Verifying Qualitative Properties of Probabilistic Programs

Benedikt Bollig[1]* and Martin Leucker[2]**

[1] Lehrstuhl für Informatik II, RWTH Aachen, Germany
bollig@informatik.rwth-aachen.de
[2] IT department, Uppsala University, Sweden
Martin.Leucker@it.uu.se

Abstract. In this chapter, we present procedures for checking linear temporal logic and automata specifications of sequential and concurrent probabilistic programs. We follow two different approaches: For LTL and sequential probabilistic programs, our method proceeds in a tableau style fashion, while the remaining procedures are based on automata theory.

1 Introduction

Randomization techniques have been employed to solve numerous problems of computing both sequentially and in parallel. Examples of probabilistic algorithms that are asymptotically better than their deterministic counterparts in solving various fundamental problems abound. It has also been shown that they allow solutions of problems which cannot be solved deterministically [7]. They have the advantages of simplicity and better performance both in theory and often in practice. An overview of the domain of randomized algorithms is given in [12].

As for any kind of hardware or software system, it is important to develop formal methods and tools for verifying their correctness, thus also for probabilistic programs. *Model checking*, introduced independently in [2] and [14], turned out to be one fruitful approach to automatically verify systems (see [3] for an overview of model checking in practice). In the model-checking approach, usually a finite system $\mathcal{M}$, often an abstraction of a real system, and a property, usually expressed as a temporal-logic formula φ or as an automaton describing the computations that adhere to the property, are given. The *model-checking procedure* decides whether the set or tree formed of all *computations* of $\mathcal{M}$ *satisfy* φ, or, in other words, whether the given system satisfies the required property. Temporal logics used for expressing requirements are usually linear temporal logic (LTL) (as initially proposed by Pnueli in [10]) or branching time logics like the computation-tree logics CTL and CTL*.

* Part of this work was done during the author's stay at the IT department, Uppsala University, Sweden. He is grateful for the hospitality and the overall support.
** This author is supported by the European Research Training Network "Games".

In the probabilistic setting, one is no longer interested in *all* but only *almost all* computations. Thus, (sets or pathes of trees of) computations with zero probability are ignored when deciding whether a given property is satisfied. More general, one is also interested whether a given property is satisfied with some given probability.

In this chapter, we describe methods for checking whether almost all runs of a finite sequential or concurrent probabilistic program in the sense of [5] and [8] satisfy a given LTL formula or all accepting runs of an automaton describing the underlying property. Thus, we are concerned about quality aspects of a given probabilistic program rather than in estimating quantity measures for given properties or in temporal logics allowing to express quantitative aspects of properties. These issues are addressed in one of the subsequent chapters.

We present a procedure for checking LTL specifications of sequential probabilistic programs, which requires running time that is exponential in the size of the formula and linear in the size of the program. It can be shown that this is optimal.

Furthermore, we describe a procedure for checking automata specifications of sequential and concurrent probabilistic programs. It can be shown that this problem can be solved in time polynomial in the size of the program and exponential in the size of the formula. However, to simplify our presentation, we present a less technical construction based on the same idea, however, with a slightly inferior complexity than the one with optimal bounds. As LTL specifications can easily be transformed into automata specifications involving an exponential blow-up [17], the procedure can also be used to check LTL specifications (in time double exponential in the size of the formula).

This chapter is mainly based on [4], in which additionally methods for an extension of LTL—known as ETL [18]—are shown and proofs showing that all constructions are optimal are given. Since we only want to give an introduction to the field of verification of qualitative properties of probabilistic programs, we refer to this paper for a more detailed study of these extensions and proof ideas.

This chapter is organized as follows. In the next section, we introduce the necessary concepts and notation of automata, linear temporal logic, and probabilistic programs. Furthermore, we recall some mathematical background in the domain of graph and probability theory. In Section 3, we describe a model-checking procedure for LTL formulas and sequential probabilistic programs. Section 4 studies this question for automata specifications and sequential as well as concurrent probabilistic programs. We draw our conclusions in Section 5. This chapter ends giving suggestions for further reading in Section 6.

2 Preliminaries

2.1 Words and Graphs

Given an alphabet Σ, Σ^* denotes the set of *finite* and Σ^ω the set of *infinite words* over Σ. For a word $w = a_0 a_1 \ldots \in \Sigma^\omega$ and a natural i, let $w(i)$ denote

$a_i a_{i+1} \ldots$, the ith suffix of w. Furthermore, take $inf(w)$ as the *infinity set* $\{a_i \mid |\{j \mid a_j = a_i\}| = \infty\}$ of letters that occur infinitely often in w.

Given a directed graph G with nodes V and edges E, we call a node $v \in V$ (a set $D \subseteq V$ of nodes) *reachable* from $v' \in V$ if there is a path from v' to v (to a node contained in D). A *strongly connected component* (SCC) of G is a maximal set D of nodes such that every two nodes contained in D are reachable from each other. A SCC is called *bottom* if there is no edge leading out of it. We define G to be *strongly connected* if V forms a SCC. Furthermore, G is called *nontrivial* if it contains at least one edge. A set $D \subseteq V$ is said to be *nontrivial* if $G[D]$, the subgraph of G *induced* by D, is nontrivial. The *size* of G, denoted by $|G|$, is defined to be $|V| + |E|$.

2.2 Automata

Let Σ be an alphabet. An *ω-automaton* over Σ is a tuple $\mathcal{A} = (S, S_0, \delta, Acc)$ where S is its nonempty finite set of *states*, $S_0 \subseteq S$ is the set of *initial states*, $\delta : S \times \Sigma \to 2^S$ is its *transition function*, and Acc is an *acceptance component*. A *run* of $\mathcal{A}$ on a word $w = a_1 a_2 \ldots \in \Sigma^\omega$ is a sequence of states $\rho = s_0 s_1 s_2 \ldots \in S^\omega$ such that $s_0 \in S_0$ and, for $i = 0, 1, \ldots$, $s_{i+1} \in \delta(s_i, a_{i+1})$.

An ω-automaton $\mathcal{A} = (S, S_0, \delta, Acc)$ is called *Büchi automaton* if Acc is a set $F \subseteq S$. We then call a run ρ of $\mathcal{A}$ *accepting* if $inf(\rho) \cap F \neq \emptyset$.

A *Streett automaton* (*Rabin automaton*) over Σ differs from a Büchi automaton only in the acceptance component. More precisely, it is a structure $\mathcal{A} = (S, S_0, \delta, \mathcal{F})$ where $\mathcal{F}$ is a subset of $(2^S)^2$. A run ρ of $\mathcal{A}$ is called *accepting* if, for all pairs $(U, V) \in \mathcal{F}$, $inf(\rho) \cap U \neq \emptyset$ implies $inf(\rho) \cap V \neq \emptyset$ (there is a pair $(U, V) \in \mathcal{F}$ such that $inf(\rho) \cap U \neq \emptyset$ and $inf(\rho) \cap V = \emptyset$).

The *language* of an ω-automaton $\mathcal{A} = (S, S_0, \delta, Acc)$, denoted by $L(\mathcal{A})$, is defined to be the set $\{w \in \Sigma^\omega \mid$ there is an accepting run of $\mathcal{A}$ on $w\}$. $\mathcal{A}$ is called *deterministic (quasi-deterministic)* if both $|S_0| = 1$ and $|\delta(s, a)| = 1$ ($|\delta(s, a)| \leq 1$) for all $s \in S$, $a \in \Sigma$. Furthermore, the *size* $|\mathcal{A}|$ of $\mathcal{A}$ is defined to be the size of its (transition) graph.

2.3 Propositional Linear Temporal Logic

In the following, let $\mathcal{P}$ be a nonempty finite set of *propositions*. The set LTL($\mathcal{P}$) of (Propositional) Linear Temporal Logic (LTL) formulas over $\mathcal{P}$ is inductively defined as follows: $\mathcal{P}$ is a subset of LTL($\mathcal{P}$), and, for LTL($\mathcal{P}$) formulas φ and ψ, $\mathcal{X}\varphi$, $\varphi \mathcal{U} \psi$, $\neg\varphi$, and $\varphi \vee \psi$ are LTL($\mathcal{P}$) formulas as well. An LTL($\mathcal{P}$) formula is inductively interpreted over $\pi = \pi_0 \pi_1 \ldots \in (2^\mathcal{P})^\omega$ as follows:

- $\pi \models p \in \mathcal{P}$ if $p \in \pi_0$
- $\pi \models \neg\varphi$ if $\pi \not\models \varphi$
- $\pi \models \varphi \vee \psi$ if $\pi \models \varphi$ or $\pi \models \psi$
- $\pi \models \mathcal{X}\varphi$ if $\pi(1) \models \varphi$
- $\pi \models \varphi \mathcal{U} \psi$ if there is an $i \geq 0$ such that $\pi(i) \models \psi$ and, for each $j \in \{0, \ldots, i-1\}$, $\pi(j) \models \varphi$

The *language* of a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$, denoted by $L(\varphi)$, is defined to be the set $\{\pi \in (2^{\mathcal{P}})^{\omega} \mid \pi \models \varphi\}$.

2.4 Probability Spaces and Probabilistic Programs

A nonempty set of possible outcomes of an experiment of chance is called *sample space*. Let Ω be a sample space. A set $\mathfrak{B} \subseteq 2^{\Omega}$ is called *Borel field* over Ω if it contains Ω, $\Omega \setminus E$ for each $E \in \mathfrak{B}$, and the union of any countable sequence of sets from $\mathfrak{B}$. A Borel field $\mathfrak{B}$ is *generated* by an at most countable set $\mathcal{E}$, denoted by $\mathfrak{B} = \langle \mathcal{E} \rangle$, if $\mathfrak{B}$ is the closure of $\mathcal{E}$'s elements under complement and countable union.

A *probability space* is a triple $\mathcal{PS} = (\Omega, \mathfrak{B}, \mu)$ where Ω is a sample space, $\mathfrak{B}$ is a Borel field over Ω, and μ is a mapping $\mathfrak{B} \to [0,1]$ such that $\mu(\Omega) = 1$ and $\mu(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \mu(E_i)$ for any sequence $E_1, E_2, \ldots$ of pairwise disjoint sets from $\mathfrak{B}$. We call μ a *probability measure*. An event $E \in \mathfrak{B}$ is said to occur *almost surely* if $\mu(E) = 1$.

A *concurrent probabilistic program* over $\mathcal{P}$, the set of propositions, is a tuple $\mathcal{M} = (Q, N, R, \Delta, P_0, P, \mathcal{V})$ where

- Q is its nonempty finite set of *states*, which is partitioned into sets N and R of *nondeterministic* and *randomizing states*, respectively,
- $\Delta \subseteq Q \times Q$ is the set of transitions with $\Delta(q) := \{q' \mid (q,q') \in \Delta\} \neq \emptyset$ for each $q \in Q$,
- $P_0 : Q \to [0,1]$ and $P : \{(q,q') \in \Delta \mid q \in R\} \to (0,1]$ are the *initial* and *transition probability distributions*[1], respectively, where $\sum_{q \in Q} P_0(q) = 1$ and, for each $q \in R$, $\sum_{q' \in \Delta(q)} P_{q,q'} = 1$, and
- $\mathcal{V} : Q \to 2^{\mathcal{P}}$ is the *valuation function*, which is extended to words over Q as expected.

A concurrent probabilistic program over $\{p\}$ is shown in Figure 1. Hereby, the nondeterministic states are represented by circles, whereas the randomizing ones are given by rectangles.

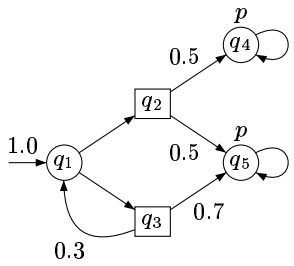

Fig. 1. A concurrent probabilistic program

[1] We usually write $P_{q,q'}$ instead of $P((q,q'))$.

In case that a concurrent probabilistic program $\mathcal{M}$ has no nondeterministic states, i.e., N is the empty set, we call it a *sequential probabilistic program* over $\mathcal{P}$ and identify $\mathcal{M}$ with $(Q, \Delta, P_0, P, \mathcal{V})$. $\mathcal{M}$ induces a probability space $\mathcal{PS}_\mathcal{M} = (\Omega_\mathcal{M}, \mathfrak{B}_\mathcal{M}, \mu_\mathcal{M})$ as follows: $\Omega_\mathcal{M} = \{q_1 q_2 \ldots \in Q^\omega \mid P_0(q_1) > 0$ and, for all $i \geq 1$, $(q_i, q_{i+1}) \in \Delta\}$ is the set of *trajectories* of $\mathcal{M}$, $\mathfrak{B}_\mathcal{M}$ is generated by $\{\mathcal{C}_\mathcal{M}(x) \mid x \in Q^*\}$ where $\mathcal{C}_\mathcal{M}(x) = \{x' \in \Omega_\mathcal{M} \mid x$ is a prefix of $x'\}$ is the *basic cylinder set* of x wrt. $\mathcal{M}$. The measure $\mu_\mathcal{M}$ is uniquely given by $\mu_\mathcal{M}(\Omega_\mathcal{M}) = 1$ and, for $n \geq 1$, $\mu_\mathcal{M}(\mathcal{C}_\mathcal{M}(q_1 \ldots q_n)) = P_0(q_1) \cdot P_{q_1,q_2} \cdot \ldots \cdot P_{q_{n-1},q_n}$ (where we assume $P_{q_i,q_{i+1}}$ to be 0 if $(q_i, q_{i+1}) \notin \Delta$).

A *scheduler* of a concurrent probabilistic program $\mathcal{M} = (Q, N, R, \Delta, P_0, P, \mathcal{V})$ over $\mathcal{P}$ is a mapping $u : Q^*N \to Q$ such that $u(xq) = q'$ implies $(q, q') \in \Delta$.

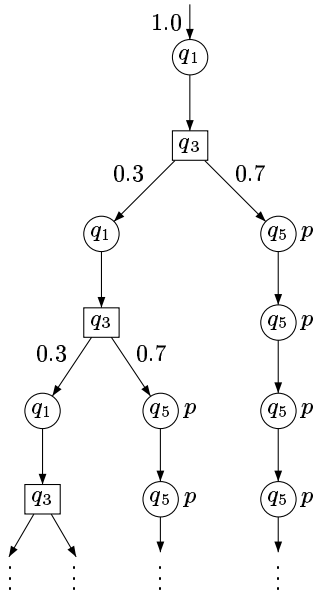

Fig. 2. A scheduler

Similarly to the sequential case, a scheduler u of a concurrent program $\mathcal{M} = (Q, N, R, \Delta, P_0, P, \mathcal{V})$ induces a probability space $\mathcal{PS}_{\mathcal{M},u} = (\Omega_{\mathcal{M},u}, \mathfrak{B}_{\mathcal{M},u}, \mu_{\mathcal{M},u})$ as follows: $\Omega_{\mathcal{M},u} = \{q_1 q_2 \ldots \in Q^\omega \mid P_0(q_1) > 0$ and, for all $i \geq 1$, $(q_i \in R$ and $(q_i, q_{i+1}) \in \Delta)$ or $(q_i \in N$ and $u(q_1 \ldots q_i) = q_{i+1})\}$ is the set of trajectories of $\mathcal{M}$ wrt. u, $\mathfrak{B}_{\mathcal{M},u} = \langle\{\mathcal{C}_{\mathcal{M},u}(x) \mid x \in Q^*\}\rangle$ where $\mathcal{C}_{\mathcal{M},u}(x) = \{x' \in \Omega_{\mathcal{M},u} \mid x$ is a prefix of $x'\}$ is the basic cylinder set of x wrt. $\mathcal{M}$ and u. The measure $\mu_{\mathcal{M},u}$ is uniquely given by $\mu_{\mathcal{M},u}(\Omega_{\mathcal{M},u}) = 1$ and, for $n \geq 1$, $\mu_{\mathcal{M},u}(\mathcal{C}_{\mathcal{M},u}(q_1 \ldots q_n)) = P_0(q_1) \cdot P'_{q_1,q_2} \cdot \ldots \cdot P'_{q_{n-1},q_n}$ where

$$P'_{q_i,q_{i+1}} = \begin{cases} 0 & \text{if } (q_i, q_{i+1}) \notin \Delta \\ P_{q_i,q_{i+1}} & \text{if } (q_i, q_{i+1}) \in \Delta \text{ and } q_i \in R \\ 1 & \text{if } (q_i, q_{i+1}) \in \Delta \text{ and } q_i \in N \end{cases}.$$

When we concentrate on topological aspects of probabilistic programs, we consider a program to be a graph where the states become (nondeterministic or randomizing) nodes and the transitions become edges. In the following, unless the context requires, a program will also denote its graph. From this point of view, we call a state *absorbing* if it forms a (nontrivial) bottom SCC. Otherwise, it is called *transient*.

3 Checking LTL Specifications of Sequential Programs

For a concurrent probabilistic program $\mathcal{M}$ over $\mathcal{P}$ with valuation function $\mathcal{V}$, a scheduler u of $\mathcal{M}$, and a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$, let $L_{\mathcal{M},u}(\varphi) := \{x \in \Omega_{\mathcal{M},u} \mid \mathcal{V}(x) \in L(\varphi)\}$. Similarly, for a sequential program $\mathcal{M}$ with valuation function $\mathcal{V}$, we define $L_{\mathcal{M}}(\varphi) := \{x \in \Omega_{\mathcal{M}} \mid \mathcal{V}(x) \in L(\varphi)\}$. It can be shown, using structural induction, that these sets are measurable:

Proposition 1 ([16]). *Given a concurrent (sequential) probabilistic program $\mathcal{M}$ over $\mathcal{P}$ and a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$, we have $L_{\mathcal{M},u}(\varphi) \in \mathfrak{B}_{\mathcal{M},u}$ for each scheduler u of $\mathcal{M}$ ($L_{\mathcal{M}}(\varphi) \in \mathfrak{B}_{\mathcal{M}}$).*

Definition 1. *Given a formula $\varphi \in \mathrm{LTL}(\mathcal{P})$, a concurrent probabilistic program $\mathcal{M}$ over $\mathcal{P}$ is said to* satisfy *φ if, for all schedulers u of $\mathcal{M}$, $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\varphi)) = 1$ (we say that u* satisfies *φ). A sequential probabilistic program is said to* satisfy *φ if $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\varphi)) = 1$.*

For example, the concurrent probabilistic program $\mathcal{M}$ shown in Figure 1 and, in particular, the scheduler of $\mathcal{M}$ from Figure 2, satisfy the $\mathrm{LTL}(\{p\})$ formula $\Diamond\Box p$. As usual, $\Diamond\varphi$ is an abbreviation for $\mathit{True}\mathcal{U}\varphi$ and $\Box\varphi$ for $\neg\Diamond\neg\varphi$.

Theorem 1. *We can test whether a sequential probabilistic program $\mathcal{M}$ satisfies a formula φ in time $O(|\mathcal{M}|2^{|\varphi|})$, or in space polynomial in φ and polylogarithmic in $\mathcal{M}$. We can compute the probability of satisfaction $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\varphi))$ in time exponential in φ and polynomial in $\mathcal{M}$.*

In the remainder of this section, we proof the previous theorem by formulating a procedure checking whether a sequential probabilistic program $\mathcal{M}$ satisfies an LTL formula φ. The procedure processes φ according to its inductive structure in a bottom-up manner, eliminating its temporal connectives one-by-one and transforms $\mathcal{M}$ in each step to preserve the probability of satisfiability.

To simplify our presentation, we sometimes say that a state satisfies an LTL formula instead of saying that all trajectories starting from this state satisfy this formula.

There are two transformations $\mathcal{T}_{\mathcal{U}}$ and $\mathcal{T}_{\mathcal{X}}$ corresponding to the two temporal connectives $\mathcal{U}$ and $\mathcal{X}$. We describe $\mathcal{T}_{\mathcal{U}}$ in detail and list the modifications for $\mathcal{T}_{\mathcal{X}}$. We illustrate the procedure with the program given in Figure 3.

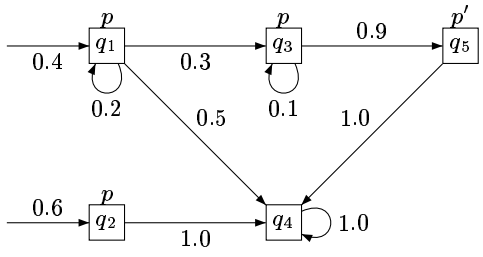

Fig. 3. A sequential probabilistic program

Transformation $\mathcal{T}_{\mathcal{U}}$ Let $p\mathcal{U}p'$ be the innermost subformula of φ.[2] Let W^+ contain all states satisfying p' and W_1^- contain all states satisfying $\neg p$ and $\neg p'$. Let W_2^- comprise all states of bottom strongly connected components (of $\mathcal{M}$ considered as a graph) which have no state in W^+. Thus, states of W_2^- possibly satisfy p but no state satisfying p' is reachable. Let $W^- = W_1^- \cup W_2^-$. Clearly, all trajectories starting in W^+ satisfy $p\mathcal{U}p'$ and all trajectories starting in W^- satisfy $\neg(pUp')$. For our example (Figure 3), we get $W^+ = \{q_5\}$, $W_1^- = W_2^- = \{q_4\}$.

Let Q^+ (Q^-) be the states q such that for all trajectories starting at q the first state in $W^+ \cup W^-$ is in W^+ (W^-). Note that $W^+ \subseteq Q^+$ and $W^- \subseteq Q^-$. For $\mathcal{M}$ of Figure 3, we have $Q^+ = \{q_3, q_5\}$ and $Q^- = \{q_2, q_4\}$.

Almost surely, every trajectory reaches eventually a bottom strongly connected component. Trajectories starting in a state of Q^+ satisfy $p\mathcal{U}p'$ and those starting in a state of Q^- never reach a state satisfying p' before reaching a state not satisfying p or never reach a state satisfying p' at all. Using P_q as a shorthand for the measure of trajectories starting in q satisfying $p\mathcal{U}p'$, we conclude that $P_q = 1$ for all $q \in Q^+$ and $P_q = 0$ for $q \in Q^-$.

Let $Q^?$ comprise the remaining states, so $Q^? = \{q_1\}$ in our example. As $q \in Q^?$ satisfies p but not p', a trajectory π starting in q satisfies $p\mathcal{U}p'$ iff $\pi(1)$ satisfies $p\mathcal{U}p'$. Thus, $P_q = \sum_{q'} P_{q,q'} P_{q'}$. We summarize:

Lemma 1.

$$\begin{aligned} P_q &= \sum_{q'} P_{q,q'} P_{q'} && \text{if } q \in Q^? \\ P_q &= 1 && \text{if } q \in Q^+ \\ P_q &= 0 && \text{if } q \in Q^- \end{aligned}$$

This set of equations has a unique solution.

Proof. It remains to show that the set of equations has a unique solution. Clearly, any two solutions can only differ on $Q^?$. Let us form a new sequential probabilistic program $\mathcal{M}^?$ whose state space consists of $Q^?$ and one additional absorbing state $\hat{q}$ with a transition to itself with probability 1. For pairs of states of $Q^?$,

[2] As Boolean combinations of propositions can be evaluated in each state, we simplify our life and think of p and p' as propositions.

the transitions and their probabilities agree with those of $\mathcal{M}$. Furthermore, for each $q \in Q^?$ with transitions to states $q' \notin Q^?$ we add a single transition q to $\hat{q}$ with probability $\sum_{q' \notin Q^?} P_{q,q'}$. The state $\hat{q}$ forms the only bottom s.c.c.: Since all states of $Q^?$ satisfy p and $\neg p'$ any further bottom s.c.c. would be in W_2^-, thus not in $Q^?$. Hence, every trajectory will reach state $\hat{q}$ with probability one.

Let T be the matrix of transition probabilities $P_{q,q'}$ restricted to states of $Q^?$. Furthermore, for T^k, the kth power of the matrix T, the entry $T^k_{q,q'}$ is equal to the probability that a path starting in q reaches q' in k steps. Since every trajectory goes eventually to $\hat{q}$ (and stays there) with probability one, we get $\lim_{k\to\infty} T^k = 0$. For any two solutions $a = (a_q)_{q\in Q^?}$ and $b = (b_q)_{q\in Q^?}$ of the equation system, we have $a - b = T(a - b)$, thus $a - b = T^k(a - b)$, and taking the limit as k tends to infinity we conclude that $a = b$. □

For the example shown in Figure 3, we get $P_{q_3} = P_{q_5} = 1$ and $P_{q_2} = P_{q_4} = 0$. Furthermore, $P_{q_1} = P_{q_1,q_1}P_{q_1} + P_{q_1,q_3}P_{q_3} + P_{q_1,q_4}P_{q_4} = 0.2 \cdot P_{q_1} + 0.3 \cdot 1 + 0.5 \cdot 0 = \frac{3}{8}$.

We are ready for the construction of the new program $\mathcal{M}'$ and the new formula φ'. $\mathcal{M}'$ has a larger state space Q' and is defined over the set of atomic propositions $\mathcal{P}$ enriched with ξ which serves in φ' as a substitute for $p\mathcal{U}p'$. Let

- $Q' = \{(q, \xi) \mid q \in Q^+ \cup Q^?\} \cup \{(q, \neg\xi) \mid q \in Q^- \cup Q^?\}$.
- $\mathcal{V}'((q, \xi)) = \mathcal{V}(q) \cup \{\xi\}$ and $\mathcal{V}'((q, \neg\xi)) = \mathcal{V}(q)$.
- we define the transitions of $\mathcal{M}'$ implicitly by giving non-zero transition probabilities. ξ_1 and ξ_2 stand for ξ or $\neg\xi$ and $\overline{P}$ for $1 - P$. We distinguish the following cases based on the states of $\mathcal{M}$:
  - $q, q' \in Q^+ \cup Q^-$. There is exactly one state (q, ξ_1) and (q', ξ_2) in Q'. We set $P'_{(q,\xi_1),(q',\xi_2)} = P_{q,q'}$.
  - $q \in Q^+ \cup Q^-$, $q' \in Q^?$. There is exactly one state with first component q and two states with first component q'. We set $P'_{(q,\xi_1),(q',\xi)} = P_{q,q'}P_{q'}$ and $P'_{(q,\xi_1),(q',\neg\xi)} = P_{q,q'}\overline{P_{q'}}$.
  - $q \in Q^?$. If $q' \in Q^?$, we define two transitions by $P'_{(q,\xi),(q',\xi)} = P_{q,q'}P_{q'}/P_q$ and $P'_{(q,\neg\xi),(q',\neg\xi)} = P_{q,q'}\overline{P_{q'}}/\overline{P_q}$. If $q' \in Q^+$ ($q' \in Q^-$), we define a single transition by $P'_{(q,\xi),(q',\xi)} = P_{q,q'}/P_q$ (resp. $P'_{(q,\neg\xi),(q',\neg\xi)} = P_{q,q'}/\overline{P_q}$).
- P'_0 is defined by $P'_0((q, \xi)) = P_0(q)P_q$ and $P'_0((q, \neg\xi)) = P_0(q)\overline{P_q}$, for all $(q, \xi), (q, \neg\xi) \in Q'$.

The new sequential probabilistic program is $\mathcal{M}' = (Q', \Delta', P', P'_0, \mathcal{V}')$ and the formula φ' is obtained by substituting $p\mathcal{U}p'$ by ξ in φ.

For the sequential probabilistic program $\mathcal{M}$ of Figure 3 and $\varphi = p\mathcal{U}p'$, we obtain $\mathcal{M}'$ as given in Figure 4 and $\varphi' = \xi$.

Let us show that the probability of $\mathcal{M}$ satisfying φ coincides with the one of $\mathcal{M}'$ satisfying φ'. Therefore, we analyze our construction in more detail.

Let $\mathcal{G}'^+$ be the subgraph of $\mathcal{M}'$ induced by states (q, ξ) with $q \in Q^+$, $\mathcal{G}'^-$ be the one induced by all states $(q, \neg\xi)$ with $q \in Q^-$. Furthermore, let $\mathcal{G}'^{?,+}$ be the subgraph induced by states (q, ξ) and $\mathcal{G}'^{?,-}$ the one induced by states $(q, \neg\xi)$, for $q \in Q^?$. By construction $\mathcal{G}'^+$ ($\mathcal{G}'^-$) is isomorphic to $\mathcal{G}^+$ ($\mathcal{G}^-$), the subgraph

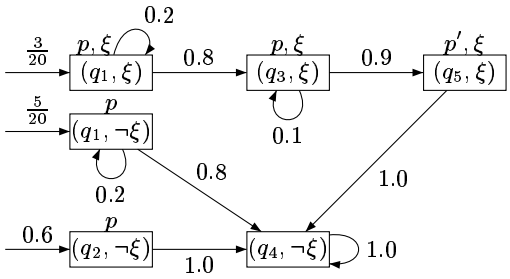

Fig. 4. A sequential probabilistic program

of $\mathcal{M}$ induced by states in Q^+ (Q^-). Furthermore, $\mathcal{G}'^{?,+}$ is isomorphic to $\mathcal{G}'^{?,-}$ and $\mathcal{G}^?$, the subgraph of $\mathcal{M}$ induced by the states of $Q^?$.

Additionally, every transition from a state of $\mathcal{G}'^{?,+}$ ($\mathcal{G}'^{?,-}$) leading to a state not in $\mathcal{G}'^{?,+}$ ($\mathcal{G}'^{?,-}$), leads to a state in $\mathcal{G}'^{+}$ ($\mathcal{G}'^{-}$). Thus, if ξ holds in a state of Q', almost surely every trajectory will eventually reach a state satisfying p' and every state in between satisfies p, and, if $\neg\xi$, almost all trajectories satisfy $\neg(p\mathcal{U}p')$. In other words, $\xi \equiv p\mathcal{U}p'$ holds almost surely in every state of $\mathcal{M}'$.

Let us see how the transition probabilities of $\mathcal{M}$ and $\mathcal{M}'$ relate: For every finite path $q'_1 \dots q'_m$ in $\mathcal{M}'$, we have by construction that $\mu_{\mathcal{M}'}(q'_1 \dots q'_m) = P_{q_1,q_2} \cdot \dots \cdot P_{q_{m-1},q_m} \cdot P_{q_m}$ or $\mu_{\mathcal{M}'}(q'_1 \dots q'_m) = P_{q_1,q_2} \cdot \dots \cdot P_{q_{m-1},q_m} \cdot \overline{P_{q_m}}$, depending on whether $q'_m = (q, \xi)$ or $q'_m = (q_m, \neg\xi)$.

Let g be the mapping that projects a trajectory of $\mathcal{M}'$ to $\mathcal{M}$ by projecting to the first component. If $g(q'_m) \in Q^+ \cup Q^-$ then $q'_1 \dots q'_m$ is the only trajectory with projection to $q_1 \dots q_m$. If $g(q_m) \in Q^?$, there are exactly two trajectories with projection $q_1 \dots q_m$: $(q_1, \xi) \dots (q_m, \xi)$ and $(q_1, \neg\xi) \dots (q_m, \neg\xi)$.

We conclude that for every measurable set E of trajectories of $\mathcal{M}$ the set $\{w' \in Q'^{\omega} \mid g(w) \in E\}$ is measurable and both sets have the same measure. This implies that $\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\varphi)) = \mu_{\mathcal{M}}(L_{\mathcal{M}}(\varphi))$. Since $p\mathcal{U}p' \equiv \xi$ for almost all trajectories in $\mathcal{M}'$, we get

Proposition 2.

$$\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\varphi')) = \mu_{\mathcal{M}}(L_{\mathcal{M}}(\varphi))$$

We conclude the presentation of $\mathcal{T}_{\mathcal{U}}$ analyzing its complexity. Let us first consider its time complexity. The construction of the new formula can be carried out in linear time with respect to $|\varphi|$. The information whether a state of $\mathcal{M}$ belongs to Q^+, Q^-, or $Q^?$ can be evaluated in linear time, as we show. We assume that $\mathcal{M}$ is given as a graph. Note that a Boolean combination of atomic propositions can be evaluated in time linear in its size in every state, so that our treatment of p and p' as propositions does not change the complexity.

1. Mark every state of $\mathcal{M}$ by Q^+ in which p' holds and by Q^- in which neither p nor p' holds.

2. Partition the remaining graph into strongly connected components. Process those in a bottom-up manner, i.e., when a s.c.c. D is processed all s.c.c. that are reachable from D have already been processed. Assign the nodes of D in the following manner:
   (a) if there is no edge leading to a state in Q^+ then add the nodes of D to Q^-.
   (b) if all edges leaving D lead to states in Q^+ then add the nodes of D to Q^+.
   (c) otherwise add the nodes of D to $Q^?$.

It is easy to see that the algorithm is correct. (1) corresponds to identifying W^+ and W_1^-. It can be done in linear time using a standard depth-first search. Partitioning a graph into strongly connected components can also be done in linear time. In (2), the bottom strongly connected components are processed first, which also finds the states of W_2^+. The further processing is obviously correct. The classification of every s.c.c. can be carried out in time linear in the size of the component.

For constructing $\mathcal{M}'$ we have to solve the linear equation system of Lemma 1, which can be done in polynomial time in the size of $\mathcal{M}$. For checking whether the formula is satisfied, however, the exact transition probabilities are not important, but it has to be decided whether they are 0 or > 0. Thus, we need either linear or polynomial time for construction $\mathcal{M}'$ depending on whether we are interested in the question of satisfaction or in the exact measure.

Let us consider the space complexity of $\mathcal{T}_{\mathcal{U}}$. First note that checking whether there is path from a node u to a node v can be tested in space $\log^2 |\mathcal{M}|$. For every state q of $\mathcal{M}$, we have to check whether (q, ξ) or $(q, \neg\xi)$ are states of $\mathcal{M}'$. $(q, \xi) \in Q'$ iff $q \in Q^+ \cup Q^?$. This is the case iff q satisfies p' or there is a node q' reachable satisfying p' by a path of which every node satisfies p. This can be checked in space $\log^2 |\mathcal{M}|$. $(q, \neg\xi) \in Q'$ iff $q \in Q^- \cup Q^?$. This is the case if q satisfies $\neg p$ and $\neg p'$, or satisfies p but not p' and there is a path by states satisfying p to a state $q' \notin Q^+ \cup Q^?$ (tested as described above). Again, this can be tested in space $\log^2 |\mathcal{M}|$. The edges of $\mathcal{M}'$ can be constructed using similar ideas.

Transformation $\mathcal{T}_{\mathcal{X}}$ Transformation $\mathcal{T}_{\mathcal{X}}$ for an innermost formula of the form $\mathcal{X}p$ is patterned after the transformation $\mathcal{T}_{\mathcal{U}}$. We first partition the states of $\mathcal{M}$ into three disjoint subsets Q^+, Q^-, and $Q^?$, defined as follows: Q^+ comprises all states q for which all transitions are into states satisfying p, Q^- contains the ones for which all transitions yield states not satisfying p, and the remaining states having transitions to states satisfying p as well as transitions to states not satisfying p are assigned to $Q^?$. Again, we will see that for a trajectory starting in a state of Q^+ (Q^-), it will satisfy $\mathcal{X}p$ with probability 1 (0, respectively). For trajectories starting in states of $Q^?$ both events have a non-zero probability.

As before, let P_q denote the probability that $\mathcal{X}p$ is satisfied starting from state q, which is given as

Lemma 2.

$$\begin{aligned} P_q &= \sum_{p \in \mathcal{V}(q')} P_{q,q'} && \text{if } q \in Q^? \\ P_q &= 1 && \text{if } q \in Q^+ \\ P_q &= 0 && \text{if } q \in Q^- \end{aligned}$$

As there is no recursion involved, the previous set of equations trivially has a unique solution.

The new sequential probabilistic program $\mathcal{M}'$ has a larger state space Q' and is defined over the set of atomic propositions $\mathcal{P}$ enriched with ξ which serves in φ' as a substitute for $\mathcal{X}p$. The state space and the propositions valid in each state are similarly defined as in the case of $\mathcal{T}_\mathcal{U}$. We define the transitions of $\mathcal{M}'$ implicitly by giving non-zero transition probabilities, as follows:

Let ξ_1 and ξ_2 stand for ξ or $\neg\xi$ and $\overline{P}$ for $1-P$. We distinguish the following cases based on the states of $\mathcal{M}$:

- $q, q' \in Q^+ \cup Q^-$. There is exactly one state (q, ξ_1) and (q', ξ_2) in Q'. We set $P'_{(q,\xi_1),(q',\xi_2)} = P_{q,q'}$.
- $q \in Q^+ \cup Q^-$, $q' \in Q^?$. There is exactly one state with first component q and two states with first component q'. We set $P'_{(q,\xi_1),(q',\xi)} = P_{q,q'}P_{q'}$ and $P'_{(q,\xi_1),(q',\neg\xi)} = P_{q,q'}\overline{P_{q'}}$.
- $q \in Q^?$.
  - If $q' \in Q^?$ and q' satisfies p, we define two transitions by $P'_{(q,\xi),(q',\xi)} = P_{q,q'}P_{q'}/P_q$ and $P'_{(q,\xi),(q',\neg\xi)} = P_{q,q'}\overline{P_{q'}}/P_q$.
  - If $q' \in Q^?$ and q' satisfies $\neg p$, we define two transitions by $P'_{(q,\neg\xi),(q',\xi)} = P_{q,q'}P_{q'}/\overline{P_q}$ and $P'_{(q,\neg\xi),(q',\neg\xi)} = P_{q,q'}\overline{P_{q'}}/\overline{P_q}$.
  - If $q' \in Q^+ \cup Q^-$ and q' satisfies p, we define a single transition by $P'_{(q,\xi),(q',\xi_2)} = P_{q,q'}/P_q$ where (q', ξ_2) is the unique state of $\mathcal{M}'$ with first component q'.
  - If $q' \in Q^+ \cup Q^-$ and q' satisfies $\neg p$, we define a single transition by $P'_{(q,\neg\xi),(q',\xi_2)} = P_{q,q'}/\overline{P_q}$ where (q', ξ_2) is the unique state of $\mathcal{M}'$ with first component q'.

The initial distribution P'_0 is defined by $P'_0((q,\xi)) = P_0(q)P_q$ and $P'_0((q,\neg\xi)) = P_0(q)\overline{P_q}$, for all $(q,\xi), (q,\neg\xi) \in Q'$.

The new sequential probabilistic program is $\mathcal{M}' = (Q', \Delta', P', P'_0, \mathcal{V}')$ and the formula φ' is obtained by substituting $\mathcal{X}p$ by ξ in φ. Similar as in case of $\mathcal{T}_\mathcal{U}$, we get

Proposition 3.

$$\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\varphi')) = \mu_\mathcal{M}(L_\mathcal{M}(\varphi))$$

Using similar arguments as in the case of $\mathcal{T}_\mathcal{U}$, we learn that $\mathcal{T}_\mathcal{X}$ can be carried out within the same time complexity of $\mathcal{T}_\mathcal{U}$.

The overall procedure If φ has k temporal operators, we can compute the measure $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\varphi))$ as follows: We apply k times the appropriate transformations $\mathcal{T}_{\mathcal{U}}$ or $\mathcal{T}_{\mathcal{X}}$ and obtain the sequence $\varphi^1, \mathcal{M}^1, \ldots, \varphi^k, \mathcal{M}^k$, where φ^k is a simple propositional formula. Then $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\varphi)) = \mu_{\mathcal{M}^k}(L_{\mathcal{M}^k}(\varphi^k))$, which is simply the sum of the initial probabilities in $\mathcal{M}^k$ over all states satisfying φ^k.

To conclude the proof of Theorem 1, it remains to show that our procedure can be carried out within the given time and space boundaries. As both transformations can be carried out in time linear in the size of the program (or polynomial if we are interested in the exact transition probabilities) and in every step the state space is at most doubled, we get $O(2^{|\varphi|}|\mathcal{M}|)$ as an upper bound for checking whether $\mathcal{M}$ satisfies φ. When we are interested in the exact measure, we need time polynomially in the size of $\mathcal{M}$ and exponentially in the size of φ. Note that it was shown in [16] that this problem is PSPACE-hard (wrt. the size of the formula). Thus, it is PSPACE-complete (as in the non-probabilistic case) and the given procedure is optimal.

It can be shown that the space required to carry out this computation is proportional to $|\varphi|(|\varphi|+\log^2(2^{|\varphi|}|\mathcal{M}|))$. Thus, the space complexity is in $O(|\varphi|^3+|\varphi|\log^2|\mathcal{M}|)$.

Note that the presented procedure can easily be extended to deal with *past tense connectives* defined in [9]. Using the same ideas, one can achieve also the same time and space boundaries.

4 Model Checking Probabilistic Programs against Automata Specifications

Let $\mathcal{A}$ be an ω-automaton over $2^{\mathcal{P}}$. Given a concurrent probabilistic program $\mathcal{M}$ over $\mathcal{P}$ with valuation function $\mathcal{V}$ and a scheduler u of $\mathcal{M}$, we denote by $L_{\mathcal{M},u}(\mathcal{A})$, similarly to the case of LTL, the set $\{x \in \Omega_{\mathcal{M},u} \mid \mathcal{V}(x) \in L(\mathcal{A})\}$. In the same manner, for a sequential program $\mathcal{M}$ over $\mathcal{P}$ with valuation function $\mathcal{V}$, we define $L_{\mathcal{M}}(\mathcal{A})$ to be $\{x \in \Omega_{\mathcal{M}} \mid \mathcal{V}(x) \in L(\mathcal{A})\}$.

In the following, probabilistic programs will be defined over $\mathcal{P}$ while ω-automata will be defined over $2^{\mathcal{P}}$. It is a precondition for model checking programs against automata specifications that languages defined by automata are measurable. This is the case:

Proposition 4 ([16]). *For a concurrent probabilistic program $\mathcal{M}$, a scheduler u of $\mathcal{M}$, and a Büchi automaton $\mathcal{A}$, $L_{\mathcal{M},u}(\mathcal{A}) \in \mathfrak{B}_{\mathcal{M},u}$.*

Of course, the corresponding holds for sequential probabilistic programs.

Given a concurrent probabilistic program $\mathcal{M}$ and a Büchi automaton $\mathcal{A}$, the *concurrent emptiness problem (for automata)* is to decide whether it holds $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\mathcal{A})) = 0$ for all schedulers u of $\mathcal{M}$. In contrast, the *concurrent universality problem* is to decide whether $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\mathcal{A})) = 1$ for all schedulers u. *Sequential emptiness* and *sequential universality for automata* are defined analogously.

From the computational point of view, the above problems become easier to handle in case the automata specification at hand is a deterministic one. So, the transformation of a Büchi automaton into an equivalent ω-automaton that behaves deterministically forms the basis of forthcoming algorithms.

Theorem 2 (Safra [15]). *For a Büchi automaton $\mathcal{B}$ over Σ with n states, there is a deterministic Streett (Rabin) automaton $\mathcal{A}$ over Σ with $2^{O(n \log n)}$ states and $O(n)$ pairs in the acceptance component such that $L(\mathcal{A}) = L(\mathcal{B})$.*

4.1 Checking Concurrent Probabilistic Programs

We assume an automata specification to represent undesired behaviour. Thus, we are interested in solving the emptiness problem for automata.

Let us make some assumptions which make life easier in the following. Without loss of generality, we can assume that, in a concurrent probabilistic program $\mathcal{M} = (Q, N, R, \Delta, P_0, P, \mathcal{V})$, it holds $Q \subseteq 2^{\mathcal{P}}$ and, for each $q \in Q$, $\mathcal{V}(q) = q$. This can be achieved by sufficiently adding propositions to the program (to distinguish states) and accordingly adding transitions to the automaton. Then $\mathcal{M}$ can easily be transformed into an equivalent (in a sense described below) concurrent probabilistic program $\mathcal{M}'$

- that has no longer an initial probability distribution but a (randomizing) initial state,
- whose transitions are additionally labelled with elements of $2^{\mathcal{P}}$, and
- that can be viewed as a quasi-deterministic ω-automaton over $2^{\mathcal{P}}$.

The transformation proceeds as follows:

1. Add a new randomizing (initial) state $q_{\mathcal{M}}$ to $\mathcal{M}$ and, for each state q of $\mathcal{M}$ with $P_0(q) > 0$, add a transition $q_{\mathcal{M}} \to q$, for which we set $P'_{q_{\mathcal{M}},q} = P_0(q)$.
2. Including the new ones, label each transition $q \to q'$ with $\mathcal{V}(q')$.

For an example how to transform a probabilistic program, look at Figure 5 where, for the sake of clarity, we assume a and b to be (different) sets of propositions.

A trajectory of $\mathcal{M}'$ is no longer a sequence of states but a sequence of transition labellings $\mathcal{V}(q)$ that $\mathcal{M}'$ finds along a path starting at $q_{\mathcal{M}}$. We will freely use the notation introduced for probabilistic programs also in the setting of such transformed programs giving them the obvious meaning. Regardless of the probabilities some transitions are equipped with, $\mathcal{M}'$ can be viewed as a quasi-deterministic Büchi automaton with single initial state $q_{\mathcal{M}}$ and acceptance component Q. The product of $\mathcal{M}'$ and a deterministic ω-automaton $\mathcal{A}$, denoted by $\mathcal{M}' \times \mathcal{A}$, is defined as usual and at least quasi-deterministic. On the other hand, preserving the probabilities, we can view $\mathcal{M}' \times \mathcal{A}$ also as a concurrent probabilistic program where a state is defined to be nondeterministic or randomizing according to its first component and trajectories are sequences of transition labellings. More precisely, $(q, s) \to (q', s')$ becomes a transition of $\mathcal{M}' \times \mathcal{A}$ if $q \to q'$ and $s \to s'$ are $\mathcal{V}(q')$-labelled transitions of $\mathcal{M}'$ and $\mathcal{A}$, respectively. In that case,

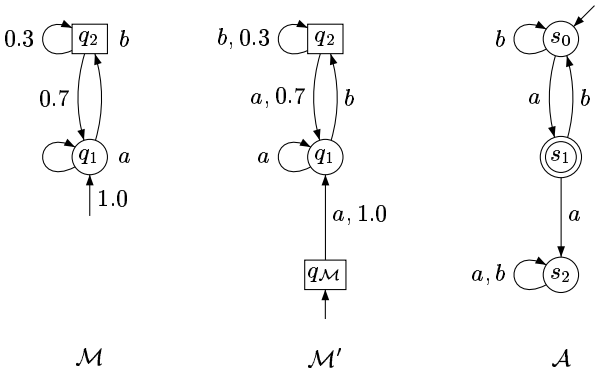

Fig. 5. A program, its transformation, and a deterministic Streett automaton

$(q, s) \to (q', s')$ is also labelled $\mathcal{V}(q')$ and, if q is randomizing, equipped with probability $P'_{q,q'}$. The product of the concurrent probabilistic program $\mathcal{M}$ and the deterministic Streett automaton $\mathcal{A}$ from Figure 5 with acceptance condition $\{(\{s_0, s_2\}, \{s_1\})\}$ is illustrated in Figure 6 (note that, for the sake of clarity, the probabilities of transitions that go out from randomizing states are omitted).

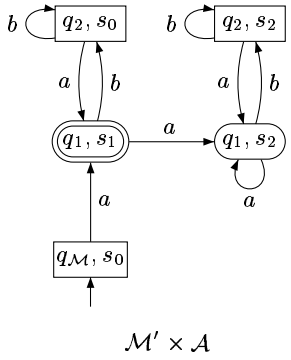

Fig. 6. The product of a program and an automaton

Given a probabilistic program $\mathcal{M}$ and a deterministic Streett automaton $\mathcal{A}$ with acceptance component $\mathcal{F}$, we want to mark some SCCs of $\mathcal{M}' \times \mathcal{A}$ to be good in some sense. We call a set D of its states *accepting* if, for all pairs $(U, V) \in \mathcal{F}$, the following holds:

$$\{s \mid (q, s) \in D \text{ for some } q\} \cap U \neq \emptyset \text{ implies } \{s \mid (q, s) \in D \text{ for some } q\} \cap V \neq \emptyset$$

Otherwise, D is called *rejecting*. We say that a state (r, f) of a rejecting set D is *rejecting* if there is a pair $(U, V) \in \mathcal{F}$ such that $f \in U$ and D contains no state (q, s) with $s \in V$.

The following proposition is crucial in this subsection and immediately leads to an algorithm that solves the concurrent emptiness problem for automata.

Proposition 5. *For a concurrent program $\mathcal{M} = (Q, N, R, \Delta, P_0, P, \mathcal{V})$ and a deterministic Streett automaton $\mathcal{A} = (S, \{s_0\}, \delta, \mathcal{F})$, there exists a scheduler u of $\mathcal{M}$ with $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\mathcal{A})) > 0$ iff there is a set D of states of $\mathcal{M}' \times \mathcal{A}$ satisfying the following:*

(1) $(\mathcal{M}' \times \mathcal{A})[D]$ is nontrivial and strongly connected,
(2) D is accepting and reachable from $(q_\mathcal{M}, s_0)$, and
(3) for all transitions $(q, s) \to (q', s')$ of $\mathcal{M}' \times \mathcal{A}$ with $(q, s) \in D$ and $(q', s') \notin D$, (q, s) is nondeterministic, i.e., it holds $q \in N$.

Proof. It is easy to see that $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\mathcal{A})) > 0$ for a scheduler u of $\mathcal{M}$ iff there is a scheduler u' of $\mathcal{M}'$ such that $\mu_{\mathcal{M}',u'}(L_{\mathcal{M}',u'}(\mathcal{A})) > 0$.

($\Leftarrow$) We fix a path $\beta \in ((Q \cup \{q_\mathcal{M}\}) \times S)^*$ through $\mathcal{M}' \times \mathcal{A}$ from $(q_\mathcal{M}, s_0)$ to a state of D. Let β' be the projection of β onto the first component. The scheduler u' of $\mathcal{M}'$ satisfying $\mu_{\mathcal{M}',u'}(L_{\mathcal{M}',u'}(\mathcal{A})) > 0$ follows β' taking $\mathcal{M}' \times \mathcal{A}$ from the initial state to D and, henceforth, forces the trajectory both to stay within D and to almost surely visit each state of D infinitely often. This can be accomplished by, for a given nondeterministic state (q, s), alternately choosing the transitions $(q, s) \to (q', s')$ of $\mathcal{M}' \times \mathcal{A}$ with $(q', s') \in D$ (recall that the history of a trajectory is at the scheduler's disposal.) Clearly, $\mu_{\mathcal{M}',u'}(\mathcal{C}_{\mathcal{M}',u'}(\beta'))$ is nonzero. Given $\mathcal{C}_{\mathcal{M}',u'}(\beta')$, the conditional probability that $\mathcal{M}'$, wrt. u', follows a trajectory that visits exactly the states of D infinitely often is one. As such a trajectory is contained in $L_{\mathcal{M}',u'}(\mathcal{A})$, we conclude $\mu_{\mathcal{M}',u'}(L_{\mathcal{M}',u'}(\mathcal{A})) > 0$.

($\Rightarrow$) Note that a trajectory x of $\mathcal{M}'$ wrt. u' unambiguously defines a path $\widetilde{x}$ through $\mathcal{M}' \times \mathcal{A}$ starting from $(q_\mathcal{M}, s_0)$. This is due to the fact that $\mathcal{A}$ is deterministic and that a transition of $\mathcal{M}'$ has a unique label. Let $\mathcal{D}$ contain the subsets D of states of $\mathcal{M}' \times \mathcal{A}$ such that $(\mathcal{M}' \times \mathcal{A})[D]$ is strongly connected. Furthermore, for $D \in \mathcal{D}$, let $E(D) := \{x \in \Omega_{\mathcal{M}',u'} \mid \mathit{inf}(\widetilde{x}) = D\}$. Now suppose that $\mu_{\mathcal{M}',u'}(L_{\mathcal{M}',u'}(\mathcal{A})) > 0$ for a scheduler u' of $\mathcal{M}'$. As

$$L_{\mathcal{M}',u'}(\mathcal{A}) = \bigcup_{D \in \mathcal{D} \text{ is accepting}} E(D),$$

we can find an accepting set $D \in \mathcal{D}$ that satisfies $\mu_{\mathcal{M}',u'}(E(D)) > 0$. (Otherwise, the probability of the countable union $L_{\mathcal{M}',u'}(\mathcal{A})$ of events would be zero.) As D is the infinity set of at least one infinite path through $\mathcal{M}' \times \mathcal{A}$ starting from $(q_\mathcal{M}, s_0)$, it is reachable from the initial state, thus satisfies condition (2), and forms a nontrivial (strongly connected) subgraph of $\mathcal{M}' \times \mathcal{A}$, satisfying condition (1). Now suppose there is a transition $(q, s) \to (q', s')$ of $\mathcal{M}' \times \mathcal{A}$ with $(q, s) \in D$, $(q', s') \notin D$, and $q \in R$. As, for every trajectory $x \in E(D)$, $\widetilde{x}$ visits (q, s) infinitely often (and each time the probability to exit D is nonzero), it will almost surely leave D infinitely often so that we have $\mu_{\mathcal{M}',u'}(E(D)) = 0$ contradicting our assumption. It follows that D also satisfies condition (3) from Proposition 5, which concludes our proof. □

Note that, in the above proof, we explicitly make use of the fact that a trajectory of $\mathcal{M}'$ determines exactly one corresponding run of $\mathcal{M}' \times \mathcal{A}$ starting from $(q_{\mathcal{M}}, s_0)$ (recall that $\mathcal{A}$ is deterministic).

Let us fall back on the example contributed by Figures 5 and 6, respectively. There is an accepting set $D = \{(q_1, s_1), (q_2, s_0)\}$ of states of $\mathcal{M}' \times \mathcal{A}$ that is reachable from $(q_{\mathcal{M}}, s_0)$ via $\beta = (q_{\mathcal{M}}, s_0)(q_1, s_1)$ and forms a nontrivial and strongly connected subgraph of $\mathcal{M}' \times \mathcal{A}$. The only transition leading from a state of D to a state outside of D is $(q_1, s_1) \to (q_1, s_2)$, thus, arises from a nondeterministic state. In fact, for the scheduler u' of $\mathcal{M}'$ with $u'(x) = q_2$ for every $x \in (Q \cup \{q_M\})^*$ that ends in a nondeterministic state (i.e. in q_1), it holds $\mu_{\mathcal{M}',u'}(L_{\mathcal{M}',u'}(\mathcal{A})) > 0$.

Based on Proposition 5, we now provide an algorithm that solves the concurrent emptiness problem, i.e., decides, given a concurrent probabilistic program $\mathcal{M}$ and a Büchi automaton $\mathcal{B}$, whether there is a scheduler u of $\mathcal{M}$ such that $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\mathcal{B})) > 0$:

1. From $\mathcal{B}$, build a deterministic Streett automaton $\mathcal{A}$ with $L(\mathcal{A}) = L(\mathcal{B})$.
2. Compute (the graph of) $\mathcal{M}' \times \mathcal{A}$, remove those states that are not reachable from $(q_{\mathcal{M}}, s_0)$, and let G denote the resulting graph.
3. Repeat
   (a) Determine the sets $\mathcal{AC}$ of nontrivial and accepting and $\mathcal{RC}$ of nontrivial and rejecting SCCs of G, respectively.
   (b) For each $C \in \mathcal{RC}$, remove the transitions going out from rejecting states.
   (c) For each $C \in \mathcal{AC}$, do the following:
      i. Find the set H of states $(q, s) \in C$ with randomizing q from where there is a transition leaving C.
      ii. If H is the empty set, then return "Yes" (thus, halt with success). Otherwise, remove the transitions going out from the states of H.

   until $\mathcal{AC} \cup \mathcal{RC} = \emptyset$.
4. Return "No".

Obviously, the algorithm terminates. Furthermore, it returns the answer "Yes" iff there is a scheduler u of $\mathcal{M}$ such that $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\mathcal{B})) > 0$. Let us discuss its complexity in the following. Starting from a Büchi automaton with n states, construct an equivalent deterministic Streett automaton with $2^{O(n \log n)}$ states and $O(n)$ pairs in the acceptance component. Assume that $\mathcal{M}'$ has m states. The number of states of $\mathcal{M}' \times \mathcal{A}$ is not greater than $m \cdot 2^{O(n \log n)}$. Thus, steps (a), (b), and (c) are repeated at most $m \cdot 2^{O(n \log n)}$-times, respectively. Determining the SCCs of G can be done in time linear in the size of $\mathcal{M}' \times \mathcal{A}$. Overall, the algorithm runs in time $O(m^3 \cdot 2^{O(n \log n)})$, i.e., it is quadratic in $|\mathcal{M}|$ and exponential in $|\mathcal{A}|$.

Theorem 3. *Given a concurrent probabilistic program $\mathcal{M}$ and a Büchi automaton $\mathcal{A}$, the concurrent emptiness problem can be solved in time $O(|\mathcal{M}|^2 \cdot 2^{O(|\mathcal{A}|)})$.*

The complexity of the above algorithm can be improved by constructing, instead of a (fully) deterministic ω-automaton, a Büchi automaton that is *deterministic in the limit* [4], i.e., that behaves deterministically as soon as it reaches

a final state. Such an automaton only requires $2^{O(n)}$ states and finally leads to an $O(m^3 \cdot 2^{O(n)})$-complexity of the algorithm. But as it does not provide further insights in the principles of probabilistic model checking, we omit the corresponding construction. Courcoubetis and Yannakakis furthermore show that to decide the concurrent emptiness problem requires exponential time in the total input size $|\mathcal{M}| + |\mathcal{A}|$ [4].

The concurrent universality problem for automata can be solved analogously, but, from the computational point of view, it seems to be more appropriate to construct a deterministic Rabin automaton from a Büchi automaton. Accordingly, we call a set D of states of $\mathcal{M}' \times \mathcal{A}$ *accepting* if there is a pair (U, V) from the acceptance component of $\mathcal{A}$ and a state $(r, f) \in D$ such that $f \in U$ and D contains no state (q, s) with $s \in V$. Otherwise, we call D *rejecting*.

Proposition 6. *Given a concurrent program $\mathcal{M} = (Q, N, R, \Delta, P_0, P, \mathcal{V})$ and a deterministic Rabin automaton $\mathcal{A} = (S, \{s_0\}, \delta, \mathcal{F})$, there is a scheduler u of $\mathcal{M}$ with $\mu_{\mathcal{M},u}(L_{\mathcal{M},u}(\mathcal{A})) < 1$ iff there is a set D of states of $\mathcal{M}' \times \mathcal{A}$ satisfying the following:*

(1) $(\mathcal{M}' \times \mathcal{A})[D]$ is nontrivial and strongly connected,
(2) D is rejecting and reachable from $(q_{\mathcal{M}}, s_0)$, and
(3) for all transitions $(q, s) \to (q', s')$ of $\mathcal{M}' \times \mathcal{A}$ with $(q, s) \in D$ and $(q', s') \notin D$, it holds $q \in N$.

Proving Proposition 6 mainly applies techniques used to prove Proposition 5. The algorithm for the concurrent universality problem as suggested by Proposition 6 runs in time $O(|\mathcal{M}|^2 \cdot 2^{O(|\mathcal{A}|)})$.

However, to solve the concurrent universality problem, our algorithm cannot fall back on using ω-automata that are deterministic in the limit for the following reason: We then cannot be sure that a path leading to D does not have an "accepting" counterpart elsewhere. In [4], it is mentioned that the concurrent universality problem for Büchi automata that are deterministic in the limit requires exponential time.

4.2 Checking Sequential Probabilistic Programs

Sequential probabilistic programs were defined as a special case of concurrent ones. Thus, the above algorithms carry over to the case the probabilistic program has no nondeterministic state. Let us formulate an algorithm explicitly for this case. Two basic properties wrt. a sequential probabilistic program $\mathcal{M}$ and a deterministic ω-automaton $\mathcal{A}$ will play a crucial role in both qualitative and quantitative analysis:

- Almost surely, a trajectory of $\mathcal{M}'$ will take $\mathcal{M}' \times \mathcal{A}$ to a bottom SCC.
- Almost surely, a trajectory of $\mathcal{M}'$ that leads $\mathcal{M}' \times \mathcal{A}$ into a bottom SCC D will visit each state of D infinitely often.

Wrt. the emptiness problem, a sequential probabilistic program does not satisfy its automata specification (and thus contains undesired behaviour), iff, within the product of the transformed program and the specification, one can find an accepting bottom SCC that is reachable from the initial state. The following characterization immediately follows from Proposition 5.

Proposition 7. *Given a sequential probabilistic program $\mathcal{M}$ and a deterministic Streett automaton $\mathcal{A}$, it holds $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A})) > 0$ iff there is a reachable (from the initial state) and accepting bottom SCC of $\mathcal{M}' \times \mathcal{A}$.*

Thus, after having computed a deterministic ω-automaton and a product automaton, the algorithm for solving the sequential emptiness problem only needs to determine whether there is an accepting bottom SCC that is reachable from the initial state. This can be done in time linear in the size of the product automaton. The algorithm works in time $O(m^2 \cdot 2^{O(n \log n)})$ where m is assumed to be the number of states of the program and n the number of states of the Büchi automaton. Again, the usage of ω-automata that are deterministic in the limit reduces the time complexity to $O(m^2 \cdot 2^{O(n)})$.

Checking universality wrt. sequential programs is closely related to checking emptiness.

Proposition 8. *For a sequential probabilistic program $\mathcal{M}$ and a deterministic Streett automaton $\mathcal{A}$, we have $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A})) = 1$ iff all the reachable bottom SCCs of $\mathcal{M}' \times \mathcal{A}$ are accepting.*

Proof. Clearly, $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A})) = 1$ iff $\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\mathcal{A})) = 1$.

($\Leftarrow$) Almost surely, a trajectory of $\mathcal{M}'$ will lead $\mathcal{M}' \times \mathcal{A}$ into a bottom SCC. Let x be a finite sequence of states of $\mathcal{M}'$ that takes $\mathcal{M}' \times \mathcal{A}$ from the initial state to a bottom SCC D. As D is nontrivial and accepting, $\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\mathcal{A}) \mid \mathcal{C}_{\mathcal{M}'}(x)) = 1$. Overall, we get $\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\mathcal{A})) = 1$.

($\Rightarrow$) Assume that $\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\mathcal{A})) = 1$ and suppose there is a reachable bottom SCC D of $\mathcal{M}' \times \mathcal{A}$ that is rejecting. We can find a finite sequence x of states of $\mathcal{M}'$ with $\mu_{\mathcal{M}'}(\mathcal{C}_{\mathcal{M}'}(x)) > 0$ that takes $\mathcal{M}' \times \mathcal{A}$ from the initial state to D. As D is rejecting, $\mu_{\mathcal{M}'}(L_{\mathcal{M}'}(\mathcal{A}) \mid \mathcal{C}_{\mathcal{M}'}(x)) = 0$. Thus, with nonzero probability, $\mathcal{M}'$ will produce a trajectory that is not contained in $L_{\mathcal{M}'}(\mathcal{A})$, contradicting the assumption. □

For example, the product of a (transformed) sequential probabilistic program $\mathcal{M}'$ and a deterministic Streett automaton $\mathcal{A}$ may have the structure as shown in Figure 7 where the bottom SCCs $\{\overline{q}_4\}$ and $\{\overline{q}_5\}$ are both assumed to be accepting. As $\{\overline{q}_4\}$ and $\{\overline{q}_5\}$ are the only bottom SCCs, it follows from Proposition 8 that $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A})) = 1$.

4.3 Quantitative Analysis

So far, we provided algorithms for checking probabilistic programs against qualitative properties, i.e., that do not compute the exact probability of satisfaction.

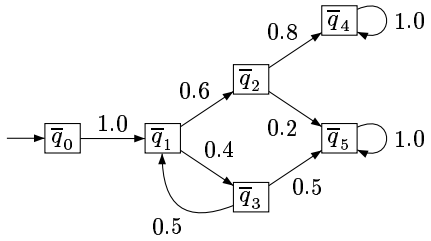

Fig. 7. A sequential probabilistic program

We have seen that the algorithms amount to simple graph analyses as satisfaction of qualitative properties does not depend on exact probabilities but only on the topology of the respective program and its specification at hand. In the following, we generalize the above results.

Proposition 9. *Given a sequential probabilistic program $\mathcal{M}$ and a deterministic Streett automaton $\mathcal{A}$, $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A}))$ is equal to the probability that a trajectory of $\mathcal{M}'$ takes $\mathcal{M}' \times \mathcal{A}$ from the initial state to an accepting bottom SCC.*

To determine $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A}))$, we compute, starting from $\mathcal{M}' \times \mathcal{A}$, a new sequential probabilistic program $abs(\mathcal{M}' \times \mathcal{A})$ by

- removing all states that lie in a bottom SCC D and do not have an immediate predecessor outside of D,
- removing all transitions inside a bottom SCC, and
- adding, for each remaining state (q, s) of a bottom SCC, a transition from (q, s) to itself, which is equipped with probability 1, respectively.

The resulting program $abs(\mathcal{M}' \times \mathcal{A})$ is *absorbing*, i.e., each of its bottom SCCs consists of a single absorbing state. Employing the *fundamental matrix* N of $abs(\mathcal{M}' \times \mathcal{A})$ [6], we can easily compute the probability that a trajectory of $abs(\mathcal{M}' \times \mathcal{A})$ ends in an absorbing state. Suppose $\{\overline{q}_0, \ldots, \overline{q}_{n-1}\}$ is the set of states of $abs(\mathcal{M}' \times \mathcal{A})$ with only initial state $\overline{q}_0$. Without loss of generality, $\{\overline{q}_0, \ldots, \overline{q}_{m-1}\}$ is the set of transient states for a natural $m < n$. Let $\mathsf{T} = (t_{ij})_{i,j \in \{0,\ldots,n-1\}}$ be the transition matrix of $abs(\mathcal{M}' \times \mathcal{A})$, i.e., t_{ij} is the transition probability of going from $\overline{q}_i$ to $\overline{q}_j$. It is of the form

$$\mathsf{T} = \begin{pmatrix} \mathsf{Q} & \mathsf{R} \\ \mathsf{0} & \mathsf{I}_{n-m} \end{pmatrix}$$

where I_{n-m} is the $(n-m) \times (n-m)$ matrix that has 1's on the main diagonal and 0's elsewhere, $\mathsf{0}$ is the matrix of dimension $(n-m) \times m$ that has all components 0, the $m \times (n-m)$ matrix R gives information about the transition probabilities of going from a transient to an absorbing state, and Q is of dimension $m \times m$ and contains the transition probabilities exclusively respecting the set of transient states. For example, assume $abs(\mathcal{M}' \times \mathcal{A})$ to be partly shown in Figure 7 (note

that, in fact, $abs(\mathcal{M}' \times \mathcal{A})$ is absorbing) and suppose the only state that emanates from an accepting bottom SCC is $\overline{q}_4$. Its transition matrix is given by

$$\mathsf{T} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The fundamental matrix of $abs(\mathcal{M}' \times \mathcal{A})$ now arises as follows:

$$\mathsf{N} = (n_{ij})_{i,j \in \{0,\ldots,m-1\}} = (\mathsf{I}_m - \mathsf{Q})^{-1}$$

Hereby, n_{ij} tells us how often a trajectory of $abs(\mathcal{M}' \times \mathcal{A})$ starting at $\overline{q}_i$ is expected to be in $\overline{q}_j$ before absorption. In our example,

$$\mathsf{Q} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \end{pmatrix}, \quad \mathsf{I}_4 - \mathsf{Q} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -0.6 & -0.4 \\ 0 & 0 & 1 & 0 \\ 0 & -0.5 & 0 & 1 \end{pmatrix}, \text{ and}$$

$$\mathsf{N} = (\mathsf{I}_4 - \mathsf{Q})^{-1} = \begin{pmatrix} 1 & 1.25 & 0.75 & 0.5 \\ 0 & 1.25 & 0.75 & 0.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0.625 & 0.375 & 1.25 \end{pmatrix}.$$

Finally,

$$\mathsf{B} = (b_{ij})_{i \in \{0,\ldots,m-1\}, j \in \{0,\ldots,n-m-1\}} = \mathsf{NR}$$

contains the probabilities b_{ij} that $abs(\mathcal{M}' \times \mathcal{A})$, starting at $\overline{q}_i$, reaches the absorbing state $\overline{q}_{j+m}$. Let A be the subset of $\{m, \ldots, n-1\}$ such that $\overline{q}_j$ is an absorbing state of $abs(\mathcal{M}' \times \mathcal{A})$ that emanates from an accepting bottom SCC of $\mathcal{M}' \times \mathcal{A}$ iff $j \in A$. If A is empty, $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A})) = 0$. Otherwise,

$$\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A})) = \sum_{j \in A} b_{0(j-m)}.$$

To complete our example,

$$\mathsf{B} = \mathsf{NR} = \begin{pmatrix} 1 & 1.25 & 0.75 & 0.5 \\ 0 & 1.25 & 0.75 & 0.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0.625 & 0.375 & 1.25 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0.8 & 0.2 \\ 0 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \\ 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$$

lets us come to the conclusion that the probability the underlying sequential probabilistic program will produce a trajectory accepted by the automata specification is 0.6.

Under consideration of the matrix operations, we get an algorithm that determines the exact probability of satisfaction in time single exponential in the size of the Büchi automaton $\mathcal{B}$ and polynomial in the size of the sequential probabilistic program $\mathcal{M}$:

1. Construct a deterministic Streett automaton $\mathcal{A}$ with $L(\mathcal{A}) = L(\mathcal{B})$.
2. Build $\mathcal{M}' \times \mathcal{A}$ and determine its accepting bottom SCCs.
3. Compute $abs(\mathcal{M}' \times \mathcal{A})$ as well as its fundamental matrix N.
4. Compute $\mathsf{B} = \mathsf{NR}$ and, within the first of its rows, sum up the probabilities that belong to states respectively emanating from an accepting bottom SCC of $\mathcal{M}' \times \mathcal{A}$.

We can summarize the above observations as follows:

Theorem 4. *For a sequential probabilistic program $\mathcal{M}$ and a Büchi automaton $\mathcal{A}$, sequential emptiness and sequential universality can be solved in time $O(|\mathcal{M}| \cdot 2^{O(|\mathcal{A}|)})$, respectively. Furthermore, we are able to determine the exact probability $\mu_{\mathcal{M}}(L_{\mathcal{M}}(\mathcal{A}))$ in time single exponential in the size of $\mathcal{A}$ and polynomial in $|\mathcal{M}|$.*

Note that both sequential emptiness and sequential universality for automata are PSPACE-complete in the size of the automaton ([16], [4]).

5 Conclusion

In this chapter, we presented procedures for checking linear temporal logic specifications and automata specifications of sequential and concurrent probabilistic programs. The complexities of these problems are summarized in Table 1. We followed two different approaches: For LTL and sequential probabilistic programs, our method proceeded in a tableau style manner, while for the remaining problems, we followed the automata theoretic approach.

	LTL formulas	ω-automata
sequential programs	$O(\lvert\mathcal{M}\rvert \cdot 2^{O(\lvert\varphi\rvert)})$	$O(\lvert\mathcal{M}\rvert \cdot 2^{O(\lvert\mathcal{A}\rvert)})$
concurrent programs	$O(\lvert\mathcal{M}\rvert^2 \cdot 2^{2^{O(\lvert\varphi\rvert)}})$	$O(\lvert\mathcal{M}\rvert^2 \cdot 2^{O(\lvert\mathcal{A}\rvert)})$

Table 1. Complexity of model checking problems

To gain further insight of probabilistic programs, one might be interested in specifying boundaries for the probabilities of properties. These extension to temporal logics are studied in the next chapters.

6 Bibliographic remarks and further reading

Initial papers studying temporal logics for probabilistic programs are [8] and [5]. Concurrent probabilistic programs were first studied in [16].

Our chapter is mainly based on [4]. The model checking procedure for LTL and sequential probabilistic programs (Theorem 1) is presented in [4]. PSPACE-hardness (wrt. length of formula) of this problem was first shown in [16]. Our algorithms for checking automata specifications are presented or at least suggested

in [4]. Note that for solving the sequential emptiness problem, the authors make use of a slightly different kind of product construction. First complexity results were achieved by Vardi [16] and extended by [4].

A different approach to study probabilistic systems is to understand randomness as a kind of *fairness*. Let us reconsider the example shown in Figure 1, and recall, that nondeterministic states are denoted by circles while probabilistic states are drawn as squares. Intuitively, nondeterminism abstracts that it is not known which possible transition the system under consideration takes. For example, it is possible that the given system chooses to move to state q_3 whenever it is in state q_1. For random states, however, we know that the system chooses a transition with a fixed probability. For example, with probability 0.3, the transition from q_3 to q_1 is taken. This implies that it is unlikely (with probability 0) that the system will choose transition q_3 to q_1 infinitely often. In almost all runs, the system will be *fair* and eventually take the other possible outcome in state q_3 and move to state q_5 (if it moves to state q_3 at all).

The idea of using concepts of fairness to study probabilistic systems first appeared in [11] where the notion of *extreme fairness* was introduced. In [13], *α-fairness* was defined to deal with checking a restricted version of linear temporal logic (but extended with past-tense modalities) specifications of concurrent probabilistic programs.[3] A notion of γ-fairness is introduced in [1] to analyze parameterized probabilistic systems.

References

1. T. Arons, L. Zuck, and A. Pnueli. Automatic verification by probabilistic abstraction. In *submitted to Fossacs'03*, 2003.
2. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. In *Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages*, pages 117–126, Austin, Texas, January 24–26, 1983. ACM SIGACT-SIGPLAN.
3. Edmund M. Clarke and Jeanette M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, December 1996.
4. Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, July 1995.
5. S. Hart and M. Sharir. Probabilistic temporal logics for finite and bounded models. In *ACM Symposium on Theory of Computing (STOC '84)*, pages 1–13, Baltimore, USA, April 1984. ACM Press.
6. J. G. Kemeny and J. L. Snell. *Finite Markov Chains.* Van Nostrand Reinhold, New York, 1960.
7. D. Lehman and M. O. Rabin. On the advantage of free choice: A fully symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of 10th ACM Symposium of Principles of Programming Languages*, pages 133–138, Williamsburg, 1981.

[3] Note that Pnueli follows a different model of sequential and concurrent probabilistic programs. Thus, [13] speaks of sequential programs which correspond to concurrent programs in our terminology.

8. Daniel Lehmann and Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–198, June 1982.
9. O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 97–107, New York, January 1985. ACM.
10. Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57, Providence, Rhode Island, October 31–November 2 1977. IEEE Computer Society Press.
11. Amir Pnueli. On the extremely fair treatment of probabilistic algorithms. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 278–290, Boston, Massachusetts, 25–27 April 1983.
12. P. Pardalos, S. Rajasekaran, J. Reif, and J. Rolim, editors. *Handbook on Randomized Computing*. Kluwer Academic Publishers, Dordrecht, The Netherlands, June 2001.
13. Amir Pnueli and Lenore D. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, March 1993.
14. J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the Fifth International Symposium in Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351, New York, 1982. Springer.
15. Shmuel Safra. On the complexity of omega-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FoCS'88*, pages 319–327, Los Alamitos, California, October 1988. IEEE Computer Society Press.
16. Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th Annual Symposium on Foundations of Computer Science*, pages 327–338, Portland, Oregon, 21–23 October 1985. IEEE.
17. Moshe Y. Vardi. *An Automata-Theoretic Approach to Linear Temporal Logic*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer, New York, NY, USA, 1996.
18. Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, January/February 1983.