Functional Synthesis: An Ideal Meeting Ground for Formal Methods and Machine Learning

Kuldeep S. Meel

Joint work with: Priyanka Golia and Subhajit Roy

1 National University of Singapore
2 Indian Institute of Technology Kanpur

Corresponding Papers: CAV-20, IJCAI-21, ICCAD-21
Synthesis

Holy Grail of Programming: *The user states the problem, the computer solves it* (Freuder, 1996)

Inputs $X$

$x_1$

$x_2$

$\ldots$

$x_n$

Specify $X$:

Outputs $Y$

$y_1$

$y_2$

$\ldots$

$y_m$
Holy Grail of Programming: *The user states the problem, the computer solves it* (Freuder, 1996)

**Specification: Relation** $\phi(X, Y)$

```cpp
int i = 0
while (i < n) {
    if ($x_i < x_{i+1}$) {
        $y_i = x_i$
    } else {
        $y_i = x_{i+1}$
    }
    i = i + 1
}
```

Inputs $X$  

$X_1$  
$X_2$  
...  
$X_n$  

Outputs $Y$  

$Y_1$  
$Y_2$  
...  
$Y_m$
**Synthesis**

*Holy Grail of Programming: The user states the problem, the computer solves it* (Freuder, 1996)

**Inputs X**

\[ x_1, x_2, ..., x_n \]

**Outputs Y**

\[ y_1, y_2, ..., y_m \]

**Specification: Relation \( \varphi(X, Y) \)**
Synthesis

Holy Grail of Programming: *The user states the problem, the computer solves it*  (Freuder, 1996)

Specification: Relation $\phi(X, Y)$

Inputs $X$

$x_1$ $\rightarrow$ $y_1$

$x_2$ $\rightarrow$ $y_2$

$\vdots$

$\vdots$

$x_n$ $\rightarrow$ $y_m$

Outputs $Y$

$y_1 := f_1(x_1, \ldots, x_n)$

$y_2 := f_2(x_1, \ldots, x_n)$

$\vdots$

$y_m := f_m(x_1, \ldots, x_n)$
Source of Specifications: Program Synthesis

\[ g_1(x_1, x_2) \geq x_1 \text{ and } \]
\[ g_1(x_1, x_2) \geq x_2 \text{ and } \]
\[ (g_1(x_1, x_2) == x_1 \text{ or } g_1(x_1, x_2) == x_2) \]

Synthesise a function \( g_1 \)
that satisfies the specification

Golia, Roy, and M. (IJCAI-21)
In order to synthesize a function $g_1$ that satisfies the specification, introduce variable $y_1$ and replace the call to $g_1(x_1, x_2)$ with $y_1$. Then, the function $g_1(x_1, x_2)$ must be defined such that:

- $g_1(x_1, x_2) \geq x_1$ and $g_1(x_1, x_2) \geq x_2$ and
- $(g_1(x_1, x_2) == x_1$ or $g_1(x_1, x_2) == x_2)$

This ensures that $y_1$ satisfies the same conditions as $x_1$ and $x_2$. Golia, Roy, and M. (IJCAI-21)
Given $\varphi(X, Y)$ over inputs $X = \{x_1, x_2, \ldots, x_n\}$ and outputs $Y = \{y_1, y_2, \ldots, y_m\}$.

Synthesize A function vector $F = \{f_1, f_2, \ldots, f_m\}$, such that $y_i := f_i(x_1, \ldots, x_n)$ such that:

$$\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$$

Each $f_i$ is called Skolem function and $F$ is called Skolem function vector.

Key Challenge: $\varphi(X, Y)$ is a relation
Non-uniqueness of Skolem Functions

Let $X = \{x_1, x_2\}$, $Y = \{y_1\}$ and $\varphi(X, Y) = x_1 \lor x_2 \lor y_1$

Possible Skolem function: $f(x_1, x_2) := \neg(x_1 \lor x_2)$
Non-uniqueness of Skolem Functions

Let $X = \{x_1, x_2\}$, $Y = \{y_1\}$ and $\varphi(X, Y) = x_1 \lor x_2 \lor y_1$

Possible Skolem function: $f(x_1, x_2) := \neg(x_1 \lor x_2)$

$$\varphi(X, F(X)) = x_1 \lor x_2 \lor (\neg(x_1 \lor x_2))$$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$\exists Y \varphi(X, Y)$</th>
<th>$\varphi(X, F(X))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 = 0, x_2 = 0$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
<tr>
<td>$x_1 = 0, x_2 = 1$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
<tr>
<td>$x_1 = 1, x_2 = 0$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
<tr>
<td>$x_1 = 1, x_2 = 1$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
</tbody>
</table>

$\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$
Non-uniqueness of Skolem Functions

Let $X = \{x_1, x_2\}$, $Y = \{y_1\}$ and $\varphi(X, Y) = x_1 \lor x_2 \lor y_1$

Possible Skolem function: $f(x_1, x_2) := \neg(x_1 \lor x_2)$

$$\varphi(X, F(X)) = x_1 \lor x_2 \lor (\neg(x_1 \lor x_2))$$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$\exists Y \varphi(X, Y)$</th>
<th>$\varphi(X, F(X))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 = 0, x_2 = 0$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
<tr>
<td>$x_1 = 0, x_2 = 1$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
<tr>
<td>$x_1 = 1, x_2 = 0$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
<tr>
<td>$x_1 = 1, x_2 = 1$</td>
<td>$y_1 = 1$</td>
<td>True</td>
</tr>
</tbody>
</table>

$\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$

Other possible Skolem functions: $f_1(x_1, x_2) = \neg x_1$  $f_2(x_1, x_2) = \neg x_2$  $f_3(x_1, x_2) = 1$
Functional synthesis is also

- *Church's Problem* (Circuit Synthesis) for Propositional Logic
- Program synthesis for propositional logic
  - No restrictions on the grammar
- *Strategy Synthesis* $\varphi(X, Y)$
  - $X$ player is trying to falsify $\varphi$ while $Y$ player is trying to satisfy $\varphi$
Diverse Approaches

- From the proof of validity of $\forall X \exists Y \varphi(X, Y)$
  - (Bendetti et al., 2005)
  - (Jussilla et al., 2007)
  - (Heule et al., 2014)

- Quantifier instantiation in SMT solvers
  - (Barrett et al., 2015)
  - (Bierre et al., 2017)

- Input-Output Separation
  - (Chakraborty et al., 2018)

- Knowledge representation
  - (Kukula et al., 2000)
  - (Trivedi et al., 2003)
  - (Jiang, 2009)
  - (Kuncak et al., 2010)
  - (Balabanov and Jiang, 2011)
  - (John et al., 2015)
  - (Fried, Tabajara, Vardi, 2016, 2017)
  - (Akshay et al., 2017, 2018)
  - (Chakraborty et al., 2019)

- Incremental determinization
  - (Rabe et al., 2015, 2018, 2019)
Diverse Approaches

- From the proof of validity of $\forall X \exists Y \varphi(X, Y)$
  (Bendetti et al., 2005)
  (Jussilla et al., 2007)
  (Heule et al., 2014)

- Quantifier instantiation in SMT solvers
  (Barrett et al., 2015)
  (Bierre et al., 2017)

- Input-Output Separation
  (Chakraborty et al., 2018)

- Knowledge representation
  (Kukula et al., 2000)
  (Trivedi et al., 2003)
  (Jiang, 2009)
  (Kuncak et al., 2010)
  (Balabanov and Jiang, 2011)
  (John et al., 2015)
  (Fried, Tabajara, Vardi, 2016, 2017)
  (Akshay et al., 2017, 2018)
  (Chakraborty et al., 2019)

- Incremental determinization
  (Rabe et al., 2015, 2018, 2019)

Scalability remains the holy grail
A Data-Driven Approach for Boolean Functional Synthesis

Machine Learning

Formal Methods

Constrained Sampling

Manthan
Machine learning excels at unlocking the creation of impressive early demos of new applications using very little development resources.

The part where it struggles is reaching the level of consistent usefulness and reliability required by production usage.
Machine learning excels at unlocking the creation of impressive early demos of new applications using very little development resources.

The part where it struggles is reaching the level of consistent usefulness and reliability required by production usage.

Formal Methods is the Answer to Machine Learning’s Struggles
Input $\phi(X, Y)$

Data Generation

Constrained Sampling
Data Generation

Learn Candidate Functions

Input $\phi(X, Y)$

Constrained Sampling

Machine Learning

Verifying

Repairing

Output $F$
Data Generation

Learn Candidate Functions

Verify

No → Repair

Yes

Output $F$

$\phi(X, Y)$

Constrained Sampling

Machine Learning

Formal Methods
Data Generation
Standing on the Shoulders of Constrained Samplers

\[ \varphi(x_1, x_2, y_1, y_2) \]

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Learn Candidate Functions
Taming the Curse of Abstractions via Learning with Errors

\[ p_1 := (\neg x_1 \land \neg x_2), \]
\[ p_2 := (x_1 \land \neg x_2) \]
\[ f_1 = \text{if } p_1 \text{ then 1} \]
\[ \text{elif } p_2 \text{ then 1} \]
\[ \text{else 0} \]

\[ p_1 := (\neg x_1 \land \neg y_1), \]
\[ p_2 := (x_1 \land y_1) \]
\[ f_1 = \text{if } p_1 \text{ then 1} \]
\[ \text{elif } p_2 \text{ then 1} \]
\[ \text{else 0} \]
Repair of Approximations
Reaping the Fruits of Formal Methods Revolution

\[ E(X, Y, F) \]

SAT, \( \sigma \)  

UNSAT  

Return \( F \)  

UNSAT Core  

Repair Cycle  

Repair Candidates  

\[ G_\sigma(X, Y) \]
Input $\phi(X, Y)$

Data Generation

Learn Candidate Functions

Verify

Yes

No

Repair

Output $F$

### Methods

- 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 $x_1, x_2$. 

\[ \varphi(x_1, x_2, y_1, y_2) : (x_1 \lor x_2 \lor y_1) \land (\neg x_1 \lor \neg x_2 \lor \neg y_2) \]
Data Generation

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

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

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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0/1</td>
<td>0</td>
</tr>
</tbody>
</table>
Data Generation

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0/1</td>
<td>0</td>
</tr>
</tbody>
</table>

Uniform Sampler

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Data Generation

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0/1</td>
<td>0</td>
</tr>
</tbody>
</table>

Uniform Sampler

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Possible Skolem functions:
  - \( f_1(x_1, x_2) = \neg(x_1 \lor x_2) \)
  - \( f_2(x_1, x_2) = \neg(x_1 \land x_2) \)
Data Generation

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1</th>
<th>0/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0/1</td>
<td>0</td>
</tr>
</tbody>
</table>

Uniform Sampler

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1</th>
<th>0/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Possible Skolem functions:
  - \( f_1(x_1, x_2) = \neg(x_1 \lor x_2) \)
  - \( f_1(x_1, x_2) = \neg x_1 \)
  - \( f_1(x_1, x_2) = \neg x_2 \)
  - \( f_1(x_1, x_2) = 1 \)
  - \( f_2(x_1, x_2) = \neg(x_1 \land x_2) \)
  - \( f_2(x_1, x_2) = \neg x_1 \)
  - \( f_2(x_1, x_2) = \neg x_2 \)
  - \( f_2(x_1, x_2) = 0 \)
Data Generation

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0/1</td>
<td>0/1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0/1</td>
<td>0</td>
</tr>
</tbody>
</table>

---

Magical Sampler

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

• Possible Skolem functions:
  - \( f_1(x_1, x_2) = \neg(x_1 \lor x_2) \)
  - \( f_1(x_1, x_2) = \neg x_1 \)
  - \( f_1(x_1, x_2) = \neg x_2 \)
  - \( f_1(x_1, x_2) = 1 \)
  - \( f_2(x_1, x_2) = \neg(x_1 \land x_2) \)
  - \( f_2(x_1, x_2) = \neg x_1 \)
  - \( f_2(x_1, x_2) = \neg x_2 \)
  - \( f_2(x_1, x_2) = 0 \)
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: $W(x_1) = 0.5$, $W(x_2) = 0.5$, $W(y_1) = 0.9$, $W(y_2) = 0.1$

  $$\sigma_1 = \{x_1 \mapsto 1, x_2 \mapsto 0, y_1 \mapsto 0, y_2 \mapsto 1\}$$

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

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

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
  
  (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 $\phi(X, Y)$

- Data Generation

- Learn Candidate Functions

- Verify
  - Yes: Output $F$
  - No: Repair

- Repair

21/39
Learn Candidate Function: Decision Tree Classifier

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

- 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 \( x_1, x_2 \)
Learning Candidate Functions

\[ p_1 := (\neg x_1 \land \neg x_2), \]
\[ p_2 := (x_1 \land \neg x_2) \]
\[ f_1 = \text{if } p_1 \text{ then } 1 \]
\[ \text{elif } p_2 \text{ then } 1 \]
\[ \text{else } 0 \]

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning Candidate Functions

\[
x_1 \quad x_2 \quad y_1 \quad y_2
\begin{array}{cccc}
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0
\end{array}
\]

\[p_1 := (\neg x_1 \land \neg x_2)\],
\[p_2 := (x_1 \land \neg x_2)\]

\[f_1 = \text{if } p_1 \text{ then } 1 \text{ elif } p_2 \text{ then } 1 \text{ else } 0\]

Can reorder \(p_1, p_2\)

Learning one level decision list
What Kind of Learning

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

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

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$p_1 := (\neg x_1 \land \neg x_2)$,
$p_2 := (x_1 \land \neg x_2)$

$f_1 = \text{if } p_1 \text{ then 1 }$
else if $p_2$ then 1
else 0
What Kind of Learning

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$p_1 := \neg x_1 \land \neg x_2$,  
$p_2 := x_1 \land \neg x_2$  
$f_1 = \text{if } p_1 \text{ then 1 }  
\text{elif } p_2 \text{ then 1 }  
\text{else 0}$

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 not Abstractions
Abstraction vs Approximation

\[
y_i = 1, \quad f_i(X) = 0
\]

\[
y_i = 0, \quad f_i(X) = 1
\]
Input $\phi(X, Y)$

Data Generation ✓

Learn Candidate Functions ✓

Verify

Yes ➔ Output $F$

No ➔ Repair
Verification of Candidate Functions

\[
E(X, Y, Y') := \varphi(X, Y) \land \neg\varphi(X, Y') \land (Y' \leftrightarrow F(X))
\]

(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

\[ E(X, Y, Y') := \varphi(X, Y) \land \neg\varphi(X, Y') \land (Y' \leftrightarrow F(X)) \]

\[ \sigma \models E(X, Y, Y') \text{ be a counterexample to fix.} \]

- 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 \( \sigma[y_i] \neq \sigma[y'_i] \).
$E(X, Y, Y') := \varphi(X, Y) \land \neg \varphi(X, Y') \land (Y' \leftrightarrow F(X))$

$\sigma \models E(X, Y, Y')$ 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 $f_i$. 
Repair Candidate Identification

\[
E(X, Y, Y') := \varphi(X, Y) \land \neg \varphi(X, Y') \land (Y' \leftrightarrow F(X))
\]

\[\sigma \models E(X, Y, Y')\] 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 \(\varphi(X, Y) \land (X \leftrightarrow \sigma[X])\).
  - Soft Clauses \((Y \leftrightarrow \sigma[Y'])\).

- Candidates to repair: \(Y\) variables in the violated soft clauses
• $\sigma = \{x_1 \mapsto 1, x_2 \mapsto 1, y_1 \mapsto 0, y_2 \mapsto 1, y'_1 \mapsto 0, y'_2 \mapsto 0\}$, and we want to repair $f_2$.

• Potential Repair: If $x_1 \land x_2 \land \neg y_1$ then $y_2 = 1$
  $\beta = \{x_1, x_2, \neg y_1\}$

• Would be nice to have $\beta = \{x_1, x_2\}$ or even $\beta = \{x_1\}$

• Challenge: How do we find small $\beta$?
  - $G_\sigma(X, Y) := \varphi(X, Y) \land x_1 \land x_2 \land \neg y_1 \land (y_2 = 0)$
  - $\beta := \text{Literals in UNSAT Core of } G_\sigma(X, Y)$
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 $p_1, p_2$
Candidaies 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, else $p_2$ then 1, else 0.
- $p_1, p_2$ can be reordered.

Suppose in repair iterations, we have learned: If $\beta_1$ then 1, $\ldots$ $\beta_2$ then 0 $\ldots$ ...

$\beta_1$ and $\beta_2$ can be reordered.

From one-level decision list to two-level decision list.
$\varphi(X, Y)$

$X = \{x_1, x_2\}$

$Y = \{y_1, y_2\}$

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Check Satisfiability of $E(X, Y, \mathcal{Y}^\prime)$

$G_\sigma(X, Y)$

Verify Candidates

SAT, $\sigma$

Data Generation

Learn Candidates

UNSAT Core-based Repair

Return $F$

31/39
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

<table>
<thead>
<tr>
<th></th>
<th>C2Syn</th>
<th>BFSS</th>
<th>CADET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benchmarks</td>
<td>206</td>
<td>247</td>
<td>280</td>
</tr>
</tbody>
</table>
Experimental Evaluations

An increase of 229 benchmarks.
Impact of Choices (I): Data Generation

- QuickSampler: 332
- CryptoMiniSAT: 399
- CMSGen: 509
### Impact of Choices (II): Use of MaxSAT

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manthan</td>
<td>0</td>
</tr>
<tr>
<td>Manthan$\text{no-maxsat}$</td>
<td>396</td>
</tr>
<tr>
<td>Manthan</td>
<td>509</td>
</tr>
</tbody>
</table>

![Graph showing runtime comparison between Manthan and Manthan$\text{no-maxsat}$](image-url)
Impact of Choices (III): Abstraction vs Approximation

<table>
<thead>
<tr>
<th>Manthan_{abstraction}</th>
<th>Manthan_{no-maxsat}</th>
<th>Manthan</th>
</tr>
</thead>
<tbody>
<tr>
<td>171</td>
<td>396</td>
<td>509</td>
</tr>
</tbody>
</table>
609 bit-vector instances from SyGuS competition

<table>
<thead>
<tr>
<th>Syntax-Guided Solvers</th>
<th>DryadSynth</th>
<th>Stochpp</th>
<th>Symbolic</th>
<th>ESolver</th>
<th>EUSolver</th>
<th>CVC4</th>
<th>Manthan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>39</td>
<td>108</td>
<td>151</td>
<td>236</td>
<td>488</td>
<td>592</td>
</tr>
</tbody>
</table>
Future work: Interesting Questions

- From Abstraction to Approximations in Verification?
- Beyond proposition 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.
Conclusion

Manthan: A Data-Driven Approach for Boolean Functional Synthesis.

- Constrained Sampling
- Decision List Classifier
- Formal Methods

Solves 509 benchmarks — state of the art could solve 280

https://github.com/meelgroup/manthan

Thanks!
• Let $X = \{x_1, x_2\}$, and $Y = \{y_1, y_2\}$

• $\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1)))$

• Skolem Functions:
  
  $f_1(x_1, x_2) := (x_1 \lor x_2)$
  
  $f_2(x_1, x_2, y_1) := (x_1 \land (x_2 \lor y_1))$

  $f_2(x_1, x_2, y_1) := (x_1 \land (x_2 \lor (x_1 \lor x_2)))$

  $f_2(x_1, x_2, y_1) := x_1$

  $\exists Y \varphi(X, Y) \equiv \varphi(X, F(X))$
Example: Data Generation

Let $X = \{x_1, x_2\}$, and $Y = \{y_1, y_2\}$

$$
\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1)))
$$
Example: Learning Candidate Functions

\[ \varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1))) \]

- Learn candidate function \( f_1 \).
- Feature set for \( y_1 \) := \( \{x_1, x_2\} \)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( y_1 )</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

\[ f_1(x_1, x_2) := x_2 \]
Example: Learning Candidate Functions

\[ \varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1))) \]

- Learn candidate function \( f_2 \).
- Feature set for \( y_2 := \{x_1, x_2, y_1\} \)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ f_2(x_1, x_2, y_1) := x_1 \]
Example: Verification of Candidate Functions

\[ \varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1))) \]

- \( E(X, Y, Y') := \varphi(X, Y) \land \neg \varphi(X, Y') \land (Y' \leftrightarrow F(X)) \)

\[ E(X, Y, Y') := \varphi(x_1, x_2, y_1, y_2) \land \neg \varphi(x_1, x_2, y'_1, y'_2) \land (y'_1 \leftrightarrow x_2) \land (y'_2 \leftrightarrow x_1) \]

\[ \text{SAT} \]

\[ \sigma \models E(X, Y, Y') \quad \Rightarrow \quad \sigma[x_1] = 1, \sigma[x_2] = 0 \]

\[ \sigma[y_1] = 1, \sigma[y_2] = 1 \quad \text{and} \quad \sigma[y'_1] = 0, \sigma[y'_2] = 1 \]
Example: Verification of Candidate Functions

\[
\varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1)))
\]

- \(E(X, Y, Y') := \varphi(X, Y) \land \neg \varphi(X, Y') \land (Y' \leftrightarrow F(X))\)

\[
E(X, Y, Y') := \varphi(x_1, x_2, y_1, y_2) \land \neg \varphi(x_1, x_2, y_1', y_2') \land (y_1' \leftrightarrow x_2) \land (y_2' \leftrightarrow x_1)
\]

\[\sigma[y_1] \neq \sigma[y_1']\]
Candidate to repair \(f_1\)

\[\sigma \models E(X, Y, Y') \rightarrow \sigma[x_1] = 1, \sigma[x_2] = 0\]

\[\sigma[y_1 = 1], \sigma[y_2] = 1 \quad \sigma[y_1' = 0], \sigma[y_2'] = 1\]
Example: Repairing candidate functions (I)

\[ \varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1))) \]

- \( G_1(X, Y) = \varphi(X, Y) \land (X \leftrightarrow \sigma[X]) \land (y_1 \leftrightarrow \sigma[y_1']) \).
- \( G_1(X, Y) = \varphi(X, Y) \land (x_1 \leftrightarrow 1) \land (x_2 \leftrightarrow 0) \land (y_1 \leftrightarrow 0) \).
- UNSAT core of \( G_1(X, Y) = \varphi(X, Y) \land (x_1 \leftrightarrow 1) \land (y_1 \leftrightarrow 0) \)
- Repair formula \( \beta = x_1 \).
Example: Repairing candidate functions (II)

\[ \varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1))) \]

<table>
<thead>
<tr>
<th>Before repair</th>
<th>Repair</th>
<th>After repair</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1(\sigma[X]) \mapsto 0 )</td>
<td>( f_1(X) \leftarrow f_1(X) \lor \beta )</td>
<td>( f_1(X) \mapsto 1 )</td>
</tr>
<tr>
<td>( f_1(X) \leftarrow x_2 \lor x_1 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Verification of Candidate Functions

\[ \varphi(X, Y) := (y_1 \leftrightarrow (x_1 \lor x_2)) \land (y_2 \leftrightarrow (x_1 \land (x_2 \lor y_1))) \]

- \[ E(X, Y, Y') := \varphi(X, Y) \land \neg \varphi(X, Y') \land (Y' \leftrightarrow F(X)) \]

\[ E(X, Y, Y') := \varphi(x_1, x_2, y_1, y_2) \land \neg \varphi(x_1, x_2, y'_1, y'_2) \land (y'_1 \leftrightarrow x_2 \lor x_1) \land (y'_2 \leftrightarrow x_1) \]

\[ \text{UNSAT} \]

\[ \text{Manthan returns } F \]
• $\Sigma_1 := \text{Sample 500 data point with } W(x_i) = 0.5 \text{ and } W(y_i) = 0.9.$

$$w_1(i) = \frac{\text{Count}(\Sigma_1 \cap (y_i = 1))}{500}$$

• $\Sigma_2 := \text{Sample 500 data point with } W(x_i) = 0.5 \text{ and } W(y_i) = 0.1.$

$$w_2(i) = \frac{\text{Count}(\Sigma_2 \cap (y_i = 0))}{500}$$

• If $0.35 < w_1(i) < 0.65 \text{ and } 0.35 < w_2(i) < 0.65,$ then $q_i = w_1(i) \text{, else } q_i = 0.9.$