Data Mining Based Decomposition for Assume Guarantee Reasoning

He Zhu, Tsinghua University
Fei He, Tsinghua University
William N. N. Hung, Synopsys Inc.
Xiaoyu Song, Portland State University
Ming Gu, Tsinghua University

Presented by William N. N. Hung
Outline

• Introduction
• Data Mining based Decomposition
• Experimental Results
• Conclusion
Compositional Verification

- Model Checking ...... state space explosion
- Divide and conquer
- Decompose properties of system \((M_1 \parallel M_2)\) in properties of its components
- Does \(M_1\) satisfy \(P\)?
  - typically a component is designed to satisfy its requirements in specific contexts / environments
- Assume-guarantee reasoning: introduces assumption \(A\) representing \(M_1\)’s “context”
- Simplest assume-guarantee rule

$$
\begin{align*}
1. & \quad \langle A \rangle M_1 \quad \langle P \rangle \\
2. & \quad \langle true \rangle M_2 \quad \langle A \rangle \\
\hline
\langle true \rangle M_1 \parallel M_2 \quad \langle P \rangle
\end{align*}
$$
Automatic Assume-Guarantee Reasoning

- 2 key steps in assume-guarantee based verification
  - Identifying an appropriate decomposition of the system,
  - Identifying simple assumptions.
- Our Goal
  - automatically decompose a system into several modules?
  - The resulting model should be convenient for assume-guarantee reasoning
    - Minimizing interactions between modules
    - It can benefit the assumption learning.
Related Works

- **Learning Assumptions for Compositional Verification** (Cobleigh et al., 2003).
  - Given a set of decomposed modules
  - Use L* algorithm to learn assumption automatically.

- **Learning-based Symbolic Assume-guarantee Reasoning with Automatic Decomposition** (Nam and Alur, 2005-2006)
  - The first paper on system decomposition for AG
  - Use hypergraph partitioning to decompose the system
Outline

• Introduction
• Data Mining based Decomposition
• Experimental Results
• Conclusion
Motivating Example

- Consider a simple example.

\[
\begin{align*}
\text{VAR } g, a, b, p, c; \\
\text{Next}(g) &:= a \& b; \\
\text{Next}(p) &:= g \mid c \\
\text{Next}(c) &:= \neg p
\end{align*}
\]

- \( g \) is dependent on \( a \) and \( b \).
Decomposition Strategy

• Target:
  • Reduce the shared variables as much as possible,
  • such that assumptions are based on a small language alphabet.

• Appropriate Decomposition:
  • Enhance inner-cohesion (within a partition)
  • Minimize inter-connection (between partitions)

• Heuristic:
  • Try to put the dependent variables together.
How to minimize inter-connection?

- Construct Weighted Hypergraph:
  - Using data mining

- Weighted Hypergraph:
  - The edge connect arbitrary vertices.
  - The edge is assigned a numerical value.

- Weighted Hypergraph partitioning:
  - Partitioning the hypergraph into $K$ parts.
  - The sum of weight of all edges connecting different parts is minimal.
How to enhance inner-cohesion?

- Using a data mining algorithm: Association rule mining.
- **Association rule mining** discovers item implications through a large data set.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>g</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_g$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_p$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_c$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- An association rule $X \Rightarrow Y$, means if $X$ occurs in a transaction, then $Y$ should occur too.
Association Rule Mining

• Two steps for using association rule mining
  – Find frequent itemsets with minimum support;
  – Generate association rules from these itemsets with minimum confidence.

• Some important concepts
  – The support of an itemset $X$: the number of records that satisfy $X$ divided by the number of records.
  – The confidence of a rule $X \Rightarrow Y$ : the number of records that satisfy $X \cup Y$ divided by the number of records that satisfy $X$. 
- Find frequent itemsets $E_{fi}$.
- Generate rules from frequent itemset.

### Frequent item sets

<table>
<thead>
<tr>
<th>V_T:</th>
<th>t_g: g a b</th>
<th>t_p: p g c</th>
<th>t_c: c p</th>
</tr>
</thead>
</table>

### Association rules

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a $\Rightarrow$ b</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>b $\Rightarrow$ a</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>b g $\Rightarrow$ a</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>g $\Rightarrow$ a</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>g $\Rightarrow$ b</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>g $\Rightarrow$ c</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>c $\Rightarrow$ g</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>p $\Rightarrow$ c</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>p $\Rightarrow$ g</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>... ...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Construct Weighted Hypergraph

• Create a hyperedge from each frequent itemset
  - Variables are the vertices
  - Hyperedge connects the variables
  - Each itemset gives a possible combination for the items.

• Weight of a hyperedge is decided by the average value of all rules derived from the corresponding itemset.
  - For example, the weight of edge \((p, g, c)\) is decided by three rules: \(p \rightarrow g \rightarrow c\), \(p \rightarrow c \rightarrow g\), and \(g \rightarrow c \rightarrow p\).

This value gives an evaluation for the interactions between items.
VAR g, a, b, p, c;
Next(g) := a & b;
Next(p) := g | c
Next(c) := !p

### Hyperedges:
- a b 100
- a b g 100
- a g 75
- b g 75
- p c 100
- p c g 50
- p g 50
- c g 50

### Weighted Hypergraph Model
Decomposition as Hypergraph Partitioning

- **Hypergraph partitioning:**
  - Partitioning the hypergraph into $K$ parts.
  - Minimize sum weights of all cut-edges

- There are some existing tools for hypergraph partitioning problem, among them, we chose hMETIS.
Hyperedges:

- a b 100
- a b g 100
- a g 75
- b g 75
- p c 100
- p c g 83.3
- p g 50
- c g 50
Decomposing the variable set into 2 partitions:

- $a$, $b$, $g$ and $p$, $c$. 
System Decomposition
• With the variable partition result

```
VAR g, a, b, p, c;
Next(g) := a & b;
Next(p) := g | c
Next(c) := !p
```

VAR p, c;
Next(p) := g | c
Next(c) := !p

VAR g, a, b;
Next(g) := a & b;
The Flow of our Approach

1. State transition system
2. Variable dependencies
3. Weighted hypergraph model
   - Weights mining
   - Partitioning into $n$ parts
4. Variable partition 1
5. Variable partition 2
6. ... Variable partition $n$
7. Decomposed sub-modules
Benefits of Our Approach

- Modules are compact and have fewer communication.
- Each module has less requirements on its environment \(\Rightarrow\) simplify assumption

\[
\begin{align*}
1. \quad & \langle A \rangle \quad M_1 \quad \langle P \rangle \\
2. \quad & \langle \text{true} \rangle \quad M_2 \quad \langle A \rangle \\
\hline
& \langle \text{true} \rangle \quad M_1 \parallel M_2 \quad \langle P \rangle
\end{align*}
\]

- Since \(A\) is reduced, the efforts for verifying these two premises are also reduced.
Outline

• Introduction
• Data Mining based Decomposition
• Experimental Results
• Conclusion
Implementation

NuSMV parser → Weighted hypergraph → Apriori → hMETIS → Symoda

Decomposition

Decomposed modules → Compositional Verification
## Experimental Results

<table>
<thead>
<tr>
<th>Benchs</th>
<th>Var</th>
<th>Weighted Hypergraph</th>
<th>Unweighted Hypergraph</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IO</td>
<td>time</td>
<td>IO</td>
<td>time</td>
</tr>
<tr>
<td>s1a</td>
<td>23</td>
<td>2</td>
<td>0.32</td>
<td>2</td>
</tr>
<tr>
<td>s1b</td>
<td>25</td>
<td>6</td>
<td>0.49</td>
<td>6</td>
</tr>
<tr>
<td>msi3</td>
<td>61</td>
<td>17</td>
<td>2.81</td>
<td>19</td>
</tr>
<tr>
<td>msi5</td>
<td>97</td>
<td>24</td>
<td>5.86</td>
<td>32</td>
</tr>
<tr>
<td>msi6</td>
<td>121</td>
<td>27</td>
<td>9.69</td>
<td>33</td>
</tr>
<tr>
<td>syncarb10</td>
<td>74</td>
<td>32</td>
<td>76.13</td>
<td>33</td>
</tr>
<tr>
<td>peterson</td>
<td>9</td>
<td>7</td>
<td>0.65</td>
<td>7</td>
</tr>
<tr>
<td>guidance</td>
<td>76</td>
<td>37</td>
<td>19.93</td>
<td>13</td>
</tr>
</tbody>
</table>

- Most of our experiments leads to good result.
- Negative result in *guidance*,
  - The variables dependencies in *guidance* are so sparse
Outline

• Introduction
• Data Mining based Decomposition
• Experimental Results
• Conclusion
New decomposition method for assume-guarantee
  - Integrates data mining to the compositional verification.
  - Using weighted hypergraph partitioning to cluster variables.

Automatic decomposition approach
  - Inner cohesion improved
  - Inter connection reduced

Experimental results show promise

Future work include:
  - Circular assume-guarantee rules.
  - Applying assorted classification methods in data mining to find even better decomposition.
Thank You!

Question & Answer