Lazy Probabilistic Model Checking without Determinisation

Ernst Moritz Hahn[1], Guangyuan Li[1], Sven Schewe[2], Andrea Turrini[1], and Lijun Zhang[1]

1 **State Key Laboratory of Computer Science, Institute of Software, CAS**
2 **University of Liverpool**

Abstract

The bottleneck in the quantitative analysis of Markov chains and Markov decision processes against specifications given in LTL or as some form of nondeterministic Büchi automata is the inclusion of a determinisation step of the automaton under consideration. In this paper, we show that full determinisation can be avoided: subset and breakpoint constructions suffice. We have implemented our approach—both explicit and symbolic versions—in a prototype tool. Our experiments show that our prototype can compete with mature tools like PRISM.

1998 ACM Subject Classification G.3: Probability and Statistics; D.2.4: Software/Program Verification

Keywords and phrases Markov Decision Processes; Model Checking; PLTL; Determinisation

1 Introduction

Markov chains (MCs) and Markov decision processes (MDPs) are widely used to study systems that exhibit both, probabilistic and nondeterministic choices. Properties of these systems are often specified by temporal logic formulas, such as the branching time logic PCTL [11], the linear time logic PLTL [3], or their combination PCTL* [3]. While model checking is tractable for PCTL [3], it is more expensive for PLTL: PSPACE-complete for Markov chains and 2EXPTIME-complete for MDPs [6].

In classical model checking, one checks whether a model $\mathcal{M}$ satisfies an LTL formula φ by first constructing a nondeterministic Büchi automaton $\mathcal{B}_{\neg\varphi}$ [20], which recognises the models of its negation $\neg\varphi$. The model checking problem then reduces to an emptiness test for the product $\mathcal{M} \otimes \mathcal{B}_{\neg\varphi}$. The translation to Büchi automata may result in an exponential blow-up compared to the length of φ. However, this translation is mostly very efficient in practice, and highly optimised off-the-shelf tools like LTL3BA [1] or SPOT [7] are available.

The quantitative analysis of a probabilistic model $\mathcal{M}$ against an LTL specification φ is more involved. To compute the maximal probability $\mathfrak{P}^{\mathcal{M}}(\varphi)$ that φ is satisfied in $\mathcal{M}$, the classic automata-based approach includes the determinisation of an intermediate Büchi automaton $\mathcal{B}_\varphi$. If such a deterministic automaton $\mathcal{A}$ is constructed for $\mathcal{B}_\varphi$, then determining the probability $\mathfrak{P}^{\mathcal{M}}(\varphi)$ reduces to solving an equation system for Markov chains, and a linear programming problem for MDPs [3], both in the product $\mathcal{M} \otimes \mathcal{A}$. Such a determinisation step usually exploits a variant of Safra's [17] determinisation construction, such as the techniques presented in [16, 18].

Kupferman, Piterman, and Vardi point out in [14] that "Safra's determinization construction has been notoriously resistant to efficient implementations." Even though analysing long LTL formulas would surely be useful as they allow for the description of more complex requirements on a system's behaviour, model checkers that employ determinisation to support LTL, such as LiQuor [5] or PRISM [15], might fail to verify such properties.

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper we argue that applying the Safra determinisation step in full generality is only required in some cases, while simpler subset and breakpoint constructions often suffice. Moreover, where full determinisation is required, it can be replaced by a combination of the simpler constructions, and it suffices to apply it locally on a small share of the places.

A subset construction is known to be sufficient to determinise finite automata, but it fails for Büchi automata. Our first idea is to construct an under- and an over-approximation starting from the subset construction. That is, we construct two (deterministic) subset automata $\mathcal{S}^u$ and $\mathcal{S}^o$ such that $\mathcal{L}(\mathcal{S}^u) \subseteq \mathcal{L}(\mathcal{B}_\varphi) \subseteq \mathcal{L}(\mathcal{S}^o)$ where $\mathcal{L}(\mathcal{B}_\varphi)$ denotes the language defined by the automaton $\mathcal{B}_\varphi$ for φ. The subset automata $\mathcal{S}^u$ and $\mathcal{S}^o$ are the same automaton $\mathcal{S}$ except for their accepting conditions. We build a product Markov chain with the subset automata. We establish the useful property that the probability $\mathfrak{P}^{\mathcal{M}}(\varphi)$ equals the probability of reaching some *accepting* bottom strongly connected components (SCCs) in this product: for each bottom SCC $\mathtt{S}$ in the product, we can first use the accepting conditions in $\mathcal{S}^u$ or $\mathcal{S}^o$ to determine whether $\mathtt{S}$ is accepting or rejecting, respectively. The challenge remains when the test is inconclusive. In this case, we first refine $\mathtt{S}$ using a breakpoint construction. Finally, if the breakpoint construction fails as well, we have two options: we can either perform a Rabin-based determinisation for the part of the model where it is required, thus avoiding to construct the larger complete Rabin product. Alternatively, a refined multi-breakpoint construction is used. An important consequence is that we no longer need to implement a Safra-style determinisation procedure: subset and breakpoint constructions are enough. From a theoretical point of view, this reduces the cost of the automata transformations involved from $n^{\mathcal{O}(k \cdot n)}$ to $\mathcal{O}(k \cdot 3^n)$ for generalised Büchi automata with n states and k accepting sets. From a practical point of view, the easy symbolic encoding admitted by subset and breakpoint constructions is of equal value. We discuss that (and how) the framework can be adapted to MDPs—with the same complexity—by analysing the end components [3, 6].

We have implemented our approach, both explicit and symbolic versions, in our IscasMC tool [10], which we applied on various Markov chain and MDP case studies (for space reasons we report on only two in this paper, cf. [9] for others). Our experimental results confirm that our new algorithm outperforms the Rabin-based approach in most of the properties considered. However, there are some cases in which the Rabin determinisation approach performs better when compared to the multi-breakpoint construction: the construction of a single Rabin automaton suffices to decide a given connected component, while the breakpoint construction may require several iterations. Our experiments also show that our prototype can compete with mature tools like PRISM.

Due to the lack of space, detailed proofs and additional case studies are provided in [9].

2 Preliminaries

2.1 ω-Automata

Nondeterministic Büchi automata are used to represent ω-regular languages $\mathcal{L} \subseteq \Sigma^\omega = \omega \to \Sigma$ over a finite alphabet Σ. In this paper, we use automata with trace-based acceptance mechanisms. We denote by $[1..k]$ the set $\{1, 2, \ldots, k\}$ and by $j \oplus_k 1$ the successor of j in $[1..k]$. I.e., $j \oplus_k 1 = j + 1$ if $j < k$ and $j \oplus_k 1 = 1$ if $j = k$.

▶ **Definition 1.** A *nondeterministic generalised Büchi automaton* (NGBA) is a quintuple $\mathcal{B} = (\Sigma, Q, I, \mathrm{T}, \mathbf{F}_k)$, consisting of a finite alphabet Σ of input letters, a finite set Q of states with a non-empty subset $I \subseteq Q$ of initial states, a set $\mathrm{T} \subseteq Q \times \Sigma \times Q$ of transitions from states through input letters to successor states, and a family $\mathbf{F}_k = \{ \mathrm{F}_j \subseteq \mathrm{T} \mid j \in [1..k] \}$ of accepting (final) sets.

Nondeterministic Büchi automata are interpreted over infinite sequences $\alpha\colon \omega \to \Sigma$ of input letters. An infinite sequence $\rho\colon \omega \to Q$ of states of $\mathcal{B}$ is called a *run* of $\mathcal{B}$ on an input word α if $\rho(0) \in I$ and, for each $i \in \omega$, $\big(\rho(i), \alpha(i), \rho(i+1)\big) \in \mathrm{T}$. We denote by $\mathrm{Run}(\alpha)$ the set of all runs ρ on α. For a run $\rho \in \mathrm{Run}(\alpha)$, we denote with $\mathrm{tr}(\rho)\colon i \mapsto \big(\rho(i), \alpha(i), \rho(i+1)\big)$ the *transitions of* ρ. We sometimes denote a run ρ by the associated states, that is, $\rho = q_0 \cdot q_1 \cdot q_2 \cdot \ldots$ where $\rho(i) = q_i$ for each $i \in \omega$ and we call a finite prefix $q_0 \cdot q_1 \cdot q_2 \cdot \ldots \cdot q_n$ of ρ a *pre-run*. A run ρ of a NGBA is *accepting* if its transitions $\mathrm{tr}(\rho)$ contain infinitely many transitions from all final sets, i.e., for each $j \in [1..k]$, $\mathrm{Inf}(\mathrm{tr}(\rho)) \cap \mathrm{F}_j \neq \emptyset$, where $\mathrm{Inf}(\mathrm{tr}(\rho)) = \{\, t \in \mathrm{T} \mid \forall i \in \omega\ \exists j > i \text{ such that } \mathrm{tr}(\rho)(j) = t \,\}$. A word $\alpha\colon \omega \to \Sigma$ is *accepted* by $\mathcal{B}$ if $\mathcal{B}$ has an accepting run on α, and the set $\mathcal{L}(\mathcal{B}) = \{\, \alpha \in \Sigma^\omega \mid \alpha \text{ is accepted by } \mathcal{B} \,\}$ of words accepted by $\mathcal{B}$ is called its *language*.

Figure 1 shows an example of Büchi automaton. The number j after the label as in the transition (x, a, y), when present, indicates that the transition belongs to the accepting set F_j, i.e., (x, a, y) belongs to F_1. The language generated by $\mathcal{B}_\mathcal{E}$ is a subset of $(ab|ac)^\omega$ and a word α is accepted if each b (and c) is eventually followed by a c (by a b, respectively).

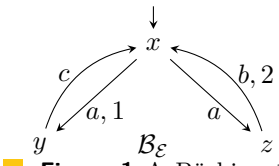


Figure 1 A Büchi automaton

We call the automaton $\mathcal{B}$ a *nondeterministic Büchi automaton* (NBA) whenever $|\mathbf{F}_k| = 1$ and we denote it by $\mathcal{B} = (\Sigma, Q, I, \mathrm{T}, \mathrm{F})$. For technical convenience we also allow for finite runs $q_0 \cdot q_1 \cdot q_2 \cdot \ldots \cdot q_n$ with $\mathrm{T} \cap \{q_n\} \times \{\alpha(n)\} \times Q = \emptyset$. In other words, a run may end with q_n if action $\alpha(n)$ is not enabled from q_n. Naturally, no finite run satisfies the accepting condition, thus it is not accepting and has no influence on the language of an automaton.

To simplify the notation, the transition set T can also be seen as a function $\mathrm{T}\colon Q \times \Sigma \to 2^Q$ assigning to each pair $(q, \sigma) \in Q \times \Sigma$ the set of successors according to T, i.e., $\mathrm{T}(q, \sigma) = \{\, q' \in Q \mid (q, \sigma, q') \in \mathrm{T} \,\}$. We extend T to sets of states in the usual way, i.e., by defining $\mathrm{T}(S, \sigma) = \bigcup_{q \in S} \mathrm{T}(q, \sigma)$.

▶ **Definition 2.** A *(transition-labelled) nondeterministic Rabin automaton* (NRA) with k accepting pairs is a quintuple $\mathcal{A} = (\Sigma, Q, I, \mathrm{T}, (\mathbf{A}_k, \mathbf{R}_k))$ where Σ, Q, I, and T are as in Definition 1 and $(\mathbf{A}_k, \mathbf{R}_k) = \{\, (\mathrm{A}_i, \mathrm{R}_i) \mid i \in [1..k],\ \mathrm{A}_i, \mathrm{R}_i \subseteq \mathrm{T} \,\}$ is a finite family of Rabin pairs. (For convenience, we sometimes use other finite sets of indices rather than $[1..k]$.)

A run ρ of a NRA is accepting if there exists $i \in [1..k]$ such that $\mathrm{Inf}(\mathrm{tr}(\rho)) \cap \mathrm{A}_i \neq \emptyset$ and $\mathrm{Inf}(\mathrm{tr}(\rho)) \cap \mathrm{R}_i = \emptyset$.

An automaton $\mathcal{A} = (\Sigma, Q, I, \mathrm{T}, \mathbf{ACC})$, where $\mathbf{ACC}$ is the acceptance condition (Rabin, Büchi, or generalised Büchi), is called *deterministic* if, for each $(q, \sigma) \in Q \times \Sigma$, $|\mathrm{T}(q, \sigma)| \leq 1$, and $I = \{q_0\}$ for some $q_0 \in Q$. For notational convenience, we denote a deterministic automaton $\mathcal{A}$ by the tuple $(\Sigma, Q, q_0, \mathrm{T}, \mathbf{ACC})$ and $\mathrm{T}\colon Q \times \Sigma \to Q$ is the partial function, which is defined at (q, σ) if, and only if, σ is enabled at q. For a given deterministic automaton $\mathcal{D}$, we denote by $\mathcal{D}_d$ the otherwise similar automaton with initial state d. Similarly, for a NGBA $\mathcal{B}$, we denote by $\mathcal{B}_R$ the NGBA with R as set of initial states.

2.2 Markov Chains and Product

A *distribution* μ over a set X is a function $\mu\colon X \to [0, 1]$ such that $\sum_{x \in X} \mu(x) = 1$. A *Markov chain (MC)* is a tuple $\mathcal{M} = (M, L, \mu_0, \mathrm{P})$, where M is a finite set of states, $L\colon M \to \Sigma$ is a *labelling function*, μ_0 is the *initial distribution*, and $\mathrm{P}\colon M \times M \to [0, 1]$ is a *probabilistic transition matrix* satisfying $\sum_{m' \in M} \mathrm{P}(m, m') \in \{0, 1\}$ for all $m \in M$. A state m is called *absorbing* if $\sum_{m' \in M} \mathrm{P}(m, m') = 0$. We write $(m, m') \in \mathrm{P}$ for $\mathrm{P}(m, m') > 0$.

A *maximal path* of $\mathcal{M}$ is an infinite sequence $\xi = m_0 m_1 \dots$ satisfying $\mathrm{P}(m_i, m_{i+1}) > 0$ for all $i \in \omega$, or a finite one if the last state is absorbing. We denote by $Paths^{\mathcal{M}}$ the set of all maximal paths of $\mathcal{M}$. An infinite path $\xi = m_0 m_1 \dots$ defines the word $\alpha(\xi) = w_0 w_1 \dots \in \Sigma^\omega$ with $w_i = L(m_i)$, $i \in \omega$.

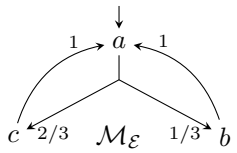


Figure 2 A MC with $L(m) = m$ for each m.

Given a finite sequence $\xi = m_0 m_1 \dots m_k$, the *cylinder* of ξ, denoted by $Cyl(\xi)$, is the set of maximal paths starting with prefix ξ. We define the probability of the cylinder set by $\mathfrak{P}^{\mathcal{M}}\big(Cyl(m_0 m_1 \dots m_k)\big) \stackrel{\text{def}}{=} \mu_0(m_0) \cdot \prod_{i=0}^{k-1} \mathrm{P}(m_i, m_{i+1})$. For a given MC $\mathcal{M}$, $\mathfrak{P}^{\mathcal{M}}$ can be uniquely extended to a probability measure over the σ-algebra generated by all cylinder sets.

In this paper we are interested in ω-regular properties $\mathcal{L} \subseteq \Sigma^\omega$ and the probability $\mathfrak{P}^{\mathcal{M}}(\mathcal{L})$ for some measurable set $\mathcal{L}$. Further, we define $\mathfrak{P}^{\mathcal{M}}(\mathcal{B}) \stackrel{\text{def}}{=} \mathfrak{P}^{\mathcal{M}}(\{\, \xi \in Paths^{\mathcal{M}} \mid \alpha(\xi) \in \mathcal{L}(\mathcal{B}) \,\})$ for an automaton $\mathcal{B}$. We write $\mathfrak{P}^{\mathcal{M}}_m$ to denote the probability function when assuming that m is the initial state. Moreover, we omit the superscript $\mathcal{M}$ whenever it is clear from the context. We follow the standard way of computing this probability in the product of $\mathcal{M}$ and a deterministic automaton for $\mathcal{L}$.

▶ **Definition 3.** Given a MC $\mathcal{M} = (M, L, \mu_0, \mathrm{P})$ and a deterministic automaton $\mathcal{A} = (\Sigma, Q, q_0, \mathrm{T}, \mathbf{ACC})$, the *product Markov chain* is defined by $\mathcal{M} \times \mathcal{A} \stackrel{\text{def}}{=} (M \times Q, L', \mu_0', \mathrm{P}')$ where $L'\big((m,d)\big) = L(m)$; $\mu_0'\big((m,d)\big) = \mu_0(m)$ if $d = \mathrm{T}(q_0, L(m))$, 0 otherwise; and $\mathrm{P}'\big((m,d),(m',d')\big)$ equals $\mathrm{P}(m,m')$ if $d' = \mathrm{T}(d, L(m'))$, and is 0 otherwise.

We denote by $\pi_{\mathcal{A}}((m,d),(m',d'))$ the *projection* on $\mathcal{A}$ of the given $((m,d),(m',d')) \in \mathrm{P}'$, i.e., $\pi_{\mathcal{A}}((m,d),(m',d')) = (d, L(m'), d')$, and by $\pi_{\mathcal{A}}(\mathrm{B})$ its extension to a set of transitions $\mathrm{B} \subseteq \mathrm{T}'$, i.e., $\pi_{\mathcal{A}}(\mathrm{B}) = \{\, \pi_{\mathcal{A}}(p,p') \mid (p,p') \in \mathrm{B} \,\}$.

As we have accepting transitions on the edges of the automata, we propose product Markov chains with accepting conditions on their edges.

▶ **Definition 4.** Given a MC $\mathcal{M}$ and a deterministic automaton $\mathcal{A}$ with accepting set $\mathbf{ACC}$, the product automaton is $\mathcal{M} \otimes \mathcal{A} \stackrel{\text{def}}{=} (\mathcal{M} \times \mathcal{A}, \mathbf{ACC}')$ where

- if $\mathbf{ACC} = \mathbf{F}_k$, then $\mathbf{ACC}' \stackrel{\text{def}}{=} \mathbf{F}'_k$ where $\mathrm{F}'_i = \{\, (p,p') \in \mathrm{P}' \mid \pi_{\mathcal{A}}(p,p') \in \mathrm{F}_i \,\} \in \mathbf{F}'_k$ for each $i \in [1..k]$ (Generalised Büchi Markov chain, GMC); and
- if $\mathbf{ACC} = (\mathbf{A}_k, \mathbf{R}_k)$, then $\mathbf{ACC}' \stackrel{\text{def}}{=} (\mathbf{A}'_k, \mathbf{R}'_k)$ where $\mathrm{A}'_i = \{\, (p,p') \in \mathrm{P}' \mid \pi_{\mathcal{A}}(p,p') \in \mathrm{A}_i \,\} \in \mathbf{A}'_k$ and $\mathrm{R}'_i = \{\, (p,p') \in \mathrm{P}' \mid \pi_{\mathcal{A}}(p,p') \in \mathrm{R}_i \,\} \in \mathbf{R}'_k$ for each $i \in [1..k]$ (Rabin Markov chain, RMC).

Thus, RMC and GMC are Markov chains extended with the corresponding accepting conditions. We remark that the labelling of the initial states of the Markov chain is taken into account in the definition of μ_0'.

▶ **Definition 5.** A *bottom strongly connected component* (BSCC) $\mathtt{S} \subseteq V$ is an SCC in the underlying digraph (V, E) of a MC $\mathcal{M}$, where all edges with source in $\mathtt{S}$ have only successors in $\mathtt{S}$ (i.e., for each $(v,v') \in E$, $v \in \mathtt{S}$ implies $v' \in \mathtt{S}$). We assume that a (bottom) SCC does not contain any absorbing state. Given an SCC $\mathtt{S}$, we denote by $\mathrm{P}_{\mathtt{S}}$ the transitions of $\mathcal{M}$ in $\mathtt{S}$, i.e., $\mathrm{P}_{\mathtt{S}} = \{\, (m,m') \in \mathrm{P} \mid m, m' \in \mathtt{S} \,\}$.

3 Lazy Determinisation

We fix an input MC $\mathcal{M}$ and a NGBA $\mathcal{B} = (\Sigma, Q, I, \mathrm{T}, \mathbf{F}_k)$ as a specification. Further, let $\mathcal{A} = \det(\mathcal{B})$ be the deterministic Rabin automaton (DRA) constructed for $\mathcal{B}$ (cf. [17–19]),

and let $\mathcal{M} \otimes \mathcal{A} = (M \times Q, L, \mu_0, \mathrm{P}, \mathbf{ACC})$ be the product RMC. We consider the problem of computing $\mathfrak{P}^{\mathcal{M}}_{m_0}(\mathcal{B})$, i.e., the probability that a run of $\mathcal{M}$ is accepted by $\mathcal{B}$.

3.1 Outline of our Methodology

We first recall the classical approach for computing $\mathfrak{P}^{\mathcal{M}}(\mathcal{B})$, see [2] for details. It is well known [3] that the computation of $\mathfrak{P}^{\mathcal{M}}(\mathcal{B})$ reduces to the computation of the probabilistic reachability in the product RMC $\mathcal{M} \otimes \mathcal{A}$ with $\mathcal{A} = \det(\mathcal{B})$. We first introduce the notion of accepting SCCs:

▶ **Definition 6.** Given a MC $\mathcal{M}$ and the DRA $\mathcal{A} = \det(\mathcal{B})$, let $\mathtt{S}$ be a bottom SCC of the product RMC $\mathcal{M} \otimes \mathcal{A}$. We say that $\mathtt{S}$ is accepting if there exists an index $i \in [1..k]$ such that $\mathrm{A}_i \cap \pi_{\mathcal{A}}(\mathrm{P}_{\mathtt{S}}) \neq \emptyset$ and $\mathrm{R}_i \cap \pi_{\mathcal{A}}(\mathrm{P}_{\mathtt{S}}) = \emptyset$; we call each $s \in \mathtt{S}$ an *accepting state*. Moreover, we call the union of all accepting BSCCs the *accepting region*.

Essentially, since a BSCC is an ergodic set, once a path enters an accepting BSCC $\mathtt{S}$, with probability 1 it will take transitions from A_i infinitely often; since A_i is finite, at least one transition from A_i is taken infinitely often. Now we have the following reduction:

▶ **Theorem 7** ([3])**.** *Given a MC $\mathcal{M}$ and a Büchi automaton $\mathcal{B}$, consider $\mathcal{A} = \det(\mathcal{B})$. Let $U \subseteq M \times Q$ be the accepting region and let $\Diamond U$ denote the set of paths containing a state of U. Then, $\mathfrak{P}^{\mathcal{M}}(\mathcal{B}) = \mathfrak{P}^{\mathcal{M} \otimes \mathcal{A}}(\Diamond U)$.*

When all bottom SCCs are evaluated, the evaluation of the Rabin MC is simple: we abstract all accepting bottom SCCs to an absorbing goal state and perform a reachability analysis, which can be solved in polynomial time [2, 3]. Thus, the outline of the traditional probabilistic model checking approach for LTL specifications is as follows: (1.) translate the NGBA $\mathcal{B}$ into an equivalent DRA $\mathcal{A} = \det(\mathcal{B})$; (2.) build (the reachable fragment of) the product automaton $\mathcal{M} \otimes \mathcal{A}$; (3.) for each BSCC $\mathtt{S}$, check whether $\mathtt{S}$ is accepting. Let U be the union of these accepting SCCs; (4.) infer the probability $\mathfrak{P}^{\mathcal{M} \otimes \mathcal{A}}(\Diamond U)$.

The construction of the deterministic Rabin automaton used in the classical approach is often the bottleneck of the approach, as one exploits some variant of the approach proposed by Safra [17], which is rather involved. The lazy determinisation technique we suggest in this paper follows a different approach. We first transform the high-level specification (e.g., given in the PRISM language [15]) into its MDP or MC semantics. We then employ some tool (e.g., LTL3BA [1] or SPOT [7]) to construct a Büchi automaton equivalent to the LTL specification. This nondeterministic automaton is used to obtain the deterministic Büchi over- and under-approximation subset automata $\mathcal{S}^u$ and $\mathcal{S}^o$, as described in Subsection 3.3. The languages recognised by these two deterministic Büchi automata are such that $\mathcal{L}(\mathcal{S}^u) \subseteq \mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{S}^o)$. We build the product of these subset automata with the model MDP or MC (cf. Lemma 13). We then compute the maximal end components or bottom strongly connected components. According to Lemma 14, we try to decide these components of the product by using the acceptance conditions F^o_i and F^u_i of $\mathcal{S}^u$ and $\mathcal{S}^o$, respectively.

For each of those components where over- and under-approximation do not agree (and which we therefore cannot decide), we employ the breakpoint construction (cf. Corollary 16), involving the deterministic Rabin over- and under-approximation breakpoint automata $\mathcal{BP}^u$ and $\mathcal{BP}^o$, such that $\mathcal{L}(\mathcal{BP}^u) \subseteq \mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{BP}^o)$. For this, we take one state of the component under consideration and start the breakpoint construction with this state as initial state. This way, we obtain a product of a breakpoint automaton with parts of the model. If the resulting product contains an accepting component (using the under-approximation), then

the original component must be accepting, and if the resulting product contains a rejecting component (using the over-approximation), then the original component must be rejecting.

The remaining undecided components are decided either by using a Rabin-based construction, restricted to the undecided component, or only by using $\mathcal{BP}^u$, where we start from possibly different states of the subset product component under consideration; this approach always decides the remaining components, and we call it the multi-breakpoint construction.

For the model states that are part of an accepting component, or from which no accepting component is reachable, the probability to fulfil the specification is now already known to be 1 or 0, respectively. To obtain the remaining state probabilities, we construct and solve a linear programming (LP) problem (or a linear equation system when we start with MCs).

Note that, even in case the multi-breakpoint procedure is necessary in some places, our method is usually still more efficient than direct Rabin determinisation, for instance based on some variation of [19]. The reason for this is twofold. First, when starting the determinisation procedure from a component rather than from the initial state of the model, the number of states in the Rabin product will be smaller, and second, we only need the multi-breakpoint determinisation to decide MECs or bottom SCCs, such that the computation of transient probabilities can still be done in the smaller subset product.

In the remainder of this section, we detail the proposed approach: we first introduce the theoretical background, and then present the incremental evaluation of the bottom SCCs.

3.2 Acceptance Equivalence

In order to be able to apply our lazy approach, we exploit a number of acceptance equivalences in the RMC. Given the DRA $\mathcal{A} = \det(\mathcal{B})$ and a state d of $\mathcal{A}$, we denote by $\mathsf{rchd}(d)$ the label of the root node ε of the labelled ordered tree associated to d (cf. [17–19]).

▶ **Proposition 8.** *Given a NGBA $\mathcal{B}$, a MC $\mathcal{M}$, and the DRA $\mathcal{A} = \det(\mathcal{B})$, (1.) a path ρ in $\mathcal{M} \otimes \mathcal{A}$ that starts from a state (m, d) is accepted if, and only if, the word it defines is accepted by $\mathcal{B}_{\mathsf{rchd}(d)}$; and (2.) if $\mathsf{rchd}(d) = \mathsf{rchd}(d')$, then the probabilities of acceptance from a state (m, d) and a state (m, d') are equal, i.e., $\mathfrak{P}^{\mathcal{M}\otimes\mathcal{A}}_{(m,d)}(\mathcal{B}) = \mathfrak{P}^{\mathcal{M}\otimes\mathcal{A}}_{(m,d')}(\mathcal{B})$.*

This property allows us to work on quotients *and* to swap between states with the same reachability set. If we ignore the accepting conditions, we have a product MC, and we can consider the quotient of such a product MC as follows.

▶ **Definition 9** (Quotient MC)**.** Given a MC $\mathcal{M}$ and a DRA $\mathcal{A} = \det(\mathcal{B})$, the *quotient MC* $[\mathcal{M} \times \mathcal{A}]$ of $\mathcal{M} \times \mathcal{A}$ is the MC $([M \times Q], [L], [\mu_0], [\mathrm{P}])$ where

- $[M \times Q] = \{\, (m, [d]) \mid (m, d) \in M \times Q,\ [d] = \{\, d' \in Q \mid \mathsf{rchd}(d') = \mathsf{rchd}(d) \,\} \,\}$,
- $[L](m, [d]) = L(m, d)$,
- $[\mu_0](m, [d]) = \mu_0(m, d)$, and
- $[\mathrm{P}]\big((m, [d]), (m', [d'])\big) = \mathrm{P}\big((m, d), (m', d')\big)$.

By abuse of notation, we define $[(m, d)] = (m, [d])$ and $[C] = \{\, [s] \mid s \in C \,\}$. It is easy to see that, for each $d \in Q$, $d \in [d]$ holds and that $[\mathrm{P}]$ is well defined: for $(m, d_1), (m, d_2) \in [(m, d)]$, $\mathrm{P}\big((m, d_1), (m', [d'])\big) = \mathrm{P}\big((m, d), (m', d')\big) = \mathrm{P}\big((m, d_2), (m', [d'])\big)$ holds.

▶ **Theorem 10.** *For a MC $\mathcal{M}$ and DRA $\mathcal{A} = \det(\mathcal{B})$, it holds that*

1. *if $\mathtt{S}$ is a bottom SCC of $\mathcal{M} \times \mathcal{A}$ then $[\mathtt{S}]$ is a bottom SCC of $[\mathcal{M} \times \mathcal{A}]$,*
2. *if $\mathtt{S}'$ is a bottom SCC of $[\mathcal{M} \times \mathcal{A}]$, then there is a bottom SCC $\mathtt{S}$ of $\mathcal{M} \times \mathcal{A}$ with $\mathtt{S}' = [\mathtt{S}]$.*

Together with Definition 6 and Proposition 8, Theorem 10 provides:

▶ **Corollary 11.** *Let* $\mathtt{S}$ *be a bottom SCC of* $[\mathcal{M} \times \mathcal{A}]$*. Then, either all states* s *of* $\mathcal{M} \otimes \mathcal{A}$ *with* $[s] \in \mathtt{S}$ *are accepting, or all states* s *of* $\mathcal{M} \otimes \mathcal{A}$ *with* $[s] \in \mathtt{S}$ *are rejecting.*

Once all bottom SCCs are evaluated, we only need to perform a standard probabilistic reachability analysis on the quotient MC.

3.3 Incremental Evaluation of Bottom SCCs

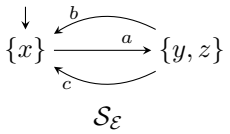
To evaluate each bottom SCC of the RMC, we use three techniques: the first one is based on evaluating the subset construction directly. We get two deterministic NGBAs that provide over- and under-approximations. If this fails, we refine the corresponding bottom SCC by a breakpoint construction. Only if both fail, a precise construction follows.

3.3.1 Subset Construction

For a given NGBA $\mathcal{B} = (\Sigma, Q, I, \mathrm{T}, \mathbf{F}_k)$, a simple way to over- and under-approximate its language by a subset construction is as follows. We build two NGBAs $\mathcal{S}^o = (\Sigma, 2^Q, \{I\}, \mathrm{T}', \mathbf{F}^o_k)$ and $\mathcal{S}^u = (\Sigma, 2^Q, \{I\}, \mathrm{T}', \mathbf{F}^u_k)$, differing only for the accepting condition, where

- $\mathrm{T}' = \{\, (R, \sigma, C) \mid \emptyset \neq R \subseteq Q, C = \mathrm{T}(R, \sigma) \,\}$,
- $\mathrm{F}^o_i = \{\, (R, \sigma, C) \in \mathrm{T}' \mid \exists (q, q') \in R \times C.\ (q, \sigma, q') \in \mathrm{F}_i \,\} \in \mathbf{F}^o_k$ for each $i \in [1..k]$, and
- $\mathrm{F}^u_i = \{\, (R, \sigma, C) \in \mathrm{T}' \mid \forall (q, q') \in R \times C.\ (q, \sigma, q') \in \mathrm{F}_i \,\} \in \mathbf{F}^u_k$ for each $i \in [1..k]$.

Essentially, $\mathcal{S}^o$ and $\mathcal{S}^u$ are the subset automata that we use to over- and under-approximate the accepting conditions, respectively. Figure 3 shows the reachable fragment of the subset construction for the NGBA $\mathcal{B}_\mathcal{E}$ depicted in Figure 1. The final sets of the two subset automata are $\mathrm{F}^o_1 = \{(\{x\}, a, \{yz\})\}$ and $\mathrm{F}^o_2 = \{(\{yz\}, b, \{x\})\}$ for $\mathcal{S}^o$ and $\mathrm{F}^u_1 = \mathrm{F}^u_2 = \emptyset$ for $\mathcal{S}^u$. The following lemma holds:



$\mathcal{S}_\mathcal{E}$

Figure 3 The subset construction for $\mathcal{B}_\mathcal{E}$

▶ **Lemma 12.** $\mathcal{L}(\mathcal{S}^u_{[d]}) \subseteq \mathcal{L}(\mathcal{A}_d) \subseteq \mathcal{L}(\mathcal{S}^o_{[d]})$.

The proof is easy as, in each F^o_i and F^u_i, the accepting transitions are over- and under-approximated. With this lemma, we are able to identify some accepting and rejecting bottom SCCs in the product.

We remark that $\mathcal{S}^o$ and $\mathcal{S}^u$ differ only in their accepting conditions. Thus, the corresponding GMCs $\mathcal{M} \otimes \mathcal{S}^u$ and $\mathcal{M} \otimes \mathcal{S}^o$ also differ only for their accepting conditions. If we ignore the accepting conditions, we have the following result:

▶ **Lemma 13.** *Let* $\mathcal{M}$ *be a MC,* $\mathcal{B}$ *a NGBA,* $\mathcal{A} = \det(\mathcal{B})$*, and* $\mathcal{S}^u$ *as defined above; let* $\mathcal{S}$ *be* $\mathcal{S}^u$ *without the accepting conditions. Then,* $\mathcal{M} \times \mathcal{S}$ *and* $[\mathcal{M} \times \mathcal{A}]$ *are isomorphic.*

The proof is rather easy—it is based on the isomorphism identifying a state (m, R) of $\mathcal{M} \times \mathcal{S}$ with the state $(m, [d])$ of $[\mathcal{M} \times \mathcal{A}]$ such that $\mathsf{rchd}(d) = R$.

Considering the accepting conditions, we can classify some bottom SCCs.

▶ **Lemma 14.** *Let* $\mathcal{M}$ *be a MC and* $\mathcal{B}$ *a NGBA. Let* $\mathcal{S}^o$ *and* $\mathcal{S}^u$ *be as defined above. Let* $\mathtt{S}$ *be a bottom SCC of* $\mathcal{M} \otimes \mathcal{S}^u$*. Then,*

- $\mathtt{S}$ *is accepting if* $\mathrm{F}^u_i \cap \pi_{\mathcal{S}^u}(\mathrm{P}_\mathtt{S}) \neq \emptyset$ *holds for all* $i \in [1..k]$*;*
- $\mathtt{S}$ *is rejecting if* $\mathrm{F}^o_i \cap \pi_{\mathcal{S}^u}(\mathrm{P}_\mathtt{S}) = \emptyset$ *holds for some* $i \in [1..k]$.

The above result directly follows by Lemma 12. Figure 4 shows the product of the MC $\mathcal{M}_\mathcal{E}$ depicted in Figure 2 and the subset automaton $\mathcal{S}_\mathcal{E}$ in Figure 3. It is easy to check that the only bottom SCC is neither accepting nor rejecting.

For the bottom SCCs, for which we cannot conclude whether they are accepting or rejecting, we continue with the breakpoint construction.

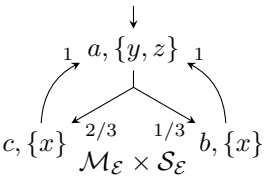


Figure 4 The product of $\mathcal{M}_\mathcal{E}$ and $\mathcal{S}_\mathcal{E}$

3.3.2 Breakpoint Construction

For a given NGBA $\mathcal{B} = (\Sigma, Q, I, \mathrm{T}, \mathbf{F}_k)$, we denote with $\mathsf{bp}(Q,k) = \{\, (R,j,C) \mid C \subsetneq R \subseteq Q,\ j \in [1..k] \,\}$ the breakpoint set. Intuitively, for a state d of the DRA $\mathcal{A} = \det(\mathcal{B})$, the corresponding breakpoint state is $\langle d \rangle = (R,j,C)$ where R contains the states labelling the root ε of labelled ordered tree associated to d, C subsumes the states labelling the lower levels of the tree, and j is the index of the accepting set F_j considered at the root of the tree.

We build two DRAs $\mathcal{BP}^o = (\Sigma, \mathsf{bp}(Q,k), (I,1,\emptyset), \mathrm{T}', \{(\mathrm{A}_\varepsilon, \emptyset), (\mathrm{T}', \mathrm{R}_0)\})$ and $\mathcal{BP}^u = (\Sigma, \mathsf{bp}(Q,k), (I,1,\emptyset), \mathrm{T}', \{(\mathrm{A}_\varepsilon, \emptyset)\})$, called the *breakpoint automata*, as follows.

From the breakpoint state (R,j,C), let $R' = \mathrm{T}(R,\sigma)$ and $C' = \mathrm{T}(C,\sigma) \cup \mathrm{F}_j(R,\sigma)$. Then an accepting transition with letter σ reaches $(R', j \oplus_k 1, \emptyset)$ if $C' = R'$. Formally,

$$\mathrm{A}_\varepsilon = \{\, ((R,j,C),\sigma,(R',j\oplus_k 1,\emptyset)) \mid (R,j,C) \in \mathsf{bp}(Q,k),\ \sigma \in \Sigma,\ \emptyset \neq R' = \mathrm{T}(R,\sigma),\ C' = \mathrm{T}(C,\sigma) \cup \mathrm{F}_j(R,\sigma),\ C' = R' \,\}.$$

The remaining transitions, for which $C' \neq R'$, are obtained in a similar way, but now the transition reaches (R', j, C'), where j remains unchanged; formally,

$$\mathrm{T}'' = \{\, ((R,j,C),\sigma,(R',j,C')) \mid (R,j,C) \in \mathsf{bp}(Q,k),\ \sigma \in \Sigma,\ \emptyset \neq R' = \mathrm{T}(R,\sigma),\ C' = \mathrm{T}(C,\sigma) \cup \mathrm{F}_j(R,\sigma),\ C' \neq R' \,\}.$$

The transition relation T' is just $\mathrm{T}'' \cup \mathrm{A}_\varepsilon$. Transitions that satisfy $C' = \emptyset$ are rejecting:

$$\mathrm{R}_0 = \{\, ((R,j,C),\sigma,d) \in \mathrm{T}'' \mid \mathrm{T}(C,\sigma) = \emptyset \,\}.$$

Figure 5 shows the reachable fragment of the breakpoint construction for the NGBA $\mathcal{B}_\mathcal{E}$ depicted in Figure 1. The double arrow transitions are in A_ε while the remaining transitions are in R_0.

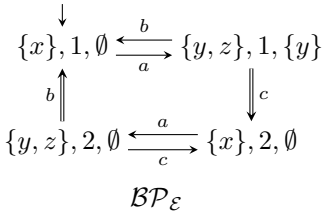


Figure 5 The breakpoint construction for $\mathcal{B}_\mathcal{E}$ (fragment reachable from $(\{x\}, 1, \emptyset)$)

▶ **Theorem 15.** *The following inclusions hold:*

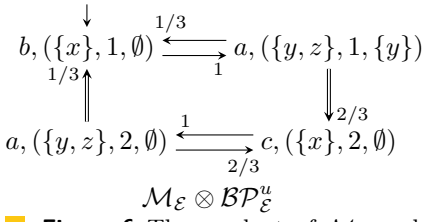
$$\mathcal{L}(\mathcal{S}^u_{[d]}) \subseteq \mathcal{L}(\mathcal{BP}^u_{\langle d \rangle}) \subseteq \mathcal{L}(\mathcal{A}_d) \subseteq \mathcal{L}(\mathcal{BP}^o_{\langle d \rangle}), \mathcal{L}(\mathcal{S}^o_{[d]}).$$

We remark that the breakpoint construction can be refined further such that it is finer than $\mathcal{L}(\mathcal{S}^o_{[d]})$. However we leave it as future work to avoid heavy technical preparations. Exploiting the above theorem, the following becomes clear.

▶ **Corollary 16.** *Let* $\mathtt{S}$ *be a bottom SCC of the quotient MC. Let* $(m,d) \in \mathtt{S}$ *be an arbitrary state of* $\mathtt{S}$*. Moreover, let* $\mathcal{BP}^o$*,* $\mathcal{BP}^u$ *be the breakpoint automata. Then,*

- $\mathtt{S}$ *is accepting if there exists a bottom SCC* $\mathtt{S}'$ *in* $\mathcal{M} \otimes \mathcal{BP}^u_{\langle d \rangle}$ *with* $\mathtt{S} = [\mathtt{S}']$*, which is accepting (i.e.,* $\mathtt{S}'$ *contains some transition in* A_ε*).*
- $\mathtt{S}$ *is rejecting if there exists a bottom SCC* $\mathtt{S}'$ *in* $\mathcal{M} \otimes \mathcal{BP}^o_{\langle d \rangle}$ *with* $\mathtt{S} = [\mathtt{S}']$*, which is rejecting (i.e.,* $\mathtt{S}'$ *contains no transition in* A_ε*, but some transition in* R_0*).*

Note that the products $\mathcal{M} \otimes \mathcal{BP}^u_{\langle d \rangle}$ and $\mathcal{M} \otimes \mathcal{BP}^o_{\langle d \rangle}$ are the same RMCs except for their accepting conditions. Figure 6 shows the product of the MC $\mathcal{M}_\mathcal{E}$ depicted in Figure 2 and the breakpoint automaton $\mathcal{BP}^u_\mathcal{E}$ in Figure 5. It is easy to see that the only bottom SCC is accepting.



$\mathcal{M}_\mathcal{E} \otimes \mathcal{BP}^u_\mathcal{E}$

Figure 6 The product of $\mathcal{M}_\mathcal{E}$ and $\mathcal{BP}^u_\mathcal{E}$

Together with Corollary 11, Lemma 14 and Corollary 16 immediately provide the following result, which justifies the incremental evaluations of the bottom SCCs.

▶ **Corollary 17.** *Given a MC $\mathcal{M}$, a NGBA $\mathcal{B}$, and $\mathcal{A} = \det(\mathcal{B})$, if $[(m,d)]$ is a state in a bottom SCC of the quotient MC and $[d] = [d']$, then*

- $\mathfrak{P}^{\mathcal{M}\otimes\mathcal{A}_d}_{(m,d)}(\mathcal{B}) = 1$ *if* $\mathfrak{P}^{\mathcal{M}\otimes\mathcal{S}^u_{[d']}}_{(m,[d])}(\mathcal{B}) > 0$ *or* $\mathfrak{P}^{\mathcal{M}\otimes\mathcal{BP}^u_{\langle d' \rangle}}_{(m,\langle d \rangle)}(\mathcal{B}) > 0$, *and*
- $\mathfrak{P}^{\mathcal{M}\otimes\mathcal{A}_d}_{(m,d)}(\mathcal{B}) = 0$ *if* $\mathfrak{P}^{\mathcal{M}\otimes\mathcal{S}^o_{[d']}}_{(m,[d])}(\mathcal{B}) < 1$ *or* $\mathfrak{P}^{\mathcal{M}\otimes\mathcal{BP}^o_{\langle d' \rangle}}_{(m,\langle d \rangle)}(\mathcal{B}) < 1$.

In case there are remaining bottom SCCs, for which we cannot conclude whether they are accepting or rejecting, we continue with a multi-breakpoint construction that is language-equivalent to the Rabin construction.

3.3.3 Multi-Breakpoint Construction

The multi-breakpoint construction we propose to decide the remaining bottom SCCs makes use of a combination of the subset and breakpoint constructions we have seen in the previous steps, but with different accepting conditions: for the subset automaton $\mathcal{S} = \mathcal{S}(\mathcal{B}) = (\Sigma, Q_{ss}, q_{ss}, \mathrm{T}_{ss}, \mathrm{F}_{ss})$, we use the accepting condition $\mathrm{F}_{ss} = \emptyset$, i.e., the automaton accepts no words; for the breakpoint automaton $\mathcal{BP} = \mathcal{BP}(\mathcal{B}) = (\Sigma, Q_{bp}, q_{bp}, \mathrm{T}_{bp}, \mathrm{F}_{bp})$, we consider $\mathrm{F}_{bp} = \mathrm{A}_\varepsilon$. Note that the Büchi acceptance condition $\mathrm{F}_{bp} = \mathrm{A}_\varepsilon$ is trivially equivalent to the Rabin acceptance condition $\{(\mathrm{A}_\varepsilon, \emptyset)\}$, so $\mathcal{BP}$ is essentially $\mathcal{BP}^u$. We remark that in general the languages accepted by $\mathcal{S}$ and $\mathcal{BP}$ are different from $\mathcal{L}(\mathcal{B})$: $\mathcal{L}(\mathcal{S}) = \emptyset$ by construction while $\mathcal{L}(\mathcal{BP}) \subseteq \mathcal{L}(\mathcal{B})$, as shown in Theorem 15. To generate an automaton accepting the same language of $\mathcal{B}$, we construct a *semi-deterministic* automaton $\mathcal{SD} = \mathcal{SD}(\mathcal{B}) = (\Sigma, Q_{sd}, q_{sd}, \mathrm{T}_{sd}, \mathrm{F}_{sd})$ by merging $\mathcal{S}$ and $\mathcal{BP}$ as follows: $Q_{sd} = Q_{ss} \cup Q_{bp}$, $q_{sd} = q_{ss}$, $\mathrm{T}_{sd} = \mathrm{T}_{ss} \cup \mathrm{T}_t \cup \mathrm{T}_{bp}$, and $\mathrm{F}_{sd} = \mathrm{F}_{bp}$, where $\mathrm{T}_t = \{\, (R, \sigma, (R', j', C')) \mid R \in Q_{ss}, (R', j', C') \in Q_{bp}$, and $R' \subseteq \mathrm{T}_{ss}(R, \sigma) \,\}$. $\mathcal{B}$ and $\mathcal{SD}$ accept the same language:

▶ **Proposition 18.** *Given a NGBA $\mathcal{B}$, let $\mathcal{SD}$ be constructed as above. Then, $\mathcal{L}(\mathcal{SD}) = \mathcal{L}(\mathcal{B})$.*

For $\mathcal{A} = \det(\mathcal{B})$, it is known by Lemma 13 that $\mathcal{M} \times \mathcal{S}$ and $\mathcal{M} \times \mathcal{A}$ are strictly related, so we can define the accepting SCC of $\mathcal{M} \times \mathcal{S}$ by means of the accepting states of $\mathcal{M} \times \mathcal{A}$.

▶ **Definition 19.** Given a MC $\mathcal{M}$ and a NGBA $\mathcal{B}$, for $\mathcal{S} = \mathcal{S}(\mathcal{B})$ and $\mathcal{A} = \det(\mathcal{B})$, we say that a bottom SCC $\mathtt{S}$ of $\mathcal{M} \times \mathcal{S}$ is accepting if, and only if, there exists a state $s = (m, d)$ in an accepting bottom SCC $\mathtt{S}'$ of $\mathcal{M} \times \mathcal{A}$ such that $(m, \mathsf{rchd}(d)) \in \mathtt{S}$.

Note that Proposition 8 ensures that the accepting SCCs of $\mathcal{M} \times \mathcal{S}$ are well defined.

▶ **Theorem 20.** *Given a MC $\mathcal{M}$ and a NGBA $\mathcal{B}$, for $\mathcal{SD} = \mathcal{SD}(\mathcal{B})$ and $\mathcal{S} = \mathcal{S}(\mathcal{B})$, the following facts are equivalent:*

1. $\mathtt{S}$ *is an accepting bottom SCC of $\mathcal{M} \times \mathcal{S}$;*
2. *there exist $(m, R) \in \mathtt{S}$ and $R' \subseteq R$ such that $(m, (R', j, \emptyset))$ belongs to an accepting SCC of $\mathcal{M} \otimes \mathcal{SD}_{(m,(R',j,\emptyset))}$ for some $j \in [1..k]$;*

3. *there exist* $(m, R) \in \mathtt{S}$ *and* $q \in R$ *such that* $(m, (\{q\}, 1, \emptyset))$ *reaches with probability* 1 *an accepting SCC of* $\mathcal{M} \otimes \mathcal{SD}_{(m,(\{q\},1,\emptyset))}$.

Theorem 20 provides a practical way to check whether an SCC $\mathtt{S}$ of $\mathcal{M} \times \mathcal{S}$ is accepting: it is enough to check whether some state (m, R) of $\mathtt{S}$ has $R \supseteq R'$ for some $(m, (R', j, \emptyset))$ in the accepting region of $\mathcal{M} \otimes \mathcal{SD}$, or whether, for a state $q \in R$, $(m, (\{q\}, 1, \emptyset))$ reaches with probability 1 the accepting region. We remark that, by construction of $\mathcal{SD}$, if we change the initial state of $\mathcal{SD}$ to (R, j, C)—i.e., if we consider $\mathcal{SD}_{(R,j,C)}$—then the run can only visit breakpoint states; i.e., it is actually a run of $\mathcal{BP}_{(R,j,C)}$.

4 Markov Decision Processes

The lazy determinisation approach proposed in this paper extends to Markov decision processes (MDPs) after minor adaptation; Markov chains have mainly been used for ease of notation. We give here an outline of the adaptation with a focus on the differences and particularities that need to be taken into consideration when we are dealing with MDPs.

An MDP is a tuple $\mathcal{M} = (M, L, \mathit{Act}, \mu_0, \mathrm{P})$ where M, L, and μ_0 are as for Markov chains, Act is a finite set of actions, and $\mathrm{P} \colon M \times \mathit{Act} \to \mathit{Dist}(M)$ is the transition probability function where $\mathit{Dist}(M)$ is the set of distributions over M. The nondeterministic choices are resolved by a scheduler υ that chooses the next action to be executed depending on a finite path. Like for Markov chains, the principal technique to analyse MDPs against a specification φ is to construct a deterministic Rabin automaton $\mathcal{A}$, build the product $\mathcal{M} \otimes \mathcal{A}$, and analyse it. This product will be referred to as a *Rabin MDP* (RMDP). According to [3], for a RMDP, it suffices to consider memoryless deterministic schedulers of the form $\upsilon \colon M \times Q \to \mathit{Act}$, where Q is the set of states of $\mathcal{A}$. Given a NGBA specification $\mathcal{B}_\varphi$, we are interested in $\sup_\upsilon \mathfrak{P}^{\mathcal{M},\upsilon}(\mathcal{B}_\varphi)$. In particular, one can use finite memory schedulers on $\mathcal{M}$. (Schedulers that control $\mathcal{M}$ can be used to control $\mathcal{M} \otimes \mathcal{A}$ for all deterministic automata $\mathcal{A}$.) The superscript $\mathcal{M}$ is omitted when it is clear from the context. We remark that the infimum can be treated accordingly, as $\inf_\upsilon \mathfrak{P}^\upsilon(\mathcal{B}_\varphi) = 1 - \sup_\upsilon \mathfrak{P}^\upsilon(\mathcal{B}_{\neg\varphi})$.

As Proposition 8 operates on words, it immediately extends to MDPs. Under the corresponding equivalence relation we obtain a *quotient MDP*. From here, it is clear that we can use the estimation of the word languages provided in Theorem 15 to estimate $\sup_\upsilon \mathfrak{P}^\upsilon(\mathcal{B}_\varphi)$.

▶ **Corollary 21.** *Given an MDP* $\mathcal{M}$ *and a NGBA* $\mathcal{B}$*, let* m *be a state of* $\mathcal{M}$ *and* d, d' *be states of* $\mathcal{A} = \det(\mathcal{B})$ *with* $[d] = [d']$*. Then* $\sup_\upsilon \mathfrak{P}^\upsilon_{(m,[d])}(\mathcal{S}^u_{[d]}) \leq \sup_\upsilon \mathfrak{P}^\upsilon_{(m,\langle d\rangle)}(\mathcal{BP}^u_{\langle d\rangle}) \leq \sup_\upsilon \mathfrak{P}^\upsilon_{(m,d)}(\mathcal{A}_d) = \sup_\upsilon \mathfrak{P}^\upsilon_{(m,d')}(\mathcal{A}_{d'}) \leq \sup_\upsilon \mathfrak{P}^\upsilon_{(m,\langle d\rangle)}(\mathcal{BP}^o_{\langle d\rangle})$*,* $\sup_\upsilon \mathfrak{P}^\upsilon_{(m,[d])}(\mathcal{S}^o_{[d]})$ *holds.*

In the standard evaluation of RMDP, the *end components* of the product $\mathcal{M} \otimes \mathcal{A}$ play a role comparable to the one played by bottom SCCs in MCs. An end component (EC) is simply a sub-MDP, which is closed in the sense that there exists a memoryless scheduler υ such that the induced Markov chain is a bottom SCC. If there is a scheduler that additionally guarantees that a run that contains all possible transitions infinitely often is accepting, then the EC is *accepting*. Thus, one can stay in the EC and traverse all of its transitions (that the scheduler allows) infinitely often, where acceptance is defined as for BSCCs in MCs.

▶ **Theorem 22.** *Given an MDP* $\mathcal{M}$ *and a NGBA* $\mathcal{B}$*, for* $\mathcal{A} = \det(\mathcal{B})$*,* $\mathcal{SD} = \mathcal{SD}(\mathcal{B})$*, and* $\mathcal{S} = \mathcal{S}(\mathcal{B})$*, if* $\mathtt{C}$ *is an accepting EC of* $\mathcal{M} \otimes \mathcal{A}$*, then (1.)* $[\mathtt{C}]$ *is an EC of* $\mathcal{M} \times \mathcal{S}$ *and (2.)* $\mathtt{C}' = \langle \mathtt{C} \rangle$ *is an accepting EC of* $\mathcal{M} \otimes \mathcal{SD}$*.* $\mathtt{C}'$ *contains a state* $(m, (R, 1, \emptyset))$ *with* $R \subseteq [d]$ *and* $(m, d) \in \mathtt{C}$*.*

Note that, since each EC $\mathtt{C}$ of $\mathcal{M} \otimes \mathcal{A}$ is either accepting or rejecting, finding an accepting EC $\mathtt{C}' = \langle \mathtt{C} \rangle$ of $\mathcal{M} \otimes \mathcal{SD}$ allows us to derive that $\mathtt{C}$ is accepting as well.

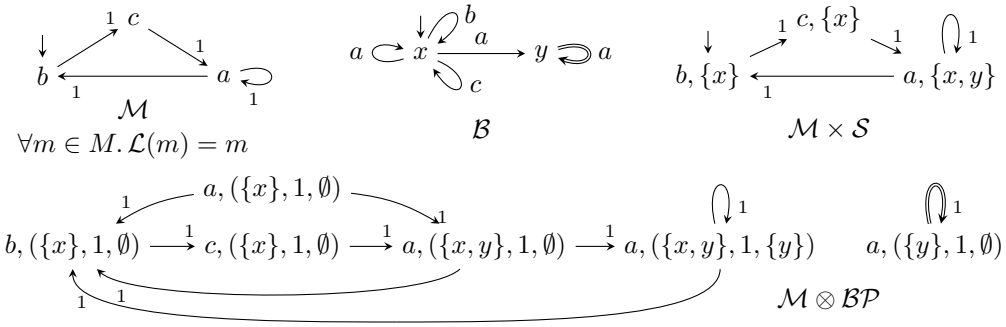


Figure 7 Finding accepting ECs in MDPs: MDP $\mathcal{M}$, NGBA $\mathcal{B}$, $\mathcal{S} = \mathcal{S}(\mathcal{B})$, $\mathcal{BP} = \mathcal{BP}(\mathcal{B})$

For RMDPs, it suffices to analyse maximal end components (MEC). We define a MEC as accepting if it contains an accepting EC. MECs are easy to construct and, for each accepting pair, they are easy to evaluate: it suffices to remove the rejecting transitions, repeat the construction of MECs on the remainder, and check if there is any that contains an accepting transition. Once accepting MECs are determined, their states are assigned a winning probability of 1, and evaluating the complete MDP reduces to a maximal reachability analysis, which reduces to solving an LP problem. It can therefore be solved in polynomial time.

These two theorems allow us to use a layered approach of lazy determinisation for MDPs, which is rather similar to the one described for Markov chains. We start with the quotient MDP, and consider an arbitrary MEC $\mathtt{C}$. By using the accepting conditions of the subset automata $\mathcal{S}^u$ and $\mathcal{S}^o$, we check whether $\mathtt{C}$ is accepting or rejecting, respectively. If this test is inconclusive, we first refine $\mathtt{C}$ by a breakpoint construction, and finally by a multi-breakpoint construction. We remark that, as for Markov chains, the breakpoint and multi-breakpoint constructions can be considered as oracles: when we have identified the accepting MECs, a plain reachability analysis is performed on the quotient MDP.

Theorem 22 makes clear what needs to be calculated in order to classify an EC—and thus a MEC—as accepting, while Corollary 21 allows for applying this observation in the quantitative analysis of an MDP, and also to smoothly combine this style of reasoning with the lazy approach. This completes the picture of [6] for the quantitative analysis of MDPs, which is technically the same as their analysis of concurrent probabilistic programs [6]. It is worthwhile to point out that, in principle, the qualitative analysis from [6] could replace Theorem 22 when starting with a Büchi automaton that recognises the complement of the models of φ and minimising instead of maximising. This detour would, however, not allow us to restrict the analysis to (M)ECs, which would, in turn, lead to a significant overhead.

For MDPs, differently from the subset and breakpoint construction, for the multi-breakpoint case testing only one $(m, R) \in \mathtt{C}$ in general is not sufficient; consider the MDP $\mathcal{M}$ and the NGBA $\mathcal{B}$ depicted in Figure 7. We first consider the product MDP $\mathcal{M} \times \mathcal{S}$, containing one MEC. We first try to decide whether it is accepting by considering the state $(c, \{x\})$. The only nonempty subset of $\{x\}$ is the set itself, thus we look for accepting MECs in $\mathcal{M} \otimes \mathcal{BP}_{(c,(\{x\},1,\emptyset))}$. It is clear that from $(c, (\{x\}, 1, \emptyset))$ no accepting MECs can be reached. In contrast to the MC setting, we cannot conclude that the original MEC is not accepting. Instead, we remove $(c, \{x\})$ from the set of states to consider, as well as $(b, \{x\})$, from which we cannot avoid reaching $(c, \{x\})$. The state left to try is $(a, \{x, y\})$, where we have two transitions available. Indeed, in $\mathcal{M} \otimes \mathcal{BP}$ the singleton MEC $\{(a, (\{y\}, 1, \emptyset))\}$ is accepting. Thus the MEC of $\mathcal{M} \times \mathcal{S}$ is accepting, though only one of its states—$\{(a, \{x, y\})\}$—allows us to conclude this, and we need to select the correct subset, $\{y\}$, to start with.

Table 1 Runtime comparison for the randomised mutual exclusion protocol

property		n	time BP expl.	BP BDD	RB expl.	RB BDD	PRISM	Rabinizer3	scaled [4]
$\mathbb{P}_{\min=?}(\mathbf{GF}p_1{=}10 \wedge \mathbf{GF}p_2{=}10 \wedge \mathbf{GF}\,p_3{=}10 \wedge \mathbf{GF}p_4{=}10)$	(3)	4	**3**	5	15	28	–	104	23
		5	**19**	21	–	104	–	1478	380
$\mathbb{P}_{\max=?}((\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_2{\neq}0) \wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_3{\neq}0))$	(4)	3	**1**	2	2	4	138	2	**1**
		4	**3**	7	4	15	–	20	18
		5	**19**	32	35	76	–	319	299
$\mathbb{P}_{\max=?}((\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_1{\neq}0) \wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_2{\neq}0))$	(5)	3	2	2	2	4	41	2	**1**
		4	**3**	8	4	17	336	19	18
		5	**19**	34	45	68	–	314	289
$\mathbb{P}_{\max=?}((\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_2{\neq}0) \wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_3{\neq}0) \wedge(\mathbf{GF}p_3{=}0 \vee \mathbf{FG}p_1{\neq}0))$	(6)	3	**1**	2	2	6	–	5	4
		4	**3**	9	7	27	–	52	47
		5	**29**	38	99	124	–	871	762
$\mathbb{P}_{\max=?}((\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_1{\neq}0) \wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_2{\neq}0) \wedge(\mathbf{GF}p_3{=}0 \vee \mathbf{FG}p_3{\neq}0))$	(7)	3	**1**	2	2	9	–	5	5
		4	**3**	9	12	41	–	50	49
		5	**29**	38	–	171	–	849	792
$\mathbb{P}_{\min=?}((\mathbf{GF}p_1{\neq}10 \vee \mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_1{=}1) \wedge \mathbf{GF}\,p_1{\neq}0 \wedge \mathbf{GF}p_1{=}1)$	(8)	3	**1**	2	**1**	3	**1**	**1**	**1**
		4	**3**	6	**3**	10	8	13	6
		5	**17**	25	**17**	41	123	208	91
$\mathbb{P}_{\max=?}((\mathbf{G}p_1{\neq}10 \vee \mathbf{G}p_2{\neq}10 \vee \mathbf{G}p_3{\neq}10) \wedge(\mathbf{FG}p_1{\neq}1 \vee \mathbf{GF}p_2{=}1 \vee \mathbf{GF}p_3{=}1) \wedge(\mathbf{FG}p_2{\neq}1 \vee \mathbf{GF}p_1{=}1 \vee \mathbf{GF}p_3{=}1))$	(9)	3	**2**	6	**2**	4	–	982	50
		4	9	16	**7**	14	–	1718	440
		5	136	60	91	**56**	–	–	–
$\mathbb{P}_{\min=?}((\mathbf{FG}p_1{\neq}0 \vee \mathbf{FG}p_2{\neq}0 \vee \mathbf{GF}p_3{=}0) \vee(\mathbf{FG}p_1{\neq}10 \wedge \mathbf{GF}p_2{=}10 \wedge \mathbf{GF}p_3{=}10)$	(10)	3	**2**	3	**2**	5	169	3	**2**
		4	79	12	**4**	18	–	32	21
		5	–	48	**44**	69	–	480	339

5 Implementation and Results

We have implemented our approach in our IscasMC tool [10] in both explicit and BDD-based symbolic versions. We use LTL formulas to specify properties, and apply SPOT [7] to translate them to NGBAs. Our experimental results suggest that our technique provides a practical approach for checking LTL properties for probabilistic systems. A web interface to IscasMC can be found at `http://iscasmc.ios.ac.cn/`. For our experiments, we used a 3.6 GHz Intel Core i7-4790 with 16GB 1600 MHz DDR3 RAM.

We consider a set of properties analysed previously in [4]. As there, we aborted tool runs when they took more than 30 minutes or needed more than 4GB of RAM. The comparison with the results from [4] cannot be completely accurate: unfortunately, their implementation is not available on request to the authors, and for their results they did not state the exact speed of the machine used. By comparing the runtimes stated for PRISM in [4] with the corresponding runtimes we obtained on our machine, we estimate that our machine is faster than theirs by about a factor of 1.6. Thus, we have included the values from [4] divided by 1.6 to take into account the estimated effect of the machine. In Table 1 we provide the results obtained. Here, "property" and "n" are as in [4] and depict the property and the size of the model under consideration. We report the total runtime in seconds ("time") for the explicit-state ("BP expl.") and the BDD-based symbolic ("BP BDD") implementations of the multi-breakpoint construction, as well as the explicit and symbolic ("RB expl.", "RB BDD") of the Rabin-based implementation. In both BP and RB cases, we first apply the subset and breakpoint steps. We also include the runtimes of PRISM ("PRISM") and of the tool used in [4] ("scaled [4]") developed for a subclass of LTL formulas and its generalisation to full LTL [8] implemented in Rabinizer 3 [13] ("Rabinizer 3"). We mark the best (rounded) runtimes with bold font.

The runtime of our approaches is almost always better than the runtime of the other

Table 2 Runtime comparison for the workstation cluster protocol

	time					
property	BP expl.	BP BDD	RB expl.	RB BDD	PRISM	Rabinizer3
$prop\mathbf{U}_{10}$	**2**	3	**2**	3	23	121
$prop\mathbf{U}_{11}$	**3**	4	**3**	4	95	686
$prop\mathbf{U}_{12}$	**4**	5	**4**	5	–	–
$prop\mathbf{U}_{13}$	**7**	8	**7**	8	–	–
$prop\mathbf{GF}\wedge_3$	**1**	**1**	**1**	**1**	48	**1**
$prop\mathbf{GF}\wedge_4$	2	**1**	**1**	**1**	–	2
$prop\mathbf{GF}\wedge_5$	**1**	**1**	**1**	2	–	14
$prop\mathbf{GF}\wedge_6$	**1**	**1**	**1**	**1**	–	177
$prop\mathbf{GF}\vee_3$	**1**	**1**	2	3	233	**1**
$prop\mathbf{GF}\vee_4$	**1**	**1**	2	3	–	2
$prop\mathbf{GF}\vee_5$	**1**	2	**1**	2	–	14
$prop\mathbf{GF}\vee_6$	**2**	**2**	**2**	4	–	180

methods. In many cases, the multi-breakpoint approach performs better than the Rabin-based approach (restricted to the single undecided end component), but not always. Broadly speaking, this can happen when the breakpoint construction has to consider many subsets as starting points for one end component, while the Rabin determinisation does not lead to a significant overhead compared to the breakpoint construction. Thus, both methods are of value. Both of them are faster than the specialised algorithm of [4] and Rabinizer 3. We assume that one reason for this is that this method is not based on the evaluation of end components in the subset product, and also its implementation might not involve some of the optimisations we apply. In most cases, the explicit-state implementation is faster than the BDD-based approach, which is, however, more memory-efficient.

As another case study, we consider a model [12] of two clusters of n=16 workstations each, so that the two clusters are connected by a backbone. Each of the workstations may fail with a given rate, as may the backbone. Though this case study is a continuous-time Markov chain, we focused on time-unbounded properties, such that we could use discrete-time Markov chains to analyse them. We give the results in Table 2, where the meaning of the columns is as for the mutual exclusion case in Table 1. The properties $prop\mathbf{U}_k = \mathbb{P}_{=?}(\mathit{left\_n{=}n}\,\mathbf{U}\,(\mathit{left\_n{=}n{-}1}\,\mathbf{U}\,(\ldots\mathbf{U}\,(\mathit{left\_n{=}n{-}k}\,\mathbf{U}\,\mathit{right\_n{\neq}n})\ldots)$ are probabilities of the event of component failures with respect to the order (first k failures on left before right) while the properties $prop\mathbf{GF}\wedge_k = \mathbb{P}_{=?}(\mathbf{GF}\mathit{left\_n{=}n}\wedge\bigvee_{i=0}^{k}\mathbf{FG}\mathit{right\_n{=}n{-}i})$ and $prop\mathbf{GF}\vee_k = \mathbb{P}_{=?}(\mathbf{GF}\mathit{left\_n{=}n}\vee\bigvee_{i=0}^{k}\mathbf{FG}\mathit{right\_n{=}n{-}i})$ describe the long-run number of workstations functional. As clearly shown from the results in the table, IscasMC outperforms PRISM and Rabinizer 3 all cases, in particular for large PLTL formulas. It is worthwhile to analyse in details the three properties and how they have been checked: for the $prop\mathbf{U}_k$ case, the subset construction suffices and returns a (rounded) probability value of 0.509642; for $prop\mathbf{GF}\wedge_k$, the breakpoint construction is enough to determine that the property holds with probability 0. This explains why the BP and RB columns are essentially the same (we remark that the reported times are the rounded actual runtimes). Property $prop\mathbf{GF}\vee_k$, instead, requires to use the multi-breakpoint or the Safra-based construction to complete the model checking analysis and obtain a probability value of 1.

Acknowledgement.

This work is supported by the Natural Science Foundation of China (NSFC) under grant No. 61472406, 61472473, 61361136002; the Chinese Academy of Sciences Fellowship for In-

ternational Young Scientists under grant No. 2013Y1GB0006, 2015VTC029; the Research Fund for International Young Scientists (Grant No. 61450110461); the CAS/SAFEA International Partnership Program for Creative Research Teams; the EPSRC through grant EP/M027287/1. We thank Jan Kretínský for providing us the source code of Rabinizer 3.

References

1 Tomás Babiak, Mojmír Kretínský, Vojtech Rehák, and Jan Strejcek. LTL to Büchi automata translation: Fast and more deterministic. In *TACAS*, volume 7214 of *LNCS*, pages 95–109, 2012.

2 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

3 Andrea Bianco and Luca de Alfaro. Model checking of probabalistic and nondeterministic systems. In *FSTTCS*, volume 1026 of *LNCS*, pages 499–513, 1995.

4 Krishnendu Chatterjee, Andreas Gaiser, and Jan Kretínský. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In *CAV*, volume 8044 of *LNCS*, pages 559–575, 2013.

5 Frank Ciesinski and Christel Baier. LiQuor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *QEST*, pages 131–132, 2006.

6 Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *JACM*, 42(4):857–907, 1995.

7 Alexandre Duret-Lutz. LTL translation improvements in SPOT. In *VECoS*, pages 72–83, 2011.

8 Javier Esparza and Jan Kretínský. From LTL to deterministic automata: A Safraless compositional approach. In *CAV*, volume 8559 of *LNCS*, pages 192–208, 2014.

9 Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. Lazy determinisation for quantitative model checking. CoRR, available at `http://arxiv.org/abs/1311.2928`, 2014.

10 Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. IscasMC: A web-based probabilistic model checker. In *FM*, volume 8442 of *LNCS*, pages 312–317, 2014.

11 Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *FAC*, 6(5):512–535, 1994.

12 Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. On the use of model checking techniques for dependability evaluation. In *SRDS*, pages 228–237, 2000.

13 Zuzana Komárková and Jan Kretínský. Rabinizer 3: Safraless translation of LTL to small deterministic automata. In *ATVA*, volume 8837 of *LNCS*, pages 235–241, 2014.

14 Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. In *CAV*, volume 4144 of *LNCS*, pages 31–44, 2006.

15 Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.

16 Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *JLMCS*, 3(3:5), 2007.

17 Shmuel Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327, 1988.

18 Sven Schewe. Tighter bounds for the determinisation of Büchi automata. In *FoSSaCS*, volume 5504 of *LNCS*, pages 167–181, 2009.

19 Sven Schewe and Thomas Varghese. Tight bounds for the determinisation and complementation of generalised Büchi automata. In *ATVA*, volume 7561 of *LNCS*, pages 42–56, 2012.

20 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344, 1986.