Analysis of Probabilistic Basic Parallel Processes

Rémi Bonnet[1], Stefan Kiefer[1⋆], and Anthony W. Lin[1,2]

[1] University of Oxford, UK
[2] Academia Sinica, Taiwan

Abstract. Basic Parallel Processes (BPPs) are a well-known subclass of Petri Nets. They are the simplest common model of concurrent programs that allows unbounded spawning of processes. In the probabilistic version of BPPs, every process generates other processes according to a probability distribution. We study the decidability and complexity of fundamental qualitative problems over probabilistic BPPs — in particular reachability with probability 1 of different classes of target sets (e.g. upward-closed sets). Our results concern both the Markov-chain model, where processes are scheduled randomly, and the MDP model, where processes are picked by a scheduler.

1 Introduction

We study probabilistic basic parallel processes (pBPP), which is a stochastic model for concurrent systems with unbounded process spawning. Processes can be of different types, and each type has a fixed probability distribution for generating new sub-processes. A pBPP can be described using a notation similar to that of stochastic context-free grammars. For instance,

$$X \xhookrightarrow{0.2} XX \qquad X \xhookrightarrow{0.3} XY \qquad X \xhookrightarrow{0.5} \varepsilon \qquad\qquad Y \xhookrightarrow{0.7} X \qquad Y \xhookrightarrow{0.3} Y$$

describes a system with two types of processes. Processes of type X can generate two processes of type X, one process of each type, or zero processes with probabilities 0.2, 0.3, and 0.5, respectively. Processes of type Y can generate one process, of type X or Y, with probability 0.7 and 0.3. The order of processes on the right-hand side of each rule is not important. Readers familiar with process algebra will identify this notation as a probabilistic version of Basic Parallel Processes (BPPs), which is widely studied in automated verification, see e.g. [7, 11, 6, 13, 12, 9],

A *configuration* of a pBPP indicates, for each type X, how many processes of type X are present. Writing Γ for the finite set of types, a configuration is thus an element of $\mathbb{N}^\Gamma$. In a configuration $\alpha \in \mathbb{N}^\Gamma$ with $\alpha(X) \geq 1$ an X-process may be scheduled. Whenever a process of type X is scheduled, a rule with X on the left-hand side is picked randomly according to the probabilities of the rules, and then an X-process is replaced by processes as on the right-hand side.

⋆ Stefan Kiefer is supported by a Royal Society University Research Fellowship.

In the example above, if an X-process is scheduled, then with probability 0.3 it is replaced by a new X-process and by a new Y-process. This leads to a new configuration, α', with $\alpha'(X) = \alpha(X)$ and $\alpha'(Y) = \alpha(Y) + 1$.

Which type is scheduled in a configuration $\alpha \in \mathbb{N}^\Gamma$ depends on the model under consideration. One possibility is that the type to be scheduled is selected randomly among those types X with $\alpha(X) \geq 1$. In this way, a pBPP induces an (infinite-state) Markov chain. We consider two versions of this Markov chain: in one version the type to be scheduled is picked using a uniform distribution on those types with at least one waiting process; in the other version the type is picked using a uniform distribution on the waiting processes. For instance, in configuration α with $\alpha(X) = 1$ and $\alpha(Y) = 2$, according to the "type" version, the probability of scheduling X is $1/2$, whereas in the "process" version, the probability is $1/3$. Both models seem to make equal sense, so we consider them both in this paper. As it turns out their difference is unimportant for our results.

In many contexts (e.g. probabilistic distributed protocols — see [15, 14]), it is more natural that this scheduling decision is not taken randomly, but by a *scheduler*. Then the pBPP induces a Markov decision process (MDP), where a scheduler picks a type X to be scheduled, but the rule with X on the left-hand side is selected probabilistically according to the probabilities on the rules.

In this paper we provide decidability results concerning *coverability* with probability 1, or "almost-sure" coverability, which is a fundamental qualitative property of pBPPs. We say a configuration $\beta \in \mathbb{N}^\Gamma$ *covers* a configuration $\phi \in \mathbb{N}^\Gamma$ if $\beta \geq \phi$ holds, where $\geq$ is meant componentwise. For instance, ϕ may model a configuration with one producer and one consumer; then $\beta \geq \phi$ means that a transaction between a producer and a consumer can take place. Another example is a critical section that can be entered only when a lock is obtained. Given a pBPP, an initial configuration α, and target configurations $\phi_1, \ldots, \phi_k$, the *coverability problem* asks whether with probability 1 it is the case that starting from α a configuration β is reached that covers some ϕ_i. One can equivalently view the problem as almost-sure reachability of an upward-closed set.

In Section 3 we show using a Karp-Miller-style construction that the coverability problem for pBPP Markov chains is decidable. We provide a nonelementary lower complexity bound. In Section 4 we consider the coverability problem for MDPs. There the problem appears in two flavours, depending on whether the scheduler is "angelic" or "demonic". In the angelic case we ask whether there *exists* a scheduler so that a target is almost-surely covered. We show that this problem is decidable, and if such a scheduler does exist one can synthesize one. In the demonic case we ask whether a target is almost-surely covered, no matter what the scheduler (an operating system, for instance) does. For the question to make sense we need to exclude unfair schedulers, i.e., those that never schedule a waiting process. Using a robust fairness notion (k-fairness), which does not depend on the exact probabilities in the rules, we show that the demonic problem is also decidable. In Section 5 we show for the Markov chain and for both versions of the MDP problem that the coverability problem becomes P-time solvable, if the target configurations ϕ_i consist of only one process each (i.e., are

unit vectors). Such target configurations naturally arise in concurrent systems (e.g. freedom from deadlock: whether *at least one* process eventually goes into a critical section). Finally, in Section 6 we show that the almost-sure reachability problem for semilinear sets, which generalizes the coverability problem, is undecidable for pBPP Markov chains and MDPs. Some missing proofs can be found in [2].

Related work. (Probabilistic) BPPs can be viewed as (stochastic) Petri nets where each transition has exactly one input place. Stochastic Petri nets, in turn, are equivalent to probabilistic vector addition systems with states (pVASSs), whose reachability and coverability problems were studied in [1]. This work is close to ours; in fact, we build on fundamental results of [1]. Whereas we show that coverability for the Markov chain induced by a pBPP is decidable, it is shown in [1] that the problem is undecidable for general pVASSs. In [1] it is further shown for general pVASSs that coverability becomes decidable if the target sets are "Q-states". If we apply the same restriction on the target sets, coverability becomes polynomial-time decidable for pBPPs, see Section 5. MDP problems are not discussed in [1].

The MDP version of pBPPs was studied before under the name *task systems* [3]. There, the scheduler aims at a "space-efficient" scheduling, which is one where the maximal number of processes is minimised. Goals and techniques of this paper are very different from ours.

Certain classes of non-probabilistic 2-player games on Petri nets were studied in [16]. Our MDP problems can be viewed as games between two players, Scheduler and Probability. One of our proofs (the proof of Theorem 11) is inspired by proofs in [16].

The notion of k-fairness that we consider in this paper is not new. Similar notions have appeared in the literature of concurrent systems under the name of "bounded fairness" (e.g. see [5] and its citations).

2 Preliminaries

We write $\mathbb{N} = \{0, 1, 2, \ldots\}$. For a countable set X we write $dist(X)$ for the set of *probability distributions* over X; i.e., $dist(X)$ consists of those functions $f : X \to [0, 1]$ such that $\sum_{x \in X} f(x) = 1$.

Markov Chains. A *Markov chain* is a pair $\mathcal{M} = (Q, \delta)$, where Q is a countable (finite or infinite) set of states, and $\delta : Q \to dist(Q)$ is a probabilistic transition function that maps a state to a probability distribution over the successor states. Given a Markov chain we also write $s \xrightarrow{p} t$ or $s \to t$ to indicate that $p = \delta(s)(t) > 0$. A *run* is an infinite sequence $s_0 s_1 \cdots \in Q^\omega$ with $s_i \to s_{i+1}$ for $i \in \mathbb{N}$. We write $Run(s_0 \cdots s_k)$ for the set of runs that start with $s_0 \cdots s_k$. To every initial state $s_0 \in S$ we associate the probability space $(Run(s_0), \mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is the σ-field generated by all basic cylinders $Run(s_0 \cdots s_k)$ with $s_0 \cdots s_k \in Q^*$, and $\mathcal{P} : \mathcal{F} \to [0, 1]$ is the unique probability measure such that $\mathcal{P}(Run(s_0 \cdots s_k)) =$

$\prod_{i=1}^{k} \delta(s_{i-1})(s_i)$. For a state $s_0 \in Q$ and a set $F \subseteq Q$, we write $s_0 \models \Diamond F$ for the event that a run started in s_0 hits F. Formally, $s_0 \models \Diamond F$ can be seen as the set of runs $s_0 s_1 \cdots$ such that there is $i \geq 0$ with $s_i \in F$. Clearly we have $\mathcal{P}(s_0 \models \Diamond F) > 0$ if and only if in $\mathcal{M}$ there is a path from s_0 to a state in F. Similarly, for $Q_1, Q_2 \subseteq Q$ we write $s_0 \models Q_1 \mathsf{U} Q_2$ to denote the set of runs $s_0 s_1 \cdots$ such that there is $j \geq 0$ with $s_j \in Q_2$ and $s_i \in Q_1$ for all $i < j$. We have $\mathcal{P}(s_0 \models Q_1 \mathsf{U} Q_2) > 0$ if and only if in $\mathcal{M}$ there is a path from s_0 to a state in Q_2 using only states in Q_1. A Markov chain is *globally coarse* with respect to a set $F \subseteq Q$ of configurations, if there exists $c > 0$ such that for all $s_0 \in Q$ we have that $\mathcal{P}(s_0 \models \Diamond F) > 0$ implies $\mathcal{P}(s_0 \models \Diamond F) \geq c$.

Markov Decision Processes. A *Markov decision process (MDP)* is a tuple $\mathcal{D} = (Q, A, En, \delta)$, where Q is a countable set of states, A is a finite set of actions, $En : Q \to 2^A \setminus \emptyset$ is an action enabledness function that assigns to each state s the set $En(s)$ of actions enabled in s, and $\delta : S \times A \to dist(S)$ is a probabilistic transition function that maps a state s and an action $a \in En(s)$ enabled in s to a probability distribution over the successor states. A *run* is an infinite alternating sequence of states and actions $s_0 a_1 s_1 a_2 \cdots$ such that for all $i \geq 1$ we have $a_i \in En(s_{i-1})$ and $\delta(s_{i-1}, a_i)(s_i) > 0$. For a finite word $w = s_0 a_1 \cdots s_{k-1} a_k s_k \in Q(AQ)^*$ we write $last(w) = s_k$. A *scheduler* for $\mathcal{D}$ is a function $\sigma : Q(AQ)^* \to dist(A)$ that maps a run prefix $w \in Q(AQ)^*$, representing the history of a play, to a probability distribution over the actions enabled in $last(w)$. We write $Run(w)$ for the set of runs that start with $w \in Q(AQ)^*$. To an initial state $s_0 \in S$ and a scheduler σ we associate the probability space $(Run(s_0), \mathcal{F}, \mathcal{P}_\sigma)$, where $\mathcal{F}$ is the σ-field generated by all basic cylinders $Run(w)$ with $w \in \{s_0\}(AQ)^*$, and $\mathcal{P}_\sigma : \mathcal{F} \to [0, 1]$ is the unique probability measure such that $\mathcal{P}(Run(s_0)) = 1$, and $\mathcal{P}(Run(was)) = \mathcal{P}(Run(w)) \cdot \sigma(w)(a) \cdot \delta(last(w), a)(s)$ for all $w \in \{s_0\}(AQ)^*$ and all $a \in A$ and all $s \in Q$. A scheduler σ is called *deterministic* if for all $w \in Q(AQ)^*$ there is $a \in A$ with $\sigma(w)(a) = 1$. A scheduler σ is called *memoryless* if for all $w, w' \in Q(AQ)^*$ with $last(w) = last(w')$ we have $\sigma(w) = \sigma(w')$. When specifying events, i.e., measurable subsets of $Run(s_0)$, the actions are often irrelevant. Therefore, when we speak of runs $s_0 s_1 \cdots$ we mean the runs $s_0 a_1 s_1 a_2 \cdots$ for arbitrary $a_1, a_2, \ldots \in A$. E.g., in this understanding we view $s_0 \models \Diamond F$ with $s_0 \in Q$ and $F \subseteq Q$ as an event.

Probabilistic BPPs and their configurations. A *probabilistic BPP (pBPP)* is a tuple $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$, where Γ is a finite set of *types*, $\hookrightarrow \subseteq \Gamma \times \mathbb{N}^\Gamma$ is a finite set of *rules* such that for every $X \in \Gamma$ there is at least one rule of the form $X \hookrightarrow \alpha$, and $Prob$ is a function that to every rule $X \hookrightarrow \alpha$ assigns its probability $Prob(X \hookrightarrow \alpha) \in (0, 1] \cap \mathbb{Q}$ so that for all $X \in \Gamma$ we have $\sum_{X \hookrightarrow \alpha} Prob(X \hookrightarrow \alpha) = 1$. We write $X \overset{p}{\hookrightarrow} \alpha$ to denote that $Prob(X \hookrightarrow \alpha) = p$. A *configuration* of $\mathcal{S}$ is an element of $\mathbb{N}^\Gamma$. We write $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ for componentwise addition and subtraction of two configurations α_1, α_2. When there is no confusion, we may identify words $u \in \Gamma^*$ with the configuration $\alpha \in \mathbb{N}^\Gamma$ such that for all $X \in \Gamma$ we have that $\alpha(X) \in \mathbb{N}$ is the number of occurrences of X in u. For instance, we write XXY or XYX for the configuration α with $\alpha(X) = 2$ and $\alpha(Y) = 1$ and

$\alpha(Z) = 0$ for $Z \in \Gamma \setminus \{X, Y\}$. In particular, we may write ε for α with $\alpha(X) = 0$ for all $X \in \Gamma$. For configurations α, β we write $\alpha \leq \beta$ if $\alpha(X) \leq \beta(X)$ holds for all $X \in \Gamma$; we write $\alpha < \beta$ if $\alpha \leq \beta$ but $\alpha \neq \beta$. For a configuration α we define the *number of types* $|\alpha|_{type} = |\{X \in \Gamma \mid \alpha(X) \geq 1\}|$ and the *number of processes* $|\alpha|_{proc} = \sum_{X \in \Gamma} \alpha(X)$. Observe that we have $|\alpha|_{type} \leq |\alpha|_{proc}$. A set $F \subseteq \mathbb{N}^\Gamma$ of configurations is called *upward-closed* (*downward-closed*, respectively) if for all $\alpha \in F$ we have that $\alpha \leq \beta$ implies $\beta \in F$ ($\alpha \geq \beta$ implies $\beta \in F$, respectively). For $\alpha \in \mathbb{N}^\Gamma$ we define $\alpha{\uparrow} := \{\beta \in \mathbb{N}^\Gamma \mid \beta \geq \alpha\}$. For $F \subseteq \mathbb{N}^\Gamma$ and $\alpha \in F$ we say that α is a *minimal element* of F, if there is no $\beta \in F$ with $\beta < \alpha$. It follows from Dickson's lemma that every upward-closed set has finitely many minimal elements; i.e., F is upward-closed if and only if $F = \phi_1{\uparrow} \cup \ldots \cup \phi_n{\uparrow}$ holds for some $n \in \mathbb{N}$ and $\phi_1, \ldots, \phi_n \in \mathbb{N}^\Gamma$.

Markov Chains induced by a pBPP. To a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ we associate the Markov chains $\mathcal{M}_{type}(\mathcal{S}) = (\mathbb{N}^\Gamma, \delta_{type})$ and $\mathcal{M}_{proc}(\mathcal{S}) = (\mathbb{N}^\Gamma, \delta_{proc})$ with $\delta_{type}(\varepsilon, \varepsilon) = \delta_{proc}(\varepsilon, \varepsilon) = 1$ and for $\alpha \neq \varepsilon$

$$\delta_{type}(\alpha, \gamma) = \sum_{\substack{X \overset{p}{\hookrightarrow} \beta \text{ s.t. } \alpha(X) \geq 1 \\ \text{and } \gamma = \alpha - X + \beta}} \frac{p}{|\alpha|_{type}} \quad \text{and} \quad \delta_{proc}(\alpha, \gamma) = \sum_{\substack{X \overset{p}{\hookrightarrow} \beta \text{ s.t.} \\ \gamma = \alpha - X + \beta}} \frac{\alpha(X) \cdot p}{|\alpha|_{proc}}.$$

In words, the new configuration γ is obtained from α by replacing an X-process with a configuration randomly sampled according to the rules $X \overset{p}{\hookrightarrow} \beta$. In $\mathcal{M}_{type}(\mathcal{S})$ the selection of X is based on the number of types in α, whereas in $\mathcal{M}_{proc}(\mathcal{S})$ it is based on the number of processes in α. We have $\delta_{type}(\alpha, \gamma) = 0$ iff $\delta_{proc}(\alpha, \gamma) = 0$. We write $\mathcal{P}_{type}$ and $\mathcal{P}_{proc}$ for the probability measures in $\mathcal{M}_{type}(\mathcal{S})$ and $\mathcal{M}_{proc}(\mathcal{S})$, respectively.

The MDP induced by a pBPP. To a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ we associate the MDP $\mathcal{D}(\mathcal{S}) = (\mathbb{N}^\Gamma, \Gamma \cup \{\bot\}, En, \delta)$ with a fresh action $\bot \notin \Gamma$, and $En(\alpha) = \{X \in \Gamma \mid \alpha(X) \geq 1\}$ for $\varepsilon \neq \alpha \in \mathbb{N}^\Gamma$ and $En(\varepsilon) = \{\bot\}$, and $\delta(\alpha, X)(\alpha - X + \beta) = p$ whenever $\alpha(X) \geq 1$ and $X \overset{p}{\hookrightarrow} \beta$, and $\delta(\varepsilon, \bot)(\varepsilon) = 1$. As in the Markov chain, the new configuration γ is obtained from α by replacing an X-process with a configuration randomly sampled according to the rules $X \overset{p}{\hookrightarrow} \beta$. But in contrast to the Markov chain the selection of X is up to a scheduler.

3 The Coverability Problem for the Markov Chain

In this section we study the *coverability problem* for the Markov chains induced by a pBPP. We say a run $\alpha_0 \alpha_1 \cdots$ of a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ *covers* a configuration $\phi \in \mathbb{N}^\Gamma$, if $\alpha_i \geq \phi$ holds for some $i \in \mathbb{N}$. The coverability problem asks whether it is almost surely the case that some configuration from a finite set $\{\phi_1, \ldots, \phi_n\}$ will be covered. More formally, the coverability problem is the following. Given a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$, an initial configuration $\alpha_0 \in \mathbb{N}^\Gamma$, and finitely many configurations $\phi_1, \ldots, \phi_n$, does $\mathcal{P}_{type}(\alpha_0 \models \Diamond F) = 1$ hold,

where $F = \phi_1\uparrow \cup \ldots \cup \phi_n\uparrow$? Similarly, does $\mathcal{P}_{proc}(\alpha_0 \models \Diamond F) = 1$ hold? We will show that those two questions always have the same answer.

In Section 3.1 we show that the coverability problem is decidable. In Section 3.2 we show that the complexity of the coverability problem is nonelementary.

3.1 Decidability

For deciding the coverability problem we use the approach of [1]. The following proposition is crucial for us:

Proposition 1. *Let $\mathcal{M} = (Q, \delta)$ be a Markov chain and $F \subseteq Q$ such that $\mathcal{M}$ is globally coarse with respect to F. Let $\bar{F} = Q \setminus F$ be the complement of F and let $\widetilde{F} := \{s \in Q \mid \mathcal{P}(s \models \Diamond F) = 0\} \subseteq \bar{F}$ denote the set of states from which F is not reachable in $\mathcal{M}$. Let $s_0 \in Q$. Then we have $\mathcal{P}(s_0 \models \Diamond F) = 1$ if and only if $\mathcal{P}(s_0 \models \bar{F} \mathsf{U} \widetilde{F}) = 0$.*

Proof. Immediate from [1, Lemmas 3.7, 5.1 and 5.2]. □

In other words, Proposition 1 states that F is almost surely reached if and only if there is no path to $\widetilde{F}$ that avoids F. Proposition 1 will allow us to decide the coverability problem by computing only reachability relations in $\mathcal{M}$, ignoring the probabilities.

Recall that for a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$, the Markov chains $\mathcal{M}_{type}(\mathcal{S})$ and $\mathcal{M}_{proc}(\mathcal{S})$ have the same structure; only the transition probabilities differ. In particular, if $F \subseteq \mathbb{N}^\Gamma$ is upward-closed, the set $\widetilde{F}$, as defined in Proposition 1, is the same for $\mathcal{M}_{type}(\mathcal{S})$ and $\mathcal{M}_{proc}(\mathcal{S})$. Moreover, we have the following proposition (full proof in [2]).

Proposition 2. *Let $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ be a pBPP. Let $F \subseteq \mathbb{N}^\Gamma$ be upward-closed. Then the Markov chains $\mathcal{M}_{type}(\mathcal{S})$ and $\mathcal{M}_{proc}(\mathcal{S})$ are globally coarse with respect to F.*

Proof (sketch). The statement about $\mathcal{M}_{type}(\mathcal{S})$ follows from [1, Theorem 4.3]. For the statement about $\mathcal{M}_{proc}(\mathcal{S})$ it is crucial to argue that starting with any configuration $\alpha \in \mathbb{N}^\Gamma$ it is the case with probability 1 that every type X with $\alpha(X) \geq 1$ is eventually scheduled. Since F is upward-closed it follows that for all $\alpha, \beta \in \mathbb{N}^\Gamma$ with $\alpha \leq \beta$ we have $\mathcal{P}_{proc}(\alpha \models \Diamond F) \leq \mathcal{P}_{proc}(\beta \models \Diamond F)$. Then the statement follows from Dickson's lemma.

For an illustration of the challenge, consider the pBPP with $X \overset{1}{\hookrightarrow} XX$ and $Y \overset{1}{\hookrightarrow} YY$, and let $F = XX\uparrow$. Clearly we have $\mathcal{P}_{proc}(X \models \Diamond F) = 1$, as the X-process is scheduled immediately. Now let $\alpha_0 = XY$. Since $\alpha_0 \geq X$, the inequality claimed above implies $\mathcal{P}_{proc}(\alpha_0 \models F) = 1$. Indeed, the probability that the X-process in α_0 is *never* scheduled is at most $\frac{1}{2} \cdot \frac{2}{3} \cdot \frac{3}{4} \cdot \ldots$, which is 0. Hence $\mathcal{P}_{proc}(\alpha_0 \models \Diamond F) = 1$. □

The following proposition follows by combining Propositions 1 and 2.

Proposition 3. *Let $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ be a pBPP. Let $F \subseteq \mathbb{N}^\Gamma$ be upward-closed. Let $\alpha_0 \in \mathbb{N}^\Gamma$. We have:*

$$\mathcal{P}_{type}(\alpha_0 \models \Diamond F) = 1 \iff \mathcal{P}_{type}(\alpha_0 \models \bar{F} \mathsf{U} \widetilde{F}) = 0$$
$$\iff \mathcal{P}_{proc}(\alpha_0 \models \bar{F} \mathsf{U} \widetilde{F}) = 0 \iff \mathcal{P}_{proc}(\alpha_0 \models \Diamond F) = 1$$

By Proposition 3 we may in the following omit the subscript from $\mathcal{P}_{type}, \mathcal{P}_{proc}, \mathcal{M}_{type}, \mathcal{M}_{proc}$ if it does not matter. We have the following theorem.

Theorem 4. *The coverability problem is decidable: given a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$, an upward-closed set $F \subseteq \mathbb{N}^\Gamma$, and a configuration $\alpha_0 \in \mathbb{N}^\Gamma$, it is decidable whether $\mathcal{P}(\alpha_0 \models \Diamond F) = 1$ holds.*

Proof. The complement of $\widetilde{F}$ (i.e., the set of configurations from which F is reachable) is upward-closed, and its minimal elements can be computed by a straightforward fixed-point computation (this is even true for the more general model of pVASS, e.g., see [1, Remark 4.2]). By Proposition 3 it suffices to decide whether $\mathcal{P}(\alpha_0 \models \bar{F} \mathsf{U} \widetilde{F}) > 0$ holds. Define $R := \{\alpha \in \bar{F} \mid \alpha \text{ is reachable from } \alpha_0 \text{ via } \bar{F}\text{-configurations}\}$. Observe that $\mathcal{P}(\alpha_0 \models \bar{F} \mathsf{U} \widetilde{F}) > 0$ if and only if $R \cap \widetilde{F} \neq \emptyset$. We can now give a Karp-Miller-style algorithm for checking that $R \cap \widetilde{F} \neq \emptyset$: (i) Starting from α_0, build a tree of configurations reachable from α_0 via $\bar{F}$-configurations (i.e., at no stage F-configurations are added to this tree) — for example, in a breadth-first search manner — but stop expanding a leaf node α_k as soon as we discover that the branch $\alpha_0 \to \cdots \to \alpha_k$ satisfies the following: $\alpha_j \leq \alpha_k$ for some $j < k$. (ii) As soon as a node $\alpha \in \widetilde{F}$ is generated, terminate and output "yes". (iii) When the tree construction is completed without finding nodes in $\widetilde{F}$, terminate and output "no".

To prove correctness of the above algorithm, we first prove termination. To this end, it suffices to show that the constructed tree is finite. To see this, observe first that every branch in the constructed tree is of finite length. This is an immediate consequence of Dickson's lemma and our policy of terminating a leaf node α that satisfies $\alpha' \leq \alpha$, for some ancestor α' of α in this tree. Now since all branches of the tree are finite, König's lemma shows that the tree itself must be finite (since each node has finite degree).

To prove partial correctness, it suffices to show that the policy of terminating a leaf node α that satisfies $\alpha' \leq \alpha$, for some ancestor α' of α in this tree, is valid. That is, we want to show that if $R \cap \widetilde{F} \neq \emptyset$ then a witnessing vector $\gamma \in R \cap \widetilde{F}$ will be found by the algorithm. We have the following lemma whose proof is in [2].

Lemma 5. *Let $\alpha_0 \in \bar{F}$ and let $\gamma \in \mathbb{N}^\Gamma$. Let $\alpha_0 \to \alpha_1 \to \ldots \to \alpha_k$ be a shortest path in $\mathcal{M}(\mathcal{S})$ such that $\alpha_0, \ldots, \alpha_k \in \bar{F}$ and $\alpha_k \leq \gamma$. Then for all i, j with $0 \leq i < j \leq k$ we have $\alpha_i \not\leq \alpha_j$.*

Let $R \cap \widetilde{F} \neq \emptyset$ and let $\gamma \in \mathbb{N}^\Gamma$ be a minimal element of $R \cap \widetilde{F}$. By Lemma 5 our algorithm does not prune any shortest path from α_0 to γ. Hence it outputs "yes". □

3.2 Nonelementary Lower Bound

We have the following lower-bound result:

Theorem 6. *The complexity of the coverability problem is nonelementary.*

The proof is technically involved.

Proof (sketch). We claim that there exists a nonelementary function f such that given a 2-counter machine M running in space $f(k)$, we can compute a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ of size $\leq k$, an upward-closed set $F \subseteq \mathbb{N}^\Gamma$ (with at most k minimal elements, described by numbers at most k), and a type $X_0 \in \Gamma$, such that $\mathcal{P}(X_0 \models \Diamond F) = 1$ holds if and only if M does not terminate. Recall that by Proposition 3 we have that $\mathcal{P}(X_0 \models \Diamond F) = 1$ is equivalent to $\mathcal{P}(X_0 \models \bar{F} \mathsf{U} \widetilde{F}) = 0$.

Since the exact values of the probabilities do not matter, it suffices to construct a (nonprobabilistic) BPP $\mathcal{S}$. Further, by adding processes that can spawn everything (and hence cannot take part in $\widetilde{F}$-configurations) one can change the condition of reaching $\widetilde{F}$ to reaching a downward closed set $G \subseteq \bar{F}$. So the problem we are reducing to is: does there exist a path in $\mathcal{S}$ that is contained in $\bar{F}$ and goes from X_0 to a downward closed set G.

By defining F suitably we can add various restrictions on the behaviour of our BPP. For example, the following example allows X to be turned into Y if and only if there is no Z present:

$$X \hookrightarrow YW \qquad W \hookrightarrow \varepsilon \qquad F = WZ\uparrow$$

Doubling the number of a given process is straightforward, and it is also possible to divide the number of a given process by two. Looking only at runs inside $\bar{F}$, the following BPP can turn all its X-processes into half as many X'-processes. (Note that more X'-processes could be spawned, but because of the monotonicity of the system, the "best" runs are those that spawn a minimal number of processes.)

$$X \hookrightarrow TP \qquad T \hookrightarrow \overline{P} \qquad P \hookrightarrow \varepsilon \qquad \overline{P} \hookrightarrow \varepsilon$$

$$P_1 \hookrightarrow \overline{P_2} \qquad \overline{P_2} \hookrightarrow P_2 \qquad P_2 \hookrightarrow \overline{P_1} \qquad \overline{P_1} \hookrightarrow P_1 X'$$

$$F = P\overline{P_1}\uparrow \cup P\overline{P_2}\uparrow \cup \overline{P}P_1\uparrow \cup \overline{P}P_2\uparrow \cup T^2\uparrow$$

$$\alpha_{init} = X^n\overline{P_1}$$

Let us explain this construction. In order to make an X-process disappear, we need to create temporary processes P and $\overline{P}$. However, these processes are incompatible, respectively, with $\overline{P_i}$ and P_i. Thus, destroying an X-process requires the process P_1 to move into $\overline{P_1}$ and then into P_2. By repeatedly destroying X-processes, this forces the creation of half as many X'-processes.

It is essential for our construction to have a loop-gadget that performs a cycle of processes $A \hookrightarrow B \hookrightarrow C \hookrightarrow A$ exactly k times ("k-loop"). By activating/disabling transitions based on the absence/presence of an A-, B- or

C-process, we can force an operation to be performed k times. For example, assuming the construction of a k-loop gadget, the following BPP doubles the number of X-processes k times:

$$X \hookrightarrow Y \qquad Y \hookrightarrow ZZ \qquad Z \hookrightarrow X$$

(rules for k-loop on $A/B/C$)

$$F = XB\uparrow \cup YC\uparrow \cup ZA\uparrow$$

For the loop to perform $A \hookrightarrow B$, all X-processes have to be turned into Y. Similarly, performing $B \hookrightarrow C \hookrightarrow A$ requires the Y-processes to be turned into Z, then into X. Thus, in order to perform one iteration of the loop, one needs to double the number of X-processes.

To implement such a loop we need two more gadgets: one for creating k processes, and one for consuming k processes. By turning a created process into a consumed process on at a time, we obtain the required cycle. Here is an example:

$$I \hookrightarrow A \qquad A \hookrightarrow B \qquad B \hookrightarrow C \qquad C \hookrightarrow \varepsilon$$

$$\overline{A} \hookrightarrow \overline{B} \qquad \overline{B} \hookrightarrow \overline{C}F \qquad \overline{C} \hookrightarrow A$$

(rules for a gadget to consume k processes F)

(rules for a gadget to spawn k processes I)

$$F = A\overline{A}\uparrow \cup B\overline{B}\uparrow \cup C\overline{C}\uparrow \cup AA\uparrow \cup BB\uparrow \cup CC\uparrow$$

$$\alpha_{init} = \overline{A}$$

By combining a k-loop with a multiplier or a divider, we can spawn or consume 2^k processes. This allows us to create a 2^k-loop. By iterating this construction, we get a BPP of exponential size (each loop requires two lower-level loops) that is able to spawn or consume $2^{2^{\cdots^k}}$ processes.

It remains to simulate our 2-counter machine M. The main idea is to spawn an initial budget b of processes, and to make sure that this number stays the same along the run. Zero-tests are easy to implement; the difficulty lies in the increments and decrements. The solution is to maintain, for each simulated counter, two pools of processes X and $\overline{X}$, such that if the counter is supposed to have value k, then we have processes $X^k\overline{X}^{b-k}$. Now, incrementing consist in turning all these processes into backup processes, except one $\overline{X}$-process. Then, we turn this process into an X-process, and return all backup process to their initial type.

In [2] we provide complete details of the proofs, including graphical representations of the processes involved. □

4 The Coverability Problem for the MDP

In the following we investigate the controlled version of the pBPP model. Recall from Section 2 that a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ induces an MDP $\mathcal{D}(\mathcal{S})$ where in a configuration $\varepsilon \neq \alpha \in \mathbb{N}^\Gamma$ a scheduler σ picks a type X with $\alpha(X) \geq 1$. The successor configuration is then obtained randomly from α according to the rules in $\mathcal{S}$ with X on the left-hand side.

We investigate (variants of) the decision problem that asks, given $\alpha_0 \in \mathbb{N}^\Gamma$ and an upward-closed set $F \subseteq \mathbb{N}^\Gamma$, whether $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$ holds for some scheduler (or for all schedulers, respectively).

4.1 The Existential Problem

In this section we consider the scenario where we ask for a scheduler that makes the system reach an upward-closed set with probability 1. We prove the following theorem:

Theorem 7. *Given a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ and a configuration $\alpha_0 \in \mathbb{N}^\Gamma$ and an upward-closed set $F \subseteq \mathbb{N}^\Gamma$, it is decidable whether there exists a scheduler σ with $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$. If such a scheduler exists, one can compute a deterministic and memoryless scheduler σ with $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$.*

Proof (sketch). The proof (in [2]) is relatively long. The idea is to abstract the MDP $\mathcal{D}(\mathcal{S})$ (with $\mathbb{N}^\Gamma$ as state space) to an "equivalent" finite-state MDP. The state space of the finite-state MDP is $Q := \{0, 1, \ldots, K\}^\Gamma \subseteq \mathbb{N}^\Gamma$, where K is the largest number that appears in the minimal elements of F. For finite-state MDPs, reachability with probability 1 can be decided in polynomial time, and an optimal deterministic and memoryless scheduler can be synthesized.

When setting up the finite-state MDP, special care needs to be taken of transitions that would lead from Q to a configuration α outside of Q, i.e., $\alpha \in \mathbb{N}^\Gamma \setminus Q$. Those transitions are redirected to a probability distribution on T_α with $T_\alpha \subseteq Q$, so that each configuration in T_α is "equivalent" to some configuration $\beta \in \mathbb{N}^\Gamma$ that could be reached from α in the infinite-state MDP $\mathcal{D}(\mathcal{S})$, if the scheduler follows a particular optimal strategy in $\mathcal{D}(\mathcal{S})$. (One needs to show that indeed with probability 1 such a β is reached in the infinite-state MDP, if the scheduler acts according to this strategy.) This optimal strategy is based on the observation that whenever in configuration $\beta \in \mathbb{N}^\Gamma$ with $\beta(X) > K$ for some X, then type X can be scheduled. This is without risk, because after scheduling X, at least K processes of type X remain, which is enough by the definition of K. The benefit of scheduling such X is that processes appearing on the right-hand side of X-rules may be generated, possibly helping to reach F. For computing T_α, we rely on decision procedures for the reachability problem in Petri nets, which prohibits us from giving an upper complexity bound. □

4.2 The Universal Problem

In this section we consider the scheduler as adversarial in the sense that it tries to avoid the upward-closed set F. We say "the scheduler wins" if it avoids F forever.

We ask if the scheduler can win with positive probability: given α_0 and F, do we have $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$ for all schedulers σ? For the question to make sense, we need to rephrase it, as we show now. Consider the pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, \mathit{Prob})$ with $\Gamma = \{X, Y\}$ and the rules $X \overset{1}{\hookrightarrow} XX$ and $Y \overset{1}{\hookrightarrow} YY$. Let $F = XX\uparrow$. If $\alpha_0 = X$, then, clearly, we have $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$ for all schedulers σ. However, if $\alpha_0 = XY$, then there is a scheduler σ with $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 0$: take the scheduler σ that always schedules Y and never X. Such a scheduler is intuitively *unfair*. If an operating system acts as a scheduler, a minimum requirement would be that waiting processes are scheduled eventually.

We call a run $\alpha_0 X_1 \alpha_1 X_2 \ldots$ in the MDP $\mathcal{D}(\mathcal{S})$ *fair* if for all $i \geq 0$ and all $X \in \Gamma$ with $\alpha_i(X) \geq 1$ we have $X = X_j$ for some $j > i$. We call a scheduler σ *classically fair* if it produces only fair runs.

Example 8. Consider the pBPP with $X \overset{1}{\hookrightarrow} Y$ and $Y \overset{0.5}{\hookrightarrow} Y$ and $Y \overset{0.5}{\hookrightarrow} X$. Let $F = YY\uparrow$. Let $\alpha_0 = XX$. In configuration $\alpha = XY$ the scheduler has to choose between two options: It can pick X, resulting in the successor configuration $YY \in F$, which is a "loss" for the scheduler. Alternatively, it picks Y, which results in α or α_0, each with probability 0.5. If it results in α, nothing has changed; if it results in α_0, we say a "a round is completed". Consider the scheduler σ that acts as follows. When in configuration $\alpha = XY$ and in the ith round, it picks Y until either the next round (the $(i+1)$st round) is completed or Y has been picked i times in this round. In the latter case it picks X and thus loses. Clearly, σ is classically fair (provided that it behaves in a classically fair way after it loses, for instances using round-robin). The probability of losing in the ith round is 2^{-i}. Hence the probability of losing is $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1 - \prod_{i=1}^{\infty}(1 - 2^{-i}) < 1$. (For this inequality, recall that for a sequence $(a_i)_{i\in\mathbb{N}}$ with $a_i \in (0,1)$ we have $\prod_{i\in\mathbb{N}}(1-a_i) = 0$ if and only if the series $\sum_{i\in\mathbb{N}} a_i$ diverges.) One can argue along these lines that any classically fair scheduler needs to play longer and longer rounds in order to win with positive probability. In particular, such schedulers need infinite memory.

It is hardly conceivable that an operating system would "consider" such schedulers. Note that the pBPP from the previous example has a finite state space.

In the probabilistic context, a commonly used alternative notion is *probabilistic fairness*, see e.g. [10, 17] or [4] for an overview (the term *probabilistic fairness* is used differently in [4]). We call a scheduler σ *probabilistically fair* if with probability 1 it produces a fair run.

Example 9. For the pBPP from the previous example, consider the scheduler σ that picks Y until the round is completed. Then $\mathcal{P}_\sigma(\alpha \models \Diamond F) = 0$ and σ is probabilistically fair.

The following example shows that probabilistic fairness for pBPPs can be unstable with respect to perturbations in the probabilities.

Example 10. Consider a pBPP with

$$X \overset{1}{\hookrightarrow} Y \qquad Y \overset{1}{\hookrightarrow} XZ \qquad Z \overset{p}{\hookrightarrow} ZZ \qquad Z \overset{1-p}{\hookrightarrow} \varepsilon \qquad \text{for some } p \in (0,1)$$

and $F = YZ\uparrow$ and $\alpha_0 = XZ$.

Let $p \leq 0.5$. Then, by an argument on the "gambler's ruin problem" (see e.g. [8, Chapter XIV]), with probability 1 each Z-process produces only finitely many other Z-processes in its "subderivation tree". Consider the scheduler σ that picks Z as long as there is a Z-process. With probability 1 it creates a run of the following form:

$$(XZ)\cdots(X)(Y)(XZ)\cdots(X)(Y)(XZ)\cdots(X)(Y)(XZ)\ldots$$

Such runs are fair, so σ is probabilistically fair and wins with probability 1.

Let $p > 0.5$. Then, by the same random-walk argument, with probability 1 some Z-process (i.e., at least one of the Z-processes created by Y) produces infinitely many other Z-processes. So any probabilistically fair scheduler σ produces, with probability 1, a Y-process before all Z-processes are gone, and thus loses.

We conclude that a probabilistically fair scheduler σ with $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) < 1$ exists if and only if $p \leq 0.5$.

The example suggests that deciding whether there exists a probabilistically fair scheduler σ with $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) < 1$ requires arguments on (in general) multidimensional random walks. In addition, the example shows that probabilistic fairness is not a robust notion when the exact probabilities are not known.

We aim at solving those problems by considering a stronger notion of fair runs: Let $k \in \mathbb{N}$. We call a run $\alpha_0 X_1 \alpha_1 X_2 \ldots$ *k-fair* if for all $i \geq 0$ and all $X \in \Gamma$ with $\alpha_i(X) \geq 1$ we have that $X \in \{X_{i+1}, X_{i+2}, \ldots, X_k\}$. In words, if $\alpha_i(X) \geq 1$, the type X has to be scheduled within time k. We call a scheduler *k-fair* if it produces only k-fair runs.

Theorem 11. *Given a pBPP $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$, an upward-closed set F, a number $k \in \mathbb{N}$, and a configuration $\alpha_0 \in \mathbb{N}^\Gamma$, it is decidable whether for all k-fair schedulers σ we have $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$.*

The proof is inspired by proofs in [16], and combines new insights with the technique of Theorem 4, see [2]. We remark that the proof shows that the exact values of the positive probabilities do not matter.

5 Q-States Target Sets

In this section, we provide a sensible restriction of input target sets which yields polynomial-time solvability of our problems. Let $Q = \{X_1, \ldots, X_n\} \subseteq \Gamma$. The *Q-states set* is the upward-closed set $F = X_1\uparrow \cup \ldots \cup X_n\uparrow$. There are two reasons to consider Q-states target sets. Firstly, Q-states target sets are sufficiently expressive to capture common examples in the literature of distributed protocols, e.g., freedom from deadlock and resource starvation (standard examples include the dining philosopher problem in which case *at least one* philosopher must eat). Secondly, Q-states target sets have been considered in the literature

of Petri nets: e.g., in [1][1] the authors showed that qualitative reachability for probabilistic Vector Addition Systems with States with Q-states target sets becomes decidable whereas the same problem is undecidable with upward-closed target sets.

Theorem 12. *Let $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ be a pBPP. Let $Q \subseteq \Gamma$ represent an upward-closed set $F \subseteq \mathbb{N}^\Gamma$. Let $\alpha_0 \in \mathbb{N}^\Gamma$ and $k \geq |\Gamma|$.*

(a) The coverability problem with Q-states target sets is solvable in polynomial time; i.e., we can decide in polynomial time whether $\mathcal{P}(\alpha_0 \models \Diamond F) = 1$ holds.

(b) We have:

$$\begin{aligned} & \mathcal{P}(\alpha_0 \models \Diamond F) = 1 \\ \iff\ & \mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1 \text{ holds for some scheduler } \sigma \\ \iff\ & \mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1 \text{ holds for all } k\text{-fair schedulers } \sigma. \end{aligned}$$

As a consequence of part (a), the existential and the k-fair universal problem are decidable in polynomial time.

Proof. Denote by $Q' \subseteq \Gamma$ the set of types $X \in \Gamma$ such that there are $\ell \in \mathbb{N}$, a path $\alpha_0, \ldots, \alpha_\ell$ in the Markov chain $\mathcal{M}(\mathcal{S})$, and a type $Y \in Q$ such that $\alpha_0 = X$ and $\beta = \alpha_\ell \geq Y$. Clearly we have $Q \subseteq Q' \subseteq \Gamma$, and Q' can be computed in polynomial time.

In the following, view $\mathcal{S}$ as a context-free grammar with empty terminal set (ignore the probabilities, and put the symbols on the right-hand sides in an arbitrary order). Remove from $\mathcal{S}$ all rules of the form: (i) $X \hookrightarrow \alpha$ where $X \in Q$ or $\alpha(Y) \geq 1$ for some $Y \in Q$, and (ii) $X \hookrightarrow \alpha$ where $X \in \Gamma \setminus Q'$. Furthermore, add rules $X \hookrightarrow \varepsilon$ where $X \in \Gamma \setminus Q'$. Check (in polynomial time) whether in the grammar the empty word ε is produced by α_0.

We have that ε is produced by α_0 if and only if $\mathcal{P}(\alpha_0 \models \Diamond F) < 1$. This follows from Proposition 3, as the complement of $\widetilde{F}$ is the Q'-states set. Hence part (a) of the theorem follows.

For part (b), let $\mathcal{P}(\alpha_0 \models \Diamond F) < 1$. By part (a) we have that ε is produced by α_0. Then for all schedulers σ we have $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) < 1$. Trivially, as a special case, this holds for some k-fair scheduler. (Note that k-fair schedulers exist, as $k \geq |\Gamma|$.)

Conversely, let $\mathcal{P}(\alpha_0 \models \Diamond F) = 1$. By part (a) we have that ε is not produced by α_0. Then, no matter what the scheduler does, the set F remains reachable. So all k-fair schedulers will, with probability 1, hit F eventually. □

6 Semilinear Target Sets

In this section, we prove that the qualitative reachability problems that we considered in the previous sections become undecidable when we extend upward-closed to semilinear target sets.

[1] Our definition seems different from [1], but equivalent from standard embedding of Vector Addition Systems with States to Petri Nets.

Theorem 13. *Let $\mathcal{S} = (\Gamma, \hookrightarrow, Prob)$ be a pBPP. Let $F \subseteq \mathbb{N}^\Gamma$ be a semilinear set. Let $\alpha_0 \in \mathbb{N}^\Gamma$. The following problems are undecidable:*

(a) Does $\mathcal{P}(\alpha_0 \models \Diamond F) = 1$ hold?
(b) Does $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$ hold for all 7-fair schedulers σ?
(c) Does $\mathcal{P}_\sigma(\alpha_0 \models \Diamond F) = 1$ hold for some scheduler σ?

The proofs are reductions from the control-state-reachability problem for 2-counter machines, see [2].

7 Conclusions and Future Work

In this paper we have studied fundamental qualitative coverability and other reachability properties for pBPPs. For the Markov-chain model, the coverability problem for pBPPs is decidable, which is in contrast to general pVASSs. We have also shown a nonelementary lower complexity bound. For the MDP model, we have proved decidability of the existential and the k-fair version of the universal coverability problem. The decision algorithms for the MDP model are not (known to be) elementary, as they rely on Petri-net reachability and a Karp-Miller-style construction, respectively. It is an open question whether there exist elementary algorithms. Another open question is whether the universal MDP problem without any fairness constraints is decidable.

We have given examples of problems where the answer depends on the exact probabilities in the pBPP. This is also true for the reachability problem for finite sets: Given a pBPP and $\alpha_0 \in \mathbb{N}^\Gamma$ and a *finite* set $F \subseteq \mathbb{N}^\Gamma$, the reachability problem for finite sets asks whether we have $\mathcal{P}(\alpha_0 \models \Diamond F) = 1$ in the Markov chain $\mathcal{M}(\mathcal{S})$. Similarly as in Example 10 the answer may depend on the exact probabilities: consider the pBPP with $X \overset{p}{\hookrightarrow} XX$ and $X \overset{1-p}{\hookrightarrow} \varepsilon$, and let $\alpha_0 = XX$ and $F = \{X\}$. Then we have $\mathcal{P}(\alpha_0 \models \Diamond F) = 1$ if and only if $p \leq 1/2$. The same is true in both the existential and the universal MDP version of this problem. Decidability of all these problems is open, but clearly decision algorithms would have to use techniques that are very different from ours, such as analyses of multidimensional random walks.

On a more conceptual level we remark that the problems studied in this paper are qualitative in two senses: (a) we ask whether certain events happen with probability 1 (rather than > 0.5 etc.); and (b) the exact probabilities in the rules of the given pBPP do not matter. Even if the system is nondeterministic and not probabilistic, properties (a) and (b) allow for an interpretation of our results in terms of nondeterministic BPPs, where the nondeterminism is constrained by the laws of probability, thus imposing a special but natural kind of fairness. It would be interesting to explore this kind of "weak" notion of probability for other (infinite-state) systems.

Acknowledgment Anthony W. Lin did this work when he was an EPSRC research fellow at Oxford University supported by grant number EP/H026878/1.

References

1. P.A. Abdulla, N. Ben Henda, and R. Mayr. Decisive Markov chains. *Logical Methods in Computer Science*, 3(4:7), 2007.
2. R. Bonnet, S. Kiefer, and A.W. Lin. Analysis of probabilistic basic parallel processes. Technical report, arxiv.org, 2014. Available at `http://arxiv.org/abs/1401.4130`.
3. T. Brázdil, J. Esparza, S. Kiefer, and M. Luttenberger. Space-efficient scheduling of stochastically generated tasks. *Information and Computation*, 210:87–110, 2012.
4. L. de Alfaro. From fairness to chance. *Electronic Notes in Theoretical Computer Science*, 22:55–87, 1999.
5. N. Dershowitz, D.N. Jayasimha, and S. Park. Bounded fairness. In *Verification: Theory and Practice*, pages 304–317, 2003.
6. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundam. Inform.*, 31(1):13–25, 1997.
7. J. Esparza and A. Kiehn. On the model checking problem for branching time logics and basic parallel processes. In *Proceedings of CAV*, number 939 in LNCS, pages 353–366, 1995.
8. W. Feller. *An introduction to probability theory and its applications*, volume I. John Wiley & Sons, 1968.
9. S.B. Fröschle, P. Jančar, S. Lasota, and Z. Sawa. Non-interleaving bisimulation equivalences on basic parallel processes. *Inf. Comput.*, 208(1):42–62, 2010.
10. S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, 5(3):356–380, 1983.
11. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theor. Comput. Sci.*, 158(1&2):143–159, 1996.
12. H. Hüttel, N. Kobayashi, and T. Suto. Undecidable equivalences for basic parallel processes. *Inf. Comput.*, 207(7):812–829, 2009.
13. P. Jančar. Strong bisimilarity on basic parallel processes is PSPACE-complete. In *Proceedings of LICS*, pages 218–227, 2003.
14. N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *PODC*, pages 314–323, 1994.
15. G. Norman. Analysing randomized distributed algorithms. In *Validation of Stochastic Systems*, pages 384–418, 2004.
16. J.-F. Raskin, M. Samuelides, and L. Van Begin. Games for counting abstractions. *Electronic Notes in Theoretical Computer Science*, 128(6):69–85, 2005.
17. M.Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proceedings of FOCS*, pages 327–338, 1985.