Markov Reward Model Checker

Joost-Pieter Katoen and Ivan S. Zapreev

MOVES, RWTH Aachen, Germany
FMT, Twente, The Netherlands

November 9, 2007
Outline

Introduction to MRMC

Model checking with MRMC

Getting MRMC models

MRMC versus PRISM, Ymer, VESTA, ETMCC

Implementation metrics

The third-party projects

The next release

Conclusions and future work
Introduction to MRMC

What is MRMC?

- A probabilistic model checker for DTMCs, CTMCs and MRM
- A command-line tool implemented in C
- The tool is:
  - available for Windows, Mac OS X and Linux
  - distributed under the GPL license

What makes MRMC so special?

- It is small and fast, perfect as a backend
- It supports:
  - Bisimulation minimization [KKZJ07]
  - Precise on-the-fly steady-state detection [KZ05]
  - Improved model checking for steady-state properties
The heart of MRMC

Sparse matrices

\[ P = \begin{bmatrix} 0.50 & 0.50 & 0.00 \\ 0.25 & 0.00 & 0.75 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \]

- **nrows**: number of rows
- **succ**: number of successors
- **pred**: number of predecessors
- **rows**: matrix rows
- **col**: non-zero column indexes
- **val**: non-zero element values
- **diag**: a diagonal value
- **back_set**: predecessors
A simple DMRM model

- Consider a dice with only four wedges: 1, 2, 3 and 4
- The outcomes have probabilities 0.4, 0.3, 0.2 and 0.1
- The outcome 4 is the \textit{goal} state
- The outcome 1 is the \textit{loss} state
- $P_{>0.5} \left( \neg \text{loss} U_{[0, 199]} \text{goal} \right)$

<table>
<thead>
<tr>
<th>game.tra</th>
<th>game.lab</th>
<th>game.rew</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATES 5</td>
<td>#DECLARATION</td>
<td>2 1</td>
</tr>
<tr>
<td>TRANSITIONS 8</td>
<td>loss goal</td>
<td>3 2</td>
</tr>
<tr>
<td>1 2 0.4</td>
<td>#END</td>
<td>4 3</td>
</tr>
<tr>
<td>1 3 0.3</td>
<td>2 loss</td>
<td>5 4</td>
</tr>
<tr>
<td>1 4 0.2</td>
<td>5 goal</td>
<td></td>
</tr>
<tr>
<td>1 5 0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 1 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 1 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 1 1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Verifying $P_{>0.5} \left( \neg \text{loss} \ U^{[0,199]}_{[5,50]} \text{goal} \right)$ with MRMC

MRMC/bin> mrmc prctl game.tra game.lab game.rewi

... Logic = PRCTL

Loading the 'game.tra' file, please wait.
States=5, Transitions=8
Loading the 'game.lab' file, please wait.
Loading the 'game.rew' file, please wait.
The Occupied Space is 992 Bytes.
Type 'help' to get help.

$>$ $P_{>0.5}[ \text{!loss} \ U_{[0,199]}^{[5,50]} \text{goal}]$

$\text{RESULT: ( 0.0647999, 0.0000000, 0.0959998, 0.1199998, 0.1199997 )}$

$\text{STATE: { } }$

The Total Elapsed Model-Checking Time is 45 milli sec(s).

$> $
Getting MRMC models

PRISM [HKNP06]

► A high-level state-based description language based on the Reactive Modules formalism.
► The underlying models are DTMCs, CTMCs, MDPS and MRMs.
► The tool allows for exporting its models into the MRMC file formats.

PEPA Workbench [TG06]

► An algebraic process-oriented language for modeling concurrent systems.
► The underlying models are CTMCs.
► The tool allows for exporting PEPA models into the MRMC file formats.
MRMC versus PRISM, Ymer, VESTA, ETMCC

Supported platforms

<table>
<thead>
<tr>
<th>Tool</th>
<th>Linux</th>
<th>Windows</th>
<th>Solaris</th>
<th>Mac OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRISM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MRMC</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ETMCC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>VESTA</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Ymer</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MRMC versus PRISM, Ymer, VESTA, ETMCC

Models and logics

<table>
<thead>
<tr>
<th>Tool</th>
<th>DTMC</th>
<th>CTMC</th>
<th>MDP</th>
<th>DMRM</th>
<th>CMRM</th>
<th>GSMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRMC</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PRISM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VESTA</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ymer</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ETMCC</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Only MRMC supports all of PCTL, CSL, PRCTL and CSRL!
The supported operators of PCTL

<table>
<thead>
<tr>
<th>Tool</th>
<th>$\mathcal{L}_{\bowtie p}[\Phi]$</th>
<th>$\mathcal{P}_{\bowtie p}[\Phi \cup \Psi]$</th>
<th>$\mathcal{P}_{\bowtie p}[\Phi \leq^k \Psi]$</th>
<th>$\mathcal{P}_{\bowtie p}[\Phi \cup^{[k_1,k_2]} \Psi]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRMC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PRISM</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VESTA</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
MRMC versus PRISM, Ymer, VESTA, ETMCC

The supported operators of CSL

<table>
<thead>
<tr>
<th>Tool</th>
<th>$\mathcal{S}[\Phi]$</th>
<th>$P_\mathcal{X}[\Phi \cup \Psi]$</th>
<th>$P_\mathcal{X}[\Phi \cup t \leq \Psi]$</th>
<th>$P_\mathcal{X}[\Phi \cup t \geq \Psi]$</th>
<th>$P_\mathcal{X}[\Phi \cup [t_1, t_2]] \Psi$</th>
<th>$P_\mathcal{X}[t \leq \Psi]$</th>
<th>$P_\mathcal{X}[t \geq \Psi]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRMC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PRISM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ymer</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VESTA</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETMCC</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Implementation metrics

<table>
<thead>
<tr>
<th>MRMC metrics</th>
<th>Value</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of code</td>
<td>6738</td>
<td>Understand C/C++</td>
</tr>
<tr>
<td>Lines of comments</td>
<td>8287</td>
<td></td>
</tr>
<tr>
<td>McCabes cyclomatic complexity</td>
<td>1399</td>
<td>CCCC</td>
</tr>
<tr>
<td>Development effort estimate</td>
<td>20.31 MM</td>
<td>SLOCCCount</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test-suite metrics</th>
<th>Value</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test coverage</td>
<td>83.46%</td>
<td>GCov</td>
</tr>
<tr>
<td>Lines of code</td>
<td>1474</td>
<td>SLOCCCount</td>
</tr>
<tr>
<td>Development effort estimate</td>
<td>3.61 MM</td>
<td></td>
</tr>
</tbody>
</table>
The third-party projects

GreatSPN v2.0
- Department information, Università di Torino, Italy
- Modeling, validation, and performance evaluation of distributed systems
- MRMC as a backend for CSL model checking [CDDS06]

Heuristics-Guided Dependability Analysis
- The chair of Software Engineering, Universität des Konstanz, Germany
- Generating diagnostics information for stochastic models [AL06]
- A prototype tool called DiPro is being linked to MRMC

Reachability analysis in uniform CTMDPs
- Dependable Systems & Software group, Universität des Saarlandes, Germany
- New timed-reachability algorithms for uCTMDPs [BHH+06]
- The tool chain: STATEMATE – extended MRMC
The next release

What to expect from MRMC v1.3?

- Model checking via discrete-event simulation (PCTL, CSL)
- Model checking of uCTMDPs (reachability properties)
- Optimized performance and memory usage
- Improved command-prompt interface:
  - Access to the intermediate model-checking results
  - Context help and run-time settings
  - *Et cetera* ...
- Simplified internal interfaces for the external developers
Conclusions and future work

Conclusions

- **MRMC** is small and fast
- It is the only tool supporting:
  - PCTL, CSL, PRCTL and CSRL
  - Bisimulation minimization
  - Precise steady-state detection
- It is available for: Windows, Linux and Max OS X
- There are several third-party projects that use **MRMC**

Future work

- State-space abstractions
- MDPs, CTMCPs, etc.
- Counter examples
Conclusions and future work

- Husain Aljazzar and Stefan Leue.
  Extended directed search for probabilistic timed reachability.

- Eckard Bode, Marc Herbstritt, Holger Hermanns, Sven Johr, Thomas Peikenkamp, Reza Pulungan, Ralf Wimmer, and Bernd Becker.
  Compositional Performability Evaluation for STATEMATE.

- D. Cerotti, D. D’Aprile, S. Donatelli, and J. Sproston.
  Verifying stochastic well-formed nets with csl model-checking tools.

- A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker.
  PRISM: A tool for automatic verification of probabilistic systems.

- Joost-Pieter Katoen, Tim Kemna, Ivan Zapreev, and David N. Jansen.
  Bisimulation Minimisation Mostly Speeds Up Probabilistic Model Checking.


- Mirco Tribastone and Stephen Gilmore.
  A New Generation PEPA Workbench.