Modelling, Reduction and Analysis of Markov Automata*

Dennis Guck[1,3], Hassan Hatefi[2], Holger Hermanns[2],
Joost-Pieter Katoen[1,3] and Mark Timmer[3]

[1] Software Modelling and Verification, RWTH Aachen University, Germany
[2] Dependable Systems and Software, Saarland University, Germany
[3] Formal Methods and Tools, University of Twente, The Netherlands

Abstract. Markov automata (MA) constitute an expressive continuous-time compositional modelling formalism. They appear as semantic backbones for engineering frameworks including dynamic fault trees, Generalised Stochastic Petri Nets, and AADL. Their expressive power has thus far precluded them from effective analysis by probabilistic (and statistical) model checkers, stochastic game solvers, or analysis tools for Petri net-like formalisms. This paper presents the foundations and underlying algorithms for efficient MA modelling, reduction using static analysis, and most importantly, quantitative analysis. We also discuss implementation pragmatics of supporting tools and present several case studies demonstrating feasibility and usability of MA in practice.

1 Introduction

Markov automata (MA, for short) have been introduced in [13] as a continuous-time version of Segala's (simple) probabilistic automata [26]. They are closed under parallel composition and hiding. An MA-transition is either labelled with an action, or with a positive real number representing the rate of a negative exponential distribution. An action transition leads to a discrete probability distribution over states. MA can thus model action transitions as in labelled transition systems, probabilistic branching, as well as delays that are governed by exponential distributions.

The semantics of MA has been recently investigated in quite some detail. Weak and strong (bi)simulation semantics have been presented in [13,12], whereas it is shown in [10] that weak bisimulation provides a sound and complete proof methodology for reduction barbed congruence. A process algebra with data for the efficient modelling of MA, accompanied with some reduction techniques using static analysis, has been presented in [29]. Although the MA model raises several challenging theoretical issues, both from a semantical and from an analysis point of view, our main interest is in their practical applicability. As MA extend

* This work is funded by the EU FP7-projects MoVeS, SENSATION and MEALS, the DFG-NWO bilateral project ROCKS, the NWO projects SYRUP (grant 612.063.817), the STW project ArRangeer (grant 12238), and the DFG Sonderforschungsbereich AVACS.

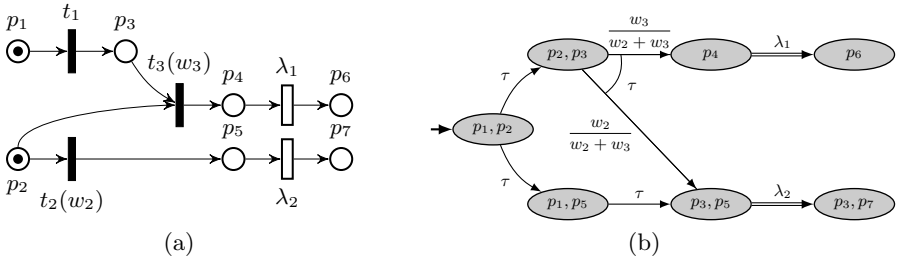

Fig. 1. (a) Confused GSPN, see [22, Fig. 21] with partial weights and (b) its MA semantics

Hermanns' interactive Markov chains (IMCs) [18], they inherit IMC application domains, ranging from GALS hardware designs [6] and dynamic fault trees [3] to the standardised modeling language AADL [4,17]. The added feature of probabilistic branching yields a natural operational model for generalised stochastic Petri nets (GSPNs) [23] and stochastic activity networks (SANs) [24], both popular modelling formalisms for performance and dependability analysis. Let us briefly motivate this by considering GSPNs. Whereas in SPNs all transitions are subject to a random delay, GSPNs also incorporate immediate transitions, transitions that happen instantaneously. The traditional GSPN semantics yields a continuous-time Markov chain (CTMC), i.e., an MA without action transitions, but is restricted to GSPNs that do not exhibit non-determinism. Such "well-defined" GSPNs occur if the net is free of confusion. It has recently been detailed in [19,11] that MA are a natural semantic model for *every* GSPN. Without going into the technical details, consider the confused GSPN in Fig. 1(a). This net is confused, as the transitions t_1 and t_2 are not in conflict, but firing transition t_1 leads to a conflict between t_2 and t_3, which does not occur if t_2 fires before t_1. Transitions t_2 and t_3 are weighted so that in a marking $\{p_2, p_3\}$ in which both transitions are enabled, t_2 fires with probability $\frac{w_2}{w_2+w_3}$ and t_3 with its complement probability. Classical GSPN semantics and analysis algorithms cannot cope with this net due to the presence of confusion (i.e., non-determinism). Figure 1(b) depicts the MA semantics of this net. Here, states correspond to sets of net places that contain a token. In the initial state, there is a non-deterministic choice between the transitions t_1 and t_2. Note that the presence of weights is naturally represented by discrete probabilistic branching. One can show that for confusion-free GSPNs, the classical semantics and the MA semantics are weakly bisimilar [11].

This paper focuses on the quantitative analysis of MA—and thus (possibly confused) GSPNs and probabilistic AADL error models. We present analysis algorithms for three objectives: expected time, long-run average, and timed (interval) reachability. As the model exhibits non-determinism, we focus on maximal and minimal values for all three objectives. We show that expected time and long-run average objectives can be efficiently reduced to well-known problems on

MDPs such as stochastic shortest path, maximal end-component decomposition, and long-run ratio objectives. This generalizes (and slightly improves) the results reported in [14] for IMCs to MA. Secondly, we present a discretisation algorithm for timed interval reachability objectives which extends [33]. Finally, we present the MAMA tool-chain, an easily accessible publicly available tool chain [1] for the specification, mechanised simplification—such as confluence reduction [31], a form of on-the-fly partial-order reduction—and quantitative evaluation of MA. We describe the overall architectural design, as well as the tool components, and report on empirical results obtained with MAMA on a selection of case studies taken from different domains. The experiments give insight into the effectiveness of our reduction techniques and demonstrate that MA provide the basis of a very expressive stochastic timed modelling approach without sacrificing the ability of time and memory efficient numerical evaluation.

Organisation of the paper. After introducing Markov Automata in Section 2, we discuss a fully compositional modelling formalism in Section 3. Section 4 considers the evaluation of expected time properties. Section 5 discusses the analysis of long run properties, and Section 6 focusses on reachability properties with time interval bounds. Implementation details of our tool as well as experimental results are discussed in detail in Section 7. Section 8 concludes the paper. Due to space constraints, we refer to [15] for the proofs of our main results.

2 Preliminaries

Markov automata. An MA is a transition system with two types of transitions: probabilistic (as in PAs) and Markovian transitions (as in CTMCs). Let Act be a universe of actions with internal action $\tau \in Act$, and $\mathsf{Distr}(S)$ denote the set of distribution functions over the countable set S.

Definition 1 (Markov automaton). *A* Markov automaton (MA) *is a tuple* $\mathcal{M} = (S, A, \longrightarrow, \Longrightarrow, s_0)$ *where* S *is a nonempty, finite set of states with* initial state $s_0 \in S$, $A \subseteq Act$ *is a finite set of actions, and*

- $\longrightarrow \subseteq S \times A \times \mathsf{Distr}(S)$ *is the* probabilistic *transition relation, and*
- $\Longrightarrow \subseteq S \times \mathbb{R}_{>0} \times S$ *is the* Markovian *transition relation.*

We abbreviate $(s, \alpha, \mu) \in \longrightarrow$ by $s \xrightarrow{\alpha} \mu$ and $(s, \lambda, s') \in \Longrightarrow$ by $s \overset{\lambda}{\Longrightarrow} s'$. An MA can move between states via its probabilistic and Markovian transitions. If $s \xrightarrow{a} \mu$, it can leave state s by executing the action a, after which the probability to go to some state $s' \in S$ is given by $\mu(s')$. If $s \overset{\lambda}{\Longrightarrow} s'$, it moves from s to s' with rate λ, except if s enables a τ-labelled transition. In that case, the MA will always take such a transition and never delays. This is the *maximal progress* assumption [13]. The rationale behind this assumption is that internal transitions are not subject to interaction and thus can happen immediately, whereas the probability for a Markovian transition to happen immediately is zero. As an example of an MA, consider Fig. 2.

[1] Stand-alone download as well as web-based interface available from `http://fmt.cs.utwente.nl/~timmer/mama`.

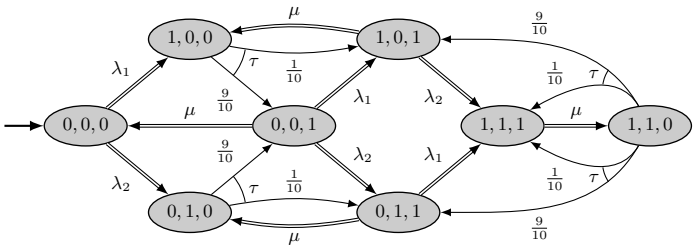

Fig. 2. A queueing system, consisting of a server and two stations. The two stations have incoming requests with rates λ_1, λ_2, which are stored until fetched by the server. If both stations contain a job, the server chooses nondeterministically (in state (1,1,0)). Jobs are processed with rate μ, and when polling a station, there is a $\frac{1}{10}$ probability that the job is erroneously kept in the station after being fetched. Each state is represented as a tuple (s_1, s_2, j), with s_i the number of jobs in station i, and j the number of jobs in the server. For simplicity we assume that each component can hold at most one job.

We briefly explain the semantics of Markovian transitions. For a state with Markovian transitions, let $\mathbf{R}(s, s') = \sum\{\lambda \mid s \overset{\lambda}{\Longrightarrow} s'\}$ be the total rate to move from state s to state s', and let $E(s) = \sum_{s' \in S} \mathbf{R}(s, s')$ be the total outgoing rate of s. If $E(s) > 0$, a competition between the transitions of s exists. Then, the probability to move from s to state s' within d time units is

$$\frac{\mathbf{R}(s, s')}{E(s)} \cdot \left(1 - e^{-E(s)d}\right).$$

This asserts that after a delay of at most d time units (second factor), the MA moves to a direct successor state s' with probability $\mathbf{P}(s, s') = \frac{\mathbf{R}(s,s')}{E(s)}$.

Paths. A path in an MA is an infinite sequence $\pi = s_0 \xrightarrow{\sigma_0, \mu_0, t_0} s_1 \xrightarrow{\sigma_1, \mu_1, t_1} \ldots$ with $s_i \in S$, $\sigma_i \in \mathit{Act} \cup \{\bot\}$, and $t_i \in \mathbb{R}_{\geq 0}$. For $\sigma_i \in \mathit{Act}$, $s_i \xrightarrow{\sigma_i, \mu_i, t_i} s_{i+1}$ denotes that after residing t_i time units in s_i, the MA has moved via action σ_i to s_{i+1} with probability $\mu_i(s_{i+1})$. Instead, $s_i \xrightarrow{\bot, \mu_i, t_i} s_{i+1}$ denotes that after residing t_i time units in s, a Markovian transition led to s_{i+1} with probability $\mu_i(s_{i+1}) = \mathbf{P}(s_i, s_{i+1})$. For $t \in \mathbb{R}_{\geq 0}$, let $\pi@t$ denote the sequence of states that π occupies at time t. Due to instantaneous action transitions, $\pi@t$ need not be a single state, as an MA may occupy various states at the same time instant. Let *Paths* denote the set of infinite paths. The time elapsed along the path π is $\sum_{i=0}^{\infty} t_i$. Path π is Zeno whenever this sum converges. As the probability of a Zeno path in an MA that only contains Markovian transitions is zero [1], an MA is non-Zeno if and only if no SCC with only probabilistic states is reachable with positive probability. In the rest of this paper, we assume MAs to be non-Zeno.

Policies. Nondeterminism occurs when there is more than one action transition emanating from a state. To define a probability space, the choice is resolved using *policies*. A policy (ranged over by D) is a measurable function which yields

for each finite path ending in state s a probability distribution over the set of enabled actions in s. The information on basis of which a policy may decide yields different classes of policies. Let GM denote the class of the general measurable policies. A stationary deterministic policy is a mapping $D\colon PS \to Act$ where PS is the set of states with outgoing probabilistic transitions; such policies always take the same decision in a state s. A time-abstract policy may decide on basis of the states visited so far, but not on their timings; we use TA denote this class. For more details on different classes of policies (and their relation) on models such as MA, we refer to [25]. Using a cylinder set construction we obtain a σ-algebra of subsets of *Paths*; given a policy D and an initial state s, a measurable set of paths is equipped with probability measure $\Pr_{s,D}$.

Stochastic shortest path (SSP) problems. As some objectives on MA are reduced to SSP problems, we briefly introduce them. A non-negative SSP problem is an MDP $(S, Act, \mathbf{P}, s_0)$ with set $G \subseteq S$ of goal states, cost function $c\colon S \setminus G \times Act \to \mathbb{R}_{\geq 0}$ and terminal cost function $g\colon G \to \mathbb{R}_{\geq 0}$. The accumulated cost along a path π through the MDP before reaching G, denoted $C_G(\pi)$, is $\sum_{j=0}^{k-1} c(s_j, \alpha_j) + g(s_k)$ where k is the state index of reaching G. Let $cR^{\min}(s, \Diamond G)$ denote the minimum expected cost reachability of G in the SSP when starting from s. This expected cost can be obtained by solving an LP problem [2].

3 Efficient modeling of Markov automata

As argued in the introduction, MA can be used as semantical model for various modeling formalisms. We show this for the process-algebraic specification language MAPA (MA Process Algebra) [29]. This language is rather expressive and supports several reductions techniques for MA specifications. In fact, it turns out to be beneficial to map a language (like GSPNs) to MAPA so as to profit from these reductions. We present the syntax and a brief informal overview of the reduction techniques.

The Markov Automata Process Algebra. MAPA relies on external mechanisms for evaluating expressions, able to handle boolean and real-valued expressions. We assume that any variable-free expression in this language can be evaluated. Our tool uses a simple and intuitive fixed data language that includes basic arithmetic and boolean operators, conditionals, and dynamic lists. For expression t in our data language and vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{d} = (d_1, \ldots, d_n)$, let $t[\boldsymbol{x} := \boldsymbol{d}]$ denote the result of substituting every x_i in t by d_i.

A *MAPA specification* consists of a set of uniquely-named *processes* X_i, each defined by a *process equation* $X_i(\boldsymbol{x_i} : \boldsymbol{D_i}) = p_i$. In such an equation, $\boldsymbol{x_i}$ is a vector of process variables with type $\boldsymbol{D_i}$, and p_i is a *process term* specifying the behaviour of X_i. Additionally, each specification has an *initial process* $X_j(\boldsymbol{t})$. We abbreviate $X((x_1, \ldots, x_n) : (D_1 \times \cdots \times D_n))$ by $X(x_1 : D_1, \ldots, x_n : D_n)$. A MAPA *process term* adheres to the grammar:

$$p ::= Y(\boldsymbol{t}) \;\mid\; c \Rightarrow p \;\mid\; p + p \;\mid\; \textstyle\sum_{\boldsymbol{x}:\boldsymbol{D}} p \;\mid\; a(\boldsymbol{t}) \textstyle\sum_{\boldsymbol{x}:\boldsymbol{D}} f : p \;\mid\; (\lambda) \cdot p$$

Here, Y is a process name, $\boldsymbol{t}$ a vector of expressions, c a boolean expression, $\boldsymbol{x}$ a vector of variables ranging over a finite type $\boldsymbol{D}$, $a \in Act$ a (parameterised)

constant $queueSize = 10, nrOfJobTypes = 3$
type $Stations = \{1, 2\},\ Jobs = \{1, \ldots, nrOfJobTypes\}$

$Station(i : Stations, q : \text{Queue}, size : \{0..queueSize\})$
$= size < queueSize \Rightarrow (2i+1) \cdot \sum_{j:Jobs} arrive(j) \cdot Station(i, \text{enqueue}(q, j), size + 1)$
$+\ size > 0 \quad \Rightarrow deliver(i, \text{head}(q)) \sum_{k \in \{1,9\}} \frac{k}{10} : k = 1 \Rightarrow Station(i, q, size)$
$+\ k = 9 \Rightarrow Station(i, \text{tail}(q), size - 1)$

$Server = \sum_{n:Stations} \sum_{j:Jobs} poll(n, j) \cdot (2 * j) \cdot finish(j) \cdot Server$

$\gamma(poll, deliver) = copy$ // actions *poll* and *deliver* synchronise and yield action *copy*
$System = \tau_{\{copy, arrive, finish\}}(\partial_{\{poll, deliver\}}(Station(1, \text{empty}, 0) \,||\, Station(2, \text{empty}, 0) \,||\, Server))$

Fig. 3. MAPA specification of a polling system.

atomic action, f a real-valued expression yielding a value in $[0, 1]$, and λ an expression yielding a positive real number. Note that, if $|\boldsymbol{x}| > 1$, $\boldsymbol{D}$ is a Cartesian product, as for instance in $\sum_{(m,i):\{m_1,m_2\}\times\{1,2,3\}} \text{send}(m, i) \ldots$. In a process term, $Y(\boldsymbol{t})$ denotes *process instantiation*, where $\boldsymbol{t}$ instantiates Y's process variables (allowing recursion). The term $c \Rightarrow p$ behaves as p if the *condition* c holds, and cannot do anything otherwise. The $+$ operator denotes *nondeterministic choice*, and $\sum_{\boldsymbol{x}:\boldsymbol{D}} p$ a *nondeterministic choice over data type* $\boldsymbol{D}$. The term $a(\boldsymbol{t}) \sum_{\boldsymbol{x}:\boldsymbol{D}} f : p$ performs the action $a(\boldsymbol{t})$ and then does a *probabilistic choice* over $\boldsymbol{D}$. It uses the value $f[\boldsymbol{x} := \boldsymbol{d}]$ as the probability of choosing each $\boldsymbol{d} \in \boldsymbol{D}$. We write $a(\boldsymbol{t}) \cdot p$ for the action $a(\boldsymbol{t})$ that goes to p with probability 1. Finally, $(\lambda) \cdot p$ can behave as p after a delay, determined by an exponential distribution with rate λ. Using MAPA processes as basic building blocks, the language also supports the modular construction of large systems via top-level parallelism (denoted $||$), encapsulation (denoted ∂), hiding (denoted τ), and renaming (denoted γ), cf. [30, App. B]. The operational semantics of a MAPA specification yields an MA; for details we refer to [29].

Example 1. Fig. 3 depicts the MAPA specification [29] of a polling system—inspired by [27]—which generalised the system of Fig. 2. Now, there are incoming requests of 3 possible types, each of which has a different service rate. Additionally, the stations store these in a queue of size 10. □

Reduction techniques. To simplify state space generation and reduction, we use a linearised format referred to as MLPPE (Markovian linear probabilistic process equation). In this format, there is precisely one process consisting of a nondeterministic choice between a set of summands. Each summand can contain a nondeterministic choice, followed by a condition, and either an interactive action with a probabilistic choice (determining the next state) or a rate and a next state. Every MAPA specification can be translated efficiently into an MLPPE [29] while preserving strong bisimulation. On MLPPEs two types of reduction techniques have been defined: simplifications and state space reductions:

– *Maximal progress reduction* removes Markovian transitions from states also having τ-transitions. It is more efficient to perform this on MLPPEs than

on the initial MAPA specification. We use heuristics (as in [32]) to omit all Markovian summands in presence of internal non-Markovian ones.
- *Constant elimination* [20] replaces MLPPE parameters that remain constants by their initial value.
- *Expression simplification* [20] evaluates functions for which all parameters are constants and applies basic laws from logic.
- *Summation elimination* [20] removes unnecessary summations, transforming e.g., $\sum_{d:\mathbb{N}} d = 5 \Rightarrow \mathit{send}(d) \cdot X$ to $\mathit{send}(5) \cdot X$, $\sum_{d:\{1,2\}} a \cdot X$ to $a \cdot X$, and $\sum_{d:D}(\lambda) \cdot X$ to $(|D| \times \lambda) \cdot X$, to preserve the total rate to X.
- *Dead-variable reduction* [32] detects states in which the value of some data variable d is irrelevant. This is the case if d will be overwritten before being used for all possible futures. Then, d is reset to its initial value.
- *Confluence reduction* [31] detects spurious nondeterminism, resulting from parallel composition. It denotes a subset of the probabilistic transitions of a MAPA specification as confluent, meaning that they can safely be given priority if enabled together with other transitions.

4 Expected time objectives

The actions of an MA are only used for composing models from smaller ones. For the analysis of MA, they are not relevant and we may safely assume that all actions are internal[2]. Due to the maximal progress assumption, the outgoing transitions of a state s are all either probabilistic transitions or Markovian transitions. Such states are called probabilistic and Markovian, respectively; let $PS \subseteq S$ and $MS \subseteq S$ denote these sets.

Let $\mathcal{M}$ be an MA with state space S and $G \subseteq S$ a set of goal states. Define the (extended) random variable $V_G \colon \mathit{Paths} \to \mathbb{R}^{\infty}_{\geq 0}$ as the elapsed time before first visiting some state in G. That is, for an infinite path $\pi = s_0 \xrightarrow{\sigma_0,\mu_0,t_0} s_1 \xrightarrow{\sigma_1,\mu_1,t_1} \cdots$, let $V_G(\pi) = \min\{t \in \mathbb{R}_{\geq 0} \mid G \cap \pi@t \neq \emptyset\}$ where $\min(\emptyset) = +\infty$. (With slight abuse of notation we use $\pi@t$ as the set of states occurring in the sequence $\pi@t$.) The minimal expected time to reach G from $s \in S$ is defined by

$$eT^{\min}(s, \Diamond G) \;=\; \inf_D \mathbb{E}_{s,D}(V_G) \;=\; \inf_D \int_{Paths} V_G(\pi)\ \Pr_{s,D}(d\pi)$$

where D is a policy on $\mathcal{M}$. Note that by definition of V_G, only the amount of time before entering the first G-state is relevant. Hence, we may turn all G-states into absorbing Markovian states without affecting the expected time reachability. In the remainder we assume all goal states to be absorbing.

Theorem 1. *The function $eT^{\min}$ is a fixpoint of the Bellman operator*

$$[L(v)](s) = \begin{cases} \dfrac{1}{E(s)} + \displaystyle\sum_{s' \in S} \mathbf{P}(s,s') \cdot v(s') & \text{if } s \in MS \setminus G \\ \min\limits_{\alpha \in Act(s)} \displaystyle\sum_{s' \in S} \mu^s_\alpha(s') \cdot v(s') & \text{if } s \in PS \setminus G \\ 0 & \text{if } s \in G. \end{cases}$$

[2] Like in the MAPA specification of the queueing system in Fig. 3, the actions used in parallel composition are explicitly turned into internal actions by hiding.

For a goal state, the expected time obviously is zero. For a Markovian state $s \notin G$, the minimal expected time to G is the expected sojourn time in s plus the expected time to reach G via its successor states. For a probabilistic state, an action is selected that minimises the expected reachability time according to the distribution μ^s_α corresponding to α. The characterization of $eT^{\min}(s, \Diamond G)$ in Thm. 1 allows us to reduce the problem of computing the minimum expected time reachability in an MA to a non-negative SSP problem [2,9].

Definition 2 (SSP for minimum expected time reachability). *The SSP of MA $\mathcal{M} = (S, Act, \to, \Longrightarrow, s_0)$ for the expected time reachability of $G \subseteq S$ is $\mathsf{ssp}_{et}(\mathcal{M}) = (S, Act \cup \{\bot\}, \mathbf{P}, s_0, G, c, g)$ where $g(s) = 0$ for all $s \in G$ and*

$$\mathbf{P}(s,\sigma,s') = \begin{cases} \frac{\mathbf{R}(s,s')}{E(s)} & \text{if } s \in MS, \sigma = \bot \\ \mu^s_\sigma(s') & \text{if } s \in PS, s \xrightarrow{\sigma} \mu^s_\sigma \\ 0 & \text{otherwise, and} \end{cases} \quad c(s,\sigma) = \begin{cases} \frac{1}{E(s)} & \text{if } s \in MS \setminus G, \sigma = \bot \\ 0 & \text{otherwise.} \end{cases}$$

Terminal costs are zero. Transition probabilities are defined in the standard way. The reward of a Markovian state is its expected sojourn time, and zero otherwise.

Theorem 2. *For MA $\mathcal{M}$, $eT^{\min}(s, \Diamond G)$ equals $cR^{\min}(s, \Diamond G)$ in $\mathsf{ssp}_{et}(\mathcal{M})$.*

Thus here is a stationary deterministic policy on $\mathcal{M}$ yielding $eT^{\min}(s, \Diamond G)$. Moreover, the uniqueness of the minimum expected cost of an SSP [2,9] now yields that $eT^{\min}(s, \Diamond G)$ is the unique fixpoint of L (see Thm. 1). The uniqueness result enables the usage of standard solution techniques such as value iteration and linear programming to compute $eT^{\min}(s, \Diamond G)$. For maximal expected time objectives, a similar fixpoint theorem is obtained, and it can be proven that those objectives correspond to the maximal expected reward in the SSP problem defined above. In the above, we have assumed MA to not contain any Zeno cycle, i.e., a cycle solely consisting of probabilistic transitions. The above notions can all be extended to deal with such Zeno cycles, by, e.g., setting the minimal expected time of states in Zeno BSCCs that do not contain G-states to be infinite (as such states cannot reach G). Similarly, the maximal expected time of states in Zeno end components (that do not containg G-states) can be defined as ∞, as in the worst case these states will never reach G.

5 Long run objectives

Let $\mathcal{M}$ be an MA with state space S and $G \subseteq S$ a set of goal states. Let $\mathbf{1}_G$ be the characteristic function of G, i.e., $\mathbf{1}_G(s) = 1$ if and only if $s \in G$. Following the ideas of [8,21], the fraction of time spent in G on an infinite path π in $\mathcal{M}$ up to time bound $t \in \mathbb{R}_{\geq 0}$ is given by the random variable (r. v.) $A_{G,t}(\pi) = \frac{1}{t}\int_0^t \mathbf{1}_G(\pi@u)\, du$. Taking the limit $t \to \infty$, we obtain the r. v.

$$A_G(\pi) = \lim_{t\to\infty} A_{G,t}(\pi) = \lim_{t\to\infty} \frac{1}{t}\int_0^t \mathbf{1}_G(\pi@u)\, du.$$

The expectation of A_G for policy D and initial state s yields the corresponding long-run average time spent in G:

$$LRA^D(s,G) = \mathbb{E}_{s,D}(A_G) = \int_{Paths} A_G(\pi)\, \Pr_{s,D}(d\pi).$$

The minimum long-run average time spent in G starting from state s is then:

$$LRA^{\min}(s,G) \;=\; \inf_{D} \; LRA^{D}(s,G) \;=\; \inf_{D} \mathbb{E}_{s,D}(A_G).$$

For the long-run average analysis, we may assume w.l.o.g. that $G \subseteq MS$, as the long-run average time spent in any probabilistic state is always 0. This claim follows directly from the fact that probabilistic states are instantaneous, i.e. their sojourn time is 0 by definition. Note that in contrast to the expected time analysis, G-states cannot be made absorbing in the long-run average analysis. It turns out that stationary deterministic policies are sufficient for yielding minimal or maximal long-run average objectives.

In the remainder of this section, we discuss in detail how to compute the minimum long-run average fraction of time to be in G in an MA $\mathcal{M}$ with initial state s_0. The general idea is the following three-step procedure:

1. Determine the maximal end components[3] $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ of MA $\mathcal{M}$.
2. Determine $LRA^{\min}(G)$ in maximal end component $\mathcal{M}_j$ for all $j \in \{1, \ldots, k\}$.
3. Reduce the computation of $LRA^{\min}(s_0, G)$ in MA $\mathcal{M}$ to an SSP problem.

The first phase can be performed by a graph-based algorithm [7,5], whereas the last two phases boil down to solving LP problems.

Unichain MA. We first show that for unichain MA, i.e., MA that under any stationary deterministic policy yield a strongly connected graph structure, computing $LRA^{\min}(s,G)$ can be reduced to determining long-ratio objectives in MDPs. Let us first explain such objectives. Let $M = (S, Act, \mathbf{P}, s_0)$ be an MDP. Assume w.l.o.g. that for each state s in M there exists $\alpha \in Act$ such that $\mathbf{P}(s,\alpha,s') > 0$. Let $c_1, c_2 \colon S \times (Act \cup \{\bot\}) \to \mathbb{R}_{\geq 0}$ be cost functions. The operational interpretation is that a cost $c_1(s,\alpha)$ is incurred when selecting action α in state s, and similar for c_2. Our interest is the *ratio* between c_1 and c_2 along a path. The *long-run ratio* $\mathcal{R}$ between the accumulated costs c_1 and c_2 along the infinite path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots$ in the MDP M is defined by[4]:

$$\mathcal{R}(\pi) \;=\; \lim_{n\to\infty} \frac{\sum_{i=0}^{n-1} c_1(s_i,\alpha_i)}{\sum_{j=0}^{n-1} c_2(s_j,\alpha_j)}.$$

The minimum long-run ratio objective for state s of MDP M is defined by:

$$R^{\min}(s) \;=\; \inf_{D} \mathbb{E}_{s,D}(\mathcal{R}) \;=\; \inf_{D} \sum_{\pi \in Paths} \mathcal{R}(\pi) \cdot \Pr_{s,D}(\pi).$$

[3] A sub-MA of MA $\mathcal{M}$ is a pair (S', K) where $S' \subseteq S$ and K is a function that assigns to each $s \in S'$ a non-empty set of actions such that for all $\alpha \in K(s)$, $s \xrightarrow{\alpha} \mu$ with $\mu(s') > 0$ or $s \overset{\lambda}{\Longrightarrow} s'$ imply $s' \in S'$. An end component is a sub-MA whose underlying graph is strongly connected; it is maximal w.r.t. K if it is not contained in any other end component (S'', K).

[4] In our setting, $\mathcal{R}(\pi)$ is well-defined as the cost functions c_1 and c_2 are obtained from non-Zeno MA. Thus for any infinite path π, $c_2(s_j,\alpha_j) > 0$ for some index j.

Here, *Paths* is the set of paths in the MDP, D an MDP-policy, and Pr the probability mass on MDP-paths. From [7], it follows that $R^{\min}(s)$ can be obtained by solving the following LP problem with real variables k and x_s for each $s \in S$: Maximize k subject to:

$$x_s \leq c_1(s,\alpha) - k \cdot c_2(s,\alpha) + \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot x_{s'} \quad \text{for each } s \in S, \alpha \in Act.$$

We now transform an MA into an MDP with 2 cost functions as follows.

Definition 3 (From MA to two-cost MDPs). *Let $\mathcal{M} = (S, Act, \rightarrow, \Longrightarrow, s_0)$ be an MA and $G \subseteq S$ a set of goal states. The MDP $\mathsf{mdp}(\mathcal{M}) = (S, Act \cup \{\bot\}, \mathbf{P}, s_0)$ with cost functions c_1 and c_2, where $\mathbf{P}$ is defined as in Def. 2, and*

$$c_1(s,\sigma) = \begin{cases} \frac{1}{E(s)} & \text{if } s \in MS \cap G \wedge \sigma = \bot \\ 0 & \text{otherwise,} \end{cases} \qquad c_2(s,\sigma) = \begin{cases} \frac{1}{E(s)} & \text{if } s \in MS \wedge \sigma = \bot \\ 0 & \text{otherwise.} \end{cases}$$

Observe that cost function c_2 keeps track of the average residence time in state s whereas c_1 only does so for states in G.

Theorem 3. *For unichain MA $\mathcal{M}$, $LRA^{\min}(s,G)$ equals $R^{\min}(s)$ in $\mathsf{mdp}(\mathcal{M})$.*

To summarise, computing the minimum long-run average fraction of time that is spent in some goal state in $G \subseteq S$ in an unichain MA $\mathcal{M}$ equals the minimum long-run ratio objective in an MDP with two cost functions. The latter can be obtained by solving an LP problem. Observe that for any two states s, s' in a unichain MA, $LRA^{\min}(s,G)$ and $LRA^{\min}(s',G)$ coincide. We therefore omit the state and simply write $LRA^{\min}(G)$ when considering unichain MA.

Arbitrary MA. Let $\mathcal{M}$ be an MA with initial state s_0 and maximal end components $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ for $k > 0$ where MA $\mathcal{M}_j$ has state space S_j. Note that each $\mathcal{M}_j$ is a unichain MA. Using this decomposition of $\mathcal{M}$ into maximal end components, we obtain the following result:

Theorem 4. [5] *For MA $\mathcal{M} = (S, Act, \rightarrow, \Longrightarrow, s_0)$ with MECs $\{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ with state spaces $S_1, \ldots, S_k \subseteq S$, and set of goal states $G \subseteq S$:*

$$LRA^{\min}(s_0,G) = \inf_D \sum_{j=1}^{k} LRA_j^{\min}(G) \cdot \Pr^D(s_0 \models \Diamond\Box S_j),$$

where $\Pr^D(s_0 \models \Diamond\Box S_j)$ is the probability to eventually reach and continuously stay in some state in S_j from s_0 under policy D and $LRA_j^{\min}(G)$ is the LRA of $G \cap S_j$ in unichain MA $\mathcal{M}_j$.

Computing minimal LRA for arbitrary MA is now reducible to a non-negative SSP problem. This proceeds as follows. In MA $\mathcal{M}$, we replace each maximal end component $\mathcal{M}_j$ by two fresh states q_j and u_j. Intuitively, q_j represents $\mathcal{M}_j$ whereas u_j represents a decision state. State u_j has a transition to q_j and contains all probabilistic transitions leaving S_j. Let U denote the set of u_j states and Q the set of q_j states.

[5] This theorem corrects a small flaw in the corresponding theorem for IMCs in [14].

Definition 4 (SSP for long run average). *The SSP of MA $\mathcal{M}$ for the LRA in $G \subseteq S$ is $\mathsf{ssp}_{lra}(\mathcal{M}) = (S \setminus \bigcup_{i=1}^{k} S_i \cup U \cup Q, Act \cup \{\bot\}, \mathbf{P}', s_0, Q, c, g)$, where $g(q_i) = LRA_i^{\min}(G)$ for $q_i \in Q$ and $c(s,\sigma) = 0$ for all s and $\sigma \in Act \cup \{\bot\}$. $\mathbf{P}'$ is defined as follows. Let $S' = S \setminus \bigcup_{i=1}^{k} S_i$. $\mathbf{P}'$ equals $\mathbf{P}$ for all $s, s' \in S'$. For the new states u_j:*

$$\mathbf{P}'(u_j,\tau,s') = \mathbf{P}(S_j,\tau,s') \quad \text{if } s' \in S' \setminus S_j \quad \text{and} \quad \mathbf{P}'(u_i,\tau,u_j) = \mathbf{P}(S_i,\tau,S_j) \quad \text{for } i \neq j.$$

Finally, we have: $\mathbf{P}'(q_j,\bot,q_j) = 1 = \mathbf{P}'(u_j,\bot,q_j)$ *and* $\mathbf{P}'(s,\sigma,u_j) = \mathbf{P}(s,\sigma,S_j)$.

Here, $\mathbf{P}(s,\alpha,S')$ is a shorthand for $\sum_{s'\in S'} \mathbf{P}(s,\alpha,s')$; similarly, $\mathbf{P}(S',\alpha,s') = \sum_{s\in S'} \mathbf{P}(s,\alpha,s')$. The terminal costs of the new q_i-states are set to $LRA_i^{\min}(G)$.

Theorem 5. *For MA $\mathcal{M}$, $LRA^{\min}(s,G)$ equals $cR^{\min}(s,\Diamond U)$ in SSP $\mathsf{ssp}_{lra}(\mathcal{M})$.*

6 Timed reachability objectives

This section presents an algorithm that approximates time-bounded reachability probabilities in MA. We start with a fixed point characterisation, and then explain how these probabilities can be approximated using digitisation.

Fixed point characterisation. Our goal is to come up with a fixed point characterisation for the maximum (minimum) probability to reach a set of goal states in a time interval. Let $\mathcal{I}$ and $\mathcal{Q}$ be the set of all nonempty nonnegative real intervals with real and rational bounds, respectively. For interval $I \in \mathcal{I}$ and $t \in \mathbb{R}_{\geq 0}$, let $I \ominus t = \{x - t \mid x \in I \wedge x \geq t\}$. Given MA $\mathcal{M}$, $I \in \mathcal{I}$ and a set $G \subseteq S$ of goal states, the set of all paths that reach some goal states within interval I is denoted by $\Diamond^I G$. Let $p^{\mathcal{M}}_{\max}(s, \Diamond^I G)$ be the maximum probability of reaching G within interval I if starting in state s at time 0. Here, the maximum is taken over all possible general measurable policies. The next result provides a characterisation of $p^{\mathcal{M}}_{\max}(s, \Diamond^I G)$ as a fixed point.

Lemma 1. *Let $\mathcal{M}$ be an MA, $G \subseteq S$ and $I \in \mathcal{I}$ with $\inf I = a$ and $\sup I = b$. Then, $p^{\mathcal{M}}_{\max}(s, \Diamond^I G)$ is the least fixed point of the higher-order operator $\Omega\colon (S \times \mathcal{I} \rightarrowtail [0,1]) \rightarrowtail (S \times \mathcal{I} \rightarrowtail [0,1])$, which for $s \in MS$ is given by:*

$$\Omega(F)(s,I) = \begin{cases} \int_0^b E(s) e^{-E(s)t} \sum_{s'\in S} \mathbf{P}(s,\bot,s') F(s', I \ominus t)\,\mathrm{d}t & s \notin G \\ e^{-E(s)a} + \int_0^a E(s) e^{-E(s)t} \sum_{s'\in S} \mathbf{P}(s,\bot,s') F(s', I \ominus t)\,\mathrm{d}t & s \in G \end{cases}$$

and for $s \in PS$ is defined by:

$$\Omega(F)(s,I) = \begin{cases} 1 & s \in G \wedge a = 0 \\ \max_{\alpha \in Act_{\setminus\bot}(s)} \sum_{s'\in S} \mathbf{P}(s,\alpha,s') F(s', I) & \text{otherwise.} \end{cases}$$

This characterisation is a simple generalisation of that for IMCs [33], reflecting the fact that taking an action from an probabilistic state leads to a distribution over the states (rather than a single state). The above characterisation yields an integral equation system which is in general not directly tractable [1]. To tackle

this problem, we approximate the fixed point characterisation using digitisation, extending ideas developed in [33]. We split the time interval into equally-sized digitisation steps, assuming a digitisation constant δ, small enough such that with high probability at most one Markovian transition firing occurs in any digitisation step. This allows us to construct a digitised MA (dMA), a variant of a semi-MDP, obtained by summarising the behaviour of the MA at equidistant time points. Paths in a dMA can be seen as time-abstract paths in the corresponding MA, implicitly still counting digitisation steps, and thus discrete time. Digitisation of MA $\mathcal{M} = (S, Act, \to, \Longrightarrow, s_0)$ and digitisation constant δ, proceeds by replacing $\Longrightarrow$ by $\Longrightarrow_\delta = \{\, (s, \mu^s) \mid s \in MS \,\}$, where

$$\mu^s(s') = \begin{cases} (1 - e^{-E(s)\delta})\mathbf{P}(s, \bot, s') & \text{if } s' \neq s \\ (1 - e^{-E(s)\delta})\mathbf{P}(s, \bot, s') + e^{-E(s)\delta} & \text{otherwise.} \end{cases}$$

Using the above fixed point characterisation, it is now possible to relate reachability probabilities in an MA $\mathcal{M}$ to reachability probabilities in its dMA $\mathcal{M}_\delta$.

Theorem 6. *Given MA $\mathcal{M} = (S, Act, \to, \Longrightarrow, s_0)$, $G \subseteq S$, interval $I = [0, b] \in \mathcal{Q}$ with $b \geq 0$ and $\lambda = \max_{s \in MS} E(s)$. Let $\delta > 0$ be such that $b = k_b\delta$ for some $k_b \in \mathbb{N}$. Then, for all $s \in S$ it holds that*

$$p^{\mathcal{M}_\delta}_{\max}(s, \Diamond^{[0,k_b]} G) \;\leq\; p^{\mathcal{M}}_{\max}(s, \Diamond^{[0,b]} G) \;\leq\; p^{\mathcal{M}_\delta}_{\max}(s, \Diamond^{[0,k_b]} G) + 1 - e^{-\lambda b}\big(1 + \lambda\delta\big)^{k_b}.$$

This theorem can be extended to intervals with non-zero lower bounds; for the sake of brevity, the details are omitted here. The remaining problem is to compute the maximum (or minimum) probability to reach G in a dMA within a step bound $k \in \mathbb{N}$. Let $\Diamond^{[0,k]} G$ be the set of infinite paths in a dMA that reach a G state within k steps, and $p^{\mathcal{D}}_{\max}(s, \Diamond^{[0,k]} G)$ denote the maximum probability of this set. Then we have $p^{\mathcal{D}}_{\max}(s, \Diamond^{[0,k]} G) = \sup_{D \in TA} \Pr_{s,D}(\Diamond^{[0,k]} G)$. Our algorithm is now an adaptation (to dMA) of the well-known value iteration scheme for MDPs.

The algorithm proceeds by backward unfolding of the dMA in an iterative manner, starting from the goal states. Each iteration intertwines the analysis of Markov states and of probabilistic states. The key issue is that a path from probabilistic states to G is split into two parts: reaching Markov states from probabilistic states in zero time and reaching goal states from Markov states in interval $[0, j]$, where j is the step count of the iteration. The former computation can be reduced to an unbounded reachability problem in the MDP induced by probabilistic states with rewards on Markov states. For the latter, the algorithm operates on the previously computed reachability probabilities from all Markov states up to step count j. We can generalize this recipe from step-bounded reachability to step interval-bounded reachability, details are described in [16].

7 Tool-chain and case studies

This section describes the implementation of the algorithms discussed, together with the modelling features resulting in our MAMA tool-chain. Furthermore, we present two case studies that provide empirical evidence of the strengths and weaknesses of the MAMA tool chain.

7.1 MaMa tool chain

Our tool chain consists of several tool components: SCOOP [28,29], IMCA [14], and GEMMA (realized in Haskell), see Figure 4. The tool-chain comprises about 8,000 LOC (without comments). SCOOP (in Haskell) supports the generation from MA from MAPA specifications by a translation into the MLPPE format. It implements all the reduction techniques described in Section 3, in particular confluence reduction. The capabilities of the IMCA tool-component (written in `C++`) have been lifted to expected time and long-run objectives for MA, and extended with timed reachability objectives. It also supports (untimed) reachability objectives which are not further treated here. A prototypical translator from GSPNs to MA, in fact MAPA specifications, has been realized (the GEMMA component). We connected the three components into a single tool chain, by making SCOOP export the (reduced) state space of an MLPPE in the IMCA input language. Additionally, SCOOP has been extended to translate properties, based on the actions and parameters of a MAPA specification, to a set of goal states in the underlying MA. That way, in one easy process systems and their properties can be modelled in MAPA, translated to an optimised MLPPE by SCOOP, exported to the IMCA tool and then analysed.

7.2 Case studies

This section reports on experiments with MaMa. All experiments were conducted on a 2.5 GHz Intel Core i5 processor with 4GB RAM, running on Mac OS X 10.8.3.

Processor grid. First, we consider a model of a 2×2 concurrent processor architecture. Using GEMMA, we automatically derived the MA model from the GSPN model in [22, Fig. 11.7]. Previous analysis of this model required weights for all immediate transitions, requiring complete knowledge of the mutual behaviour of all these transitions. We allow a weight assignment to just a (possibly empty) subset of the immediate transitions—reflecting the practical scenario of only knowing the mutual behaviour for a selection of the transitions. For this case study we indeed kept weights for only a few of the transitions, obtaining probabilistic behaviour for them and nondeterministic behaviour for the others.

Table 1 reports on the time-bounded and time-interval bounded probabilities for reaching a state such that the first processor has an empty task queue. We vary the degree of multitasking K, the error bound ϵ and the interval I. For each setting, we report the number of states $|S|$ and goal states $|G|$, and the generation time with SCOOP (both with and without the reductions from Section 3).

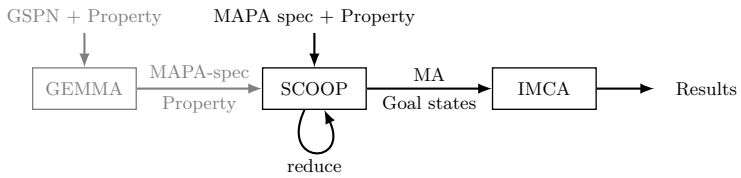

Fig. 4. Analysing Markov Automata using the MaMa tool chain.

K	unreduced $\|S\|$	unreduced $\|G\|$	unreduced time	reduced $\|S\|$	reduced $\|G\|$	reduced time	ϵ	I	$p^{\min}(s_0, \Box^I G)$	time(unred)	time(red)	$p^{\max}(s_0, \Box^I G)$	time(unred)	time(red)
2	2,508	1,398	0.6	1,789	1,122	0.8	10^{-2}	$[0,3]$	0.91	58.5	31.0	0.95	54.9	21.7
							10^{-2}	$[0,4]$	0.96	103.0	54.7	0.98	97.3	38.8
							10^{-2}	$[1,4]$	0.91	117.3	64.4	0.96	109.9	49.0
							10^{-3}	$[0,3]$	0.910	580.1	309.4	0.950	544.3	218.4
3	10,852	4,504	3.1	7,201	3,613	3.5	10^{-2}	$[0,3]$	0.18	361.5	202.8	0.23	382.8	161.1
							10^{-2}	$[0,4]$	0.23	643.1	360.0	0.30	681.4	286.0
							10^{-2}	$[1,4]$	0.18	666.6	377.3	0.25	696.4	317.7
							10^{-3}	$[0,3]$	0.176	3,619.5	2,032.1	0.231	3,837.3	1,611.9
4	31,832	10,424	9.8	20,021	8,357	10.5	10^{-2}	$[0,3]$	0.01	1,156.8	614.9	0.03	1,196.5	486.4

Table 1. Interval reachability probabilities for the grid. (Time in seconds.)

K	$eT^{\min}(s_0, \Box G)$	time(unred)	time(red)	$eT^{\max}(s_0, \Box G)$	time(unred)	time(red)	$LRA^{\min}(s_0, G)$	time(unred)	time(red)	$LRA^{\max}(s_0, G)$	time(unred)	time(red)
2	1.0000	0.3	0.1	1.2330	0.7	0.3	0.8110	1.3	0.7	0.9953	0.5	0.2
3	11.1168	18.3	7.7	15.2768	135.4	40.6	0.8173	36.1	16.1	0.9998	4.7	2.6
4	102.1921	527.1	209.9	287.8616	6,695.2	1,869.7	0.8181	505.1	222.3	1.0000	57.0	34.5

Table 2. Expected times and long-run averages for the grid. (Time in seconds.)

The runtime demands grow with both the upper and lower time bound, as well as with the required accuracy. The model size also affects the per-iteration cost and thus the overall complexity of reachability computation. Note that our reductions speed-up the analysis times by a factor between 1.7 and 3.5: even more than the reduction in state space size. This is due to our techniques significantly reducing the degree of nondeterminism.

Table 2 displays results for expected time until an empty task queue, as well as the long-run average that a processor is active. Whereas [22] fixed all nondeterminism, obtaining for instance an LRA of 0.903 for $K = 2$, we are now able to retain nondeterminism and provide the more informative interval $[0.8810, 0.9953]$. Again, our reduction techniques significantly improve runtimes.

Polling system. Second, we consider the polling system from Fig. 3 with two stations and one server. We varied the queue sizes Q and the number of job types N, analysing a total of six different settings. Since—as for the previous case—analysis scales proportionally with the error bound, we keep this constant here.

Table 3 reports results for time-bounded and time-interval bounded properties, and Table 4 displays probabilities and runtime results for expected times and long-run averages. For all analyses, the goal set consists of all states for which both station queues are full.

8 Conclusion

This paper presented new algorithms for the quantitative analysis of Markov automata (MA) and proved their correctness. Three objectives have been considered: expected time, long-run average, and timed reachability. The MAMA tool-chain supports the modelling and reduction of MA, and can analyse these three objectives. It is also equipped with a prototypical tool to map GSPNs onto MA. The MAMA is accessible via its easy-to-use web interface that can be found

Q	N	unreduced $\lvert S\rvert$	unreduced $\lvert G\rvert$	unreduced time	reduced $\lvert S\rvert$	reduced $\lvert G\rvert$	reduced time	ϵ	I	$p^{\min}(s_0, \Box^{\leq} G)$	time(unred)	time(red)	$p^{\max}(s_0, \Box^{\leq} G)$	time(unred)	time(red)
2	3	1,497	567	0.4	990	324	0.2	10^{-3}	$[0,1]$	0.277	4.7	2.9	0.558	4.6	2.5
								10^{-3}	$[1,2]$	0.486	22.1	14.9	0.917	22.7	12.5
2	4	4,811	2,304	1.0	3,047	1,280	0.6	10^{-3}	$[0,1]$	0.201	25.1	14.4	0.558	24.0	13.5
								10^{-3}	$[1,2]$	0.344	106.1	65.8	0.917	102.5	60.5
3	3	14,322	5,103	3.0	9,522	2,916	1.7	10^{-3}	$[0,1]$	0.090	66.2	40.4	0.291	60.0	38.5
								10^{-3}	$[1,2]$	0.249	248.1	180.9	0.811	241.9	158.8
3	4	79,307	36,864	51.6	50,407	20,480	19.1	10^{-3}	$[0,1]$	0.054	541.6	303.6	0.291	578.2	311.0
								10^{-3}	$[1,2]$	0.141	2,289.3	1,305.0	0.811	2,201.5	1,225.9
4	2	6,667	1,280	1.1	4,745	768	0.8	10^{-3}	$[0,1]$	0.049	19.6	14.0	0.118	19.7	12.8
								10^{-3}	$[1,2]$	0.240	83.2	58.7	0.651	80.9	53.1
4	3	131,529	45,927	85.2	87,606	26,244	30.8	10^{-3}	$[0,1]$	0.025	835.3	479.0	0.118	800.7	466.1
								10^{-3}	$[1,2]$	0.114	3,535.5	2,062.3	0.651	3,358.9	2,099.5

Table 3. Interval reachability probabilities for the polling system. (Time in seconds.)

Q	N	$eT^{\min}(s_0, \Box G)$	time(unred)	time(red)	$eT^{\max}(s_0, \Box G)$	time(unred)	time(red)	$LRA^{\min}(s_0, G)$	time(unred)	time(red)	$LRA^{\max}(s_0, G)$	time(unred)	time(red)
2	3	1.0478	0.2	0.1	2.2489	0.3	0.2	0.1230	0.8	0.5	0.6596	0.2	0.1
2	4	1.0478	0.2	0.1	3.2053	2.0	1.0	0.0635	9.0	5.2	0.6596	1.3	0.6
3	3	1.4425	1.0	0.6	4.6685	8.4	5.0	0.0689	177.9	123.6	0.6600	26.2	13.0
3	4	1.4425	9.7	4.6	8.0294	117.4	67.2	0.0277	7,696.7	5,959.5	0.6600	1,537.2	862.4
4	2	1.8226	0.4	0.3	4.6032	2.4	1.6	0.1312	45.6	32.5	0.6601	5.6	3.9
4	3	1.8226	29.8	14.2	9.0300	232.8	130.8	– timeout (18 hours) –			0.6601	5,339.8	3,099.0

Table 4. Expected times and long-run averages for the polling system. (Time in seconds.)

at `http://wwwhome.cs.utwente.nl/~timmer/mama`. Experimental results on a processor grid and a polling system give insight into the accuracy and scalability of the presented algorithms. Future work will focus on efficiency improvements and reward extensions.

References

1. C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE TSE*, 29(6):524–541, 2003.
2. D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.
3. H. Boudali, P. Crouzen, and M. I. A. Stoelinga. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE Trans. Dependable Sec. Comput.*, 7(2):128–143, 2010.
4. M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, and M. Roveri. Safety, dependability and performance analysis of extended AADL models. *The Computer Journal*, 54(5):754–775, 2011.
5. K. Chatterjee and M. Henzinger. Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In *SODA*, pages 1318–1336. SIAM, 2011.
6. N. Coste, H. Hermanns, E. Lantreibecq, and W. Serwe. Towards performance prediction of compositional models in industrial GALS designs. In *CAV*, volume 5643 of *LNCS*, pages 204–218. Springer, 2009.
7. L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, 1997.
8. L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *LICS*, pages 454–465. IEEE, 1998.

9. L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *CONCUR*, volume 1664 of *LNCS*, pages 66–81. Springer, 1999.
10. Y. Deng and M. Hennessy. On the semantics of Markov automata. *Inf. Comput.*, 222:139–168, 2013.
11. C. Eisentraut, H. Hermanns, J.-P. Katoen, and L. Zhang. Every GSPN is semantically well-defined. In *ICATPN*, LNCS. Springer, 2013. (to appear).
12. C. Eisentraut, H. Hermanns, and L. Zhang. Concurrency and composition in a stochastic world. In *CONCUR*, volume 6269 of *LNCS*, pages 21–39. Springer, 2010.
13. C. Eisentraut, H. Hermanns, and L. Zhang. On probabilistic automata in continuous time. In *LICS*, pages 342–351. IEEE, 2010.
14. D. Guck, T. Han, J.-P. Katoen, and M. R. Neuhäußer. Quantitative timed analysis of interactive Markov chains. In *NFM*, volume 7226 of *LNCS*, pages 8–23. Springer, 2012.
15. D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen, and M. Timmer. Modelling, reduction and analysis of Markov automata (extended version). Technical Report 1305.7050, ArXiv e-prints, 2013.
16. H. Hatefi and H. Hermanns. Model checking algorithms for Markov automata. In *ECEASST (AVoCS proceedings)*, volume 53, 2012. To appear.
17. B. R. Haverkort, M. Kuntz, A. Remke, S. Roolvink, and M. I. A. Stoelinga. Evaluating repair strategies for a water-treatment facility using Arcade. In *DSN*, pages 419–424. IEEE, 2010.
18. H. Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *LNCS*. Springer, 2002.
19. J.-P. Katoen. GSPNs revisited: Simple semantics and new analysis algorithms. In *ACSD*, pages 6–11. IEEE, 2012.
20. J.-P. Katoen, J. C. van de Pol, M. I. A. Stoelinga, and M. Timmer. A linear process-algebraic format with data for probabilistic automata. *TCS*, 413(1):36–57, 2012.
21. G. López, H. Hermanns, and J.-P. Katoen. Beyond memoryless distributions: Model checking semi-Markov chains. In *PAPM-PROBMIV*, number 2165 in LNCS, pages 57–70. Springer, 2001.
22. M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.
23. M. A. Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, 1984.
24. J. F. Meyer, A. Movaghar, and W. H. Sanders. Stochastic activity networks: Structure, behavior, and application. In *PNPM*, pages 106–115. IEEE, 1985.
25. M. R. Neuhäußer, M. I. A. Stoelinga, and J.-P. Katoen. Delayed nondeterminism in continuous-time Markov decision processes. In *FOSSACS*, volume 5504 of *LNCS*, pages 364–379. Springer, 2009.
26. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.
27. M. M. Srinivasan. Nondeterministic polling systems. *Management Science*, 37(6):667–681, 1991.
28. M. Timmer. SCOOP: A tool for symbolic optimisations of probabilistic processes. In *QEST*, pages 149–150. IEEE, 2011.
29. M. Timmer, J.-P. Katoen, J. C. van de Pol, and M. I. A. Stoelinga. Efficient modelling and generation of Markov automata. In *CONCUR*, volume 7454 of *LNCS*, pages 364–379. Springer, 2012.
30. M. Timmer, J.-P. Katoen, J. C. van de Pol, and M. I. A. Stoelinga. Efficient modelling and generation of Markov automata (extended version). Technical Report TR-CTIT-12-16, CTIT, University of Twente, 2012.
31. M. Timmer, M. I. A. Stoelinga, and J. C. van de Pol. Confluence reduction for Markov automata. In *FORMATS*, LNCS, 2013 (to be published).
32. J. C. van de Pol and M. Timmer. State space reduction of linear processes using control flow reconstruction. In *ATVA*, LNCS 5799, pages 54–68. Springer, 2009.
33. L. Zhang and M. R. Neuhäußer. Model checking interactive Markov chains. In *TACAS*, volume 6015 of *LNCS*, pages 53–68. Springer, 2010.