Scalable Exploration of Functional Dependency by Interpolation and Incremental SAT Solving

Chih-Chun Lee§, J.-H. Roland Jiang§, C.-Y. Ric Huang§, and Alan Mishchenko¶

§National Taiwan University
¶University of California, Berkeley
Outline

- Introduction
- Prior work
- Our approach
- Experimental results
- Conclusions
Introduction

- Functional dependency
  - \( f(x) = h(g_1(x), g_2(x), \ldots, g_m(x)) = h(G(x)) \)

  Under what condition can a function \( f \) be expressed as some function \( h \) over a set of functions \( G \)?

  - \( h \) exists \( \iff \not\exists a, b \) such that \( f(a) \neq f(b) \) and \( G(a) = G(b) \)

    i.e., \( G \) is more distinguishing than \( f \)
Applications

- Resynthesis/rewiring [KK04]
- Redundant register removal [LN91,STB96]
- BDD minimization [HD93]
- Verification reduction [JB04]
- ...

Boolean Network

- target function
- base functions
Prior Work

- BDD-based computation of $h$

\[
h^{on} = \{ y \in B^m : y = G(x) \text{ and } f(x) = 1, \ x \in B^n \}
\]
\[
h^{off} = \{ y \in B^m : y = G(x) \text{ and } f(x) = 0, \ x \in B^n \}
\]
BDD-based Computation

- Not robust
  - 2 image computations for every choice of $G$
  - Inefficient when $|G|$ is large or when there are many choices of $G$

- Move on to SAT?
SAT-based Computation

- h exists ⇔

  \[ \exists a, b \text{ such that } f(a) \neq f(b) \text{ and } G(a) = G(b), \]
  
i.e., (f(x) \neq f(x^*)) \land (G(x) \equiv G(x^*)) \text{ is UNSAT}

- But how to derive h?
- Moreover how to select G?
Craig Interpolation

- $A \land B$ is UNSAT for clause sets $A$ and $B$.
  Then there exists an interpolant $I$ of $A$ such that
  1. $A \Rightarrow I$
  2. $I \land B$ is UNSAT
  3. $I$ refers only to the common variables of $A$ and $B$

- Now what?
  Connecting $I$ with $h$

$I$ is an abstraction of $A$
The Magic

\[ (f(x) \neq f(x^*)) \land (G(x) \equiv G(x^*)) \text{ is } \text{UNSAT} \]
Premises

- Interpolant is of size linear to the refutation size
  - Resolution refutations of UNSAT CNF formulas can be generated from DPLL-style SAT solvers, and so can interpolants

- Circuit-to-CNF translation is doable in linear time
  - Intermediate variables are introduced
It Works!

- Clause set $A$: $C_{DFNon}, y_0$
- Clause set $B$: $C_{DFNoff}, \neg y_0^*, (y_i = y_i^*)$ for $i = 1, \ldots, m$
- $I$ is an overapproximation of $\text{Img}(f_{on})$ and is disjoint from $\text{Img}(f_{off})$
- $I$ only refers to $y_1, \ldots, y_m$
- Therefore, $I$ corresponds to a feasible implementation of $h$
Incremental SAT Solving

- Controlled equality constraints
  \[(y_i \equiv y_i^*) \rightarrow (\neg y_i \lor y_i^* \lor \alpha_i)(y_i \lor \neg y_i^* \lor \alpha_i)\]
  with auxiliary variables \(\alpha_i\)

\[\alpha_i = \text{true} \Rightarrow i^{th} \text{ equality constraint is disabled}\]

- Fast switch between target and base functions by unit assumptions over control variables
- Fast enumeration of different base functions
- Share learned clauses
## SAT vs. BDD

<table>
<thead>
<tr>
<th>SAT</th>
<th>BDD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td><strong>Cons</strong></td>
</tr>
<tr>
<td>- Detect multiple choices of $G$ automatically</td>
<td>- Detect one choice of $G$ at a time</td>
</tr>
<tr>
<td>- Scalable to large $</td>
<td>G</td>
</tr>
<tr>
<td>- Fast enumeration of different target functions $f$</td>
<td>- Slow enumeration of different target functions $f$</td>
</tr>
<tr>
<td>- Fast enumeration of different base functions $G$</td>
<td>- Slow enumeration of different base functions $G$</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td><strong>Pros</strong></td>
</tr>
<tr>
<td>- Single feasible implementation of $h$</td>
<td>- All possible implementations of $h$</td>
</tr>
</tbody>
</table>
Experimental Setup

- Implemented in ABC using MiniSAT
- Detect functional dependency among transition functions of sequential circuits
Experimental Results

### SAT vs. BDD

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#Nodes</th>
<th>Original</th>
<th>Retimed</th>
<th>SAT (original)</th>
<th>BDD (original)</th>
<th>SAT (retimed)</th>
<th>BDD (retimed)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#FF.</td>
<td>#Dep-S</td>
<td>#Dep-B</td>
<td>Time</td>
<td>Mem</td>
<td>Time</td>
</tr>
<tr>
<td>s5378</td>
<td>2794</td>
<td>179</td>
<td>52</td>
<td>25</td>
<td>398</td>
<td>283</td>
<td>173</td>
</tr>
<tr>
<td>s9234.1</td>
<td>5597</td>
<td>211</td>
<td>46</td>
<td>x</td>
<td>459</td>
<td>301</td>
<td>201</td>
</tr>
<tr>
<td>s13207.1</td>
<td>8022</td>
<td>638</td>
<td>190</td>
<td>136</td>
<td>1930</td>
<td>802</td>
<td>x</td>
</tr>
<tr>
<td>s15850.1</td>
<td>9785</td>
<td>534</td>
<td>18</td>
<td>9</td>
<td>907</td>
<td>402</td>
<td>x</td>
</tr>
<tr>
<td>s35932</td>
<td>16065</td>
<td>1728</td>
<td>0</td>
<td>--</td>
<td>2026</td>
<td>1170</td>
<td>--</td>
</tr>
<tr>
<td>s38417</td>
<td>22397</td>
<td>1636</td>
<td>95</td>
<td>--</td>
<td>5016</td>
<td>243</td>
<td>--</td>
</tr>
<tr>
<td>s38584</td>
<td>19407</td>
<td>1452</td>
<td>24</td>
<td>--</td>
<td>4350</td>
<td>2569</td>
<td>--</td>
</tr>
<tr>
<td>b12</td>
<td>946</td>
<td>121</td>
<td>4</td>
<td>2</td>
<td>170</td>
<td>66</td>
<td>33</td>
</tr>
<tr>
<td>b14</td>
<td>9847</td>
<td>245</td>
<td>2</td>
<td>--</td>
<td>245</td>
<td>2</td>
<td>--</td>
</tr>
<tr>
<td>b15</td>
<td>8367</td>
<td>449</td>
<td>0</td>
<td>--</td>
<td>1134</td>
<td>793</td>
<td>--</td>
</tr>
<tr>
<td>b17</td>
<td>30777</td>
<td>1415</td>
<td>0</td>
<td>--</td>
<td>3967</td>
<td>2350</td>
<td>--</td>
</tr>
<tr>
<td>b18</td>
<td>111241</td>
<td>3320</td>
<td>5</td>
<td>--</td>
<td>9254</td>
<td>5723</td>
<td>--</td>
</tr>
<tr>
<td>b19</td>
<td>224624</td>
<td>6642</td>
<td>0</td>
<td>--</td>
<td>7164</td>
<td>337</td>
<td>--</td>
</tr>
<tr>
<td>b20</td>
<td>19682</td>
<td>490</td>
<td>4</td>
<td>--</td>
<td>1604</td>
<td>1167</td>
<td>--</td>
</tr>
<tr>
<td>b21</td>
<td>20027</td>
<td>490</td>
<td>4</td>
<td>--</td>
<td>1950</td>
<td>1434</td>
<td>--</td>
</tr>
<tr>
<td>b22</td>
<td>29162</td>
<td>735</td>
<td>6</td>
<td>--</td>
<td>3013</td>
<td>2217</td>
<td>--</td>
</tr>
</tbody>
</table>
Experimental Results (cont’d)

circuit size vs. runtime

R² = 0.9664

R² = 0.909

Number of nodes (log)

Time (log)

Original
Retimed

Experimental Results (cont’d)

Incremental SAT

![Graph showing time versus iteration for different node counts: b19 (200k nodes), b18 (100k nodes), b17 (30k nodes), and b15 (10k nodes).]
Experimental Results (cont’d)

#total input vs. #redundant inputs

- Number of input variables: 0, 50, 100, 150, 200
- Number of spurious variables: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
- Total input values: 16858
- Redundant input values: 174, 68, 14, 4, 2, 9, 1, 6, 5, 2, 2, 0, 1, 2

11/6/2007 ICCAD 2007 18
Experimental Results (cont’d)

interpolant size vs. support size

R² = 0.861
R² = 0.8506

Original
Retimed

Number of variables (log)

Interpolant size (log)
Conclusions and Future Work

- Have shown
  - Functional dependency is computable with pure SAT-solving and is scalable to large designs

- To try
  - Don’t-care generation for \( h \)
  - Other applications of Craig interpolation
  - Hybrid approach combining BDD and SAT
Thank You