Data Mining Based Decomposition for Assume Guarantee Reasoning

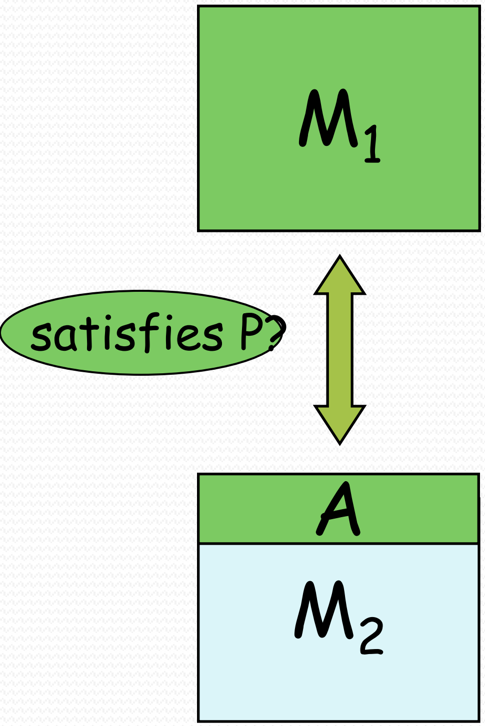
He Zhu	Tsinghua University
Fei He	Tsinghua University
William N. N. Hung	Synopsys Inc.
Xiaoyu Song	Portland State University
Ming Gu	Tsinghua University

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 || 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

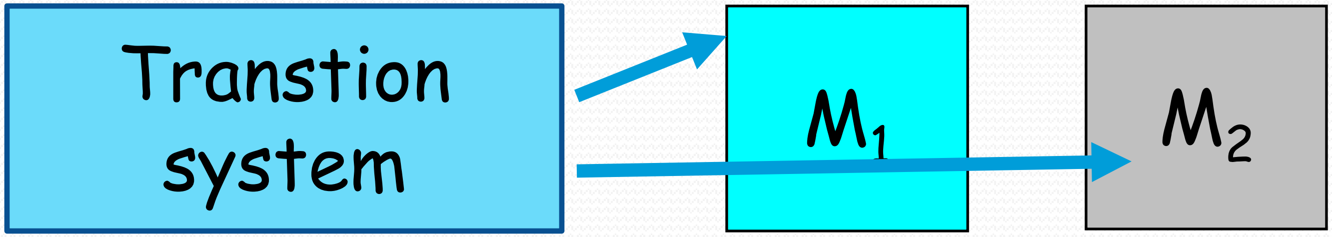
$$\frac{\begin{array}{llll} 1. & \langle A \rangle & M_1 & \langle P \rangle \\ 2. & \langle \mathit{true} \rangle & M_2 & \langle A \rangle \end{array}}{\langle \mathit{true} \rangle\ M_1 \mid\mid M_2\ \langle P \rangle}$$

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.

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

g is dependent on *a* and *b*.

X:

a, b, g, p, c

T:

t_g: g a b

t_p: p g c

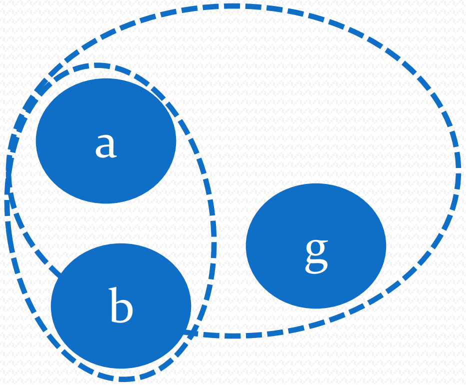
t_c: c p

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.

item

transaction

	a	b	c	g	p
t_g	1	1	0	1	0
t_p	0	0	1	1	1
t_c	0	0	1	0	1

- 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.

V_T:

t_g: g a b

t_p: p g c

t_c: c p

Frequent item sets

a b

a b g

a g

b g

p c

p g

p g c

c g

Association rules

$a \Rightarrow b$	100
$b \Rightarrow a$	100
$b\ g \Rightarrow a$	100
$g \Rightarrow a$	50
$g \Rightarrow b$	50
$c \Rightarrow g$	50
$p \Rightarrow c$	100
$p \Rightarrow g$	50
… …	

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: $p\ g \Rightarrow c$, $p\ c \Rightarrow g$, and $g\ c \Rightarrow p$.

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
```

variable transactions

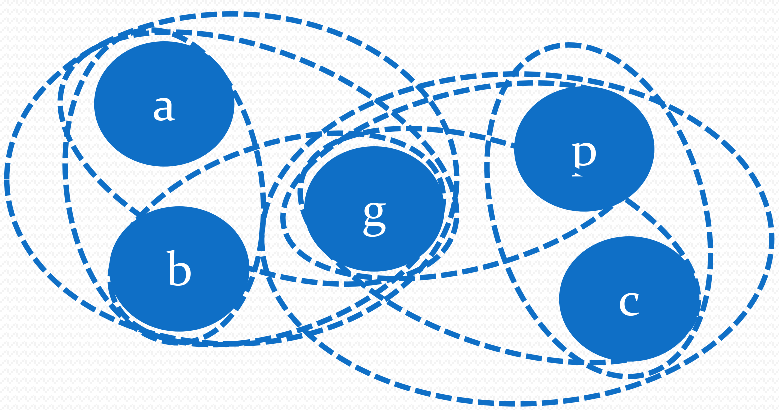
g	a	b
p	g	c
c	p	

frequent item set

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

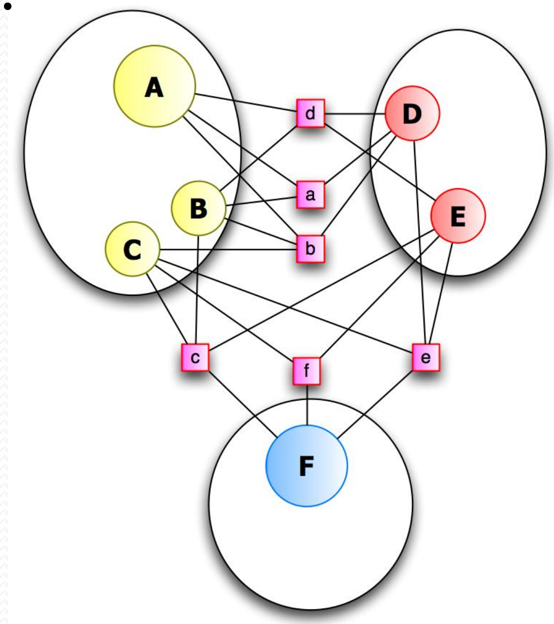
modeling



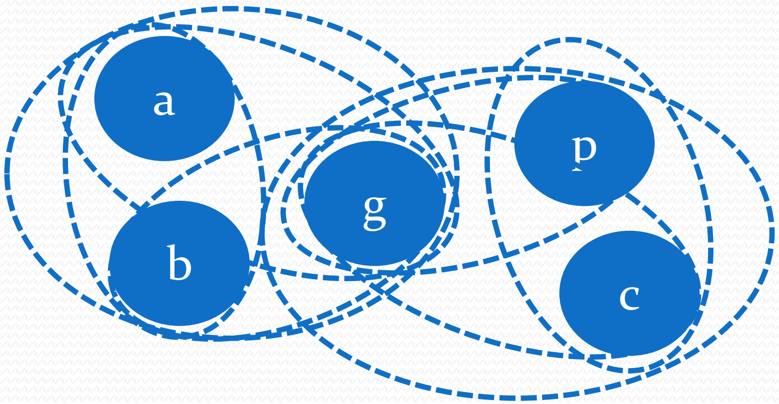
Weighted Hypergraph Model

Decomposition as Hypergraph Partitioning

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

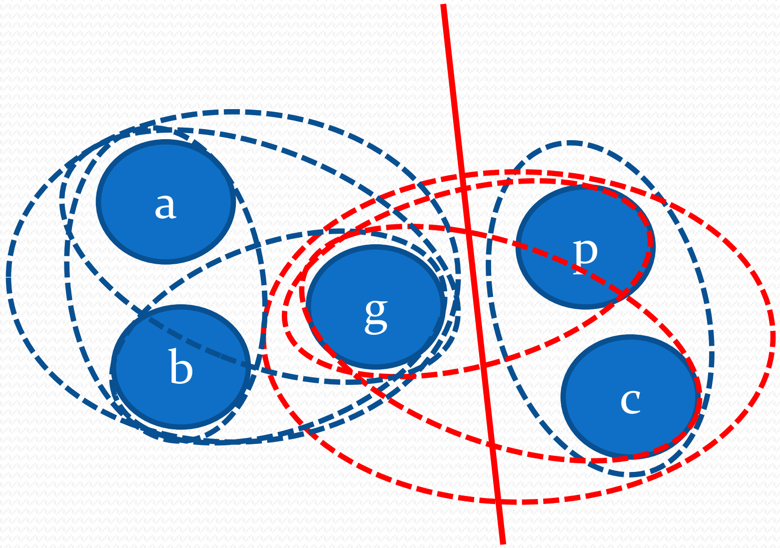


- There are some existing tools for hypergraph partitioning problem, among them, we chose hMETIS.



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



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
```

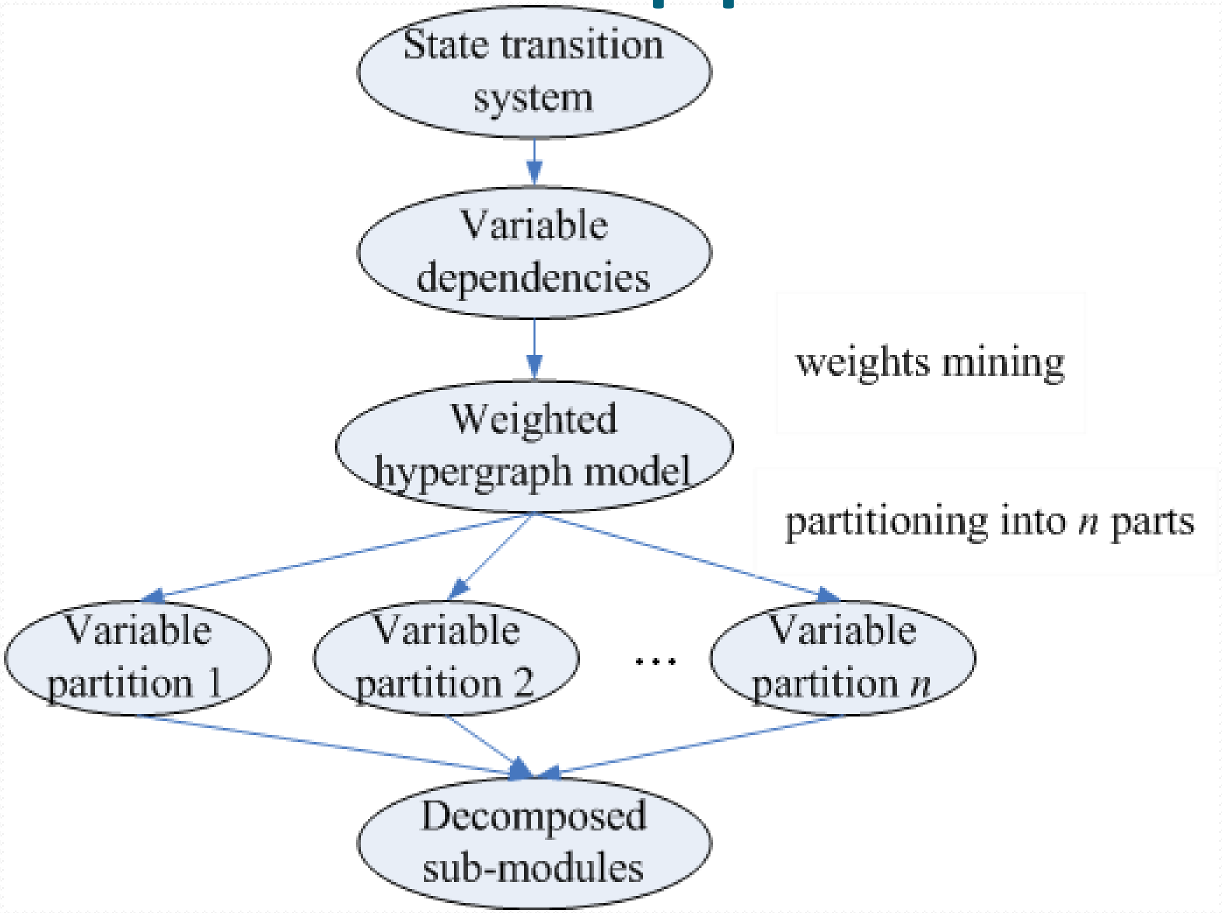
p,c

g,a,b

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

```
VAR g, a, b;
Next (g) :=  a & b;
```

The Flow of our Approach



Benefits of Our Approach

- Modules are compact and have fewer communication.
- Each module has less requirements on its environment ➔ simplify assumption

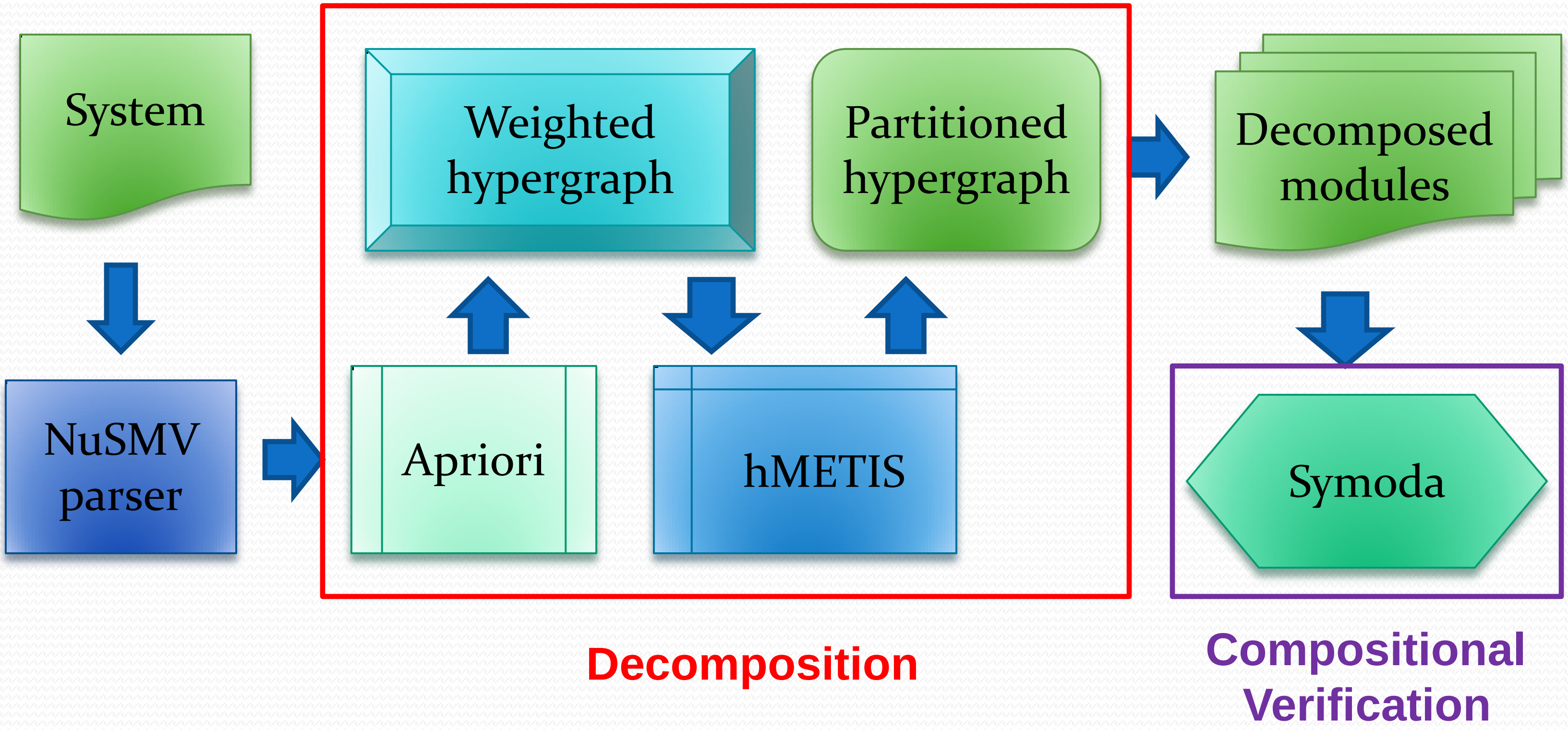
$$\frac{\begin{array}{llll} 1. & \langle A \rangle & M_1 & \langle P \rangle \\ 2. & \langle true \rangle & M_2 & \langle A \rangle \end{array}}{\langle true \rangle\ M_1 \,||\, M_2\ \langle P \rangle}$$

- Since *A* is reduced, the efforts for verifying these two premises are also reduced.

Outline

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

Implementation



Experimental Results

Benchs	Var	Weighted Hypergraph		Unweighted Hypergraph		General
		IO	time	IO	time	
s1a	23	2	0.32	2	0.31	15.77
s1b	25	6	0.49	6	0.60	16.03
msi3	61	17	2.81	19	3.53	10.23
msi5	97	24	5.86	32	8.81	27.17
msi6	121	27	9.69	33	12.11	43.80
syncarb10	74	32	76.13	33	129.20	Timeout
peterson	9	7	0.65	7	113.8	27.67
guidance	76	37	19.93	13	4.11	18.75

- 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

Conclusion and Future work

- 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