Bandwidth and Wavefront Reduction for Static Variable Ordering in Symbolic Reachability Analysis

Nasa Formal Methods 2016

Jeroen Meijer Jaco van de Pol

Formal Methods and Tools Group, University of Twente, the Netherlands

June 9, 2016

Table of Contents

- 1 Symbolic Reachability Analysis
- 2 Bandwidth and Wavefront Reduction
- 3 Results
- 4 Conclusion

- 1 Symbolic Reachability Analysis
- 2 Bandwidth and Wavefront Reduction
- 3 Results
- 4 Conclusion

A Petri net

Symbolic storage of all states

- A set of states
- Multiple transition relations
- Decision diagrams as data structure
- Also PNML, B, mCRL2, PROMELA, DVE,

Storing the States in Decision Diagrams

Storing the States in Decision Diagrams

Variable Ordering

Alphanumeric order: 16 nodes

Cuthill-McKee order: 12 nodes

Permutation

$$\pi = (2, 3, 4, 5, 1)$$

Variable Ordering

- Good variable orders save time and memory.
- What are good heuristics for variable orders?
- How do we get good variable orders?

Heuristics for Variable Ordering

Exploit **event locality**

- 1 Place related variables near each other.
- 2 Change the bottom of the decision diagram as much as possible (saturation).

Dependency Matrix

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

Dependency Matrix

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

Dependency Matrix

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

Measuring the Quality of the Order

span

The distance between the minimum and maximum nonzero of a row. Event span is the sum of all distances.

	p_1	p_2	p_3	p_4	p_5	span
t_1	0	1	0	1	1	4
t_2	0	1	1	0	0	2
t_3	0	1	1	0	0	2
t_4	1	0	0	0	1	5
t_5	1	0	0	0	1	5
t_6	1	0	1	1	0	4
event span:						22

Measuring the Quality of the Order

span

The **distance** between the minimum and maximum nonzero of a row. Event span is the sum of all distances.

	p_1	p_2	p_3	p_4	p_5	span
t_1	0	1	0	1	1	4
t_2	0	1	1	0	0	2
t_3	0	1	1	0	0	2
t_4	1	0	0	0	1	5
t_5	1	0	0	0	1	5
t_6	1	0	1	1	0	4
event span:						22

Measuring the Quality of the Order

span

The distance between the minimum and maximum nonzero of a row. Event span is the **sum** of all distances.

	p_1	p_2	p_3	p_4	p_5	span
t_1	0	1	0	1	1	4
t_2	0	1	1	0	0	2
t_3	0	1	1	0	0	2
t_4	1	0	0	0	1	5
t_5	1	0	0	0	1	5
t_6	1	0	1	1	0	4
event span:						22

Measuring the Quality of the Order (2)

weighted span (Siminiceanu et al., 2006)

- moment = $\frac{\text{Distance of the leftmost nonzero from the rightmost column}}{\text{half the width of the matrix}}$
- weighted span = span $\cdot$ moment
- weighted event span = sum of all weighted spans

	p_1	p_2	p_3	p_4	p_5	span	mom.	w.s.
t_1	0	1	0	1	1	4	1.2	4.8
t_2	0	1	1	0	0	2	1.2	2.4
t_3	0	1	1	0	0	2	1.2	2.4
t_4	1	0	0	0	1	5	1.6	8
t_5	1	0	0	0	1	5	1.6	8
t_6	1	0	1	1	0	4	1.6	6.4
weighted event span:								32

Measuring the Quality of the Order (2)

weighted span (Siminiceanu et al., 2006)

- $\text{moment} = \frac{\text{Distance of the leftmost nonzero from the rightmost column}}{\text{half the width of the matrix}}$
- **weighted span** = span · **moment**
- weighted event span = sum of all weighted spans

	p_1	p_2	p_3	p_4	p_5	span	mom.	w.s.
t_1	0	1	0	1	1	4	1.2	4.8
t_2	0	1	1	0	0	2	1.2	2.4
t_3	0	1	1	0	0	2	1.2	2.4
t_4	1	0	0	0	1	5	1.6	8
t_5	1	0	0	0	1	5	1.6	8
t_6	1	0	1	1	0	4	1.6	6.4
weighted event span:								32

Measuring the Quality of the Order (2)

weighted span (Siminiceanu et al., 2006)

- $\text{moment} = \frac{\text{Distance of the leftmost nonzero from the rightmost column}}{\text{half the width of the matrix}}$
- $\text{weighted span} = \text{span} \cdot \text{moment}$
- **weighted event span** = sum of all weighted spans

	p_1	p_2	p_3	p_4	p_5	span	mom.	w.s.
t_1	0	1	0	1	1	4	1.2	4.8
t_2	0	1	1	0	0	2	1.2	2.4
t_3	0	1	1	0	0	2	1.2	2.4
t_4	1	0	0	0	1	5	1.6	8
t_5	1	0	0	0	1	5	1.6	8
t_6	1	0	1	1	0	4	1.6	6.4
weighted event span:								32

Span Reduction

Existing *heuristic* algorithms

- weighted event span: not available?
- event span: FORCE (Aloul, 2003) and Noack (1999, Petri nets only), both subquadratic complexity.

Alphanumeric: WES = 32

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

FORCE: WES = 19

	p_3	p_2	p_4	p_1	p_5
t_1	0	1	1	0	1
t_2	1	1	0	0	0
t_3	1	1	0	0	0
t_4	0	0	0	1	1
t_5	0	0	0	1	1
t_6	1	0	1	1	0

Span Reduction

Existing *heuristic* algorithms

- weighted event span: not available?
- event span: FORCE (Aloul, 2003) and Noack (1999, Petri nets only), both subquadratic complexity.

Alphanumeric: WES = 32

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

FORCE: WES = 19

	p_3	p_2	p_4	p_1	p_5
t_1	0	1	1	0	1
t_2	1	1	0	0	0
t_3	1	1	0	0	0
t_4	0	0	0	1	1
t_5	0	0	0	1	1
t_6	1	0	1	1	0

Span Reduction

Existing *heuristic* algorithms

- weighted event span: not available?
- event span: FORCE (Aloul, 2003) and Noack (1999, Petri nets only), both subquadratic complexity.

Alphanumeric: WES = 32

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

FORCE: WES = 19

	p_3	p_2	p_4	p_1	p_5
t_1	0	1	1	0	1
t_2	1	1	0	0	0
t_3	1	1	0	0	0
t_4	0	0	0	1	1
t_5	0	0	0	1	1
t_6	1	0	1	1	0

- 1 Symbolic Reachability Analysis
- 2 Bandwidth and Wavefront Reduction**
- 3 Results
- 4 Conclusion

- Sparse matrix solvers (decades old field of study ≥ 1960)
- Bandwidth: proportional to memory usage
- Wavefront: proportional to time usage

main obstacle

Only algorithms for **symmetric** matrices.
Dependency matrix is **asymmetric**.

Symmetrization (Reid, 2006)

Matrix A: asymmetric

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

Matrix $\hat{A}$: symmetric

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	0	1	0	0	0	1
t_5	0	0	0	0	0	0	1	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

Procedure

$$\hat{A} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

the algorithm

- Start from the smallest vertex with minimum degree.
- Perform breadth-first traversal.
- Visit neighbors in increasing order of degree.

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	0	1	0	0	0	1
t_5	0	0	0	0	0	0	1	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

the algorithm

- Start from the smallest vertex with minimum degree.
- Perform breadth-first traversal.
- Visit neighbors in increasing order of degree.

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	0	1	0	0	0	1
t_5	0	0	0	0	0	0	1	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

the algorithm

- Start from the smallest vertex with minimum degree.
- Perform breadth-first traversal.
- Visit neighbors in increasing order of degree.

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	0	1	0	0	0	1
t_5	0	0	0	0	0	0	1	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

the algorithm

- Start from the smallest vertex with minimum degree.
- Perform breadth-first traversal.
- Visit neighbors in increasing order of degree.

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	0	1	0	0	0	1
t_5	0	0	0	0	0	0	1	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

the algorithm

- Start from the smallest vertex with minimum degree.
- Perform breadth-first traversal.
- Visit neighbors in increasing order of degree.

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	0	1	0	0	0	1
t_5	0	0	0	0	0	0	1	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

the algorithm

- Start from the smallest vertex with minimum degree.
- Perform breadth-first traversal.
- Visit neighbors in increasing order of degree.

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	0	1	0	0	0	1
t_5	0	0	0	0	0	0	1	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

Alphanumeric

- Bandwidth: 10
- Profile: 76

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	1	0	0	0	0	1
t_5	0	0	0	0	0	1	0	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

Cuthill McKee

- Bandwidth: 3
- Profile: 29

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

Alphanumeric

- Bandwidth: 10
- Profile: 76

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	1	0	0	0	0	1
t_5	0	0	0	0	0	1	0	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

Cuthill McKee

- Bandwidth: 3
- Profile: 29

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

Alphanumeric

- Bandwidth: 10
- Profile: 76

	t_1	t_2	t_3	t_4	t_5	t_6	p_1	p_2	p_3	p_4	p_5
t_1	0	0	0	0	0	0	0	1	0	1	1
t_2	0	0	0	0	0	0	0	1	1	0	0
t_3	0	0	0	0	0	0	0	1	1	0	0
t_4	0	0	0	0	0	1	0	0	0	0	1
t_5	0	0	0	0	0	1	0	0	0	0	1
t_6	0	0	0	0	0	0	1	0	1	1	0
p_1	0	0	0	1	1	1	0	0	0	0	0
p_2	1	1	1	0	0	0	0	0	0	0	0
p_3	0	1	1	0	0	1	0	0	0	0	0
p_4	1	0	0	0	0	1	0	0	0	0	0
p_5	1	0	0	1	1	0	0	0	0	0	0

Cuthill McKee

- Bandwidth: 3
- Profile: 29

	t_2	p_2	p_3	t_3	t_1	t_6	p_4	p_5	p_1	t_4	t_5
t_2	0	1	1	0	0	0	0	0	0	0	0
p_2	1	0	0	1	1	0	0	0	0	0	0
p_3	1	0	0	1	0	1	0	0	0	0	0
t_3	0	1	1	0	0	0	0	0	0	0	0
t_1	0	1	0	0	0	0	1	1	0	0	0
t_6	0	0	1	0	0	0	1	0	1	0	0
p_4	0	0	0	0	1	1	0	0	0	0	0
p_5	0	0	0	0	1	0	0	0	0	1	1
p_1	0	0	0	0	0	1	0	0	0	1	1
t_4	0	0	0	0	0	0	0	1	1	0	0
t_5	0	0	0	0	0	0	0	1	1	0	0

De-symmetrization

Total order $\rightarrow$ partial order

Example decision diagram

$p_2 < p_3 < p_4 < p_5 < p_1$ is the order we saw earlier.

De-symmetrization

Total order $\rightarrow$ partial order

Example decision diagram

$p_2 < p_3 < p_4 < p_5 < p_1$ is the order we saw earlier.

Result

Partial order

$$t_2 < t_3 < t_1 < t_6 < t_4 < t_5 \cup p_2 < p_3 < p_4 < p_5 < p_1$$

Alphanumeric: ES=22, WES=32

	p_1	p_2	p_3	p_4	p_5
t_1	0	1	0	1	1
t_2	0	1	1	0	0
t_3	0	1	1	0	0
t_4	1	0	0	0	1
t_5	1	0	0	0	1
t_6	1	0	1	1	0

Cuthill McKee: ES=16, WES=19

	p_2	p_3	p_4	p_5	p_1
t_2	1	1	0	0	0
t_3	1	1	0	0	0
t_1	1	0	1	1	0
t_6	0	1	1	0	1
t_4	0	0	0	1	1
t_5	0	0	0	1	1

unknown symmetric graph algorithms

Algorithm	Package	Reducing type	Year
Cuthill McKee	Boost	bandwidth	1969
King		bandwidth, profile	1970
Sloan		profile, wavefront	1989
Cuthill McKee	ViennaCL	bandwidth	1969
adv. Cuthill McKee		bandwidth	~
GPS		bandwidth, profile	1976

known algorithms for span reduction

- Noack1, Noack2 (for Petri nets).
- FORCE

database10UNFOLD.pnml, WES = 612585

Boost's Cuthill McKee, $WES = 275274$

Boost's King, WES = 264108

Boost's Sloan, WES = 289062

Boost's Sloan, WES = 289062

ViennaCL's Cuthill McKee, WES = 338263

ViennaCL's advanced Cuthill McKee, WES = 327478

ViennaCL's Gibbs Poole Stockmeyer, WES = 297214

FORCE, WES = 149719

FORCE, WES = 149719

Noack1, WES = 235904

Noack1, WES = 235904

Noack2, WES = 217641

Noack2, WES = 217641

Notable feature

- Presented algorithms can be run on any symmetric matrix.
- Total graph sometimes produces better results (Kaveh, 2006).
- Edges become vertices; incident edges and adjacent vertices are connected.

graph

	<i>a</i>	<i>b</i>	<i>c</i>
<i>a</i>	0	1	0
<i>b</i>	1	0	1
<i>c</i>	0	1	0

total graph

	<i>a</i>	<i>b</i>	<i>c</i>	(a, b)	(b, c)
<i>a</i>	0	1	0	1	0
<i>b</i>	1	0	1	1	1
<i>c</i>	0	1	0	0	1
(a, b)	1	1	0	0	1
(b, c)	0	1	1	1	0

- 1 Symbolic Reachability Analysis
- 2 Bandwidth and Wavefront Reduction
- 3 Results**
- 4 Conclusion

What are we interested in?

research questions

- What algorithm produces the best WES?
- How do the algorithms perform for state space generation?

What are we interested in?

research questions

- What algorithm produces the best WES?
- How do the algorithms perform for state space generation?

standardizing

- We benchmark 785 model specifications, written in B, DVE, mCRL2, PNML and Promela: *lots of data*.
- We can only compare to a mean value (e.g. mean WES).
- We must *standardize* values produced by benchmarks.

Mean Standard Score

standard score

- $z_s = \frac{x_s - \mu_s}{\sigma_s}$
- $s \in \{\text{database10UNFOLD.pnml}, \dots\}$
- μ_s, σ_s is a mean/standard deviation over all reordering algorithms (for e.g. time, WES, ...).

mean standard score

- $MSS = \frac{\sum_{s \in \{\text{database10UNFOLD.pnml}, \dots\}} z_s}{|\{\text{database10UNFOLD.pnml}, \dots\}|}$
- Question, e.g.: how many standard deviations is Sloan better than the mean on WES?
- Lower MSS is better

Best Algorithm for WES Reduction (1)

314 Petri nets (Petri net model checking contest, 2015)

Best Algorithm for WES Reduction (2)

47 B, 264 DVE, 142 mCRL2, 314 PNML, 18 Promela (= 785)

Best Algorithm for Reachability

- 30 minutes timeout
- *peak nodes* = maximum number of nodes during reachability.

- 1 Symbolic Reachability Analysis
- 2 Bandwidth and Wavefront Reduction
- 3 Results
- 4 Conclusion

- Three new metrics: bandwidth, profile, wavefront.
- Four new algorithms: Cuthill McKee, King, Sloan, GPS.
- Six implementations: Boost, ViennaCL.
- Two new graph representations: symmetric, total.
- Language independent approach; works for DVE, PNML, mCRL2, B, Promela,

- Implement algorithms in other model checkers: SMART, ITS-TOOLS, Marcie, PetriDotNet,
- Implement more graph representations (Kaveh): Element Clique Graph, Skeleton Graph, Element Wheel Graph,
- Metaheuristic algorithms for bandwidth, profile, wavefront.
- Algorithms can be applied to SAT/SMT solving as well.

Thank you