DTMC Model Checking by SCC Reduction

Erika Ábrahám, Nils Jansen, Ralf Wimmer, Joost-Pieter Katoen, and Bernd Becker

Theory of Hybrid Systems
RWTH Aachen University Germany

Computer Architecture
University of Freiburg, Germany

September 16, 2010

QEST 2010
Probabilistic Model Checking
Counterexamples for Probabilistic Systems

“It is impossible to overestimate the importance of the counterexample feature. The counterexamples are invaluable in debugging complex systems. Some people use model checking just for this feature.”

Edmund Clarke, 2008
Contents

1 Motivation

2 SCC-based Model Checking

3 Interactive Counterexample Refinement

4 Implementation and Case Studies

5 Contribution and Perspectives
(CTL or LTL)-Model Checking provides counterexamples.

Classical probabilistic Model Checking provides no diagnostic information.

State-of-the-art probabilistic counterexample generation needs Model Checking result for completeness.
(CTL or LTL)-Model Checking provides counterexamples.

Classical probabilistic Model Checking provides no diagnostic information.

State-of-the-art probabilistic counterexample generation needs Model Checking result for completeness.

Goal: Development of a Model Checking algorithm with simultaneous counterexample generation

- Counterexamples may consist of a large number of paths.
- Abstract but refinable counterexamples
Contents

1 Motivation

2 SCC-based Model Checking

3 Interactive Counterexample Refinement

4 Implementation and Case Studies

5 Contribution and Perspectives
Problem
- Model Checking DTMCs against unbounded reachability properties
- Target nodes are absorbing.

Observation
- Paths through a non-bottom-SCC will leave it with probability 1.
- For unbounded reachability we look at the whole probability mass provided by SCCs.

Idea
- Reduce each SCC to an abstract node whose outgoing edges carry the whole probability mass.
Related approaches

- Computation of probabilities for SCCs towards counterexamples
  - Andrés, D’Argenio and van Rossum, 2008

- State elimination
  - Finite Automaton: regular expression
  - Markov Chain: Compute probabilities during elimination
    - Daws 2004
    - Damman, Han, and Katoen 2008
    - Hahn, Hermanns, and Zhang, 2009

- Our approach: Recursive algorithm
  - Bottom-Up computing starting with “minimal SCCs”
  - Exploiting specific properties of Markov Chains
Example SCC

Input node: $s_1$

Output nodes: $s_2$ and $s_3$

Probabilities $p_2$, $p_3$ of reaching output nodes.
Example SCC

- **Input node:** $s_1$

---

SCCs - Point of View

Example SCC

- **Input node:** $s_1$
Example SCC

- **Input node:** $s_1$
- **Output nodes:** $s_2$ and $s_3$
Example SCC - abstract view

- **Input node:** $s_1$
- **Output nodes:** $s_2$ and $s_3$
Example SCC - more abstract view

- **Input node:** $s_1$
- **Output nodes:** $s_2$ and $s_3$
- **Probability** $p_r$ of returning to input node
- **Probabilities** $p_2, p_3$ of reaching output nodes
Every path through a DTMC will eventually reach a Bottom-SCC (and stay there).
Every path through a DTMC will eventually reach a Bottom-SCC (and stay there).

Non-Bottom-SCC will be left at some point in time with probability 1.
Every path through a DTMC will eventually reach a Bottom-SCC (and stay there).

Non-Bottom-SCC will be left at some point in time with probability 1.

Probability of reaching one output node is composed of the probabilities of all finite paths through the SCC.
Probability mass of all paths entering and eventually leaving SCC:

$$\sum_{i \in \mathbb{N}} p_r^i \cdot (p_2 + p_3) = \frac{1}{1 - p_r} \cdot (p_2 + p_3)$$

Probability of coming back to input node can be computed without considering paths that lead back to it!
Probability mass of all paths entering and eventually leaving SCC:

\[ \sum_{i \in \mathbb{N}} p_r^i \cdot (p_2 + p_3) = \frac{1}{1 - p_r} \cdot (p_2 + p_3) \]

Prob. for leaving SCC is 1:

\[ \frac{1}{1 - p_r} \cdot (p_2 + p_3) = 1 \]

\[ \iff p_r = 1 - p_2 - p_3 \]
Probability mass of all paths entering and eventually leaving SCC:

\[ \sum_{i \in \mathbb{N}} p_r^i \cdot (p_2 + p_3) = \frac{1}{1 - p_r} \cdot (p_2 + p_3) \]

Prob. for leaving SCC is 1:

\[ \frac{1}{1 - p_r} \cdot (p_2 + p_3) = 1 \]

⇔ \( p_r = 1 - p_2 - p_3 \)

Probability of coming back to input node can be computed without considering paths that lead back to it!
SCCs - One Input Node

Probability of reaching $s_2$:

$$\frac{1}{1 - p_r \cdot p_2}$$
Probability of reaching $s_2$:

$$\frac{1}{1 - p_r} \cdot p_2$$

Discard $p_r$ and scale $p_2$ and $p_3$:

$$p_2' = \frac{p_2}{p_2 + p_3}$$

$$p_3' = \frac{p_3}{p_2 + p_3}$$
Probability of reaching $s_2$:

\[ \frac{1}{1 - p_r} \cdot p_2 \]

Discard $p_r$ and scale $p_2$ and $p_3$:

\[ p_2' = \frac{p_2}{p_2 + p_3} \]

\[ p_3' = \frac{p_3}{p_2 + p_3} \]

Scaling preserves reachability probabilities:

\[ p_2' = \frac{p_2}{p_2 + p_3} = \frac{p_2}{p_2 + (1 - p_2 - p_1)} = \frac{1}{1 - p_r} \cdot p_2 \]
Paths returning to input node are ignored during computations.
Example

Whole graph:
- Search for SCCs
Example

SCC 1

Nils Jansen - SCC-based Model Checking
Example

SCC 1

SCC 1:
- Ignore Input-Node
- Search for SCCs
Example

SCC 1

SCC 1.1

SCC 1.2

Nils Jansen - SCC-based Model Checking
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.1:
- Ignore Input-Node
- Search for SCCs

SCC 1.2:
- Ignore Input-Node
- Search for SCCs
No SCCs found!
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1

SCC 1.2:
- Ignore Input-Node
- Search for SCCs

Nils Jansen - SCC-based Model Checking
SCC 1.2.1:
- No SCCs found!
- Compute probabilities for reaching output nodes from input node.
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1:

- Scale probabilities.

Nils Jansen - SCC-based Model Checking
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1

SCC 1.1

SCC 1.2

Nils Jansen - SCC-based Model Checking
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1:
- Reduced to abstract node.

Nils Jansen - SCC-based Model Checking
SCC 1.2:
- Compute probabilities for reaching output nodes from input node.
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1

SCC 1.2:

- Scale probabilities.
Example
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1

SCC 1.2:
- Reduced to abstract node.
Example

SCC 1

SCC 1.1

SCC 1.1:
- Compute probabilities for reaching output nodes from input node.
Example

SCC 1

SCC 1.1

SCC 1.1:

- Scale probabilities.

Nils Jansen - SCC-based Model Checking
Example

SCC 1

SCC 1.1

1.1

SCC 1.2

SCC 1.2.1

SCC 1.1:
- Reduced to abstract node.
Example

SCC 1:
- Compute probabilities for reaching output nodes from input node.
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1

SCC 1.

SCC 1.

SCC 1.

SCC 1:

- Scale probabilities.
Example

SCC 1

SCC 1.1

SCC 1.2

SCC 1.2.1

\[ 0.5339 \]

\[ 0.4661 \]

\[ 1 \]
Example

SCC 1:
- Reduced to abstract node.
- Model Checking result for reachability
Towards Counterexamples

- SCCs are represented by abstract nodes.
- Abstract paths may contain both abstract and concrete nodes.
- All intermediate results are saved.
Towards Counterexamples

- SCCs are represented by abstract nodes.
- Abstract paths may contain both abstract and concrete nodes.
- All intermediate results are saved.
  - a) Probability mass for all paths walking through SCC

\[ p_r, p_1', p_2' \]: Previously calculated scaled outgoing probabilities

\[ p_r \]: Saved probability of coming back to ingoing node
Towards Counterexamples

- SCCs are represented by abstract nodes.
- Abstract paths may contain both abstract and concrete nodes.
- All intermediate results are saved.
  - a) Probability mass for all paths walking through SCC
  - b) Probabilities for returning to input node and leaving

```
b)

pr

p1

p2

pr

SCC 1

s2

s3
```
Towards Counterexamples

- SCCs are represented by abstract nodes.
- Abstract paths may contain both abstract and concrete nodes.
- All intermediate results are saved.
  a) Probability mass for all paths walking through SCC
  b) Probabilities for returning to input node and leaving
- Interactive counterexample refinement
Property to refute:
Probability of reaching $s_5$ is less or equal than $0.3$
Interactive Counterexample Refinement - Example

Returning to $s_1$ is not needed.
Visit of abstract node 1.1 and hidden SCC is not needed
Interactive Counterexample Refinement - Example

Returning to abstract node 1.2 is not needed
Interactive Counterexample Refinement - Example

Abstract path $s_1 \rightarrow s_6 \rightarrow 1.2.1 \rightarrow s_5$ induces probability mass of 0.33264.
At least one revisit of 1.2.1 is needed
The paths

\[ S_1 \rightarrow S_6 \rightarrow S_7 \rightarrow S_5 \]
\[ S_1 \rightarrow S_6 \rightarrow S_7 \rightarrow S_8 \rightarrow S_7 \rightarrow S_5 \]

induce probability mass 0.31806 and are therefore sufficient as counterexample.
Contents

1 Motivation

2 SCC-based Model Checking

3 Interactive Counterexample Refinement

4 Implementation and Case Studies

5 Contribution and Perspectives
Implementation

- Prototype in C++
- Goal: Comparison to leading Probabilistic Model Checking tools like
  - MRMC (Katoen et al.)
  - PRISM (Kwiatkowska et al.)
- Same input format as MRMC
- Model Checking for case studies
  - Synchronous leader election protocol (Itai, Rodeh, 1990)
  - Crowds protocol (Reiter, Rubin, 1998)
Results for the leader election benchmark

<table>
<thead>
<tr>
<th></th>
<th>states</th>
<th>transitions</th>
<th>MRMC</th>
<th>SCC-MC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3902</td>
<td>5197</td>
<td>0.002</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>30014</td>
<td>40013</td>
<td>0.01</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>115262</td>
<td>153677</td>
<td>0.035</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>131101</td>
<td>163868</td>
<td>0.04</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>236225</td>
<td>295273</td>
<td>0.054</td>
<td>0</td>
</tr>
</tbody>
</table>
Results for the crowds benchmark

<table>
<thead>
<tr>
<th></th>
<th>3515</th>
<th>18817</th>
<th>68740</th>
<th>198199</th>
<th>485941</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>6035</td>
<td>32677</td>
<td>120220</td>
<td>348349</td>
<td>857221</td>
</tr>
<tr>
<td>transitions</td>
<td>0.001</td>
<td>0.014</td>
<td>0.085</td>
<td>0.344</td>
<td>0.877</td>
</tr>
<tr>
<td>MRMC s</td>
<td>0.01</td>
<td>0.01</td>
<td>0.06</td>
<td>0.19</td>
<td>0.44</td>
</tr>
<tr>
<td>SCC-MC s</td>
<td>0.81</td>
<td>4.29</td>
<td>15.67</td>
<td>45.19</td>
<td>110.82</td>
</tr>
<tr>
<td>MRMC MB</td>
<td>0.64</td>
<td>3.50</td>
<td>13.16</td>
<td>37.29</td>
<td>88.68</td>
</tr>
<tr>
<td>SCC-MC MB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Contents

1 Motivation

2 SCC-based Model Checking

3 Interactive Counterexample Refinement

4 Implementation and Case Studies

5 Contribution and Perspectives
Perspectives

SCC-based abstraction, Model Checking, and interactive counterexample refinement

- Recursive SCC-based algorithm
- User can control structure and abstraction-level of counterexamples.
Perspectives

SCC-based abstraction, Model Checking, and interactive counterexample refinement

- Recursive SCC-based algorithm
- User can control structure and abstraction-level of counterexamples.

Future work

- Automation of counterexample-search
- User interface and optimization of implementation
- Extension to MDPs (not straightforward)
- Next step:
  - Extension to parametric Markov Chains
  - Parameter synthesis for Markov Chains
Perspectives

SCC-based abstraction, Model Checking, and interactive counterexample refinement

- Recursive SCC-based algorithm
- User can control structure and abstraction-level of counterexamples.

Future work

- Automation of counterexample-search
- User interface and optimization of implementation
- Extension to MDPs (not straightforward)
- Next step:
  - Extension to parametric Markov Chains
  - Parameter synthesis for Markov Chains

Thank you for your attention!