Data Mining Based Decomposition for Assume Guarantee Reasoning

Fei He (贺飞)
joint work with He Zhu, William N.N Hung, Xiaoyu Song and Ming Gu

School of Software
Tsinghua University
Outline

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

• Model checking has been successfully applied to many areas.

• It suffers from the state space explosion problem,
  – Larger state systems,
  – Infinite state systems.

• To tackle the state space explosion problem, one of the popular techniques is to use the divide and conquer strategy, i.e. compositional reasoning.
Compositional Verification

- 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

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

- There are two key steps in an assume-guarantee based verification strategy,
  - Identifying an appropriate decomposition of the system,
  - Identifying a simple assumption.
Our Goal

• How to automatically decompose a system into several modules?

• The resulting model should be convenient for assume-guarantee reasoning, i.e.
  – The interactions between modules should be as little as possible,
  – Because this can do benefit to the assumption learning.
Related Works

• Learning Assumptions for Compositional Verification, (Cobleigh etc, 2003).
  – 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 solver this problem.
Outline

• Introduction
• Data Mining based Decomposition
• Experimental Results
• Conclusion
State Transition System

- Component modeled as a state transition system $M_i = <X_i, I_i, O_i, Init_i, T_i>$, where
  - $X_i$ is a set of state variables controlled by $M_i$;
  - $I_i$ is a set of input variables that are controlled by other modules and are readable by $M_i$;
  - $O_i$ is a set of output variables that are accessible to other modules;
  - $Init_i$ gives the initial condition;
  - $T_i$ defines the transition condition.
Decomposition

• Given a system $M = <X, NULL, NULL, Init, T>$ and an integer $n$, the decomposition problem is to decompose $M$ into $n$ sub-modules $M_i = <X_i, I_i, O_i, Init_i, T_i>$, such that $X = \bigcup_{1 \leq i \leq n} X_i$, and $X_i \cap X_j = \emptyset$ for $i \neq j$.

• The key for decomposition of a system is to partition the system variables.
Motivating Example

• Consider a simple circuit.

\[
\begin{align*}
X: & \quad a, b, g, p, c \\
\text{Init:} & \quad \text{NULL} \\
T: & \quad \\
& \quad \begin{align*}
&t_g: g & a & b \\
&t_p: p & g & c \\
&t_c: c & p
\end{align*}
\end{align*}
\]

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

• Target:
  – Reduce the *shared* variables as great as possible,
  – Such that the assumption is based on a small language alphabet.

• Heuristic:
  – Try to put the *dependent* variables together.

• Our approach: *Hypergraph partitioning*. 
Hypergraph

- A **hypergraph** is a special graph, which can be defined as $G(V,E,W)$, where
  - $V$ is a set of *vertices*.
  - $E$ is a set of *hyperedges* that connect arbitrary number of vertices.
  - $W : E \rightarrow R$ defines a *weight* value for each hyperedge.
Modeling in Hypergraph

• Given a system $M = <X, \text{Init}, T>$,
  – each variable $x \in X$ corresponds to a vertex $v_x$,
  – and the dependent variables refer to a hyperedge in the hypergraph.

It’s not easy to assign weights to the hyperedges.
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 rules from these itemset 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 \( X \).
Frequent item sets

\(V_T:\)

\(t_g: \{g, a, b\}\)
\(t_p: \{p, g, c\}\)
\(t_c: \{c, p\}\)

\(a\ b\)
\(a\ b\ g\)
\(a\ g\)
\(b\ g\)
\(p\ c\)
\(p\ g\)
\(p\ g\ c\)
\(c\ g\)

Association rules

\(a \Rightarrow b \quad 100\)
\(b \Rightarrow a \quad 100\)
\(b\ g \Rightarrow a \quad 100\)
\(g \Rightarrow a \quad 50\)
\(g \Rightarrow b \quad 50\)
\(c \Rightarrow g \quad 50\)
\(p \Rightarrow c \quad 100\)
\(p \Rightarrow g \quad 50\)
\(... ...\)
How to Assign Weights?

• With each itemset, we define a hyperedge
  – 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: \(p \land g \Rightarrow c\), \(p \land c \Rightarrow g\), and \(g \land c \Rightarrow p\). This value gives an evaluation for the interactions between items.
Weighted Hypergraph Model

Hyperedges:
- \(a \ b\) \(100\)
- \(a \ b \ g\) \(100\)
- \(a \ g\) \(75\)
- \(b \ g\) \(75\)
- \(p \ c\) \(100\)
- \(p \ c \ g\) \(83.3\)
- \(p \ g\) \(50\)
- \(c \ g\) \(50\)

Frequent item set:
- \(g \ a \ b\)
- \(p \ g \ c\)
- \(c \ p\)
Decomposition as Hypergraph Partitioning

- Hypergraph partitioning:
  - Partitioning the hypergraph into $K$ parts.
  - The sum weights of all cut-edges is minimal.

- 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.

Hyperedges:

<table>
<thead>
<tr>
<th>Hyperedge</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b</td>
<td>100</td>
</tr>
<tr>
<td>a b g</td>
<td>100</td>
</tr>
<tr>
<td>a g</td>
<td>75</td>
</tr>
<tr>
<td>b g</td>
<td>75</td>
</tr>
<tr>
<td>p c</td>
<td>100</td>
</tr>
<tr>
<td>p c g</td>
<td>83.3</td>
</tr>
<tr>
<td>p g</td>
<td>50</td>
</tr>
<tr>
<td>c g</td>
<td>50</td>
</tr>
</tbody>
</table>
System Decomposition

- With the variable partition result

```
System Decomposition

• With the variable partition result

p, c

p, c

g, a, b

a

b

g
```
The Flow of our Approach

State transition system

Variable dependencies

Weighted hypergraph model

Variable partition 1
Variable partition 2
... Variable partition n

Decomposed sub-modules

weights mining
partitioning into n parts
The Benefits of our Approach

- Modules are compact and have fewer communication.
- The module has less requirements on its environment, then the assumption can be greatly reduced.

1. \( \langle A \rangle \ M_1 \ \langle P \rangle \)
2. \( \langle true \rangle \ M_2 \ \langle A \rangle \)

\[
\langle true \rangle M_1 \ || \ M_2 \ \langle P \rangle 
\]

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

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

System \rightarrow \text{Decomposition} \rightarrow \text{Comp. Verif.}

- NuSMV
- Symoda

External tools
- Apriori, hMetis
## 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></td>
<td>IO</td>
<td>time</td>
<td>IO</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>
Experimental Results

• Compositional verification leads to better performance than general model checking.

• In most of the benchmarks, our approach outperforms the unweighted hypergraph method.
Outline

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

• A new decomposition method for the assume-guarantee reasoning
  – It integrates the data mining method to the compositional verification.
  – It utilizes the weighted hypergraph partitioning to cluster the variables.

• Experimental results show the promising performance of this approach.
Thank You!

Question & Answer