Linear-Time Model Checking Branching Processes

Stefan Kiefer
University of Oxford, UK

Pavel Semukhin
University of Oxford, UK

Cas Widdershoven
University of Oxford, UK

Abstract

(Multi-type) branching processes are a natural and well-studied model for generating random infinite trees. Branching processes feature both nondeterministic and probabilistic branching, generalizing both transition systems and Markov chains (but not generally Markov decision processes). We study the complexity of model checking branching processes against linear-time omega-regular specifications: is it the case almost surely that every branch of a tree randomly generated by the branching process satisfies the omega-regular specification? The main result is that for LTL specifications this problem is in PSPACE, subsuming classical results for transition systems and Markov chains, respectively. The underlying general model-checking algorithm is based on the automata-theoretic approach, using unambiguous Büchi automata.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Verification by model checking

Keywords and phrases model checking, Markov chains, branching processes, automata, computational complexity

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2021.6

Related Version *Full Version*: https://arxiv.org/abs/2107.01687

Funding *Stefan Kiefer*: supported by a Royal Society University Research Fellowship.

1 Introduction

Checking whether a (labelled) transition system satisfies a linear-time specification is a staple in verification. The specification is often given as a formula of linear temporal logic (LTL). While early procedures for LTL model checking work directly with the formula [25], the automata-theoretic approach translates LTL formulas into finite automata on infinite words, such as Büchi automata, and analyzes a product of the system and the automaton [37]. This approach can lead to clean and modular model-checking algorithms.

Although LTL captures only a subset of ω-regular languages, model-checking algorithms based on the automata-theoretic approach can be made optimal from the point of view of computational complexity. In particular, model checking finite transition systems against LTL specifications is PSPACE-complete [32], and the algorithm [37] that, loosely speaking, translates (the negation of) the LTL formula into a Büchi automaton and checks the product with the transition system for emptiness can indeed be implemented in PSPACE.

The same approach does not directly work for probabilistic systems modelled as finite Markov chains: intuitively, the nondeterminism in a Büchi automaton causes issues in a stochastic setting where the specification should hold with probability 1, i.e., almost surely but not necessarily surely. A possible remedy is to translate the nondeterministic Büchi automaton further into a deterministic automaton, e.g., a deterministic Rabin automaton (deterministic Büchi automata are less expressive), with which the Markov chain can be naturally instrumented and subsequently analyzed. This determinization step causes a (second) exponential blowup and does not lead to algorithms that are optimal from a computational-complexity point of view. However, for *Markov decision processes (MDPs)*, which allow for nondeterminism in the probabilistic system, this approach is adequate and leads to an optimal, double-exponential time, model-checking algorithm.

Checking whether a Markov chain satisfies an LTL specification with probability 1 is PSPACE-complete, but membership in PSPACE was proved only in [10, 11], not using the automata-theoretic approach but by a recursive procedure on the formula. This raised the question if there is also an optimal algorithm based on the automata-theoretic approach; see [36] for a survey of the state of the art at the end of the 90s.

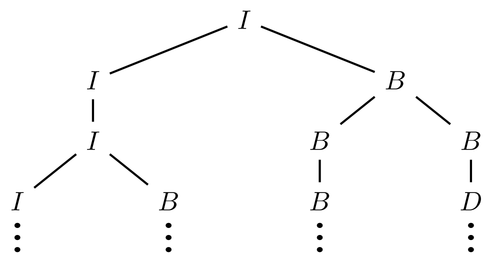
The answer is yes and was first given in [12], using a single-exponential translation from LTL to *separated* Büchi automata. Such automata are special *unambiguous* Büchi automata, which restrict nondeterministic Büchi automata by requiring that every word have at most one accepting run. Another algorithm, using alternating Büchi automata, was proposed in [6], exploiting *reverse determinism*, a property also related to unambiguousness. A polynomial-time (even NC) model-checking algorithm for Markov chains against general unambiguous Büchi automata was given in [2]. These works all imply optimal PSPACE algorithms for LTL model checking of Markov chains via the automata-theoretic approach.

In this paper we exhibit an LTL model checking algorithm that has the following features: (1) it applies to (multi-type) branching processes, a well established model for random trees, generalizing both nondeterministic transition systems and Markov chains; (2) it runs in PSPACE, which is the optimal complexity both for nondeterministic transition systems and Markov chains; and (3) it is based on the automata-theoretic approach (using unambiguous Büchi automata). The fact that there exists an algorithm with the first two features might seem surprising, as one might think that any system model that encompasses both nondeterminism and probability will generalize MDPs, for which LTL model checking is 2EXPTIME-complete [11].

Branching processes (BPs) are a well-studied model in mathematics with applications in numerous fields including biology, physics and natural language processing; see, e.g., [23, 1, 22]. BPs randomly generate infinite trees, and, from a computer-science point of view, they might be the most natural model to do so: (multi-type) BPs can be thought of as a version of stochastic context-free grammars without terminal symbols, randomly generating infinite derivation trees. For example, consider the following BP, taken from [8], with 3 *types* I, B, D:

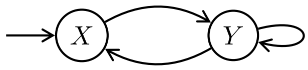
$$\begin{array}{lll} I \xhookrightarrow{0.9} I & B \xhookrightarrow{0.2} D & D \xhookrightarrow{1} D \\ I \xhookrightarrow{0.1} IB & B \xhookrightarrow{0.5} B & \\ & B \xhookrightarrow{0.3} BB & \end{array} \tag{1}$$

This BP might generate a tree with the following prefix:

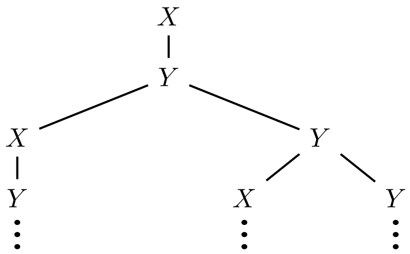


The probability that the BP generates a tree with the shown prefix is the product of the probabilities of the fired transition rules, i.e., (in breadth-first order) $0.1 \cdot 0.9 \cdot 0.3 \cdot 0.1 \cdot 0.5 \cdot 0.2$.

BPs generalize transition systems. Consider the following transition system:

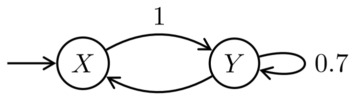


It is equivalent to the BP with $X \overset{1}{\hookrightarrow} Y$ and $Y \overset{1}{\hookrightarrow} XY$, which generates with probability 1 the following unique tree:

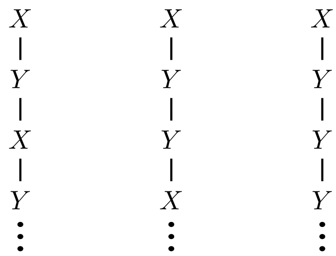


The branches of this unique tree are exactly the executions of the transition system. As a consequence, any LTL formula holds on all executions of the transition system if and only if it holds (with probability 1) on all branches of the generated tree.

BPs also generalize Markov chains. Consider the following Markov chain:



It is equivalent to the BP with $X \overset{1}{\hookrightarrow} Y$ and $Y \overset{0.3}{\hookrightarrow} X$ and $Y \overset{0.7}{\hookrightarrow} Y$, which generates, with probabilities 0.3, $0.7 \cdot 0.3$, $0.7 \cdot 0.7$, respectively, the following prefixes of (degenerated) trees:



Here, each possible "tree" has only a single branch, and the possible "trees" are distributed in the same way as the possible executions of the Markov chain. As a consequence, any LTL formula holds with probability 1 on a random execution of the Markov chain if and only if it holds with probability 1 on the (single) branch of the generated tree.

Hence, both for the transition system and for the Markov chain, the respective model-checking question reduces to the *BP model-checking problem* which asks whether with probability 1 the property holds on *all* branches.

For LTL specifications, we refer to this BP model-checking problem as $\mathbb{P}(\text{LTL}) = 1$. Our main result is that it is in PSPACE, generalizing the corresponding classical results on transition systems and Markov chains. As mentioned, our model-checking algorithm is based on the automata-theoretic approach, in particular on unambiguous Büchi automata. Another important technical ingredient is the algorithmic analysis of certain nonnegative matrices in terms of their spectral radius.

The latter points to the fact that the numbers in the system generally matter, even though we only consider the qualitative problem of comparing the satisfaction probability with 1. For example, for the BP given in (1), one can show that the probability that all branches eventually hit a node of type D is less than 1 (in fact, it is 0). Intuitively, this is because the probability of "branching" via $B \xhookrightarrow{0.3} BB$ is larger than the probability of "dying" via $B \xhookrightarrow{0.2} D$. Were the probabilities 0.3 and 0.2 swapped, the probability that all branches eventually hit a node of type D would be 1; cf. [8, Section 1].

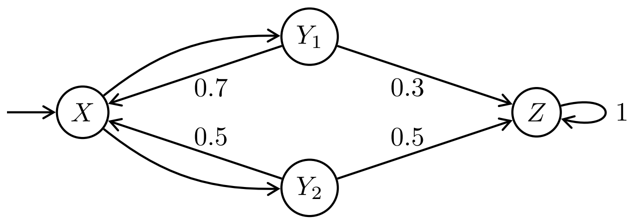
We also consider the problem $\mathbb{P}(\text{LTL} = 0)$, which asks whether the probability that *all* branches satisfy a given LTL formula is 0. Even though it is trivial to negate an LTL formula, this problem is (unlike in Markov chains) not equivalent to the complement of $\mathbb{P}(\text{LTL} = 1)$, because even when the probability is less than 1 that the formula holds on all branches, the probability may still be 0 that the negated formula holds on all branches. We will show that $\mathbb{P}(\text{LTL} = 0)$ is much more computationally complex than $\mathbb{P}(\text{LTL} = 1)$: it is 2EXPTIME-complete.

Besides LTL, we also consider automata-based specifications. Büchi automata are relevant from a verification point of view, as a way of specifying desired or undesired executions of the system. Unambiguous Büchi automata are useful from a technical point of view, in particular, to facilitate our main result on $\mathbb{P}(\text{LTL} = 1)$. See Section 2 and Table 1 for definitions of our problems and a map of our results.

▶ Remark 1. Readers familiar with MDPs may wonder how the problem $\mathbb{P}(\text{LTL}) = 1$ can have lower computational complexity than the problem whether all schedulers of an MDP satisfy an LTL specification almost surely. Consider the BP

$$X \xhookrightarrow{1} Y_1Y_2 \qquad Y_1 \xhookrightarrow{0.7} X \qquad Y_1 \xhookrightarrow{0.3} Z \qquad Y_2 \xhookrightarrow{0.5} X \qquad Y_2 \xhookrightarrow{0.5} Z \qquad Z \xhookrightarrow{1} Z\,,$$

which might be depicted graphically as follows:



One might view this BP as an MDP where in an X-node the scheduler nondeterministically picks either the Y_1- or the Y_2-successor, and in an Y_i-node, the X- or the Z-successor is chosen randomly. In such an MDP, regardless of the scheduler, a random run reaches with probability 1 a Z-node. However, in the BP above, the probability is positive that *some* branch of a random tree never reaches a Z-node. Although each branch of a random tree could be thought of as being witnessed by at least one scheduler, this is not a contradiction, as there are uncountably many schedulers (over which one cannot take a sum). Hence, if an MDP is interpreted as a BP in the way sketched above, then the requirement that the BP satisfy an LTL formula almost surely on *all* branches is *stronger*, and computationally less complex to check, than the requirement that the MDP satisfy, for *each* scheduler, the formula almost surely.

Related work. We have already discussed related work concerning model checking transition systems and Markov chains.

In addition to the mentioned applications of BPs in various fields, there has also been work on BPs in computer science, especially in the last 10 years. This paper builds on [8], where specifications in terms of deterministic parity *tree* automata are considered. The work [8] implies decidability of the problems considered in this paper and some basic upper complexity bounds. For example, it is not hard to derive from [8] that $\mathbb{P}(\text{LTL} = 1)$ is in 2EXPTIME. Lowering this to PSPACE is the main achievement of this paper.

A related strand of work considers *regular tree languages*; i.e., the specification is not in terms of a word automaton that is run on each branch but in terms of tree automata. Even measurability is not easy to show in this case [20], and fundamental decidability questions around computing the measure have been answered positively only for subclasses of regular tree languages [26, 27].

Fundamental results on the complexity of algorithmically analyzing BPs have been obtained in [18]. Indeed, in Section 3.1 we build on and improve results from [18] on *finiteness* (more often called "extinction" in the literature) of BPs.

Another recent line of work considers extensions of BPs with nondeterminism, focusing on algorithmic questions about properties such as reachability. *Branching MDPs*, which are BPs where a controller chooses actions to influence the evolution of the tree, have been investigated, e.g., in [16, 17]. Even branching *games*, featuring two adversarial controllers, have been studied recently [14].

The work [21] also considers BPs with "internal" nondeterminism (as opposed to the "external" nondeterminism manifested as branching in the generated tree), along with model-checking problems against the logic *GPL*. This expressive, μ-calculus based modal logic had been introduced in [9]. The system model therein, called *reactive probabilistic labeled transition systems (RPLTSs)*, is essentially equivalent to BPs as considered in this paper.

BPs are related to models for probabilistic programs with recursion, such as *Recursive Markov chains*, for which model-checking problems have been studied in detail; see, in particular, [19]. Very loosely speaking, a run of a ("1-exit") Recursive Markov chain can be viewed as a depth-first traversal of a tree generated by a BP. Indeed, for a lower bound in the present paper (Theorem 11) we adapt a proof from [19]. However, most qualitative model-checking problems for Recursive Markov chains are EXPTIME-complete [19], and so many of the BP problems we study turn out to have different computational complexity.

As a key technical tool we use unambiguous Büchi automata, as recently proposed for Markov chains [2]. It is non-trivial to extend their use to random trees, as the branching behaviour of BPs interferes with the spectral-radius based analysis from [2]. One may view as the main technical insight of this paper that the limited nondeterminism in unambiguous automata can be combined with the tree branching of BPs, so that, in a sense, BP model checking reduces to comparing the spectral radius of a certain nonnegative matrix with 1 (Proposition 16).

2 Preliminaries

Let $\mathbb{N}$ and $\mathbb{N}_0$ denote the set of positive and nonnegative integers, respectively. For a finite set Γ, we write Γ^* (resp., Γ^+) for the set of words (resp., nonempty words) over Γ.

Branching processes. A *(multi-type) branching process (BP)* is a tuple $\mathcal{B} = (\Gamma, \hookrightarrow, \mathit{Prob}, X_0)$, where Γ is a finite set of types, $\hookrightarrow \subseteq \Gamma \times \Gamma^+$ is a finite set of transition rules, Prob is a function assigning positive rational probabilities to transition rules so that for every $X \in \Gamma$ we have $\sum_{X \hookrightarrow w} \mathit{Prob}(X \hookrightarrow w) = 1$, and $X_0 \in \Gamma$ is the start type. We write $X \overset{p}{\hookrightarrow} w$ to denote that $Prob(X \hookrightarrow w) = p$. Given a BP $\mathcal{B}$ and a type $X \in \Gamma$ we write $\mathcal{B}[X]$ for the BP obtained from $\mathcal{B}$ by making X the start type. For $X, Y \in \Gamma$ we call Y a *successor* of X if there is a rule $X \hookrightarrow uYv$ for some $u, v \in \Gamma^*$.

A BP *with ε-rules allowed* relaxes the requirement $\hookrightarrow \subseteq \Gamma \times \Gamma^+$ to $\hookrightarrow \subseteq \Gamma \times \Gamma^*$, i.e., there may be rules of the form $X \hookrightarrow \varepsilon$, where ε denotes the empty word. In the following, we disallow ε-rules unless specified otherwise; but the definitions generalize in a natural way.

Fix a BP $\mathcal{B} = (\Gamma, \hookrightarrow, Prob, X_0)$ for the rest of the section.

Trees. Write $[\![\mathcal{B}]\!]$ for the set of trees *generated* by $\mathcal{B}$; i.e., $[\![\mathcal{B}]\!]$ denotes the set of ordered Γ-labelled trees t such that for each $X \in \Gamma$ and each X-labelled node v in t, there is a rule $X \hookrightarrow X_1 \cdots X_k$, denoted by $rule(v)$, such that the k ordered children of v are labelled with $X_1, \ldots, X_k$, respectively. We say a node *has type* $X \in \Gamma$ if the node is labelled with X. A *finite prefix* of a tree $t \in [\![\mathcal{B}]\!]$ is an ordered Γ-labelled *finite* tree obtained from t by designating some nodes as leaves, and removing all their children, grandchildren, etc. Write $(\!|\mathcal{B}|\!)$ for the set of finite prefixes of trees generated by $\mathcal{B}$. For $t \in (\!|\mathcal{B}|\!)$ write $t{\downarrow} \subseteq [\![\mathcal{B}]\!]$ for the ("cylinder") set of trees $t' \in [\![\mathcal{B}]\!]$ such that t is a finite prefix of t'. For $X \in \Gamma$ write $[\![\mathcal{B}]\!]_X \subseteq [\![\mathcal{B}]\!]$ and $(\!|\mathcal{B}|\!)_X \subseteq (\!|\mathcal{B}|\!)$ for the subsets of trees whose root has type X; the trees in $[\![\mathcal{B}]\!]_X$ are called *X-trees.* A *branch* of a tree t is a sequence $v_0 v_1 \cdots$ of nodes in t, where v_0 is the root of t and v_{i+1} is a child of v_i for all $i \in \mathbb{N}_0$. See [8] for equivalent, more formal tree-related definitions.

Probability space. For each $X \in \Gamma$ we define the probability space $([\![\mathcal{B}]\!]_X, \Sigma_X, \mathbb{P}_X)$, where Σ_X is the σ-algebra generated by $\{t{\downarrow} \mid t \in (\!|\mathcal{B}|\!)_X\}$, and $\mathbb{P}_X$ is the probability measure generated by $\mathbb{P}_X(t{\downarrow}) := \prod_v Prob(rule(v))$ for all $t \in (\!|\mathcal{B}|\!)_X$, where the product extends over all non-leaf nodes v in t. This is analogous to the standard definition of the probability space of a Markov chain. We may write $\mathbb{P}_\mathcal{B}$ for $\mathbb{P}_{X_0}$, omitting the subscript when $\mathcal{B}$ is understood. We often talk about events (i.e., measurable sets of trees) and their probability in text form. For example, by saying "a $\mathcal{B}$-tree has with positive probability infinitely many nodes of type X" we mean that $\mathbb{P}_\mathcal{B}(E) > 0$ where $E \subseteq [\![\mathcal{B}]\!]_{X_0}$ is the set of X_0-trees with infinitely many nodes of type X.

Linear-Time Properties. We are particularly interested in sets of trees *all whose branches* (more precisely, their associated sequences of types) satisfy an ω-regular linear-time property $L \subseteq \Gamma^\omega$. Given $L \subseteq \Gamma^\omega$, we write $\mathbb{P}_\mathcal{B}(L)$ for the probability that *all* branches of a $\mathcal{B}$-tree satisfy L. *Linear temporal logic (LTL)* formulas specify linear-time properties; see, e.g., [34] for a definition of LTL. An important example for us are formulas of the form $\mathsf{F}T$, where $T \subseteq \Gamma$, which denotes the linear-time property $\{uXw \mid u \in \Gamma^*,\ X \in T,\ w \in \Gamma^\omega\}$. Accordingly, $\mathbb{P}_\mathcal{B}(\mathsf{F}T)$ denotes the probability that all branches of a $\mathcal{B}$-tree have a node whose type is in T (equivalently, the probability that a $\mathcal{B}$-tree has a finite prefix all whose leaves have a type in T).

Automata. We use finite automata on infinite words over Γ, where Γ is the set of types of a BP. We use *deterministic parity automata (DPAs)*, *deterministic Büchi automata (DBAs)*, *nondeterministic Büchi automata (NBAs)*, and *unambiguous Büchi automata (UBAs)*. The definitions are standard; see, e.g., [34]. In the following we fix some terms and notation. Let $\mathcal{A} = (Q, \Gamma, \delta, Q_0, F)$ be an NBA, where Q is a finite set of states, Γ is the alphabet, $\delta \subseteq Q \times \Gamma \times Q$ is the transition relation, $Q_0 \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of accepting states. We write $q \xrightarrow{X} r$ to denote that $(q, X, r) \in \delta$. A finite sequence $q_0 \xrightarrow{X_1} q_1 \xrightarrow{X_2} \cdots \xrightarrow{X_n} q_n$ is called a *path* and can be summarized as $q_0 \xrightarrow{X_1 \cdots X_n}{}^{*}\, q_n$. An

Table 1 Results and organization of the paper. The complexity classes indicate completeness results, except "in NC", which only means membership in NC.

	$=1$	$=0$		$=1$	$=0$
$\mathbb{P}$(finite) Section 3.1	in NC Proposition 6		$\mathbb{P}$(coNBA) Section 4	PSPACE Theorem 14	EXPTIME Theorem 15
$\mathbb{P}$(DPA) Section 3.2	in NC Theorem 8	P Theorem 9	$\mathbb{P}$(coUBA) Section 5	in NC Proposition 16	
$\mathbb{P}$(NBA) Section 3.3	PSPACE Theorem 10	EXPTIME Theorem 11	$\mathbb{P}$(LTL) Section 6	PSPACE Theorem 18	2EXPTIME Theorem 19

infinite sequence $q_0 \xrightarrow{X_1} q_1 \xrightarrow{X_2} \cdots$ is called a *run* of $X_1 X_2 \cdots$. We call the run *accepting* if $q_0 \in Q_0$ and $q_i \in F$ holds for infinitely many q_i. The NBA $\mathcal{A}$ *accepts* (resp., *rejects*) an infinite word $w \in \Gamma^\omega$ if w has (resp., does not have) an accepting run in $\mathcal{A}$. The NBA $\mathcal{A}$ is called an *unambiguous Büchi automaton (UBA)* if every $w \in \Gamma^\omega$ has at most one accepting run. An automaton $\mathcal{A}$ defines ω-regular linear-time properties $\{w \in \Gamma^\omega \mid \mathcal{A} \text{ accepts } w\}$ and $\{w \in \Gamma^\omega \mid \mathcal{A} \text{ rejects } w\}$. In keeping with previous definitions, we write $\mathbb{P}_\mathcal{B}(\mathcal{A} \text{ accepts})$ (resp., $\mathbb{P}_\mathcal{B}(\mathcal{A} \text{ rejects})$) for the probability that *all* branches of a $\mathcal{B}$-tree (more precisely, their associated sequences of types) are accepted (resp., rejected) by $\mathcal{A}$.

Problems. We consider the following computational problems. The problem $\mathbb{P}(\text{finite}) = 1$ asks, given a BP $\mathcal{B}$ with ε-rules allowed, whether the probability that a $\mathcal{B}$-tree is finite is 1. The problem $\mathbb{P}(\text{LTL}) = 1$ asks, given a BP $\mathcal{B}$ and an LTL formula φ, whether $\mathbb{P}_\mathcal{B}(\varphi) = 1$. The problems $\mathbb{P}(\text{DPA}) = 1$ (resp., $\mathbb{P}(\text{NBA}) = 1$) ask, given a BP $\mathcal{B}$ and a DPA (resp., NBA) $\mathcal{A}$, whether $\mathbb{P}_\mathcal{B}(\mathcal{A} \text{ accepts}) = 1$. The problems $\mathbb{P}(\text{coNBA}) = 1$ (resp., $\mathbb{P}(\text{coUBA}) = 1$)[1] ask, given a BP $\mathcal{B}$ and an NBA (resp., UBA) $\mathcal{A}$, whether $\mathbb{P}_\mathcal{B}(\mathcal{A} \text{ rejects}) = 1$. The problems $\mathbb{P}(\text{LTL}) = 0, \mathbb{P}(\text{DPA}) = 0, \ldots$ are defined similarly, where "$=1$" is replaced with "$=0$". See Table 1 for a map of our results in those terms, as well as for an overview of the rest of the paper. As explained in the introduction, the problem $\mathbb{P}(\text{LTL}) = 1$ is of particular interest from a model-checking point of view, and the technically most challenging one.

Complexity Classes. In addition to standard complexity classes between P and 2EXPTIME, we use the class NC, the subclass of P comprising those problems solvable in polylogarithmic time by a parallel random-access machine using polynomially many processors; see, e.g., [28, Chapter 15]. To prove membership in PSPACE in a modular way, we will use the following pattern:

▶ **Lemma 2.** *Let P_1, P_2 be two problems, where P_2 is in NC. Suppose there is a reduction from P_1 to P_2 implemented by a PSPACE transducer, i.e., a Turing machine whose work tape (but not necessarily its output tape) is PSPACE-bounded. Then P_1 is in PSPACE.*

Proof. Note that the output of the transducer is (at most) exponential. Problems in NC can be decided in polylogarithmic space [4, Theorem 4]. Using standard techniques for composing space-bounded transducers (see, e.g., [28, Proposition 8.2]), it follows that P_1 is in PSPACE. ◀

[1] We do not explicitly define or use a notion of "co-Büchi automata" to avoid possible confusion about accepting/rejecting. If one were to do so, one would define a "co-NBA" $\mathcal{A}$ like an NBA $\mathcal{A}$, but the "co-NBA" $\mathcal{A}$ would accept a word $w \in \Gamma^\omega$ if and only if $\mathcal{A}$ viewed as an NBA rejects w. Similarly for "co-UBAs".

Matrices. We use finite sets S to index matrices $M \in \mathbb{R}^{S\times S}$ and vectors $v \in \mathbb{R}^S$. The *graph* of a nonnegative matrix $M \in [0,\infty)^{S\times S}$ is the directed graph (S,E) with $E = \{(s,t) \in S\times S \mid M_{s,t} > 0\}$. The *spectral radius* of a matrix is the largest absolute value of its eigenvalues. The following lemma allows to efficiently compare the spectral radius of a nonnegative matrix with 1.

▶ **Lemma 3.** *Given a nonnegative rational matrix M, one can determine in NC whether $\rho < 1$ or $\rho = 1$ or $\rho > 1$, where ρ denotes the spectral radius of M.*

Proof. Use the algorithm from [13, Proposition 2.2], but not with Gaussian elimination as suggested there, but by solving the systems of linear equations described in [13, Proposition 2.2] in NC. The latter is possible in NC [5, Theorem 5]. ◀

3 Basic Results

In this section we develop the more basic results indicated in Table 1, on finiteness (Section 3.1), deterministic parity automata (Section 3.2), and Büchi automata (Section 3.3), on the one hand rounding off the complexity map in Table 1, and on the other hand building the foundation for more challenging results in the following sections. In particular, Proposition 6 is indirectly used throughout the paper.

3.1 Finiteness

In this section we consider BPs with ε-rules allowed, i.e., rules of the form $X \hookrightarrow \varepsilon$. Such BPs may generate finite trees. We are interested in the *almost-sure finiteness* problem, also denoted as $\mathbb{P}(\text{finite}) = 1$, i.e., the problem whether the probability that a given BP with ε-rules allowed generates a finite tree is equal to 1. In Proposition 6 below we show that this problem is in NC. All upper bounds on the complexity of $\mathbb{P}(\cdot) = 1$ problems in this paper build directly or indirectly on this result.

While the almost-sure finiteness (or "extinction") problem has often been studied and is known to be in (strongly) polynomial time [18, 13], its membership in NC is, to the best of the authors' knowledge, new. For instance, since linear programming is P-complete, one cannot use linear programming (as in [18]) to show membership in NC. Nor can one directly use the strongly polynomial-time algorithm of [13], as it computes, in a sub-procedure, the set of types X for which there *exists* a finite X-tree. But the latter problem is P-complete.

For the rest of the section, fix a BP $\mathcal{B} = (\Gamma, \hookrightarrow, \mathit{Prob}, X_0)$ with ε-rules allowed. Define a directed graph $G = (\Gamma, E)$ (i.e., the types of $\mathcal{B}$ are the vertices of G) with an edge $(X,Y) \in E$ if and only if Y is a successor of X (i.e., there is a rule $X \hookrightarrow uYv$ for some $u,v \in \Gamma^*$). Given a strongly connected component (SCC) $S \subseteq \Gamma$ of G and $X \in S$, define a BP $\mathcal{B}[S,X] = (S, \hookrightarrow_S, \mathit{Prob}_S, X)$ obtained from $\mathcal{B}$ by restricting the types to S and deleting on all right-hand sides of the rules those types not in S. The following lemma is straightforward:

▶ **Lemma 4.** *A $\mathcal{B}$-tree is infinite with positive probability if and only if there exist an SCC $S \subseteq \Gamma$ of G and $X \in S$ such that X is reachable from X_0 in G and a $\mathcal{B}[S,X]$-tree is infinite with positive probability.*

Let $M \in \mathbb{Q}^{\Gamma\times\Gamma}$ be the nonnegative $\Gamma \times \Gamma$-matrix with $M_{X,Y} = \sum_{X \overset{p}{\hookrightarrow} w} p|w|_Y$, where $|w|_Y \in \mathbb{N}_0$ is the number of occurrences of Y in w. That is, $M_{X,Y}$ is the expected number of direct Y-successors of the root of a $\mathcal{B}[X]$-tree. By induction, M^i, the ith power of M, is such that $(M^i)_{X,Y}$ is the expected number of Y-nodes that are exactly i levels under the root of a $\mathcal{B}[X]$-tree. The graph of M is exactly the previously defined graph G.

Let $S \subseteq \Gamma$ be an SCC of G. Denote by $M_S \in \mathbb{Q}^{S \times S}$ the (square) principal submatrix obtained from M by restricting it to the rows and columns indexed by elements of S. Let ρ_S denote the spectral radius of M_S. Call S *supercritical* if $\rho_S > 1$. Call S *linear* if for all rules $X \hookrightarrow w$ with $X \in S$ there is exactly one occurrence in w of a type in S. Observe that if S is linear then M_S is *stochastic*, i.e., $M_S\vec{1} = \vec{1}$ where $\vec{1}$ is the all-1 vector, i.e., the element of $\{1\}^S$. In that case, by the Perron-Frobenius theorem [3, Theorem 2.1.4 (b)], we have $\rho_S = 1$ and, thus, S is not supercritical.

The following characterization can be proved using [13, Section 3] (which builds on [18, Section 8.1]):

▶ **Lemma 5.** *A $\mathcal{B}$-tree is infinite with positive probability if and only if there exist an SCC $S \subseteq \Gamma$ of G and $X \in S$ such that X is reachable from X_0 in G and S is supercritical or linear.*

It follows:

▶ **Proposition 6.** *The problem $\mathbb{P}(\mathit{finite}) = 1$ is in NC.*

3.2 Deterministic Parity Automata

In this section we consider deterministic parity automata (DPAs) on words. In [8, Section 3] it was shown that the problem $\mathbb{P}(\mathrm{DPA}) = 1$ can be decided in polynomial time. We improve this to membership in NC.

By the following lemma we can check in NC whether a $\mathcal{B}$-tree almost surely has a finite prefix all whose leaves have types in a given set T. The proof is by reduction to almost-sure finiteness.

▶ **Lemma 7.** *Given a BP $\mathcal{B} = (\Gamma, \hookrightarrow, \mathit{Prob}, X_0)$ and a set of types $T \subseteq \Gamma$, the problem whether $\mathbb{P}_{X_0}(\mathsf{F}T) = 1$ is in NC.*

By combining Lemma 7 with results from [8] we obtain:

▶ **Theorem 8.** *The problem $\mathbb{P}(\mathrm{DPA}) = 1$ is in NC.*

The hardness result in the following theorem highlights the different complexities of $\mathbb{P}(\cdot) = 0$ and $\mathbb{P}(\cdot) = 1$ problems in this paper.

▶ **Theorem 9.** *The problem $\mathbb{P}(\mathrm{DPA}) = 0$ is P-complete. It is P-hard even for deterministic Büchi automata with two states, the accepting state being a sink.*

3.3 Büchi Automata

▶ **Theorem 10.** *The problem $\mathbb{P}(\mathrm{NBA}) = 1$ is PSPACE-complete.*

Proof. PSPACE-hardness is immediate in two different ways. It follows from the PSPACE-hardness of model checking Markov chains against NBAs [35]. It also follows from the PSPACE-hardness of model checking transition systems against NBAs. (The latter follows easily from the PSPACE-hardness of NBA universality [32].) Both model-checking problems are special cases of $\mathbb{P}(\mathrm{NBA}) = 1$.

Towards membership in PSPACE, we use a translation from NBA to DPA [29]. This translation causes an exponential blow-up, but an inspection of the construction [29, Section 3.2] reveals that it can be computed by a PSPACE transducer. By Theorem 8 the problem $\mathbb{P}(\mathrm{DPA}) = 1$ is in NC. By Lemma 2 it follows that $\mathbb{P}(\mathrm{NBA}) = 1$ is in PSPACE. ◀

▶ **Theorem 11.** *The problem $\mathbb{P}(\text{NBA}) = 0$ is EXPTIME-complete. It is EXPTIME-hard even for NBAs whose only accepting state is a sink.*

Proof. Towards membership in EXPTIME, an NBA can be translated, in exponential time, to a DPA of exponential size; see, e.g., [29]. Since $\mathbb{P}(\text{DPA}) = 0$ is in P by Theorem 9, it follows that $\mathbb{P}(\text{NBA}) = 0$ is in EXPTIME.

Concerning EXPTIME-hardness, we adapt the proof (in the online appendix) of [19, Theorem 17] on model checking *recursive Markov chains* against NBAs. The details are in [24]. ◀

4 Co-Büchi Automata

In this section we consider the problem $\mathbb{P}(\text{coNBA}) = 1$, which asks, given a BP $\mathcal{B}$ and a Büchi automaton $\mathcal{A}$, whether $\mathcal{B}$ almost surely generates a tree whose branches are all rejected by $\mathcal{A}$; i.e., whether $\mathbb{P}_{\mathcal{B}}(\mathcal{A} \text{ rejects}) = 1$. Dually, one might ask whether the probability is positive that a $\mathcal{B}$-tree has a branch accepted by $\mathcal{A}$. Intuitively, we view the Büchi automaton $\mathcal{A}$ as specifying "bad" branches, and we would like the tree almost surely not to have any bad branches.

This problem is in PSPACE, which can be shown via a translation to DPAs, as in Theorem 10. However, with a view on the following sections, in particular on LTL specifications, we pursue a different approach to the problem $\mathbb{P}(\text{coNBA}) = 1$. In this section we lay the groundwork for arbitrary Büchi automata $\mathcal{A}$. By building on these results, we will show in the next section that if $\mathcal{A}$ is *unambiguous* then the problem is in NC, which will allow us to derive our headline result, namely that $\mathbb{P}(\text{LTL}) = 1$ is in PSPACE.

Let $\mathcal{B} = (\Gamma, \hookrightarrow, \mathit{Prob}, X_0)$ be a BP and $\mathcal{A} = (Q, \Gamma, \delta, Q_0, F)$ a (not necessarily unambiguous) Büchi automaton.

Define a Büchi automaton, $\mathcal{A} \times \mathcal{B}$, by $\mathcal{A} \times \mathcal{B} := (Q \times \Gamma, \Gamma, \delta_{\mathcal{A}\times\mathcal{B}}, Q_0 \times \{X_0\}, F \times \Gamma)$, where

$$\delta_{\mathcal{A}\times\mathcal{B}}((q_1, X_1), X_2) = \begin{cases} \delta(q_1, X_1) \times \{X_2\} & \text{if } X_2 \text{ is a successor of } X_1 \\ \emptyset & \text{otherwise.} \end{cases}$$

The remainder of the section is organized as follows. In Section 4.1 we show that the problem $\mathbb{P}(\text{coNBA}) = 1$ reduces to the analysis of certain SCCs within $\mathcal{A} \times \mathcal{B}$. In Section 4.2 we introduce a key lemma, Lemma 13, which allows us to "forget" about the distinction between accepting and non-accepting states: the lemma reduces $\mathbb{P}(\text{coNBA}) = 1$ to a pure reachability problem in an exponential-sized BP, $\mathcal{B}_{det}$. This leads us to prove PSPACE-completeness of $\mathbb{P}(\text{coNBA}) = 1$, but more importantly, Lemma 13 plays a key role in the rest of the paper. We prove it in [24].

4.1 The Automaton $\mathcal{A}[f, X_f]$

For any $(f, X_f) \in F \times \Gamma$ on a cycle of the transition graph of $\mathcal{A} \times \mathcal{B}$, define the Büchi automaton

$$\mathcal{A}[f, X_f] \;:=\; (\{\bar{q}_0\} \cup Q[f, X_f], \Gamma, \delta[f, X_f], \{\bar{q}_0\}, \{(f, X_f)\})$$

as the Büchi automaton obtained from $\mathcal{A} \times \mathcal{B}$ by

1. making (f, X_f) the only accepting state,
2. restricting the set of states, $Q[f, X_f] \subseteq Q \times \Gamma$, to those (q, X) that, in the transition graph of $\mathcal{A} \times \mathcal{B}$, are reachable from (f, X_f) and can reach (f, X_f), i.e., those (q, X) in the SCC containing (f, X_f),

3. restricting the transition function $\delta[f, X_f]$ accordingly, i.e.,

$$\delta[f, X_f]((q, X), Y) \;:=\; \delta_{\mathcal{A}\times\mathcal{B}}((q, X), Y) \cap Q[f, X_f]\,,$$

4. making $\bar{q}_0$ the only initial state, and
5. setting $\delta[f, X_f](\bar{q}_0, X_f) := \{(f, X_f)\}$ and $\delta[f, X_f](\bar{q}_0, X) := \emptyset$ for all $X \in \Gamma \setminus \{X_f\}$.

The following lemma follows from the pigeonhole principle and basic probability arguments:

▶ **Lemma 12.** *The probability that some branch of a $\mathcal{B}$-tree is accepted by $\mathcal{A}$ is positive if and only if there are $q_0 \in Q_0$ and $f \in F$ and $X_f \in \Gamma$ such that (f, X_f) is reachable from (q_0, X_0) in the transition graph of $\mathcal{A} \times \mathcal{B}$ and the probability that some branch of a $\mathcal{B}[X_f]$-tree is accepted by $\mathcal{A}[f, X_f]$ is positive.*

For the rest of the section let $(f, X_f) \in F \times \Gamma$ be on a cycle of the transition graph of $\mathcal{A} \times \mathcal{B}$.

4.2 The Determinization $\mathcal{A}_{det}$ and the BP $\mathcal{B}_{det}$

Let

$$\mathcal{A}_{det} := (2^{\{\bar{q}_0\}\cup Q[f,X_f]}, \Gamma, \delta_{det}, \{\bar{q}_0\}, 2^{\{\bar{q}_0\}\cup Q[f,X_f]} \setminus \{\emptyset\})$$

be the determinization of $\mathcal{A}[f, X_f]$ obtained by the standard subset construction. Which states are accepting will not actually be relevant. Note that every state reachable via a nonempty path from $\{\bar{q}_0\}$ is of the form $P \times \{X\}$ with $P \subseteq Q$ and $X \in \Gamma$.

Define a BP $\mathcal{B}_{det}$ based on $\mathcal{A}_{det}$ as

$$\mathcal{B}_{det} \;:=\; (\Gamma', \hookrightarrow', \mathit{Prob}', \{(f, X_f)\})\,,$$

where the set of types $\Gamma' \subseteq 2^{Q[f,X_f]}$ is the set of those states in $\mathcal{A}_{det}$ that are reachable (in $\mathcal{A}_{det}$) from $\{\bar{q}_0\}$ via a nonempty path (recall that they are of the form $P \times \{X\}$ with $P \subseteq Q$ and $X \in \Gamma$), and

$$X' \overset{p}{\hookrightarrow}{}' \; \delta_{det}(X', X_1) \cdots \delta_{det}(X', X_k)$$

for all $X' = P \times \{X\} \in \Gamma'$ with $P \neq \emptyset$ and all $X \overset{p}{\hookrightarrow} X_1 \cdots X_k$, and $\emptyset \overset{1}{\hookrightarrow}{}' \emptyset$. Here is the key lemma of this section:

▶ **Lemma 13.** *The following statements are equivalent:*

(i) *The probability that some branch of a $\mathcal{B}[X_f]$-tree is accepted by $\mathcal{A}[f, X_f]$ is positive.*

(ii) *The probability that some branch of a $\mathcal{B}_{det}$-tree does not have any nodes of type $\emptyset$ is positive.*

We prove Lemma 13 in [24]. It will be used in the proof of Theorem 14 below; but more importantly, Lemma 13 is the foundation of Section 5.

Given that Lemma 13 reflects the key insight of this section, let us comment further. Considering that condition (ii) does not mention a notion of acceptance, one might have two concerns at this point:

(a) Condition (ii) does not obviously imply that with positive probability there is even a branch with infinitely many nodes of types containing (f, X_f).

(b) Even if with positive probability there is such a branch, it is not obvious that such branches would necessarily correspond to branches of $\mathcal{B}[X_f]$ that are accepted by $\mathcal{A}[f, X_f]$.

Even for the special case of Markov chains (i.e., every tree has only a single branch), Lemma 13 is not at all obvious, and both concerns (a) and (b) apply. Indeed, for Markov chains, Courcoubetis and Yannakakis prove a statement related to Lemma 13, namely [11, Proposition 4.1.4], with a proof related to ours and dealing explicitly with concern (b) above. For the special case of transition systems (i.e., the BP generates exactly one tree), Lemma 13 is simple though: consider the branch that follows a cycle around (f, X_f). For the general case, we need a result on BPs from [8], dealing with concern (a) above. The high-level principle behind the proof of Lemma 13 is often used in the analysis of Markov chains: if it is possible, infinitely often, to reach a state with a probability bounded away from 0, then this state is almost surely reached infinitely often. See [24] for a full proof of Lemma 13.

We can now derive a PSPACE procedure for the problem $\mathbb{P}(\text{coNBA}) = 1$ without resorting to DPAs:

▶ **Theorem 14.** *The problem* $\mathbb{P}(\text{coNBA}) = 1$ *is PSPACE-complete.*

Theorem 11 (for NBAs) has a coNBA-analogue:

▶ **Theorem 15.** *The problem* $\mathbb{P}(\text{coNBA}) = 0$ *is EXPTIME-complete. It is EXPTIME-hard even for NBAs all whose states are accepting.*

5 Co-Unambiguous Büchi Automata

In this section we build on the previous section, in particular on Lemma 13, to derive our main technical result: given a BP $\mathcal{B}$ and an *unambiguous* Büchi automaton (UBA) $\mathcal{A}$, one can decide in NC whether $\mathcal{B}$ almost surely generates a tree all whose branches are rejected by $\mathcal{A}$:

▶ **Proposition 16.** *The problem* $\mathbb{P}(\text{coUBA}) = 1$ *is in NC.*

The rest of the section is devoted to the proof of this theorem. Fix a BP $\mathcal{B}$ and a UBA $\mathcal{A}$. Since NC is closed under complement, we can focus on the problem whether the probability is positive that a $\mathcal{B}$-tree has some branch accepted by $\mathcal{A}$. We use Lemma 12. Since reachability in a graph is in NL and, hence, in NC, it suffices to decide in NC whether the probability that some branch of a $\mathcal{B}[X_f]$-tree is accepted by $\mathcal{A}[f, X_f]$ is positive. By Lemma 13 it suffices to decide in NC whether the probability that some branch of a $\mathcal{B}_{det}$-tree does not have any nodes of type $\emptyset$ is positive. The challenge is that $\mathcal{B}_{det}$ may be exponentially larger than $\mathcal{A}$, so we need to exploit the unambiguousness of $\mathcal{A}$ and the regular structure it gives to $\mathcal{B}_{det}$.

Let $\mathcal{B}''_{det}$ be the BP (with ε-rules allowed) obtained from $\mathcal{B}_{det}$ by removing the type $\emptyset$ and eliminating all occurrences of type $\emptyset$ from all right-hand sides. The probability that a $\mathcal{B}_{det}$-tree has an infinite branch of non-$\emptyset$ nodes is equal to the probability that a $\mathcal{B}''_{det}$-tree is infinite. Hence, it remains to show that one can decide in NC whether the probability that a $\mathcal{B}''_{det}$-tree is infinite is positive.

Define a matrix $M \in \mathbb{Q}^{Q[f,X_f] \times Q[f,X_f]}$ whose rows and columns are indexed with the non-$\bar{q}_0$ states of $\mathcal{A}[f, X_f]$:

$$M_{(q,X),(r,Y)} := \begin{cases} \sum_{X \xrightarrow{p} u} p|u|_Y & \text{if } (q, X) \xrightarrow{Y} (r, Y) \text{ in } \mathcal{A}[f, X_f] \\ 0 & \text{otherwise}, \end{cases}$$

where $|u|_Y \in \mathbb{N}_0$ is the number of occurrences of Y in u. (Think of $M_{(q,X),(r,Y)}$ as the expected number of (r, Y)-"successors" of (q, X).) The graph of M is equal to the transition graph of $\mathcal{A}[f, X_f]$ (excluding $\bar{q}_0$), which is strongly connected.

Say that $\mathcal{A}[f, X_f]$ has *proper branching* if there exist $(q, Y) \xrightarrow{Z_1} (r_1, Z_1)$ and $(q, Y) \xrightarrow{Z_2} (r_2, Z_2)$ in $\mathcal{A}[f, X_f]$ and a rule $Y \overset{p}{\hookrightarrow} u_1 Z_1 u_2 Z_2 u_3$ in $\mathcal{B}$ with $u_1, u_2, u_3 \in \Gamma^*$. Now we can state the key lemma:

▶ **Lemma 17.** *Let ρ be the spectral radius of M. The probability that a $\mathcal{B}''_{det}$-tree is infinite is positive if and only if either $\rho > 1$ or $\rho = 1$ and $\mathcal{A}[f, X_f]$ does not have proper branching.*

Observe the similarity between Lemmas 5 and 17. In fact, the proof of Lemma 17, given below, is based on Lemma 5. Lemma 17 shows that properties of $\mathcal{A}[f, X_f]$ and M (which are polynomial-sized objects) determine a property of the exponential-sized BP $\mathcal{B}''_{det}$. Unambiguousness of $\mathcal{A}[f, X_f]$ is crucial for that connection.

Given that Lemma 17 reflects the key insight of this section (if not of this paper), let us comment further. Suppose $\mathcal{A}[f, X_f]$ has two outgoing transitions in a state (q, Y), say $(q, Y) \xrightarrow{Z_1} (r_1, Z_1)$ and $(q, Y) \xrightarrow{Z_2} (r_2, Z_2)$. This branching could be "proper branching" as defined before Lemma 17, or the original UBA $\mathcal{A}$ could be nondeterministic when reading Y in q and have transitions $q \xrightarrow{Y} r_1$ and $q \xrightarrow{Y} r_2$. Either type of branching causes non-0 entries in the matrix M and, intuitively, increases its spectral radius ρ. Lemma 17 tells us that the probability that a $\mathcal{B}''_{det}$-tree is infinite is governed by the *combined* effect on ρ of both types of branching: if $\rho > 1$ then a $\mathcal{B}''_{det}$-tree is infinite with positive probability; only in the borderline case, $\rho = 1$, the type of branching matters. Again, this characterization is only correct if the nondeterminism in $\mathcal{A}$ does not cause ambiguousness.

Let us consider what Lemma 17 states for the special case of Markov chains. In that case, clearly there is no proper branching. One can show, using unambiguousness, that for Markov chains the spectral radius ρ of the matrix M is at most 1. Hence, Lemma 17 states for Markov chains that the probability that a $\mathcal{B}''_{det}$-tree (consisting of a single branch) is infinite is positive if and only if $\rho = 1$. Indeed, a related statement can be found in [2, Lemma 6].

To finish the proof of Proposition 16 it suffices to show that we can check the conditions of Lemma 17 in NC. Indeed, for comparing the spectral radius with 1, we employ Lemma 3. One can check for proper branching in logarithmic space, hence in NC. This completes the proof of Proposition 16.

6 LTL

With Proposition 16 from the previous section, we can now show our headline result:

▶ **Theorem 18.** *The problem $\mathbb{P}(\mathrm{LTL}) = 1$ is PSPACE-complete.*

Proof. PSPACE-hardness is immediate in two different ways. It follows both from the PSPACE-hardness of model checking Markov chains against LTL and from the PSPACE-hardness of model checking transition systems against LTL [31]. Both model-checking problems are special cases of $\mathbb{P}(\mathrm{LTL}) = 1$.

Towards membership in PSPACE, there is a classical PSPACE procedure that translates an LTL formula into an (exponential-sized) Büchi automaton [37]. As noted by several authors (e.g., [12, 7]), this procedure can easily be adapted to ensure that the Büchi automaton be a UBA. By applying this translation to the negation $\neg\varphi$ of the input formula φ, we obtain a UBA that rejects exactly those words that satisfy φ. By Proposition 16 the problem $\mathbb{P}(\mathrm{coUBA}) = 1$ is in NC. By Lemma 2 it follows that $\mathbb{P}(\mathrm{LTL}) = 1$ is in PSPACE. ◀

Finally we show the following result, exhibiting a big complexity gap between the problems $\mathbb{P}(\mathrm{LTL}) = 1$ and $\mathbb{P}(\mathrm{LTL}) = 0$.

▶ **Theorem 19.** *The problem* $\mathbb{P}(\mathrm{LTL}) = 0$ *is 2EXPTIME-complete.*

Proof. For membership in 2EXPTIME, we use again the classical procedure that translates an LTL formula into an exponential-sized Büchi automaton [37] and then invoke Theorem 11.

For 2EXPTIME-hardness we adapt the reduction from [11, Theorem 3.2.1] for MDPs. The details are in [24]. ◀

7 Conclusions

We have devised a PSPACE procedure for $\mathbb{P}(\mathrm{LTL}) = 1$, i.e., qualitative LTL model checking of BPs. The best previously known procedure ran in 2EXPTIME [8]. Since BPs naturally generalize both transition systems and Markov chains (for both of which LTL model checking is PSPACE-complete), one might view our model-checking algorithm as an optimal general procedure. The same holds for NBA-specifications instead of LTL.

The main technical ingredients have been the automata-theoretic approach and the algorithmic analysis of UBAs, nonnegative matrices, and finiteness of BPs. Our proofs were inspired by the observation that the spectral radii of certain nonnegative matrices are central to model checking Markov chains against UBAs, and also determine fundamental properties of BPs. Very loosely speaking, when model checking Markov chains against UBAs, the spectral radius measures the amount of nondeterministic branching in the UBA, whereas when analyzing BPs, the spectral radius measures the amount of tree branching. The "general case", i.e., model checking BPs, features both kinds of branching. Serendipitously, an analysis of spectral radii still leads, as we have seen, to optimal algorithms.

We have also established the complexities of related problems, partially as a tool for the mentioned LTL and NBA problems and partially to map out the landscape. We have shown that the $\mathbb{P}(\cdot) = 0$ variants are more complex than their $\mathbb{P}(\cdot) = 1$ counterparts. An intuitive explanation of this phenomenon is that for an instance of an $\mathbb{P}(\cdot) = 1$ problems to be negative, tree branching and probabilistic branching "work together" to falsify the specification on some branch. In contrast, for $\mathbb{P}(\cdot) = 0$ problems, tree branching and probabilistic branching are "adversaries", like in MDPs. Indeed, for lower bounds on $\mathbb{P}(\cdot) = 0$ problems we have encoded alternation in various forms.

One might ask about the complexity of $\mathbb{P}(\mathrm{UBA}) = 1$. Indeed, in trying to solve $\mathbb{P}(\mathrm{LTL}) = 1$ efficiently, the authors set out to solve $\mathbb{P}(\mathrm{UBA}) = 1$ efficiently (perhaps in P or even NC), with the PSPACE transduction from LTL to UBA in mind. However, the complexity of UBA universality is an open problem [30]; only membership in PSPACE is known. So even for the fixed transition system with $a \overset{1}{\hookrightarrow} ab$ and $b \overset{1}{\hookrightarrow} ab$ the problem $\mathbb{P}(\mathrm{UBA}) = 1$ cannot be placed in P without improving the complexity of UBA universality. A PSPACE-hardness proof of $\mathbb{P}(\mathrm{UBA}) = 1$ might have to make use of both types of branching in BPs, as $\mathbb{P}(\mathrm{UBA}) = 1$ is in NC for Markov chains [2].

Model checking BPs quantitatively, i.e., computing the satisfaction probability, comparing it with a threshold, or approximating it, is left for future work. Exact versions of these problems are computationally complex, as they are at least as hard as the corresponding $\mathbb{P}(\cdot) = 0$ problem. The paper [8] describes, for DPAs, nonlinear equation systems whose least nonnegative solution characterizes the satisfaction probabilities. Newton's method is efficient for approximating the solution of such equation systems; see [33, 15].

References

1 K.B. Athreya and P.E. Ney. *Branching Processes*. Springer, 1972.

2 C. Baier, S. Kiefer, J. Klein, S. Klüppelholz, D. Müller, and J. Worrell. Markov chains and unambiguous Büchi automata. In *Proceedings of the 28th International Conference on Computer Aided Verification (CAV)*, volume 9779 of *LNCS*, pages 23–42, 2016.

3 A. Berman and R.J. Plemmons. *Nonnegative matrices in the mathematical sciences*. SIAM, 1994.

4 A. Borodin. On relating time and space to size and depth. *SIAM Journal of Computing*, 6(4):733–744, 1977. doi:10.1137/0206054.

5 A. Borodin, J. von zur Gathen, and J.E. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52(3):241–256, 1982. doi:10.1016/S0019-9958(82)90766-5.

6 D. Bustan, S. Rubin, and M.Y. Vardi. Verifying omega-regular properties of Markov chains. In *Computer Aided Verification (CAV)*, volume 3114 of *Lecture Notes in Computer Science*, pages 189–201. Springer, 2004.

7 S. Chakraborty and J.-P. Katoen. Parametric LTL on Markov chains. In *8th IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science*, volume 8705 of *Lecture Notes in Computer Science*, pages 207–221. Springer, 2014.

8 T. Chen, K. Dräger, and S. Kiefer. Model checking stochastic branching processes. In *Mathematical Foundations of Computer Science (MFCS)*, volume 7464 of *Lecture Notes in Computer Science*, pages 271–282. Springer, 2012.

9 R. Cleaveland, S.P. Iyer, and M. Narasimha. Probabilistic temporal logics via the modal mu-calculus. *Theoretical Computer Science*, 342(2-3):316–350, 2005.

10 C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Symposium on Foundations of Computer Science (FOCS)*, pages 338–345. IEEE Computer Society, 1988.

11 C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

12 J.-M. Couvreur, N. Saheb, and G. Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, volume 2850 of *Lecture Notes in Computer Science*, pages 361–375. Springer, 2003.

13 J. Esparza, A. Gaiser, and S. Kiefer. A strongly polynomial algorithm for criticality of branching processes and consistency of stochastic context-free grammars. *Information Processing Letters*, 113(10-11):381–385, 2013.

14 K. Etessami, E. Martinov, A. Stewart, and M. Yannakakis. Reachability for branching concurrent stochastic games. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132 of *LIPIcs*, pages 115:1–115:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

15 K. Etessami, A. Stewart, and M. Yannakakis. A polynomial time algorithm for computing extinction probabilities of multitype branching processes. *SIAM Journal of Computing*, 46(5):1515–1553, 2017.

16 K. Etessami, A. Stewart, and M. Yannakakis. Greatest fixed points of probabilistic min/max polynomial equations, and reachability for branching Markov decision processes. *Information and Computation*, 261:355–382, 2018.

17 K. Etessami, A. Stewart, and M. Yannakakis. Polynomial time algorithms for branching Markov decision processes and probabilistic min(max) polynomial Bellman equations. *Mathematics of Operations Research*, 45(1):34–62, 2020.

18 K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1:1–1:66, 2009.

19 K. Etessami and M. Yannakakis. Model checking of recursive probabilistic systems. *ACM Transactions on Computational Logic*, 13(2):12:1–12:40, 2012. doi:10.1145/2159531.2159534.

20 T. Gogacz, H. Michalewski, M. Mio, and M. Skrzypczak. Measure properties of regular sets of trees. *Information and Computation*, 256:108–130, 2017.

21 A. Gorlin and C.R. Ramakrishnan. Separable GPL: decidable model checking with more non-determinism. In *International Conference on Concurrency Theory (CONCUR)*, volume 118 of *LIPIcs*, pages 36:1–36:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

22 P. Haccou, P. Jagers, and V.A. Vatutin. *Branching Processes: Variation, Growth, and Extinction of Populations.* Cambridge University Press, 2005.

23 T.E. Harris. *The Theory of Branching Processes.* Springer, 1963.

24 S. Kiefer, P. Semukhin, and C. Widdershoven. Linear-time model checking branching processes. Technical report, arxiv.org, 2021. Available at arXiv:2107.01687.

25 O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Principles of Programming Languages (POPL)*, pages 97–107. ACM Press, 1985.

26 H. Michalewski and M. Mio. On the problem of computing the probability of regular sets of trees. In *Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 45 of *LIPIcs*, pages 489–502. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

27 D. Niwiński, M. Przybyłko, and M. Skrzypczak. Computing measures of weak-MSO definable sets of trees. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 168 of *LIPIcs*, pages 136:1–136:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

28 C.M. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

29 N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods Computer Science*, 3(3), 2007. doi:10.2168/LMCS-3(3:5)2007.

30 A. Rabinovich. Complementation of finitely ambiguous Büchi automata. In *Developments in Language Theory (DLT)*, volume 11088 of *Lecture Notes in Computer Science*, pages 541–552. Springer, 2018.

31 A.P Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985. doi:10.1145/3828.3837.

32 A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with appplications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987. doi:10.1016/0304-3975(87)90008-9.

33 A. Stewart, K. Etessami, and M. Yannakakis. Upper bounds for Newton's method on monotone polynomial systems, and P-time model checking of probabilistic one-counter automata. *Journal of the ACM*, 62(4):30:1–30:33, 2015.

34 W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science (Vol. B): Formal Models and Semantics*, chapter 4, pages 133–191. MIT Press, 1990.

35 M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 327–338. IEEE Computer Society, 1985. doi:10.1109/SFCS.1985.12.

36 M.Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *Formal Methods for Real-Time and Probabilistic Systems*, pages 265–276. Springer, 1999.

37 M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *1st Symposium on Logic in Computer Science (LICS)*, pages 332–344. IEEE Computer Society Press, 1986.