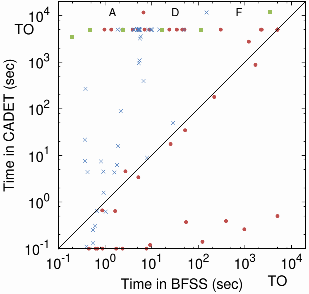


Q: QBFEval, A: Arithmetic, F: Factorization, D: Disjunctive Decomposition. TO: Timeout (3600 sec)

Comparison with other tools

BFSS vis-a-vis *CADET* [Rabe & Seshia'16]
[Comparisons with other tools in paper]

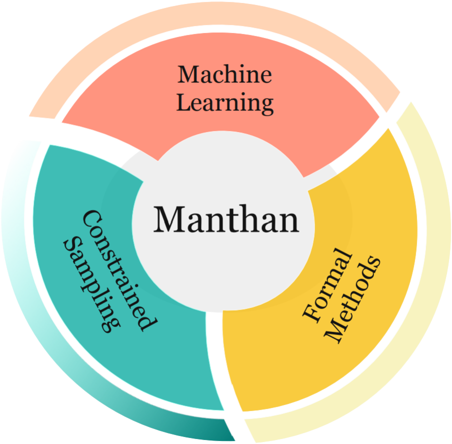


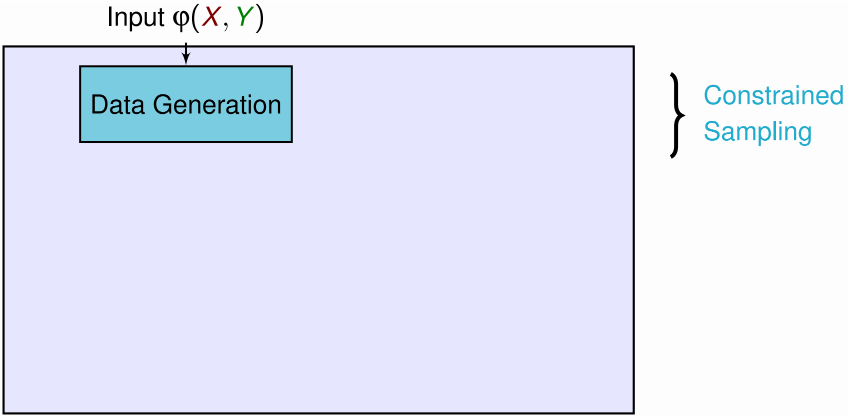


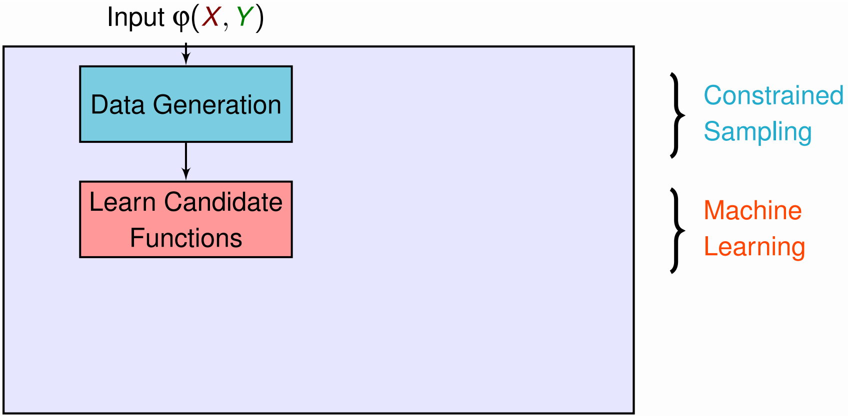
Q: QBFEval, A: Arithmetic, F: Factorization, D: Disjunctive Decomposition. TO: Timeout (3600 sec)

- Mixed results: tools have orthogonal strengths
- Using *CADET* and *BFSS* as a portfolio solver sounds promising

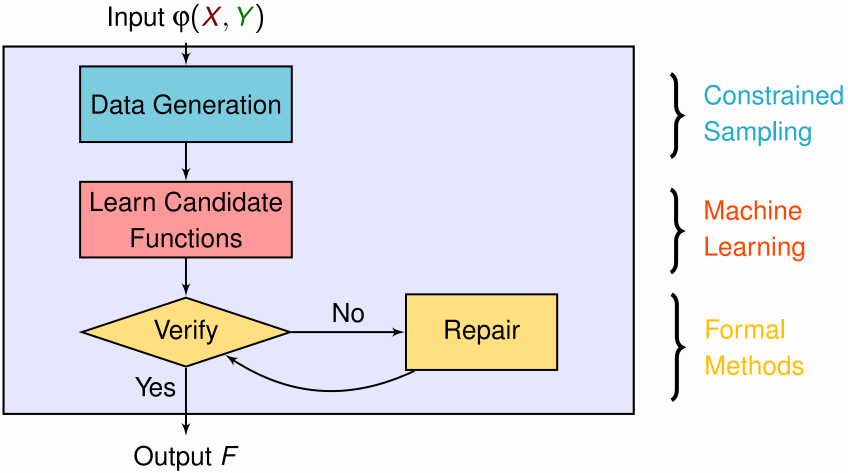
Another Flavour of Guess-Check-Repair




Input φ(X, Y)
Data Generation
Constrained Sampling


Input φ(X, Y)
Data Generation
Learn Candidate Functions
Constrained Sampling
Machine Learning

Manthan


Input φ(X, Y)
Data Generation
Learn Candidate Functions
Verify
No
Repair
Yes
Output F
Constrained Sampling
Machine Learning
Formal Methods

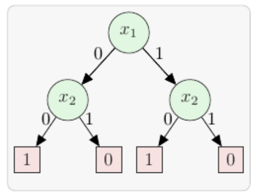
$\varphi(x_1, x_2, y_1, y_2) \longrightarrow$

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0

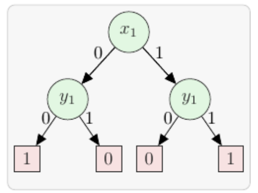
Learn Candidate Functions

Taming the Curse of Abstractions via Learning with Errors

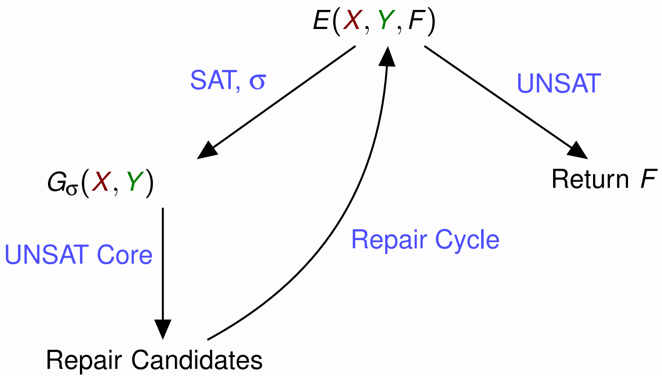
x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



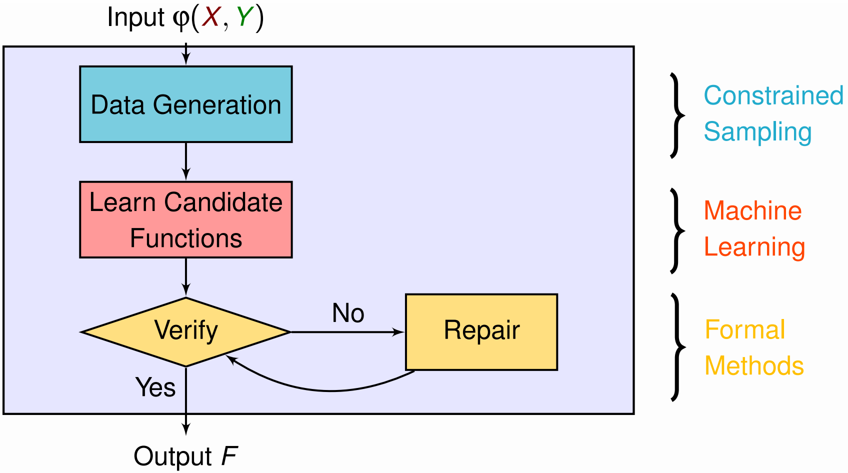
$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0



$p_1 := (\neg x_1 \wedge \neg y_1)$,
$p_2 := (x_1 \wedge y_1)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0



$E(X, Y, F)$
SAT, σ
UNSAT
$G_\sigma(X, Y)$
Return F
UNSAT Core
Repair Cycle
Repair Candidates


Input φ(X, Y)
Data Generation
Learn Candidate Functions
Verify
No
Repair
Yes
Output F
Constrained Sampling
Machine Learning
Formal Methods

Data Generation

Potential Strategy: Randomly sample satisfying assignment of $\varphi(X, Y)$.

Challenge: Multiple valuations of y_1, y_2 for same valuation of x_1, x_2.

Data Generation

Potential Strategy: Randomly sample satisfying assignment of $\varphi(X, Y)$.

Challenge: Multiple valuations of y_1, y_2 for same valuation of x_1, x_2.

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

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

Data Generation

$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \lor x_2 \lor y_1) \land (\neg x_1 \lor \neg x_2 \lor \neg y_2)$$

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

$\xrightarrow{\text{Uniform Sampler}}$

x_1	x_2	y_1	y_2
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

Data Generation

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

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

$\xrightarrow{\text{Uniform Sampler}}$

x_1	x_2	y_1	y_2
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

- Possible Skolem functions:
  - $f_1(x_1, x_2) = \neg(x_1 \vee x_2)$
  - $f_2(x_1, x_2) = \neg(x_1 \wedge x_2)$

Data Generation

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

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

$\xrightarrow{\text{Uniform Sampler}}$

x_1	x_2	y_1	y_2
0	0	1	1
0	1	0	1
1	0	0	1
1	1	0	0

- Possible Skolem functions:
  - $f_1(x_1, x_2) = \neg(x_1 \vee x_2)$ $f_1(x_1, x_2) = \neg x_1$ $f_1(x_1, x_2) = \neg x_2$ $f_1(x_1, x_2) = 1$
  - $f_2(x_1, x_2) = \neg(x_1 \wedge x_2)$ $f_2(x_1, x_2) = \neg x_1$ $f_2(x_1, x_2) = \neg x_2$ $f_2(x_1, x_2) = 0$

Data Generation

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

x_1	x_2	y_1	y_2
0	0	1	0/1
0	1	0/1	0/1
1	0	0/1	0/1
1	1	0/1	0

$\xrightarrow{\text{Magical Sampler}}$

x_1	x_2	y_1	y_2
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	0

- Possible Skolem functions:
  - $f_1(x_1, x_2) = \neg(x_1 \vee x_2)$ $\quad f_1(x_1, x_2) = \neg x_1$ $\quad f_1(x_1, x_2) = \neg x_2$ $\quad f_1(x_1, x_2) = 1$
  - $f_2(x_1, x_2) = \neg(x_1 \wedge x_2)$ $\quad f_2(x_1, x_2) = \neg x_1$ $\quad f_2(x_1, x_2) = \neg x_2$ $\quad f_2(x_1, x_2) = 0$

Weighted Sampling to Rescue

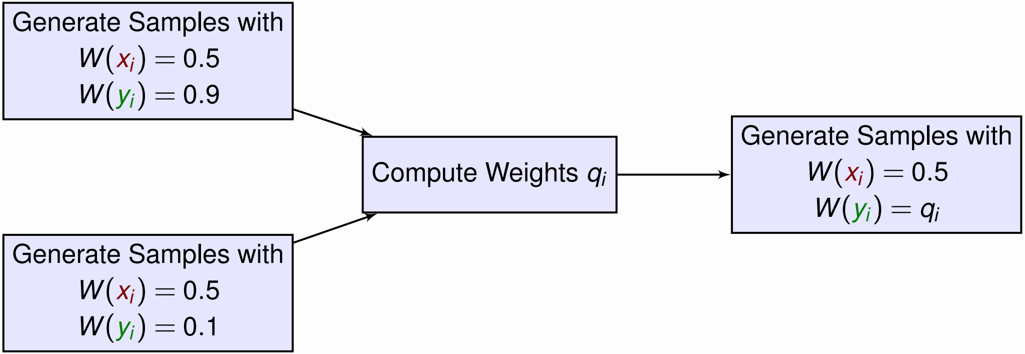
- $W : X \cup Y \mapsto [0,1]$
- The probability of generation of an assignment is proportional to its weight.

$$W(\sigma) = \prod_{\sigma(z_i)=1} W(z_i) \prod_{\sigma(z_i)=0} (1 - W(z_i))$$

- Example: $W(x_1) = 0.5 \quad W(x_2) = 0.5 \quad W(y_1) = 0.9 \quad W(y_2) = 0.1$
  $\sigma_1 = \{x_1 \mapsto 1, x_2 \mapsto 0, y_1 \mapsto 0, y_2 \mapsto 1\}$

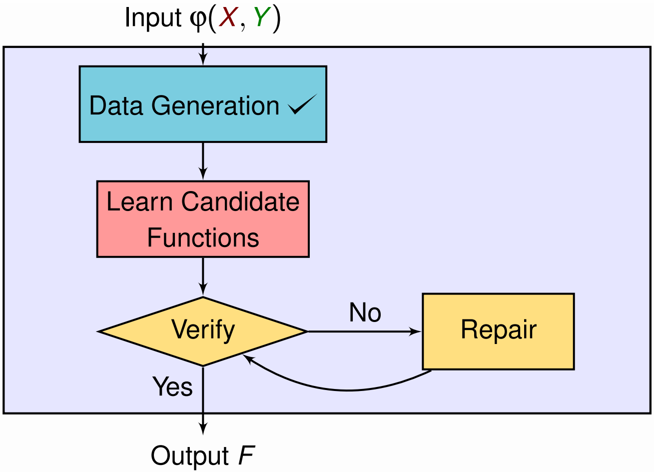
$$W(\sigma_1) = 0.5 \times (1 - 0.5) \times (1 - 0.9) \times 0.1 = 0.0025$$

- Uniform sampling is a special case where all variables are assigned weight of 0.5.


Generate Samples with
$W(x_i) = 0.5$
$W(y_i) = 0.9$
Generate Samples with
$W(x_i) = 0.5$
$W(y_i) = 0.1$
Compute Weights q_i
Generate Samples with
$W(x_i) = 0.5$
$W(y_i) = q_i$

Different Sampling Strategies

- Knowledge representation based techniques
  - (Yuan,Shultz, Pixley,Miller,Aziz 1999)
  - (Yuan,Aziz, Pixley,Albin, 2004)
  - (Kukula and Shiple, 2000)
  - (Sharma, Gupta, M., Roy, 2018)
  - (Gupta, Sharma, M., Roy, 2019)
- Hashing based techniques
  - (Chakraborty, M., and Vardi 2013, 2014,2015)
  - (Soos, M., and Gocht 2020)
- Mutation based techniques
  - (Dutra, Laeufer, Bachrach, Sen, 2018)
- Markov Chain Monte Carlo based techniques
  - (Wei and Selman,2005)
  - (Kitchen,2010)
- Constraint solver based techniques
  - (Ermon, Gomes, Sabharwal, Selman,2012)
- Belief networks based techniques
  - (Dechter, Kask, Bin, Emek,2002)
  - (Gogate and Dechter,2006)


Input φ(X, Y)
Data Generation ✓
Learn Candidate Functions
Verify
No
Repair
Yes
Output F

Learn Candidate Function: Decision Tree Classifier

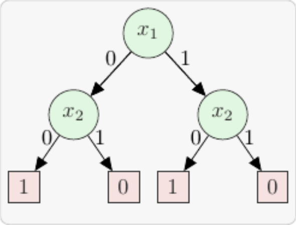
$$\varphi(x_1, x_2, y_1, y_2) : (x_1 \vee x_2 \vee y_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg y_2)$$

- To learn y_2
  - Feature set: valuation of x_1, x_2, y_1
  - Label: valuation of y_2
  - Learn decision tree to represent y_2 in terms of x_1, x_2, y_1
- To learn y_1
  - Feature set: valuation of x_1, x_2
  - Label: valuation of y_1
  - Learn decision tree to represent y_1 in terms of x_1, x_2

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0

Learning Candidate Functions

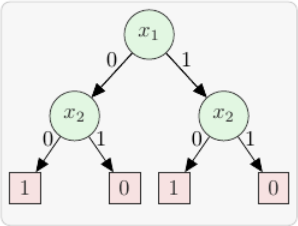
x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0

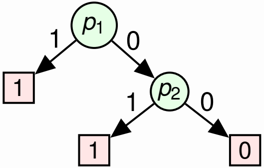
Learning Candidate Functions

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



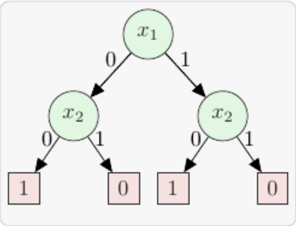
$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0

Can reorder p_1, p_2
Learning one level decision list



What Kind of Learning

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



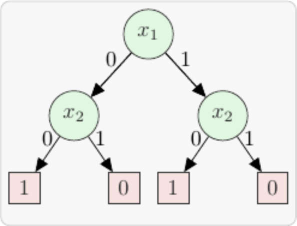
$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0

Learning without Error
Every row is a solution of $\varphi(X, Y)$

Learning with Errors
The data is only a subset of solutions.

What Kind of Learning

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



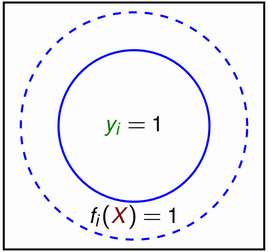
$p_1 := (\neg x_1 \wedge \neg x_2)$,
$p_2 := (x_1 \wedge \neg x_2)$
f_1 = if p_1 then 1
elif p_2 then 1
else 0

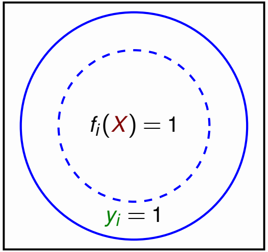
Learning without Error
Every row is a solution of $\varphi(X, Y)$

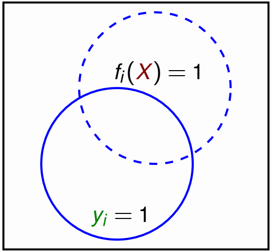
Learning with Errors
The data is only a subset of solutions.

Learn with Errors: Approximations <u>not</u> Abstractions

Abstraction vs Approximation







$y_i \rightarrow f_i(X)$

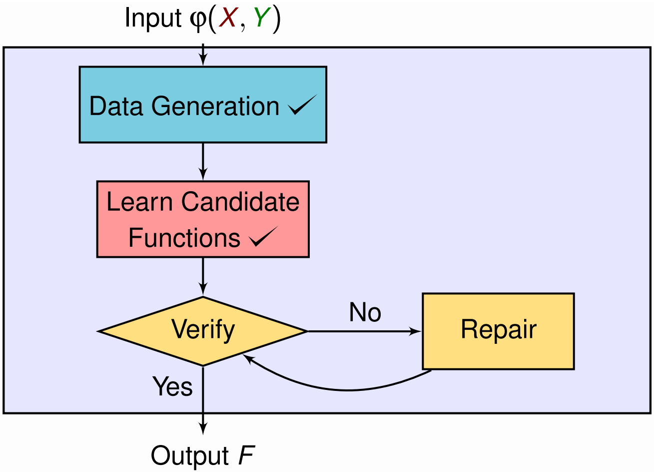
$f_i(X) \rightarrow y_i$

Abstraction

Approximation

$y_i = 1, f_i(X) = 0$

$y_i = 0, f_i(X) = 1$


Input φ(X, Y)
Data Generation ✓
Learn Candidate Functions ✓
Verify
No
Repair
Yes
Output F

Verification of Candidate Functions

$$E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$$

(JSCTA'15)

- If $E(X, Y, Y')$ is UNSAT: $\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$
  - Return F
- If $E(X, Y, Y')$ is SAT: $\exists Y \varphi(X, Y) \not\equiv \varphi(X, F(X))$
  - Let $\sigma \models E(X, Y, Y')$ be a counterexample to fix.

Repair Candidate Identification

$$E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$$

$\sigma \models E(X, Y, Y')$ be a counterexample to fix.

- Let $\sigma := \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 1, y_2 \mapsto 1, y_1' \mapsto 0, y_2' \mapsto 0\}$.
- Potential repair candidates: All y_i where $\sigma[y_i] \neq \sigma[y_i']$.

Repair Candidate Identification

$$E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$$

$\sigma \models E(X, Y, Y')$ be a counterexample to fix.

- Let $\sigma := \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 1, y_2 \mapsto 1, y'_1 \mapsto 0, y'_2 \mapsto 0\}$.
- Potential repair candidates: All y_i where $\sigma[y_i] \neq \sigma[y'_i]$.
- $\varphi(X, Y)$ is Boolean Relation.
  - So it can be $\hat{\sigma} = \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 0, y_2 \mapsto 1, y'_1 \mapsto 0, y'_2 \mapsto 0\}$
  - We would not repair f_1.

Repair Candidate Identification

$$E(X, Y, Y') := \varphi(X, Y) \wedge \neg\varphi(X, Y') \wedge (Y' \leftrightarrow F(X))$$

$\sigma \models E(X, Y, Y')$ be a counterexample to fix.

- Let $\sigma := \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 1, y_2 \mapsto 1, y_1' \mapsto 0, y_2' \mapsto 0\}$.
- Potential repair candidates: All y_i where $\sigma[y_i] \neq \sigma[y_i']$.
- $\varphi(X, Y)$ is Boolean Relation.
  - So it can be $\hat{\sigma} = \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 0, y_2 \mapsto 1, y_1' \mapsto 0, y_2' \mapsto 0\}$
  - We would not repair f_1.
- MaxSAT-based Identification of *nice counterexamples*:
  - Hard Clauses $\varphi(X, Y) \wedge (X \leftrightarrow \sigma[X])$.
  - Soft Clauses $(Y \leftrightarrow \sigma[Y'])$.
- Candidates to repair: Y variables in the violated soft clauses

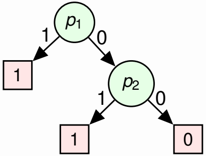
Repairing Approximations

- $\sigma = \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 0, y_2 \mapsto 1, y_1' \mapsto 0, y_2' \mapsto 0\}$, and we want to repair f_2.
- Potential Repair: If $\underbrace{x_1 \wedge x_2 \wedge \neg y_1}_{\beta=\{x_1, x_2, \neg y_1\}}$ then $y_2 = 1$
- Would be nice to have $\beta = \{x_1, x_2\}$ or even $\beta = \{x_1\}$
- Challenge: How do we find small β?
  - $G_\sigma(X, Y) := \varphi(X, Y) \wedge x_1 \wedge x_2 \wedge \neg y_1 \wedge \neg y_2$
  - β:= Literals in UNSAT Core of $G_\sigma(X, Y)$

Repair: Adding Level to Decision List

- Candidates are from one level decision list:
  - Say we have paths p_1, p_2 with the leaf node label as 1.
  - Learned decision tree: If p_1 then 1, elif p_2 then 1, else 0.
  - p_1, p_2 can be reordered.

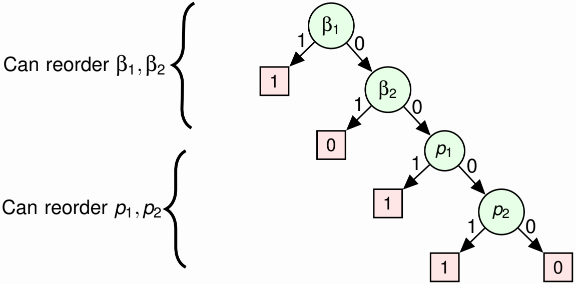
Can reorder p_1, p_2



Repair: Adding Level to Decision List

- Candidates are from one level decision list:
  - Say we have paths p_1, p_2 with the leaf node label as 1.
  - Learned decision tree: If p_1 then 1, elif p_2 then 1, else 0.
  - p_1, p_2 can be reordered.
- Suppose in repair iterations, we have learned: If β_1 then 1, . . . β_2 then 0

  . . . ...
- β_1 and β_2 can be reordered.
- From one-level decision list to two-level decision list.



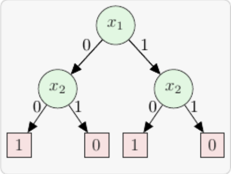
Manthan

$\varphi(X, Y)$
$X = \{x_1, x_2\}$
$Y = \{y_1, y_2\}$

$\xrightarrow{\text{Data Generation}}$

x_1	x_2	y_1	y_2
0	0	1	0
0	1	0	1
1	0	1	0
1	1	0	1

$\xrightarrow{\text{Learn Candidates}}$



Verify Candidates

Check Satisfiability of $E(X, Y, Y')$

SAT, σ

$G_\sigma(X, Y)$

UNSAT Core-based Repair

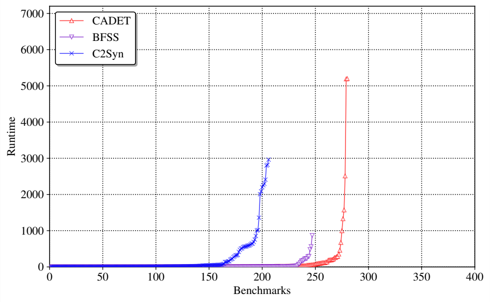
UNSAT

Return F

Experimental Evaluations

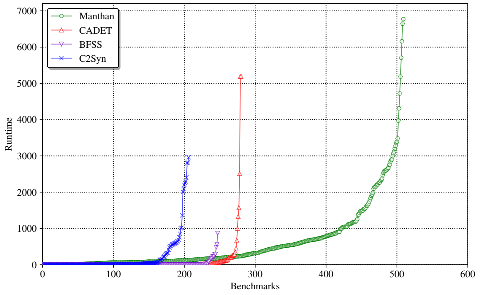
- 609 Benchmarks from:
  - QBFEval competition
  - Arithmetic
  - Disjunctive decomposition
  - Factorization
- Compared Manthan with State-of-the-art tools: CADET (Rabe et al., 2019), BFSS (Akshay et al. ,2018), C2Syn (Chakraborty et al., 2019).
- Timeout: 7200 seconds.

Experimental Evaluations



C2Syn	BFSS	CADET
206	247	280

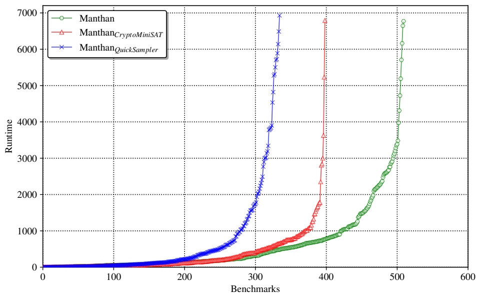
Experimental Evaluations



C2Syn	BFSS	CADET	Manthan
206	247	280	509

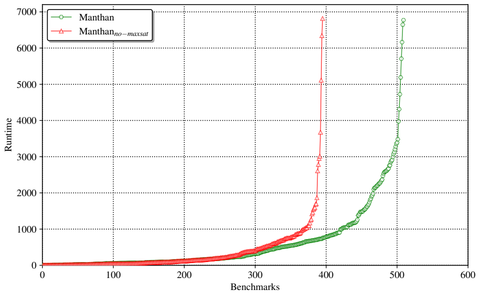
An increase of 229 benchmarks.

Impact of Choices (I): Data Generation



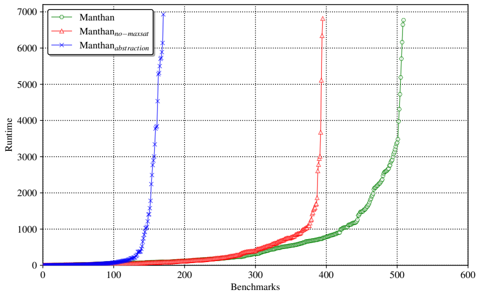
QuickSampler	CryptoMiniSAT	CMSGen
332	399	509

Impact of Choices (II): Use of MaxSAT



Manthan$_{\text{no-maxsat}}$	Manthan
396	509

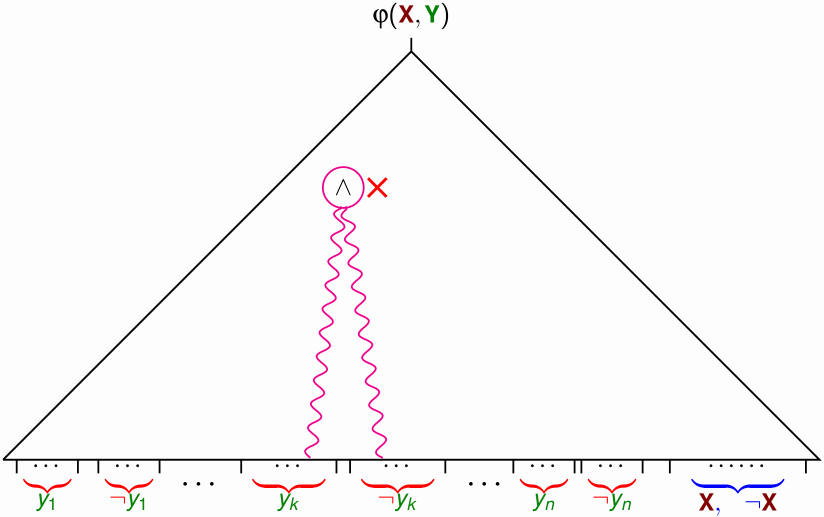
Impact of Choices (III): Abstraction vs Approximation



$\text{Manthan}_{\text{abstraction}}$	$\text{Manthan}_{\text{no-maxsat}}$	Manthan
171	396	509

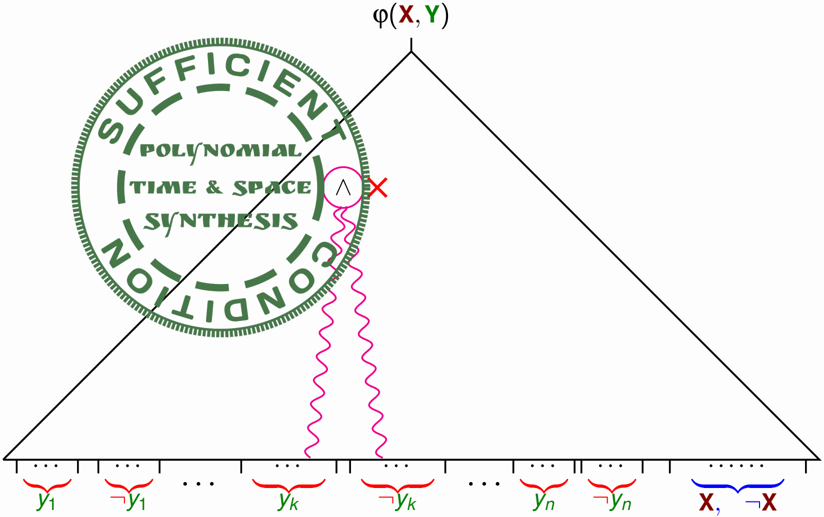
A Flavour of Knowledge Compilation Based Approach

Weak DNNF (wDNNF): Forbidden **structure**



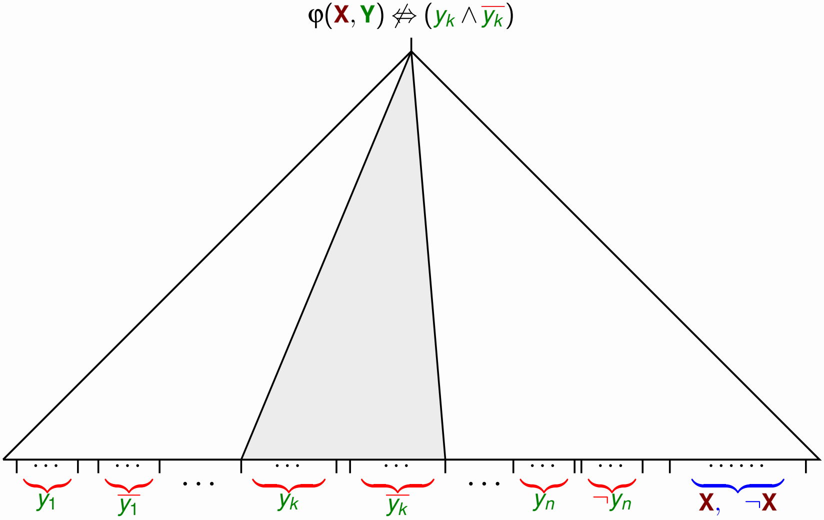
A Flavour of Knowledge Compilation Based Approach

Weak DNNF (wDNNF): Forbidden **structure**



Special Normal Forms (Prior Work)

Synthesis Negation Normal Form (SynNNF): Forbidden **semantics**



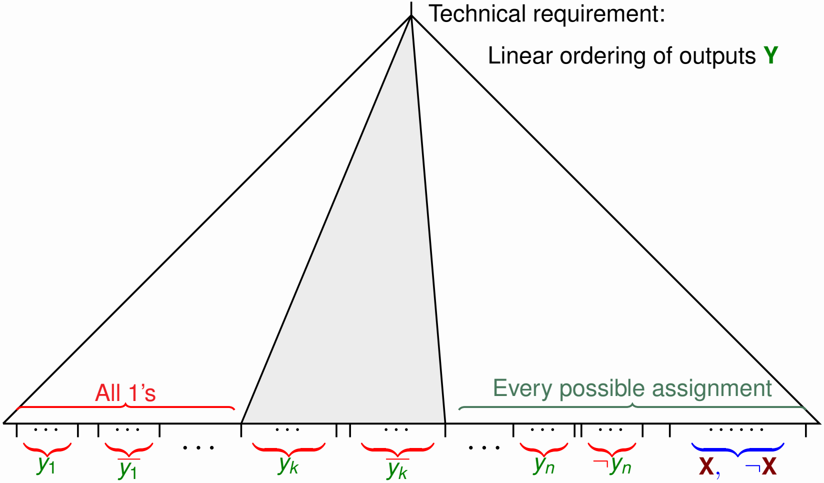
Special Normal Forms (Prior Work)

Synthesis Negation Normal Form (SynNNF): Forbidden **semantics**

$$\varphi(\mathbf{X}, \mathbf{Y}) \not\Leftarrow (y_k \wedge \overline{y_k})$$

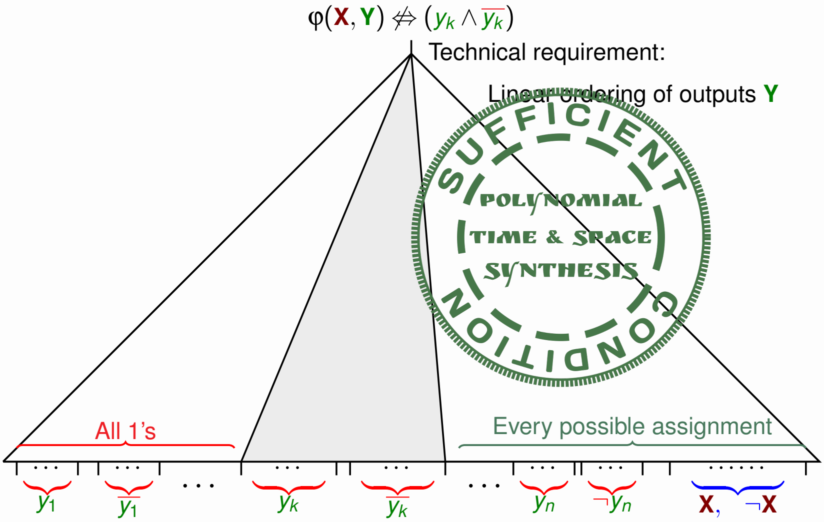
Technical requirement:

Linear ordering of outputs **Y**



Special Normal Forms (Prior Work)

Synthesis Negation Normal Form (SynNNF): Forbidden **semantics**



Can we get necessary & sufficient condition?

Characterizing poly-time and poly-size BFnS

Does there exist a ”semantically universal” class $\mathcal{C}^\star$ of ckts s.t.:

P1 : BFnS is poly-time for $\mathcal{C}^\star$

Can we get necessary & sufficient condition?

Characterizing poly-time and poly-size BFnS

Does there exist a ”semantically universal” class $\mathcal{C}^\star$ of ckts s.t.:

P1 : BFnS is poly-time for $\mathcal{C}^\star$

P2 : For every class $\mathcal{C}$ of ckts:

1. BFnS is poly-time for $\mathcal{C}$ iff $\mathcal{C}$ compiles to $\mathcal{C}^\star$ in poly-time.

Can we get necessary & sufficient condition?

Characterizing poly-time and poly-size BFnS

Does there exist a "semantically universal" class $\mathcal{C}^\star$ of ckts s.t.:

P1 : BFnS is poly-time for $\mathcal{C}^\star$

P2 : For every class $\mathcal{C}$ of ckts:

1. BFnS is poly-time for $\mathcal{C}$ iff $\mathcal{C}$ compiles to $\mathcal{C}^\star$ in poly-time.
2. BFnS is poly-size for $\mathcal{C}$ iff $\mathcal{C}$ compiles to poly-size ckts in $\mathcal{C}^\star$

Can we get necessary & sufficient condition?

Characterizing poly-time and poly-size BFnS

Does there exist a ”semantically universal” class $\mathcal{C}^\star$ of ckts s.t.:

P1 : BFnS is poly-time for $\mathcal{C}^\star$

P2 : For every class $\mathcal{C}$ of ckts:

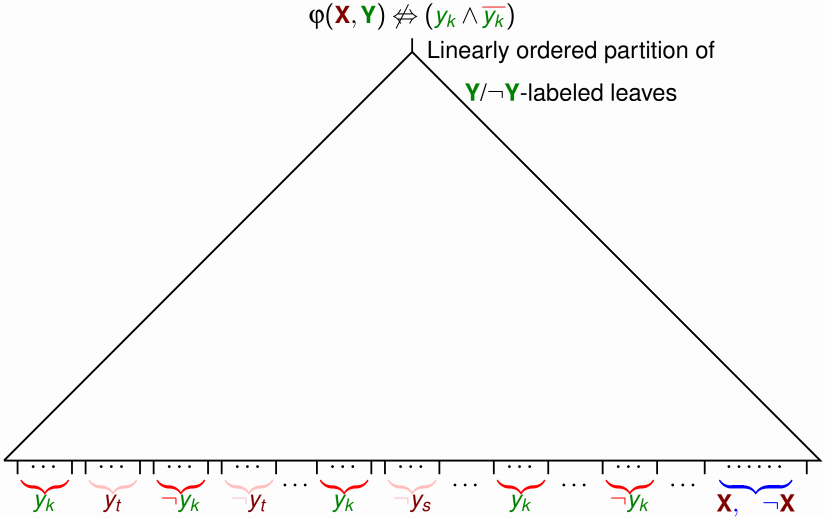
1. BFnS is poly-time for $\mathcal{C}$ iff $\mathcal{C}$ compiles to $\mathcal{C}^\star$ in poly-time.
2. BFnS is poly-size for $\mathcal{C}$ iff $\mathcal{C}$ compiles to poly-size ckts in $\mathcal{C}^\star$

Our Main Result

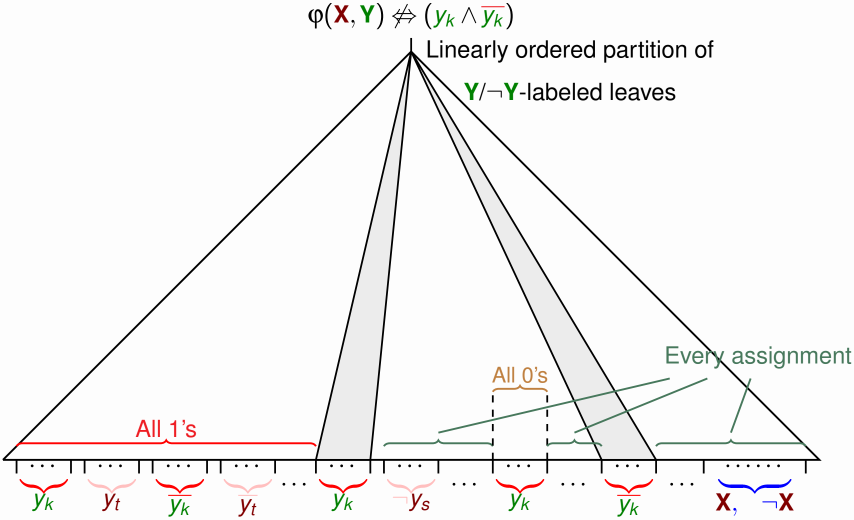
Yes, there exists such a class! Subset-And-Unrealizable Normal Form (SAUNF)

SAUNF: A Very Special Normal Form

Generalizing forbidden **semantics** of SynNNF



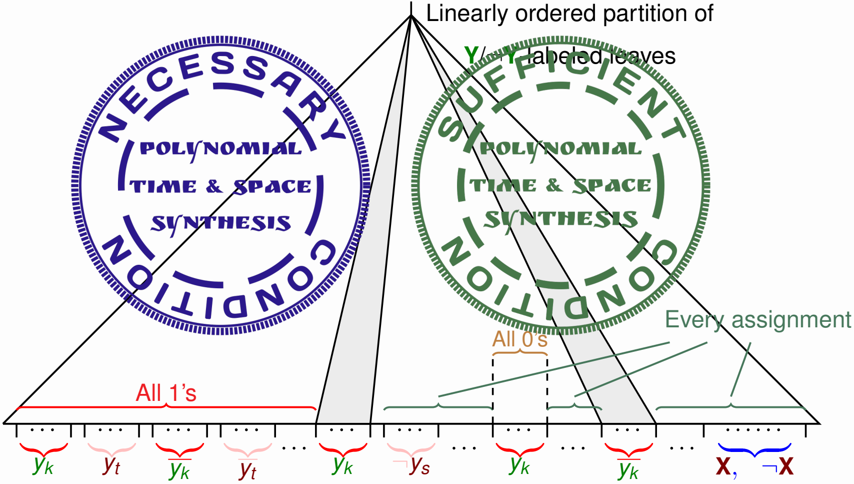
Generalizing forbidden **semantics** of SynNNF



SAUNF: A Very Special Normal Form

Generalizing forbidden **semantics** of SynNNF

$\varphi(\mathbf{X}, \mathbf{Y}) \not\Leftrightarrow (y_k \wedge \overline{y_k})$



SAUNF vs Existing Popular Normal Forms

Proposition

- Every SynNNF, wDNNF, DNNF circuit is also in SAUNF.
- Every FBDD, ROBDD can be compiled in linear time to SAUNF.

SAUNF vs Existing Popular Normal Forms

Proposition

- Every SynNNF, wDNNF, DNNF circuit is also in SAUNF.
- Every FBDD, ROBDD can be compiled in linear time to SAUNF.

Proposition

SAUNF is strictly weaker/more succinct than SynNNF, wDNNF, DNNF, FBDD, ROBDD

SAUNF vs Existing Popular Normal Forms

Proposition

- Every SynNNF, wDNNF, DNNF circuit is also in SAUNF.
- Every FBDD, ROBDD can be compiled in linear time to SAUNF.

Proposition

SAUNF is strictly weaker/more succinct than SynNNF, wDNNF, DNNF, FBDD, ROBDD

Proposition

SAUNF is exponentially more succinct than DNNF/dDNNF

SAUNF vs Existing Popular Normal Forms

Proposition

- Every SynNNF, wDNNF, DNNF circuit is also in SAUNF.
- Every FBDD, ROBDD can be compiled in linear time to SAUNF.

Proposition

SAUNF is strictly weaker/more succinct than SynNNF, wDNNF, DNNF, FBDD, ROBDD

Proposition

SAUNF is exponentially more succinct than DNNF/dDNNF, which are themselves exponentially more succinct than ROBDDs/FBDD.

Properties of SAUNF

Operations on SAUNF

Given $\varphi_1(\mathbf{X}, \mathbf{Y})$ and $\varphi_2(\mathbf{X}, \mathbf{Y})$ in SAUNF

- Computing $\varphi_1 \vee \varphi_2$ in SAUNF takes constant time

Properties of SAUNF

Operations on SAUNF

Given $\varphi_1(\mathbf{X}, \mathbf{Y})$ and $\varphi_2(\mathbf{X}, \mathbf{Y})$ in SAUNF

- Computing $\varphi_1 \vee \varphi_2$ in SAUNF takes constant time
- Computing $\varphi_1 \wedge \varphi_2$ in SAUNF
  - Takes constant time if every pair of **Y**-labeled leaves of φ_1 and φ_2 are consistent.

Properties of SAUNF

Operations on SAUNF

Given $\varphi_1(\mathbf{X}, \mathbf{Y})$ and $\varphi_2(\mathbf{X}, \mathbf{Y})$ in SAUNF

- Computing $\varphi_1 \vee \varphi_2$ in SAUNF takes constant time
- Computing $\varphi_1 \wedge \varphi_2$ in SAUNF
  - Takes constant time if every pair of **Y**-labeled leaves of φ_1 and φ_2 are consistent.
  - Otherwise,
    - Not possible in poly-time unless P = NP
    - Not possible in poly-size unless $\Sigma_2^p = \Pi_2^p$

Properties of SAUNF

Operations on SAUNF

Given $\varphi_1(\mathbf{X}, \mathbf{Y})$ and $\varphi_2(\mathbf{X}, \mathbf{Y})$ in SAUNF

- Computing $\varphi_1 \vee \varphi_2$ in SAUNF takes constant time
- Computing $\varphi_1 \wedge \varphi_2$ in SAUNF
  - Takes constant time if every pair of **Y**-labeled leaves of φ_1 and φ_2 are consistent.
  - Otherwise,
    - Not possible in poly-time unless P = NP
    - Not possible in poly-size unless $\Sigma_2^p = \Pi_2^p$
- Existentially quantifying $y_1, \ldots y_m$ takes linear time.

Properties of SAUNF

Operations on SAUNF

Given $\varphi_1(\mathbf{X}, \mathbf{Y})$ and $\varphi_2(\mathbf{X}, \mathbf{Y})$ in SAUNF

- Computing $\varphi_1 \vee \varphi_2$ in SAUNF takes constant time
- Computing $\varphi_1 \wedge \varphi_2$ in SAUNF
  - Takes constant time if every pair of **Y**-labeled leaves of φ_1 and φ_2 are consistent.
  - Otherwise,
    - Not possible in poly-time unless P = NP
    - Not possible in poly-size unless $\Sigma_2^p = \Pi_2^p$
- Existentially quantifying $y_1, \ldots y_m$ takes linear time.
  - Quantifying subset of **Y** not possible in linear time in general.

Properties of SAUNF

Operations on SAUNF

Given $\varphi_1(\mathbf{X}, \mathbf{Y})$ and $\varphi_2(\mathbf{X}, \mathbf{Y})$ in SAUNF

- Computing $\varphi_1 \vee \varphi_2$ in SAUNF takes constant time
- Computing $\varphi_1 \wedge \varphi_2$ in SAUNF
  - Takes constant time if every pair of **Y**-labeled leaves of φ_1 and φ_2 are consistent.
  - Otherwise,
    - Not possible in poly-time unless P = NP
    - Not possible in poly-size unless $\Sigma_2^p = \Pi_2^p$
- Existentially quantifying $y_1, \ldots y_m$ takes linear time.
  - Quantifying subset of **Y** not possible in linear time in general.

Checking if a given specification is in SAUNF

- Is Co-NP complete, given linearly ordered partition of **Y**-labeled leaves

Properties of SAUNF

Operations on SAUNF

Given $\varphi_1(\mathbf{X}, \mathbf{Y})$ and $\varphi_2(\mathbf{X}, \mathbf{Y})$ in SAUNF

- Computing $\varphi_1 \vee \varphi_2$ in SAUNF takes constant time
- Computing $\varphi_1 \wedge \varphi_2$ in SAUNF
  - Takes constant time if every pair of **Y**-labeled leaves of φ_1 and φ_2 are consistent.
  - Otherwise,
    - Not possible in poly-time unless P = NP
    - Not possible in poly-size unless $\Sigma_2^p = \Pi_2^p$
- Existentially quantifying $y_1, \ldots y_m$ takes linear time.
  - Quantifying subset of **Y** not possible in linear time in general.

Checking if a given specification is in SAUNF

- Is Co-NP complete, given linearly ordered partition of **Y**-labeled leaves
- Is Co-NP hard and in Σ_2^P, otherwise

Future work: Interesting Questions

- Closing the complexity gap for checking if a specification is in SAUNF.
- From Abstraction to Approximations in Verification?
- Beyond propositional synthesis: SMT
- Learning Theoretic Foundations for Functional Synthesis
  - What is the ideal distribution to generate the data?
  - Mistake bounds/complexity of learning functions from relations?
- The Future of Formal Methods (FM) +Machine Learning (ML)
  - The proposed solutions by ML do not need to be fully correct.
  - Use FM for correctness and ML to quickly find the solution.

Thanks !