Extending Markov Automata with State and Action Rewards*

Dennis Guck Mark Timmer Stefan Blom

Formal Methods and Tools, Faculty of EEMCS
University of Twente, The Netherlands
{d.guck,m.timmer,s.c.c.blom}@utwente.nl

This presentation introduces the Markov Reward Automaton (MRA), an extension of the Markov automaton that allows the modelling of systems incorporating *rewards* in addition to nondeterminism, discrete probabilistic choice and continuous stochastic timing. Our models support both rewards that are acquired instantaneously when taking certain transitions (action rewards) and rewards that are based on the duration that certain conditions hold (state rewards).

In addition to introducing the MRA model, we extend the process-algebraic language MAPA to easily specify MRAs. Also, we provide algorithms for computing the expected reward until reaching one of a certain set of goal states, as well as the long-run average reward. We extended the MAMA tool chain (consisting of the tools SCOOP and IMCA) to implement the reward extension of MAPA and these algorithms.

1 Introduction

The Markov automaton [7] is a novel formalism for modelling systems incorporating nondeterminism, discrete probabilistic choice as well as continuous stochastic timing. They provide a well-defined semantics for generalised stochastic Petri nets (GSPNs) [6], dynamic fault trees [3] and the domain-specific language AADL [4]. Recent work demonstrated that they are suitable for modelling and analysing distributed algorithms such as a leader election protocol, performance models such as a polling system and even hardware models such as a processor grid [15].

Notions of strong, weak and branching bisimulation have been defined to equate behaviourally equivalent Markov automata [7, 13, 5, 15], and the process algebra MAPA has been defined for easily specifying large Markov automata in a concise manner [16]. Several types of reduction techniques [12, 17] have been defined for the MAPA language and implemented in the tool SCOOP, optimising specifications to decrease the state space of the corresponding Markov automaton while staying bisimilar [14, 9]. This way, Markov automata can be generated efficiently in a direct way (as opposed to first generating a large model and then reducing), thus partly circumventing the omnipresent state space explosion. The tool IMCA [8, 9] was developed to analyse the concrete Markov automata that are generated by SCOOP. It includes algorithms for computing time-bounded reachability probabilities, expected times and long-run averages for sets of goal states within an MA.

While the framework in place already works well for computing probabilities and expected durations until certain events, it did not yet support *rewards* or *costs*. Such extensions have shown valuable in the past, for instance leading to the tool MRMC [11] for model checking reward-based properties over CTMCs [10] and DTMCs [1] with rewards.

*This work has been supported by the NWO project SYRUP (612.063.817), by the STW-ProRail partnership program ExploRail under the project ArRangeer (12238), and by the DFG/NWO bilateral project ROCKS (DN 63-257).

Submitted to:
QAPL 2014

2 Approach

We extend the formalism of Markov automata with two reward functions: an *action-reward function* and a *state-reward function*. The action-reward function assigns a rational number to each transition, representing an instantaneous reward that is obtained directly when taking that transition[1]. The state-reward function assigns a rational number to each state, representing a reward that is obtained over time during a stay of one time unit in that state.

We lifted most of the existing framework for efficiently modelling, generating and analysing Markov automata to the realm of rewards. Three main improvements were made: (1) the MAPA language was extended to include rewards, (2) the SCOOP tool including most of its reduction techniques was extended to efficiently generate MRAs, and (3) the IMCA tool was extended with algorithms for computing expected and long-run rewards. We discuss these extensions in more detail below. The only aspect that was not generalised yet was confluence reduction. It requires a generalised notion of weak or branching bisimulation that is beyond the scope of this paper and left for future work.

2.1 Modelling MRAs with MAPA

The MAPA language is based on a small core of process operators, modelling nondeterministic choice, probabilistic choice and stochastic delay. Additionally, it supports multiple processes to be defined, each having its own set of data parameters. Processes can call each other sequentially or be composed in parallel. Additionally, the data parameters of a process can be used for conditionally restricting behaviour and for parameterising actions. We refer to [16] for the detailed syntax and semantics.

We extended the MAPA language in two ways: (1) actions can now be decorated with a numerical expression to generate action-based rewards, and (2) state-based rewards can now be modelled by providing pairs of conditions and numerical expressions — to each state that satisfies one of these conditions, the corresponding reward is assigned. All conditions and numerical expressions may depend on one or more of a process' data variables, allowing rewards to be modelled concisely. If a state satisfies multiple conditions, the sum of their rewards is taken.

Example 2.1. Consider a salesman that has a certain hourly rate and a bonus for every sale he makes. He can concisely be modelled by the following MAPA specification (with the extensions underlined).

```
Salesman(hourlyRate : {10..20}, working : Boolean) =
     not(working)  ⟹  startWorking · Salesman(working := true )
   +     working   ⟹    (0.2) · stopWorking · Salesman(working := false)
                       + (1.0) · sell@10     · Salesman
stateReward working -> hourlyRate
init Salesman(15, false)
```

Note that we included an action-based reward of 10 for the action `sell` and a state reward equal to the hourly rate for each state in which the variable `working` is set to `true`. The parenthesized rational numbers indicate Markovian rates; a sale is made on average once per time unit, while the salesman on average works $\frac{1}{0.2} = 5$ time units.

[1] In the formalism of Markov automata there may be several transitions from a state all having the same action label. We allow these to have distinct rewards. Hence, the term *transition-reward function* would have made more sense, but for historical reasons we decided to stick with the more common terminology of *action rewards*.

2.2 Generating MRAs with SCOOP

The framework implemented by SCOOP was extended with rewards relatively easily. The action-based rewards are stored as part of the transitions, and the state-based rewards are represented internally by self-loops. During state space generation (constructing an IMCA input file), these self-loops are removed again and the state rewards are incorporated in the output.

We generalised most of the reduction techniques implemented by SCOOP: constant elimination, summation elimination, expression simplification and dead variable reduction. These techniques mainly reduce MAPA specifications by either omitting or resetting variables, based on their value (temporarily) not being relevant anymore. We updated the heuristics for checking this relevance by including the use of variables in rewards. Additionally, at this point we excluded confluence reduction from being used in the presence of rewards, since the correctness of this technique relies branching bisimulation — a concept that has not yet been defined for MRAs.

2.3 Analysing MRAs with IMCA

The analysis capabilities of IMCA are extended to deal with MRAs. It is possible to compute the expected reward — the expected reward gained until reaching a set of goal states — as well as the expected long-run average reward — the expected average reward gained in a set of goal states in the long run.

The algorithm for expected reward is an extension of the existing expected time computation [9]. The basic idea is to accumulate the action and state rewards until a goal state is reached. Therefore, we distinguish between probabilistic and Markovian states. For probabilistic states we have to consider the reward of an action when resolving a nondeterministic choice and for Markovian states the sojourn time must be weighted with the state reward.

The algorithm for the expected long-run average reward is an extension of the long-run average computation [9]. The main idea here is to not only consider the time spent in a state, but also the reward gained by staying in that state, as well as the impact of taking a specific action. Therefore, the cost function of the long-run average computation must be adjusted, such that we consider the average reward in the goal states against the average time spent in all states in the long run. Additional, we can give the long-run reward ratio by comparing the average reward in the goal states against the average reward in all states in the long run.

3 Conclusions and Challenges

We extended a framework for efficient modelling, generation and analysis of Markov automata with state-based and action-based rewards. We generalised the formal model, as well as the MAPA language and most of its reduction techniques. Also, we implemented algorithms for computing expected rewards as well as long-run average rewards.

Future work may focus on case studies demonstrating the benefits of our extensions. Additionally, it would be useful to define notions of weak and branching bisimulation for MRAs and generalise confluence reduction for an even more efficient framework for working with rewards. We are also currently working on incorporating SCOOP in the LTSmin toolset [2]. This tool has extensive functionality for very efficient state space generation, optimising memory usage and employing multi-core and distributed model checking techniques. Preliminary results indicate significant speedups when generating MRAs.

References

[1] S. Andova, H. Hermanns & J.-P. Katoen (2003): *Discrete-Time Rewards Model-Checked*. In: *Proc. of the 1st Int. Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS)*, *LNCS* 2791, Springer, pp. 88–104, doi:10.1007/978-3-540-40903-8_8.

[2] S. C. C. Blom, J. C. van de Pol & M. Weber (2010): *LTSmin: Distributed and Symbolic Reachability*. In: *Proc. of the 22nd Int. Conf. on Computer Aided Verification (CAV)*, *LNCS* 6174, Springer, pp. 354–359.

[3] H. Boudali, P. Crouzen & M. I. A. Stoelinga (2010): *A Rigorous, Compositional, and Extensible Framework for Dynamic Fault Tree Analysis*. *IEEE Transactions on Dependable and Secure Computing* 7(2), pp. 128–143, doi:10.1109/TDSC.2009.45.

[4] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll & M. Roveri (2011): *Safety, Dependability and Performance Analysis of Extended AADL Models*. *The Computer Journal* 54(5), pp. 754–775, doi:10.1093/comjnl/bxq024.

[5] Y. Deng & M. Hennessy (2013): *On the semantics of Markov automata*. *Information and Computation* 222, pp. 139–168, doi:10.1016/j.ic.2012.10.010.

[6] C. Eisentraut, H. Hermanns, J.-P. Katoen & L. Zhang (2013): *A Semantics for Every GSPN*. In: *Proc. of the 34th Int. Conf. on Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN)*, *LNCS* 7927, Springer, pp. 90–109, doi:10.1007/978-3-642-38697-8_6.

[7] C. Eisentraut, H. Hermanns & L. Zhang (2010): *On Probabilistic Automata in Continuous Time*. In: *Proc. of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, pp. 342–351, doi:10.1109/LICS.2010.41.

[8] D. Guck, T. Han, J.-P. Katoen & M. R. Neuhäußer (2012): *Quantitative Timed Analysis of Interactive Markov Chains*. In: *Proc. of the 4th Int. NASA Formal Methods Symposium (NFM)*, *LNCS* 7226, Springer, pp. 8–23, doi:10.1007/978-3-642-28891-3_4.

[9] D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen & M. Timmer (2013): *Modelling, Reduction and Analysis of Markov Automata*. In: *Proc. of the 10th Int. Conf. on Quantitative Evaluation of Systems (QEST)*, *LNCS* 8054, Springer, pp. 55–71, doi:10.1007/978-3-642-40196-1_5.

[10] B. R. Haverkort, L. Cloth, H. Hermanns, J.-P. Katoen & C. Baier (2002): *Model Checking Performability Properties*. In: *Proc. of the 2002 Int. Conf. on Dependable Systems and Networks (DSN)*, IEEE Computer Society, pp. 103–112, doi:10.1109/DSN.2002.1028891.

[11] J.-P. Katoen, I. S. Zapreev, E. Moritz Hahn, H. Hermanns & D. N. Jansen (2011): *The ins and outs of the probabilistic model checker MRMC*. *Performance Evaluation* 68(2), pp. 90–104, doi:10.1016/j.peva.2010.04.001.

[12] J. C. van de Pol & M. Timmer (2009): *State Space Reduction of Linear Processes using Control Flow Reconstruction*. In: *Proc. of the 7th Int. Symposium on Automated Technology for Verification and Analysis (ATVA)*, *LNCS* 5799, Springer, pp. 54–68, doi:10.1007/978-3-642-04761-9_5.

[13] L. Song, L. Zhang & J. C. Godskesen (2012): *Late Weak Bisimulation for Markov Automata*. Technical Report, ArXiv e-prints. Available at `http://arxiv.org/abs/1202.4116`.

[14] M. Timmer (2011): *SCOOP: A Tool for SymboliC Optimisations of Probabilistic Processes*. In: *Proc. of the 8th Int. Conf. on Quantitative Evaluation of Systems (QEST)*, IEEE, pp. 149–150, doi:10.1109/QEST.2011.27.

[15] M. Timmer (2013): *Efficient Modelling, Generation and Analysis of Markov Automata*. Ph.D. thesis, University of Twente, doi:10.3990/1.9789036505925.

[16] M. Timmer, J.-P. Katoen, J. C. van de Pol & M. I. A. Stoelinga (2012): *Efficient Modelling and Generation of Markov Automata*. In: *Proc. of the 23rd Int. Conf. on Concurrency Theory (CONCUR)*, *LNCS* 7454, Springer, pp. 364–379, doi:10.1007/978-3-642-32940-1_26.

[17] M. Timmer, M. I. A. Stoelinga & J. C. van de Pol (2013): *Confluence Reduction for Markov Automata*. In: *Proc. of the 11th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS)*, *LNCS* 8053, Springer, pp. 243–257, doi:10.1007/978-3-642-40229-6_17.