Automatic Verification of Liveness Properties of Randomized Systems (Extended Abstract)

Christel Baier
Fakultät für Mathematik & Informatik
Universität Mannheim
68131 Mannheim, Germany
baier@pi1.informatik.uni-mannheim.de

Marta Kwiatkowska*
School of Computer Science
University of Birmingham, Edgbaston
Birmingham B15 2TT, UK
M.Z.Kwiatkowska@cs.bham.ac.uk

June 13, 1997

Abstract

We consider concurrent probabilistic systems which distinguish between nondeterministic and probabilistic choice. Such systems typically arise as formal models of randomized distributed systems as considered by Segala & Lynch [29, 34, 36]. By resolving only the nondeterministic choices, these models can be decomposed into a collection of "computation trees". The presence of nondeterminism means that certain liveness properties cannot be established unless fairness is assumed. We introduce a probabilistic branching time logic $PBTL$ whose formulas express properties such as "every request is eventually granted with probability at least p" and give an interpretation for $PBTL$ on concurrent probabilistic systems, where the branching time quantifiers are taken to range over a certain kind of fair computation trees. We then present a model checking algorithm for verifying whether a concurrent probabilistic system satisfies a $PBTL$ formula assuming fairness constraints, and also discuss how to modify the algorithm of Bianco & de Alfaro, which can be applied to $PBTL^*$, to cater for fairness.

1 Introduction

Randomization has proved useful in designing and building distributed systems, in many cases yielding more efficient algorithms, see e.g. [7, 27]. However, as already observed in [29, 34, 36], randomized distributed algorithms are notoriously difficult to verify: the proofs of their correctness are complex, and so often argued informally, presenting the programmer with the difficult task of developing an implementation potentially only having partial confidence in the algorithm's guarantees. Formal methods for specifying and verifying randomized systems such as those of [29, 34, 36], and particularly automatic procedures for checking of the required properties, are called for.

The particular difficulty in establishing correctness of randomized algorithms is due to the fact that they exhibit both the probabilistic choice (which comes from random assignment and is considered internal to the system) as well as nondeterminism. The latter arises e.g. from the asynchronicity of certain subprocesses, or external intervention such as action taken by environment. The scheduling of nondeterministic choices is made by an *adversary*. Adversaries are thought capable of selecting one of possibly many probabilistic distributions, with each such distribution giving the probability of reaching a particular successor state.

As argued in [29], the clear separation of probabilistic and nondeterministic choice is advantageous as it allows to decompose the system into a "computation tree", which in turn admits formal verification techniques. The choices made by an adversary can use the knowledge of the past history, but are not, and should *not*, in general, be resolved probabilistically. This means that, as well as having to deal with the technical difficulty of not having at hand the obvious probability space on execution sequences of the system, we also have to introduce explicit *fairness assumptions* as certain liveness properties (such as e.g. "the probability that the message is eventually delivered is at least p") can only be proved if fairness (of nondeterministic choice) is ensured. In summary, fairness assumptions

*Supported in part by EPSRC grant GR/K42028.

allow us to prove more properties, but at a cost of qualifying the correctness statement in the following sense: the system has been shown to satisfy a property *on condition that* the scheduler or user complies with the given fairness assumptions.

In this extended abstract (for the full paper containing proofs and additional material see [9]) we consider models of concurrent probabilistic systems which exhibit both the probabilistic and nondeterministic choice (and thus are particularly apt for the analysis of randomized distributed algorithms), and formulate a model checking procedure for a probabilistic extension of the branching-time temporal logic CTL which we call PBTL (Probabilistic Branching Time Logic) to distinguish it from other syntactically similar probabilistic logics such as PCTL [35] or pCTL [4, 11]. Much has been published in the area of probabilistic logics and the corresponding verification techniques, see e.g. [2, 3, 14, 19, 20, 21, 22, 28, 31, 32, 33, 38]. Our work is closest to that of: Segala & Lynch [29, 34, 36], where formal modelling and verification methods with fairness for a superset of our models (but no model checking) are considered; Hansson & Jonsson [20], where model checking involving linear programming techniques is proposed for a more restricted logic and models without nondeterminism; and Bianco & de Alfaro [11, 4], where the logic and models are similar to ours but fairness is not considered. In addition, our algorithm allows to verify *quantitative* properties such as "the property holds with probability at least p" for any real number p in the $[0, 1]$-interval, and not just the *qualitative* properties for which satisfaction with probability 1 is required. To our knowledge, this paper constitutes the first attempt to provide an *automatic* verification method for proving *quantitative* properties of concurrent probabilistic systems assuming fair scheduling. It should be emphasised that the notion of fairness that we use is w.r.t. nondeterminism, and so is based on that of Hart, Sharir & Pnueli [22] and Vardi [38], and not on [31, 32, 33] where fairness w.r.t. probabilistic choice is imposed.

Technical contribution: The main contribution of this paper (the interested reader is referred to the full paper [9] for more details) is the formulation of a probabilistic extension of a branching-time temporal logic and the corresponding *automatic* model checking procedure for this logic subject to the given fairness assumptions. We consider the logic $PBTL$, based on the logics of [4, 5, 11, 19, 20, 35], which we interpret over concurrent probabilistic systems as described above. $PBTL$ contains atomic propositions, the usual boo-lean connectives, a next step X, an until operator $\mathcal{U}$, and the usual branching time quantifiers E ("there exists a computation tree") and A ("for all computation trees"). The next-step and the until operators are "lifted" to the probabilistic case by means of real numbers $p \in [0, 1]$ which act as probability thresholds for truth. We adapt Vardi's notion of fairness to our model for concurrent probabilistic processes and define an execution sequence π to be fair if none of the nondeterministic alternatives in a state occurring infinitely often on π is refused continuously[1]. As in [22, 38], we define a scheduler to be fair iff the measure of its fair execution sequences is 1.

In contrast to the standard interpretation, where the branching time quantifiers E and A range over all computation trees, we deal with an interpretation involving fairness, i.e. in our interpretation the quantifiers A and E range over *all computation trees that arise from fair schedulers*. For the fair interpretation, we give a model checking algorithm for $PBTL$: it takes a formula Φ and a concurrent probabilistic systems as its input, and returns the set of states satisfying Φ. It turns out that standard techniques can be applied to verify all classes of subformulas with the exception of those containing until as their outermost operator. The handling of the latter class is non-trivial and is justified by a series of technical results omitted from this version of the paper. Until formulas assert something about the set of fair computation trees, which in general is, as the fair computation trees themselves, infinite. We show that – instead of ranging over *all fair* computation trees – an investigation of a certain *finite* subclass of (in general unfair) computation trees (that can be represented by a Markov chain) suffices. For Markov chains, the probability of the set of paths fulfilling an until property can be computed by solving a linear equation system (cf. [14, 20]).

Related work: Apart from the temporal logic frame-

[1] Our notion of fairness can be viewed as a special case of Vardi's [38]. In the full paper [9] we show that our methods extend to the more general notion of fairness à la Vardi, where we distinguish between states in which fairness is required and possibly unfair states, at a cost of an extra level of complexity.

work, probabilistic extensions of equivalence relations and preorders have been investigated and applied to the specification and verification of probabilistic processes, see e.g. [8, 23, 25, 26, 35, 39]. Several authors presented verification methods for proving qualitative properties of concurrent probabilistic systems, but only a minority of them deal with fairness. [22] proposes a decision procedure for checking that a given liveness property is fulfilled with probability 1 in all fair computation trees of a concurrent probabilistic system. [38] extends the results of [22] and presents a verification method for concurrent Markov chains which aims to establish whether or not a linear time formula holds with probability 1 for all fair computation trees. Verification methods for proving quantitative properties of probabilistic systems can be found in [4, 5, 11, 14, 15, 20, 24]. [5, 14, 20, 24] deal with sequential Markov chains as models for probabilistic systems. [15] introduces a model for concurrent probabilistic processes based on concurrent Markov chains and shows that the verification problem w.r.t. a specification given in the form of a set of ω-regular languages can be reduced to a linear programming problem (and thus yields a polynomial time method for the verification of concurrent probabilistic systems w.r.t. a specification expressed in some temporal logic). [4, 11] consider a model for probabilistic systems which, similarly to ours, distinguishes between probabilistic choice and nondeterminism. In essence, the logic $pCTL$ considered in [11] agrees with our logic $PBTL$, and is extended in [4] by an operator to express bounds on the average time between events. [11] (and [4]) present model checking algorithms for $pCTL$ and $pCTL^*$ which extends $pCTL$ by more complex path formulas similar to the way CTL^* extends CTL. The issue of fairness is not treated in [4, 11]. Although the logic $pCTL^*$ considered in [11] is capable of expressing fairness of execution sequences, formulas stating the existence or nonexistence of fair computation trees with certain properties *cannot* be expressed in $pCTL^*$. Hence, the model checking algorithm for $pCTL^*$ of [11] does not yield a method for proving quantitative properties of concurrent probabilistic systems assuming fairness of schedulers resolving nondeterministic choices. [29, 34, 36] present proof rules for establishing quantitative (timed) progress properties for randomized distributed systems which can be combined with several notions of fairness.

2 Concurrent probabilistic systems

As pointed out in [38], certain states of a concurrent system whose components work asynchronously are inherently nondeterministic. The nondeterministic choices are beyond the control of the process and can be supposed to be resolved by a scheduler, whereas the probabilistic choices are made by the system itself. Our model generalizes the "concurrent Markov chains" considered in [14, 38] and essentially agrees with the "simple deterministic automata" of [35].

Definition 2.1 *A* concurrent probabilistic system *is a pair $\mathcal{C} = (S, Steps)$ where S is a finite set of states and $Steps$ a function which assigns to each state s a finite and nonempty set $Steps(s)$ of distributions on S*[2].

$Steps(\cdot)$ can be viewed as a total transition relation on S which assigns to each state a set of nondeterministic alternatives. Given a state $s \in S$, a scheduler chooses some transition $\mu \in Steps(s)$. The process itself resolves the probabilistic choice, i.e. selects some state t with positive probability ($\mu(t) > 0$).

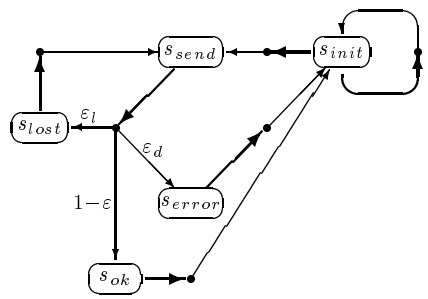
Example 2.2 Let P be a concurrent process which sends messages to another process Q along an uncertain medium M, which possibly loses or destroys messages. In cases where the messages are lost, M sends a signal to P and P attempts to send the message again. We suppose that neither P nor M nor Q are able to detect whether or not the message is destroyed in cases when a message is actually delivered to Q. Apart from sending messages, P can also perform some internal actions which we do not specify; note that this gives rise to a nondeterministic choice between sending a message (if one is available) and performing internal actions (which leads back to the initial state), each with probability 1[3]. We describe the system from the point of view of an "observer" who does not have access to the local variables of P. The state space S consists of the following states:

[2] A distribution on a finite set S is a function $\mu : S \to [0, 1]$ with $\sum_{t \in S} \mu(t) = 1$.

[3] In general, distributions other than point distributions are allowed.

s_{init}	initial state
s_{send}	sending message
s_{lost}	message lost
s_{ok}	successful delivery
s_{error}	error state

We assume that M delivers the correct message with probability $1-\varepsilon$ (for some "small" ε) and that it loses (resp. destroys) the message with probability ε_l (resp. $\varepsilon_d = \varepsilon - \varepsilon_l$).



Let μ_s be the distribution with $\mu_s(s) = 1$. μ is given by $\mu(s_{ok}) = \varepsilon$, $\mu(s_{lost}) = \varepsilon_l$, $\mu(s_{error}) = \varepsilon_d$. Then,
$Steps(s_{init}) = \{\mu_{send}, \mu_{init}\}$,
$Steps(s_*) = \{\mu_{init}\}$, $* \in \{error, ok\}$,
$Steps(s_{lost}) = \{\mu_{send}\}$,
$Steps(s_{send}) = \{\mu\}$. ■

Execution sequences (which we call paths) arise by resolving both the nondeterministic and probabilistic choices. A *path* is a nonempty (finite or infinite) alternating sequence

$$\pi \;=\; s_0 \;\stackrel{\mu_1}{\rightarrow}\; s_1 \;\stackrel{\mu_2}{\rightarrow}\; s_2 \;\ldots$$

where s_i are states, $\mu_i \in Steps(s_{i-1})$ and $\mu_i(s_i) > 0$, $i = 1, 2, \ldots$[4]. A path π is called a *fulpath* iff it is infinite. The first state of a path π is denoted by $first(\pi)$. If π is finite then the last state of π is denoted by $last(\pi)$. The length $|\pi|$ of a path π is defined in the usual way as follows: if $\pi = s_0 \stackrel{\mu_1}{\rightarrow} s_1 \stackrel{\mu_2}{\rightarrow} \ldots \stackrel{\mu_n}{\rightarrow} s_n$ then $|\pi| = n$, else if π infinite we put $|\pi| = \infty$. Let π be a finite or infinite path as above. If $k \leq |\pi|$ then $\pi(k)$ denotes the k-th state of π (i.e. $\pi(k) = s_k$). $\pi^{(k)}$ is the k-th prefix of π (i.e. if $k < |\pi|$ then $\pi^{(k)} = s_0 \stackrel{\mu_1}{\rightarrow} s_1 \stackrel{\mu_2}{\rightarrow} \ldots \stackrel{\mu_k}{\rightarrow} s_k$, and if $k \geq |\pi|$ then $\pi^{(k)} = \pi$.) If $i < |\pi|$ then we put $step(\pi, i) = \mu_{i+1}$. If π is a fulpath then $inf(\pi)$ denotes the set of states $s \in S$ with $s = \pi(i)$ for infinitely many i. $Path(\mathcal{C})$ denotes the set of all fulpaths in $\mathcal{C}$, $Path_{\rm fin}(\mathcal{C})$ the set of all finite paths in $\mathcal{C}$. For $s \in S$, $Path(s, \mathcal{C})$ denotes the set of fulpaths π in $\mathcal{C}$ with $first(\pi) = s$, $Path_{\rm fin}(s, \mathcal{C})$ the set of finite paths ω with $first(\omega) = s$. When clear from the context we abbreviate $Path(\mathcal{C})$ by $Path$, $Path_{\rm fin}(\mathcal{C})$ by $Path_{\rm fin}$, $Path(s, \mathcal{C})$ by $Path(s)$ and $Path_{\rm fin}(s, \mathcal{C})$ by $Path_{\rm fin}(s)$.

[4]The case $\pi = s_0$ is allowed.

We split a concurrent probabilistic system into its "computation trees" (called "execution trees" in [22] and "maximal resolutions" in [25]). The computation trees arise by resolving the nondeterministic choices (but not the probabilistic choices). It is convenient to suppose that an adversary decides which step is performed next.

Definition 2.3 *An* adversary *(or* scheduler*) of a concurrent probabilistic system $\mathcal{C} = (S, Steps)$ is a function $A : Path_{\rm fin} \rightarrow \bigcup_s Steps(s)$ with $A(\omega) \in Steps(last(\omega))$ for all $\omega \in Path_{\rm fin}$. An adversary A of $\mathcal{C}$ is called* simple *iff*

$$\forall s \in S \exists \mu_s \in Steps(s) \forall \omega \in Path_{\rm fin} : A(\omega) = \mu_{last(\omega)}$$

$\mathcal{A}(\mathcal{C})$, abbrev. $\mathcal{A}$, denotes the set of all adversaries of $\mathcal{C}$, $\mathcal{A}_{simple}(\mathcal{C})$, abbrev. $\mathcal{A}_{simple}$, the set of simple adversaries.

An adversary chooses for every finite path ω in $\mathcal{C}$ an outgoing "transition" from $last(\omega)$. Simple adversaries resolve the nondeterminism by selecting for every state a next step which is executed whenever the state s is reached – independent of the past history. In some sense, simple adversaries are extremely unfair and would be ruled out for practical purposes. We need them only for the sake of convenience.

Example 2.4 For the system in Example 2.2, nondeterministic choice is possible only in the state s_{init}: the system can either send the message or loop back to s_{init}. Thus, we have exactly two simple adversaries: A_{send}, the unique simple adversary with $A_{send}(\omega) = \mu_{send}$ if $last(\omega) = s_{init}$, and A_{init}, the unique simple adversary with $A_{init}(\omega) = \mu_{init}$ if $last(\omega) = s_{init}$. ■

We suppose the reader to be familiar with basic notions of measure and probability theory (see e.g. [18]). Each adversary A induces a probability space on $Path^A(s)$, the set of fulpaths π starting in s where $step(\pi, i) = A(\pi^{(i)})$ for all $i \geq 1$. Let $Path^A_{\rm fin}(s)$ be the set of all finite paths that are prefices of some $\pi \in Path^A(s)$

and let $\Sigma^A(s)$ be the smallest σ-field on $Path^A(s)$ which contains the basic cylinders $\{\pi \in Path^A(s) : \omega \text{ is a prefix of } \pi\}$, $\omega \in Path^A_{\text{fin}}(s)$. The probability measure $\mathcal{P}$ on $\Sigma^A(s)$ is the unique measure with

$$\mathcal{P}\left\{\pi \in Path^A(s) : \omega \text{ is a prefix of } \pi\right\} = \mathcal{P}(\omega)$$

where $\mathcal{P}\left(s_0 \xrightarrow{\mu_1} \ldots \xrightarrow{\mu_k} s_k\right) = \mu_1(s_1) \cdot \ldots \cdot \mu_k(s_k)$.

Our notion of fairness requires (strong) fairness of the adversaries, or, in other words, fairness with respect to the nondeterministic choices as in [22, 38] (rather than the probabilistic choices as in [31, 32, 33]). We adapt Vardi's notion of fair paths in concurrent Markov chains to our model of concurrent probabilistic systems with the simplification that we require fairness in all states. We define a fulpath π of a concurrent probabilistic system to be *fair* iff for each state s occurring infinitely often on π, each nondeterministic alternative which is enabled in s (i.e. each distribution $\mu \in Steps(s)$) is taken infinitely often in π.

Definition 2.5 *Let π be a fulpath in a concurrent probabilistic system $\mathcal{C} = (S, Steps)$. π is called* fair *iff for each $s \in inf(\pi)$ and each $\mu \in Steps(s)$, there are infinitely many indices i with $\pi(i) = s$ and $step(\pi, i) = \mu$. For A to be an adversary of $\mathcal{C}$ and $s \in S$, we define $Fair^A(s)$ to be the set of all fair fulpaths $\pi \in Path^A(s)$.*

The above notion of fairness is stronger than fairness of fulpaths in [22]. In [22] "process fairness" is considered, in the sense that, for π to be fair, all sequential processes (whose composition is the concurrent probabilistic system under consideration) are activated infinitely many times in π. If $\mathcal{C}$ is a concurrent probabilistic system which arises through the interleaving of sequential processes without shared variables then fairness in the sense of Definition 2.5 implies fairness in the sense of [22].

Example 2.6 For the system in Example 2.2, a fulpath π is fair iff $s_{send} \in inf(\pi)$. For instance, the fulpath π with $\pi(i) = s_{init}$ for all $i \geq 0$ is not fair, while the fulpath

$$s_{init} \to s_{send} \to s_{lost} \to s_{send} \to s_{lost} \to s_{send} \to \ldots$$

is fair. (We omit the names of the distributions.) ■

Definition 2.7 *Let F be an adversary of a concurrent probabilistic system $\mathcal{C} = (S, Steps)$. F is called* fair *iff $\mathcal{P}(Fair^F(s)) = 1$ for all $s \in S$. $\mathcal{A}_{fair}(\mathcal{C})$, abbrev. $\mathcal{A}_{fair}$, denotes the set of fair adversaries.*

If F is a fair adversary then for each $\omega \in Path^F_{\text{fin}}(s)$ there exists $\pi \in Fair^F(s)$ such that ω is a prefix of π. This reflects "liveness" in the sense of [1] which states that every finite computation can be extended to an infinite computation which is "live".

3 Probabilistic branching time temporal logic

In this section we introduce the logic $PBTL$ (Probabilistic Branching Time Logic). $PBTL$ is a probabilistic extension of CTL which allows to express quantitative properties of probabilistic processes such as "the system terminates with probability at least 0.75". In essence, the syntax of $PBTL$ agrees with the logic $PCTL$ considered by Segala & Lynch [35] and the logic $pCTL$ considered by Bianco & de Alfaro [11]. As in [11, 35], $PBTL$ formulas are interpreted over the states of a concurrent probabilistic system. The logics (also called $PCTL$) considered by Hansson & Jonsson [20] and Aziz et al [5] are interpreted over sequential Markov chains and can be viewed as linear time versions of $PBTL$. $PBTL$ contains atomic propositions and operators: next-step X and until $\mathcal{U}$. The operators X and $\mathcal{U}$ are used in connection with the branching time quantifiers E (the existence-quantor) and A (the all-quantor) and an interval of probabilities. From now on we suppose a fixed set of atomic propositions. The syntax of $PBTL$ is as follows:

$$\Phi \quad ::= \quad tt \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid [EX\Phi]_{\sqsupseteq p} \mid [AX\Phi]_{\sqsupseteq p} \mid [\Phi_1 E\mathcal{U}\Phi_2]_{\sqsupseteq p} \mid [\Phi_1 A\mathcal{U}\Phi_2]_{\sqsupseteq p}$$

where a is an atomic proposition, $p \in [0, 1]$, $\sqsupseteq$ is either $\geq$ or $>$. Formulas of the form $X\Phi$ or $\Phi_1\mathcal{U}\Phi_2$, where Φ, Φ_1, Φ_2 are $PBTL$ formulas, are called *path formulas*. $PBTL$ formulas are interpreted over states of concurrent probabilistic processes, whereas path formulas over paths. Informally, $X\Phi$ asserts that Φ holds in the next state. $\Phi_1\mathcal{U}\Phi_2$ is true iff from now on Φ_1 is fulfilled until Φ_2 holds and Φ_2 holds sometime in the future. The subscript $\sqsupseteq p$ denotes that the probability for paths (in

an adversary) fulfilling the path formula is $\sqsupseteq p$. Depending on the interpretation, the branching time quantifiers E and A denote quantification over the computations which result from adversaries of a certain type: E "there exists an adversary of this type", and A "for all adversaries of this type". In the standard interpretation $\models$, no restrictions on the adversaries are made, while in our interpretation $\models_{fair}$, the quantifiers E and A range over the fair adversaries.

The usual derived constants and operators are: $ff = \neg tt$, $\Phi_1 \vee \Phi_2 = \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, $\Phi_1 \rightarrow \Phi_2 = \neg\Phi_1 \vee \Phi_2$. Operators for modelling "eventually" or "always" can be derived by:

$$[E\Diamond\Phi]_{\sqsupseteq p} = [tt E\mathcal{U}\Phi]_{\sqsupseteq p},\ [E\Box\Phi]_{\sqsupseteq p} = \neg[A\Diamond\neg\Phi]_{\overline{\sqsupseteq} 1-p},$$

$$[A\Diamond\Phi]_{\sqsupseteq p} = [tt A\mathcal{U}\Phi]_{\sqsupseteq p},\ [\,A\Box\Phi]_{\sqsupseteq p} = \neg[E\Diamond\neg\Phi]_{\overline{\sqsupseteq} 1-p},$$

where $\overline{\geq} \ = >$ and $\overline{>} \ = \geq$. For instance, $[E\Diamond\Phi]_{\geq p}$ states the existence of an adversary where the probability for paths in which Φ eventually holds is $\geq p$. Formulas of the form $[A\Box\Phi]_{\geq p}$ express safety properties asserting that for all adversaries the probability for paths where Φ holds continuously is $\geq p$.

A *$PBTL$-structure* is a tuple $M = (\mathcal{C}, I)$ where $\mathcal{C} = (S, Steps)$ is a concurrent probabilistic system and I an interpretation for the atomic propositions, i.e. I is a mapping which assigns to each state s of S a set $I(s)$ of atomic propositions. For a given $PBTL$-structure $M = (\mathcal{C}, I)$, the satisfaction relation $\models_{fair} \subseteq S \times PBTL$ is shown in Table 1[5]. The fact that for a path formula Ψ the set $\{\pi \in Path^A(s) : \pi \models_{fair} \Psi\}$ is measurable is an easy verification. If $s \models_{fair} \Phi$ then we say s fulfills Φ (or also s satisfies Φ or Φ holds in s). The truth value of formulas involving the (linear time) quantifiers $\Diamond$ and $\Box$ can be derived. For example, $s \models_{fair} [\ E\Diamond\ \Phi\ ]_{\geq p}$ if and only if there exists $A \in \mathcal{A}_{fair}$ with

$$\mathcal{P}\left\{\pi \in Path^A(s) : \pi(k) \models_{fair} \Phi \text{ for some } k \geq 0\right\} \geq p.$$

Hence, $s \models_{fair} [A\Box\Phi]_{>p}$ iff

$$\mathcal{P}\{\pi \in Path^A(s) : \pi(k) \models_{fair} \Phi \text{ for all } k \geq 0\} > p$$

for all $A \in \mathcal{A}_{fair}$.

Given a probabilistic process P, described by a concurrent probabilistic system $\mathcal{C}$ with an initial state s and an interpretation I which assigns to each state s a set $I(s)$ of atomic propositions, we say P satisfies a $PBTL$ formula Φ iff $s \models_{fair} \Phi$ w. r. t. the $PBTL$-structure $(\mathcal{C}, I)$. For instance, if a is an atomic proposition which stands for termination and P satisfies $[A\Diamond a]_{\geq p}$ then P terminates with probability p; more precisely, every fair computation of P terminates with probability at least p. If c_i, $i = 1, 2$, are atomic propositions stating that subprocesses P_i of P are in their critical sections and P satisfies $[A\Box(\neg c_1 \vee \neg c_2)]_{\geq p}$ then P fulfills mutual exclusion with probability p; more formally, in each fair computation, the probability of an execution in which P_1 and P_2 are never simultanously in their critical sections is at least p.

Example 3.1 We consider the atomic propositions $init$, $send$ and ok and the $PBTL$-structure $(\mathcal{C}, I)$ where $\mathcal{C}$ is the system in Example 2.2 and I the interpretation with $a \in I(s)$ iff $s = s_a$. The liveness property

$$\Lambda \ = \ [\ A\Diamond send\ ]_{\geq 1}$$

states that, in all fair computations, P will eventually try to send a message with probability 1. For each fair adversary F of $\mathcal{C}$, the state s_{send} occurs on almost all fulpaths of F (cf. Example 2.6). Hence, $s_{init} \models_{fair} \Lambda$. On the other hand, $s_{init} \not\models \Lambda$ since, for the simple adversary A_{init} no fulpath π starting in s_{init} contains the state s_{send}[6]; in other words, without fairness assumptions we *cannot* ensure that P will try to send a message.

The quantitative property $\Phi = [\ (init \vee send)\ A\mathcal{U}\ ok\ ]_{\geq \varepsilon}$ states that, in all fair computations, with probability at least ε, P will eventually send a message which is immediately delivered correctly. Then, $s_{init} \models_{fair} \Phi$ (cf. Example 4.4) while $s_{init} \not\models \Phi$ (since no fulpath in A_{init} fulfills $(init \vee send)\mathcal{U}ok$). ■

4 Model checking for $PBTL$

In [11] a model checking algorithm for $pCTL$ w.r.t. the standard interpretation $\models$ (where the quantifiers E and A range over all computations rather than the fair ones) can be found, which is time polynomial in the size of

[5] We write $s \models_{fair} \Phi$ instead of $(s, \Phi) \in \models_{fair}$.

[6] Thus, $\mathcal{P}\{\pi \in Path^{A_{init}}(s_{init}) : \pi \models \Diamond send\} = 0$.

$s \models_{fair} tt$ for all $s \in S$, $\quad s \models_{fair} a$ iff $a \in I(s)$

$s \models_{fair} \Phi_1 \wedge \Phi_2$ iff $s \models_{fair} \Phi_i$, $i = 1, 2$

$s \models_{fair} \neg\Phi$ iff $s \not\models_{fair} \Phi$

$s \models_{fair} [EX\Phi]_{\sqsupseteq p}$ iff $\mathcal{P}\left\{\pi \in Path^A(s) : \pi \models_{fair} X\Phi\right\} \sqsupseteq p$ for some $A \in \mathcal{A}_{fair}$.

$s \models_{fair} [AX\Phi]_{\sqsupseteq p}$ iff $\mathcal{P}\left\{\pi \in Path^A(s) : \pi \models_{fair} X\Phi\right\} \sqsupseteq p$ for all $A \in \mathcal{A}_{fair}$.

$s \models_{fair} [\Phi_1 E\mathcal{U}\Phi_2]_{\sqsupseteq p}$ iff $\mathcal{P}\left\{\pi \in Path^A(s) : \pi \models_{fair} \Phi_1\mathcal{U}\Phi_2\right\} \sqsupseteq p$ for some $A \in A_{fair}$.

$s \models_{fair} [\Phi_1 A\mathcal{U}\Phi_2]_{\sqsupseteq p}$ iff $\mathcal{P}\left\{\pi \in Path^A(s) : \pi \models_{fair} \Phi_1\mathcal{U}\Phi_2\right\} \sqsupseteq p$ for all $A \in \mathcal{A}_{fair}$.

$\pi \models_{fair} X\Phi$ iff $\pi(1) \models_{fair} \Phi$

$\pi \models_{fair} \Phi_1\mathcal{U}\Phi_2$ iff there is some $k \geq 0$ with $\pi(k) \models_{fair} \Phi_2$ and $\pi(i) \models_{fair} \Phi_1$, $i = 0, \ldots, k-1$.

Table 1: The satisfaction relation $\models_{fair}$

the underlying concurrent probabilistic system. In addition, [11] contains a model checking algorithm for a much richer logic $pCTL^*$ which allows arbitrary combinations of path formulas by the boolean connectives and path operators such as next step and until (see Section 5). In the non-probabilistic case, e.g. when using CTL, fairness of fulpaths can be expressed by path formulas of the full branching time logic, e.g. CTL^*. Typically, this is achieved by means of formulas of the form $\varphi_{fair} = \bigvee_i \bigwedge_j (\Diamond\Box\varphi_{i,j} \vee \Box\Diamond\psi_{i,j})$, and the model checking for CTL under fairness assumptions (e.g. w.r.t. a non-probabilistic version of our satisfaction relation $\models_{fair}$) can be reduced to the model checking problem for CTL^* since one has an equivalence of the form $s \models_{fair} A\varphi$ iff $s \models A(\varphi_{fair} \rightarrow \varphi)$ (cf. [17]). Unfortunately, this equivalence does *not* hold in the probabilistic case. The problem is that formulas of the form $[\, A(\varphi_{fair} \rightarrow \varphi)\,]_{\sqsupseteq p}$ interpreted over $\models$ state that in *all adversaries* the measure of fulpaths which are either unfair or fair and satisfy φ is $\sqsupseteq p$ (rather than that in all fair adversaries the probability for φ is $\sqsupseteq p$). Hence, the model checking algorithm of [11] for $pCTL^*$ cannot be used (at least not in a straightforward manner) to handle fairness.

In this section we present a model checking algorithm for $PBTL$ (which essentially coincides with $pCTL$) for the satisfaction relations $\models_{fair}$. The starting point is a $PBTL$-structure $M = (\mathcal{C}, I)$ and a $PBTL$-formula Φ. The algorithm is similar to the model checking algorithms of [12] for CTL and that of [11] for $pCTL$. First, the algorithm builds the parse tree of Φ whose nodes stand for subformulas of Φ; more precisely, the leaves are labelled by *tt* or an atomic proposition, and the internal nodes are labelled by one of the operators $\wedge$, $\neg$, $[\, EX\, _\,]_{\sqsupseteq p}$, $[\, AX\, _\,]_{\sqsupseteq p}$, $[\, _\, E\mathcal{U}\, _\,]_{\sqsupseteq p}$ or $[\, _\, A\mathcal{U}\, _\,]_{\sqsupseteq p}$. Nodes labelled by $\neg$ or the next-step operator have exactly one son, representing the argument of the negation, resp. next step operator, in the corresponding subformula. Nodes labelled by $\wedge$ or the until operator have exactly two sons (their arguments). For each node v we calculate the set $Sat(\Psi)$ consisting of the states where the subformula Ψ holds. The cases where the associated formula of a node v is *tt*, a, $\neg\Phi'$ or $\Phi = \Phi_1 \wedge \Phi_2$ is clear as we have: $Sat(tt) = S$, $Sat(a) = \{s \in S : a \in I(s)\}$, $Sat(\neg\Phi') = S \setminus Sat(\Phi')$ and $Sat(\Phi_1 \wedge \Phi_2) = Sat(\Phi_1) \cap Sat(\Phi_2)$. For the next-step operator we have:

$s \models_{fair} [EX\Phi]_{\sqsupseteq p}$ iff there is some $\mu \in Steps(s)$ with $\sum_{t \in Sat(\Phi)} \mu(t) \sqsupseteq p$

$s \models_{fair} [AX\Phi]_{\sqsupseteq p}$ iff $\sum_{t \in Sat(\Phi)} \mu(t) \sqsupseteq p$ for all $\mu \in Steps(s)$.

The remainder of this section is concerned with the until operator. We fix a $PBTL$-model $M = (\mathcal{C}, I)$ where $\mathcal{C} = (S, Steps)$ and two $PBTL$ formulas Φ_1, Φ_2 and suppose that the sets $Sat(\Phi_1)$, $Sat(\Phi_2)$ are already computed. For an adversary A of $\mathcal{C}$ and $s \in S$ we define:

$$p_s^A(\phi) = \mathcal{P}\{\pi \in Path^A(s) : \pi \models_{fair} \phi\}$$
$$p_s^{max}(\phi) = \max\{p_s^A(\phi) : A \in \mathcal{A}_{simple}\}$$
$$p_s^{min}(\phi) = \min\{p_s^A(\phi) : A \in \mathcal{A}_{simple}\}$$

where ϕ is the path formula $\Phi_1\mathcal{U}\Phi_2$. Note that $p_s^{max}(\phi)$ and $p_s^{min}(\phi)$ are well-defined since $\mathcal{A}_{simple}$ is finite.

Theorem 1 (cf. [11, 15]) *For all $s \in S$:*

(a) $s \models [\Phi_1 E\mathcal{U}\Phi_2]_{\sqsupseteq p}$ *iff* $p_s^{max}(\Phi_1\mathcal{U}\Phi_2) \sqsupseteq p$.

(b) $s \models [\Phi_1 A \mathcal{U} \Phi_2]_{\sqsupseteq p}$ *iff* $p_s^{min}(\Phi_1 \mathcal{U} \Phi_2) \sqsupseteq p$.

Theorem 1 can be derived from Corollary 20 (part 1) of [11], which uses the results of [15]. The reason we state it here is that part (a) of this theorem carries over to the satisfaction relation $\models_{fair}$ (Theorem 2), while part (b) does not (cf. Example 3.1 which shows that $s \models_{fair} [\Phi_1 A \mathcal{U} \Phi_2]_{\sqsupseteq p}$ while $s \not\models [\Phi_1 A \mathcal{U} \Phi_2]_{\sqsupseteq p}$ is possible).

Theorem 2 *For all* $s \in S$: $s \models_{fair} [\Phi_1 \; E\mathcal{U} \; \Phi_2]_{\sqsupseteq p}$ *iff* $p_s^{max}(\Phi_1 \mathcal{U} \Phi_2) \sqsupseteq p$.

In order to reduce the question whether or not a formula of the form $[\Phi_1 A \mathcal{U} \Phi_2]_{\sqsupseteq p}$ is fulfilled w.r.t. $\models_{fair}$ to an investigation of certain simple adversaries we must find a subclass of simple adversaries A where the probabilities $p_s^A(\Phi_1 \mathcal{U} \Phi_2)$ can be "approximated" by fair adversaries. In Example 3.1 we saw that the liveness property $[A \Diamond send]_{\geq 1}$ cannot be established unless fairness is assumed. The problem with the simple adversary A_{init} is that it forces P to stay forever in a "non-successful" state (s_{init}) from which a "successful" state (s_{send}) can be reached. In fair adversaries, with probability 1, all states that are reachable from a state that is visted infinitely often are also visited infinitely often. This explains why $p_s^{A_{init}}(\Diamond send)$ cannot be "approximated" by fair adversaries. Thus, in order to handle formulas of the form $[\Phi_1 A \mathcal{U} \Phi_2]_{\sqsupseteq p}$ we restrict our attention to those simple adversaries A where almost all fulpaths π contain a state s that either fulfills Φ_2 (i.e. $s \models_{fair} \Phi_2$) or where $\Phi_1 \mathcal{U} \Phi_2$ cannot be satisfied with a non-zero probability (i.e. $p_s^{max}(\Phi_1 \mathcal{U} \Phi_2) = 0$).

We define $Reach_{\Phi_1 \wedge \neg \Phi_2}(s)$ to be the set of states which are *reachable* in $\mathcal{C}$ from s via a path where all states – possibly except the last one – fulfill the formula $\Phi_1 \wedge \neg \Phi_2$. $S^+(\Phi_1, \Phi_2)$ is the set of all states from which one can reach a Φ_2-state via a path through Φ_1-states. Formally:

Notation 4.1 $Reach_{\Phi_1 \wedge \neg \Phi_2}(s)$ *is the set of states* $t \in S$ *such that there is some* $\omega \in Path_{\text{fin}}(s)$ *with* $last(\omega) = t$ *and* $\omega(i) \models \Phi_1 \wedge \neg \Phi_2$, $i = 0, 1, \ldots, |\omega| - 1$. *Let* $S^+(\Phi_1, \Phi_2)$ *be the set of states* $s \in S$ *where*

$$Reach_{\Phi_1 \wedge \neg \Phi_2}(s) \cap Sat(\Phi_2) \neq \emptyset.$$

For a simple adversary A, we define $Reach^A_{\Phi_1 \wedge \neg \Phi_2}(s)$ *to be set of states* t *such that there is some* $\omega \in Path^A_{\text{fin}}(s)$ *with* $last(\omega) = t$ *and* $\omega(i) \models \Phi_1 \wedge \neg \Phi_2$ *for all* $i < |\omega|$.

Then, $s \in S^+(\Phi_1, \Phi_2)$ if and only if $p_s^A(\Phi_1 \mathcal{U} \Phi_2) > 0$ for some $A \in \mathcal{A}$ if and only if $p_s^{max}(\Phi_1 \mathcal{U} \Phi_2) > 0$.

Example 4.2 For the system in Example 2.2 we have:

$Reach_{(init \vee send) \wedge \neg ok}(s_{init}) = \{s_{init}, s_{send}, s_{ok}\}$,
$Reach_{(init \vee send) \wedge \neg ok}(s_{send}) = \{s_{send}, s_{ok}\}$,
$Reach_{(init \vee send) \wedge \neg ok}(s_{ok}) = \{s_{ok}\}$,
$Reach_{(init \vee send) \wedge \neg ok}(s_*) = \emptyset$ for $* \in \{lost, error\}$.

Thus, $S^+(init \vee send, ok) = \{s_{init}, s_{send}, s_{ok}\}$. ■

We call a simple adversary A *admissible for* (Φ_1, Φ_2) iff for all $s \in S^+(\Phi_1, \Phi_2)$ there is a finite path $\omega \in Path^A_{\text{fin}}(s)$ with $\omega(i) \models_{fair} \Phi_1 \wedge \neg \Phi_2$, $i = 0, 1, \ldots, |\omega| - 1$, and either $last(\omega) \models_{fair} \Phi_2$ or $last(\omega) \notin S^+(\Phi_1, \Phi_2)$. This definition ensures that, in an admissible adversary, the probability of the set of fulpaths $\pi \in Path^A(s)$ where $\pi(i) \in S^+(\Phi_1, \Phi_2) \setminus Sat(\Phi_2)$ for all $i \geq 0$ is 0 (which is the case for all fair adversaries).

Definition 4.3 *A simple adversary* A *of* $\mathcal{C}$ *is called* admissible *for* (Φ_1, Φ_2) *iff for all* $s \in S^+(\Phi_1, \Phi_2)$*:*

$$Reach^A_{\Phi_1 \wedge \neg \Phi_2}(s) \cap (Sat(\Phi_2) \cup (S \setminus S^+(\Phi_1, \Phi_2))) \neq \emptyset.$$

Let $\mathcal{A}_{adm}(\Phi_1, \Phi_2)$ *be the set of simple adversaries that are admissible for* (Φ_1, Φ_2)*. For* $s \in S$*, let*

$$p_s^{adm}(\phi) = \min\{p_s^A(\phi) : A \in \mathcal{A}_{adm}(\Phi_1, \Phi_2)\}$$

where $\phi = \Phi_1 \mathcal{U} \Phi_2$.

The following theorem states that to handle formulas of the type $[\Phi_1 A \mathcal{U} \Phi_2]_{\sqsupseteq p}$ we need only consider the admissible adversaries.

Theorem 3 *For all* $s \in S$: $s \models_{fair} [\Phi_1 \; A\mathcal{U} \; \Phi_2]_{\sqsupseteq p}$ *iff* $p_s^{adm}(\Phi_1 \mathcal{U} \Phi_2) \sqsupseteq p$.

Example 4.4 For the system in Example 2.2, A_{send} is admissible for $(init \vee send, ok)$ while A_{init} is not[7]. Thus, $\mathcal{A}_{adm}(init \vee send, ok) = \{A_{send}\}$ and

$$p_{s_{init}}^{adm}(\phi) \; = \; p_{s_{init}}^{A_{send}}(\phi) = \varepsilon$$

where $\phi = (init \vee send) \mathcal{U} ok$. Hence,

$$s_{init} \models_{fair} [(init \vee send) A \mathcal{U} ok]_{\geq \varepsilon}$$

by Theorem 3. ■

[7] Note that $Reach^{A_{init}}_{init \vee send, ok}(s_{init}) = \{s_{init}\}$ and $s_{init} \notin \{s_{ok}\} \cup S \setminus S^+(init \vee send, ok) = \{s_{ok}, s_{error}, s_{lost}\}$.

If $Reach_{\Phi_1 \wedge \neg \Phi_2}(s) \subseteq S^+(\Phi_1, \Phi_2)$ then (by Theorem 3) $s \models_{fair} [\Phi_1 A \mathcal{U} \Phi_2]_{\geq 1}$[8]. Vice versa, if (at least one state of) $Reach_{\Phi_1 \wedge \neg \Phi_2}(s)$ is not contained in $S^+(\Phi_1, \Phi_2)$ then there is a finite path $\omega \in Path_{\text{fin}}(s)$ with $\omega(i) \models \Phi_1 \wedge \neg \Phi_2$, $i = 0, 1, \ldots, |\omega| - 1$, and $last(\omega) \notin S^+(\Phi_1, \Phi_2)$. Hence, for a fair adversary F with $\omega \in Path^F_{\text{fin}}(s)$, we have $p^F_s(\Phi_1 \mathcal{U} \Phi_2) \leq 1 - \mathcal{P}(\omega) < 1$. Thus:

Corollary 4.5 *For all $s \in S$: $s \models_{fair} [\ \Phi_1\ A\mathcal{U}\ \Phi_2\ ]_{\geq 1}$ iff $Reach_{\Phi_1 \wedge \neg \Phi_2}(s) \subseteq S^+(\Phi_1, \Phi_2)$.*

In particular, a concurrent probabilistic system $\mathcal{C}$ with initial state s_{init} satisfies "qualitative progress properties" expressed by $PBTL$ formulas of the form $[\ A \Diamond \Phi\ ]_{\geq 1}$ if and only if the system is "safe" in the sense that no "deadlocked" state (a state t from which no Φ-state can be reached, i.e. $Reach(t) \cap Sat(\Phi) = \emptyset$) can be reached from the initial state. Hence, $s_{init} \models_{fair} [A \Diamond \Phi]_{\geq 1}$ iff $Reach(s) \cap Sat(\Phi) \neq \emptyset$ for all $s \in Reach(s_{init})$. Thus, for verifying "qualitative progress properties" as explained above an analysis of the "topology" of the system suffices. This result was first established in [22].

Example 4.6 In Example 2.2 we have $S^+(tt, ok) = S$. Thus, $s \models_{fair} [A \Diamond ok]_{\geq 1}$ for all states s. ■

Remark 4.7 Our methods can be modified to handle fairness à la Vardi [38] which supposes the state space to be divided into a set of states where fairness is required and a set of possibly unfair states. Since Vardi's notion of fairness subsumes ours as a special case, Theorem 2 also holds for Vardi's notion. To generalise Theorem 3 to the more general setting a modification to the definition of admissible adversaries w.r.t. (Φ_1, Φ_2) is required; see the full paper [9] for more details. ■

Complexity of model checking: The size of the parse tree (the number of nodes) is linear in the length $|\Phi|$. Let n be the set of states and m the number of "transitions" (i.e. $m = \sum_s |Steps(s)|$) in the underlying $PBTL$-structure. For every node v of the parse tree where the associated formula Φ is *tt*, an atomic proposition or of the form $\neg \Phi'$ or $\Phi_1 \wedge \Phi_2$, the costs for computing the set of states fulfilling Φ is $\mathcal{O}(n)$. The nodes which represent formulas whose outermost operator is the next-step operator require $\mathcal{O}(n \cdot m)$ time, since for every "transition" $\mu \in Steps(s)$ we have to compute the sum $\sum \mu(t)$. Theorems 2 and 3 show that in order to compute the sets of states fulfilling formulas of the form $[\ \Phi_1\ E\mathcal{U}\ \Phi_2\ ]_{\sqsupseteq p}$, resp. $[\ \Phi_1\ A\mathcal{U}\ \Phi_2\ ]_{\sqsupseteq p}$, one has to calculate the probabilities $p^{max}_s(\Phi_1 \mathcal{U} \Phi_2)$, resp. $p^{adm}_s(\Phi_1 \mathcal{U} \Phi_2)$. As in [11, 15], $p^{max}_s(\Phi_1 \mathcal{U} \Phi_2)$ can be computed by solving a linear programming problem by means of well-known methods. The algorithms with ellipsoid methods have time complexity which is polynomial in n and m.

Computation of $S^+(\Phi_1, \Phi_2)$: In time $\mathcal{O}(n \cdot m)$ we compute the directed graph $G = (S, E)$ where $(s, t) \in E$ iff $t \models \Phi_1 \wedge \neg \Phi_2$ and $\mu(s) > 0$ for some $\mu \in Steps(t)$. Then $S^+(\Phi_1, \Phi_2)$ is the set of states which are reachable in G from a state $s \in Sat(\Phi_2)$, and so $S^+(\Phi_1, \Phi_2)$ can be derived in time $\mathcal{O}(n \cdot m)$ by a depth-first search in G.

Computation of $p^{adm}_s(\Phi_1 \mathcal{U} \Phi_2)$: For each simple adversary A we calculate the set V of states $s \notin Sat(\Phi_2)$ which are reachable in G^A from a state in $Sat(\Phi_2)$ and the set W of states $s \in S \setminus V$ which are reachable in G^A from a state in $S \setminus S^+(\Phi_1, \Phi_2)$ in time $\mathcal{O}(n^2)$ by a depth-first search in the directed graph $G^A = (S, E)$ where $(s, t) \in E$ if and only if $t \in S^+(\Phi_1, \Phi_2) \setminus Sat(\Phi_2)$ and $A(t)(s) > 0$. Then, $U = V \cup W$ is the set of states $u \in S$ where for at least one state $t \in Reach^A_{\Phi_1 \wedge \neg \Phi_2}(u)$ either Φ_2 holds in t or t belongs to $S \setminus S^+(\Phi_1, \Phi_2)$. A is admissible if and only if $S^+(\Phi_1, \Phi_2) \subseteq U$. For an admissible adversary A, we apply the method described in [14, 20] and compute $p^A_s(\Phi_1 \mathcal{U} \Phi_2)$, $s \in V$, by solving the linear equation system $(\mathbf{I} - \mathbf{A}) \cdot \mathbf{x} = \mathbf{b}$. Here, $\mathbf{I}$ is the $|V| \times |V|$-identity matrix, $\mathbf{A} = (A(s, v))_{s,v \in V}$, $\mathbf{b} = (b^A_s)_{s \in V}$, $b^A_s = \sum_{t \in Sat(\Phi_2)} A(s, t)$. Solving a linear equation system with at most n variables and n equations can be done in time $\mathcal{O}(n^{2.8})$ [6]. Thus, the computation of $p^{adm}_s(\Phi_1 \mathcal{U} \Phi_2)$ takes $\mathcal{O}\left(|\mathcal{A}_{adm}(\Phi_1, \Phi_2)| \cdot n^{2.8}\right)$ time.

Given an upper bound l for the number of "transitions" in every state (i.e. $|Steps(s)| \leq l$ for all s) we obtain $|\mathcal{A}_{adm}(\Phi_1, \Phi_2)| \leq l^n$. Hence, worst-case time complexity is exponential in the number of states and linear in the size of the formula. The space complexity is $\mathcal{O}(|\Phi| \cdot n + n \cdot m)$.

[8] Note that for all adversaries A that are admissible for (Φ_1, Φ_2) and all $t \in Reach^A_{\Phi_1 \wedge \neg \Phi_2}(s)$: $Reach^A_{\Phi_1 \wedge \neg \Phi_2}(t) \cap Sat(\Phi_2) \neq \emptyset$. This yields $p^A_s(\Phi_1 \mathcal{U} \Phi_2) = 1$. Hence, $p^{adm}_s(\Phi_1 \mathcal{U} \Phi_2) = 1$.

5 Model checking for $PBTL^*$

Similarly to the way in which CTL^* extends CTL, our logic $PBTL$ can be extended to a much richer logic $PBTL^*$ which allows more complex path formulas, i.e. arbitrary combinations of path formulas by the boolean connectives and the operators X and $\mathcal{U}$. This logic (more precisely, the logic $pCTL^*$ which essentially coincides with $PBTL^*$) was already considered in [11]. While [11] deals with a satisfaction relation that does not involve aspects of fairness, we interpret $PBTL^*$ over fair computations. [11] presents a model checking algorithm for $pCTL^*$ which uses the model checker for $pCTL$. We briefly explain how the model checking procedure of [11] can be modified, thus yielding a model checking algorithm for $PBTL^*$ when interpreted by means of a satisfaction relation that involves fairness. $PBTL^*$ consists of *state formulas* and *path formulas* which are recursively defined:

- *tt* and all atomic propositions are state formulas.
- If Φ, Φ_1, Φ_2 are state formulas then $\neg\Phi$ and $\Phi_1 \wedge \Phi_2$ are state formulas.
- If φ is a path formula and $p \in [0,1]$ then $[\ E\varphi\ ]_{\sqsupseteq p}$ and $[\ A\varphi\ ]_{\sqsupseteq p}$ are state formulas.
- If Φ is a state formula then Φ is a path formula.
- If $\varphi, \varphi_1, \varphi_2$ are path formulas then $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $X\varphi$ and $\varphi_1 \mathcal{U} \varphi_2$ are path formulas.

Identifying e.g. $[A(\Phi_1 \mathcal{U} \Phi_2)]_{\sqsupseteq p}$ with the $PBTL$ formula $[\Phi_1 A \mathcal{U} \Phi_2]_{\sqsupseteq p}$, $PBTL$ can be viewed as a sublogic of $PBTL^*$. As usual, $\Diamond\varphi = tt\,\mathcal{U}\,\varphi$ and $\Box\varphi = \neg\Diamond\neg\varphi$. Given a $PBTL$-structure $M = (\mathcal{C}, I)$, the satisfaction relations $\models$ and $\models_{fair}$ are defined in the obvious way. For instance, $s \models_{fair} [\ A(\ X(\Phi_1 \mathcal{U} \Phi_2)\ )\ ]_{\sqsupseteq p}$ iff

$$\mathcal{P}\{\pi \in Path^A(s) : \pi \models_{fair} X(\Phi_1 \mathcal{U} \Phi_2)\} \sqsupseteq p$$

for all $A \in \mathcal{A}_{fair}$ where $\pi \models_{fair} X(\Phi_1 \mathcal{U} \Phi_2)$ iff there is some $j \geq 1$ with $\pi(j) \models_{fair} \Phi_2$ and $\pi(i) \models_{fair} \Phi_1$ for all $1 \leq i < j$.

Before we explain how the model checker of [11] applied to $PBTL^*$ w.r.t. $\models$ can be modified, thus yielding a model checking procedure w.r.t. $\models_{fair}$, we give an example which shows that – in contrast to our results of Section 4 – an analysis of the simple adversaries is not sufficient when using $PBTL^*$.

Example 5.1 Consider the system in Example 2.2 and the path formula $\varphi = (init \wedge Xinit) \rightarrow \Box init$. Then, $p^{A_*}_{s_{init}}(\varphi) = 1$ for A_* equal to either of the simple adversaries A_{send} and A_{init}. On the other hand, $p^{F}_{s_{init}}(\varphi) = 0$ for each fair adversary F with $F(s_{init}) = \mu_{init}$. ■

The model checker of [11] when applied to $PBTL^*$ w.r.t. $\models$ successively computes the set of states satisfying a state subformula of the given state formula Φ. Since $s \models_* [\ E\varphi\ ]_{\sqsupseteq p}$ iff $s \models_* \neg[\ A(\neg\varphi)\ ]_{\overline{\sqsupseteq} 1-p}$ only the case where Φ has the form $\Phi = [A\varphi]_{\sqsupseteq p}$ is of interest. The state subformulas occurring in Φ can be viewed as atomic propositions. We adapt the first two steps of [11]. In the first step, the formula φ is transformed into a canonical form

$$\psi \;=\; \bigvee_{i=1}^{l} \Diamond\Box(\chi_i \wedge \Diamond\kappa_i)$$

where χ_i, κ_i are *past formulas*, i.e. linear time formulas that are built from atomic propositions (including those that represent the state subformulas of Φ), boolean connectives and the past operators "since" and "previous"[9]. In the second step, a new $PBTL$-structure M' is constructed whose states keep track of the truth-values of the past formulas χ_i and κ_i. When interpreted over M', the past formulas χ_i and κ_i can be viewed as atomic propositions. Moreover, there is a one-to-one mapping $S \rightarrow S'$, $s \mapsto s'$, and a bijection $\mathcal{A}(\mathcal{C}) \rightarrow \mathcal{A}(\mathcal{C}')$, $A \mapsto A'$, such that $p^A_s(\psi) = p^{A'}_{s'}(\psi)$ for each adversary $A \in \mathcal{A}(\mathcal{C})$ and state $s \in S$. Here $p^A_s(\psi) = P\{\pi \in Path^A(s) : \pi \models \psi\}$ (cf. Lemma 12 and 13 in [11]). In the third step, we have to modify the procedure of [11]. It is easy to see that an adversary A of $\mathcal{C}$ is fair if and only if the induced adversary A' of $\mathcal{C}'$ is fair. Hence, $s \models_{fair} [\ A\varphi\ ]_{\sqsupseteq p}$ iff $s \models_{fair} [\ A\psi\ ]_{\sqsupseteq p}$ iff $s' \models_{fair} [\ A\psi\ ]_{\sqsupseteq p}$. Thus, we need a method for computing the set of states fulfilling a formula of the form $[\ A\psi\ ]_{\sqsupseteq p}$, where $\psi = \bigvee_{i=1}^{l} \Diamond\Box(a_i \vee \Diamond b_i)$ with atomic propositions $a_1, \ldots, a_l, b_1, \ldots, b_l$.

Lemma 5.2 *Let $M = (\mathcal{C}, I)$ be a $PBTL$-structure, $\mathcal{C} = (S, Steps)$, and ψ the path formula $\bigvee_{i=1}^{l} \Diamond\Box(a_i \vee \Diamond b_i)$ with atomic propositions $a_1, \ldots, a_l, b_1, \ldots, b_l$. Let U_i be the largest subset of $Sat(a_i)$ with $Reach(u) \subseteq U_i$ and $Reach(u) \cap Sat(b_i) \neq \emptyset$ for all $u \in U_i$[10]. Let*

[9] In this step the result of [30] stating that each linear time formula is equivalent to a "canonical reactivity formula" is used.

[10] Here, $Reach(u) = Reach_{tt \wedge \neg ff}(u)$, i.e. $Reach(u)$ is the set of states that are reachable from u via a finite path in $\mathcal{C}$.

$U = \bigcup_{1 \le i \le l} U_i$ and let a_U be an atomic proposition with $a_U \in I(s)$ iff $s \in U$. Then for all $s \in S$:

$$s \models_{fair} [A\psi]_{\sqsupseteq p} \text{ iff } s \models_{fair} [A \Diamond a_U]_{\sqsupseteq p}$$

We apply Lemma 5.2 to M' (which is constructed in the second step), rather than M, and first compute the set U (in time linear in the size of M' by an analysis of the strongly connected components of the underlying directed graph of M') and then compute the set of states satisfying $[\ A \Diamond a_U\ ]_{\sqsupseteq p}$ with the methods of Section 4. Of course, in general, a_U is not an atomic proposition, but if the set U is computed we can introduce a_U as a "new" atomic proposition and extend $I'(s)$ by a_U iff $s \in U$.

6 Concluding remarks and further direction

The algorithm for model checking of *quantitative* probabilistic properties w.r.t. fairness constraints presented in this paper has applications in the specification and verification of concurrent probabilistic systems such as fault-tolerant systems or distributed randomized systems. In many such systems it is essential to allow nondeterminism, which arises through a scheduler or external intervention, as well as probabilistic choice. The notion of fairness w.r.t. scheduling that we use is natural and reasonable from a practical point of view. The main advantage of our approach is that through $\models_{fair}$ we can automatically verify certain (probabilistic) liveness properties (e.g. the progress property in Example 3.1) which simply do not hold if fairness is not assumed, although they are normally ensured at the level of implementation. The algorithm can be extended to cater for the requirement of only a subset of states being fair and the case of bounded until [9]; the latter has applications in the calculation of time bounds (in the sense of "soft-deadlines" as in [20] rather than real-time). Its complexity depends on the size of the formula and the size of the model, and is in line with the expected complexity of probabilistic verification [38]. It is an open question whether the time complexity to compute $p_s^{adm}(\Phi_1 \mathcal{U} \Phi_2)$ can be improved. It may be possible to reduce the calculation of $p_s^{adm}(\Phi_1 \mathcal{U} \Phi_2)$ to a linear programming problem, as is the case for the computation of $p_s^{max}(\Phi_1 \mathcal{U} \Phi_2)$ [11].

Our algorithm is based on the reachability graph analysis, and its complexity is necessarily exponential in the number of states and linear in the size of the formula. In [10] we have formulated an efficient, MTBDD-based [13], model checking procedure for the more restrictive logic of [20], which is performing well in practical applications. We intend to extend the symbolic techniques to the more general case discussed here.

References

1. B. Alpern, F. Schneider: Defining Liveness, Information Processing Letters, Vol. 21, 1985.
2. R. Alur, C. Courcoubetis, D. Dill: Model-Checking for Probabilistic Real-Time Systems, Proc. ICALP, LNCS 510, pp 115-127, 1991.
3. R. Alur, C. Courcoubetis, D. Dill: Verifying Automata Specifications of Probabilistic Real-Time Systems, Proc. REX, LNCS 600, pp 27-44, 1991.
4. L. de Alfaro: Formal Verification of Performance and Reliability of Real-Time Systems, Techn. Report, Stanford, 1996.
5. A. Aziz, V. Singhal, F. Balarin, R. Brayton, A. Sangiovanni-Vincentelli: It usually works: The Temporal Logic of Stochastic Systems, Proc. CAV, LNCS 939, pp 155-165, 1995.
6. A. Aho, J. Hopcroft, J. Ullman: The Design and Analysis of of Computer Algorithms, Addison-Wesley Publishing Company, 1974.
7. J. Aspnes, M.P. Herlihy: Fast randomized consensus using shared memory, Journal of Algorithms, Vol. 15(1):441-460, 1990.
8. C. Baier: Polynomial Time Algorithms for Testing Probabilistic Bisimulation and Simulation, Proc. CAV, LNCS 1102, pp 38-49, 1996.
9. C. Baier, M. Kwiatkowska: Model checking for a probabilistic branching time logic with fairness. Currently under revision for *Distributed Computing*. Also available as Techn. Report CSR-96-12, University of Birmingham.
10. C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, M. Ryan: Symbolic model checking for probabilistic processes. In *Proceedings, 24th ICALP* (1997), LNCS, Springer-Verlag. To appear.
11. A. Bianco, L. de Alfaro: Model Checking of Probabilistic and Nondeterministic Systems, Proc. Foundations of Software Technology and Theoretical Computer Science, LNCS 1026, pp 499-513, 1995.
12. E.M. Clarke, E.A. Emerson, A.P. Sistla: Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications: A Pratical Approach, Proc. POPL, 1983.

13. E. Clarke, M. Fujita, P. McGeer, K. McMillan, J. Yang, X. Zhao: Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation, preprint.

14. C. Courcoubetis, M. Yannakakis: Verifying Temporal Properties of Finite-State Probabilistic Programs, Proc. FOCS, pp 338-345, 1988.

15. C. Courcoubetis, M. Yannakakis: Markov Decision Processes and Regular Events, Proc. ICALP, LNCS 443, pp 336-349, 1990.

16. E.A. Emerson, E.M. Clarke: Using Branching Time Logic to Synthesize Synchronization Skeletons, Sci. Comput. Programming 2, pp 241-266, 1982.

17. E.A. Emerson, C. Lei: Modalities for Model Checking: Branching Time Strikes Back, Proc. POPL, 1985.

18. P. Halmos: Measure Theory, Springer-Verlag, 1950.

19. H. Hansson: Time and Probability in Formal Design of Distributed Systems, Real-Time Safety Critical Systems, Elsevier, 1994.

20. H. Hansson, B. Jonsson: A Logic for Reasoning about Time and Probability, Formal Aspects of Computing, Vol. 6, pp 512-535, 1994.

21. S. Hart, M. Sharir: Probabilistic Temporal Logic for Finite and Bounded Models, Proc. 16th ACM Symposium on Theory of Computing, pp 1-13, 1984.

22. S. Hart, M. Sharir, A. Pnueli: Termination of Probabilistic Concurrent Programs, ACM Transactions on Programming Languages and Systems, Vol. 5, pp 356-380, 1983.

23. T. Huynh, L. Tian: On some Equivalence Relations for Probabilistic Processes, Fundamenta Informaticae, Vol. 17, pp 211-234, 1992.

24. P. Iyer, M. Narasima: "Almost always" and "definitely sometime" are not enough: Probabilistic quantifiers and Probabilistic model-checking, Techn. Report TR-96-16, North Carolina State University, 1996.

25. B. Jonsson, K.G. Larsen: Specification and Refinement of Probabilistic Processes, Proc. LICS, 1991.

26. B. Jonsson, W. Yi: Compositional Testing Preorders for Probabilistic Processes, Proc. LICS, pp 431-443, 1995.

27. E. Kushilevitz, M. Rabin: Randomized mutual exclusion algorithms revisited, Proc. PODC, ACM Press 1992.

28. D. Lehmann, S. Shelah: Reasoning with Time and Chance, Information and Control, Vol. 53, pp 165-198, 1982.

29. N. Lynch, I. Saias, R. Segala: Proving Time Bounds for Randomized Distributed Algorithms, Proc. PODC, pp 314-323, 1994.

30. Z. Manna, A. Pnueli: A Hierarchy of Temporal Properties, Proc. PODC, 1990.

31. A. Pnueli: On the Extremely Fair Treatment of Probabilistic Algorithms, Proc. 15th ACM Symposium on Theory of Computing, 1983.

32. A. Pnueli, L. Zuck: Verification of Multiprocess Probabilistic Protocols, Distributed Computing, Vol. 1, No. 1, pp 53-72, 1986.

33. A. Pnueli, L. Zuck: Probabilistic Verification, Information and Computation, Vol. 103, pp 1-29, 1993.

34. A. Pogosyants, R. Segala: Formal Verification of Timed Properties of Randomized Distributed Algorithms, Proc. PODC, 1995.

35. R. Segala, N. Lynch: Probabilistic Simulations for Probabilistic Processes, Proc. CONCUR, LNCS 836, pp 481-496, 1994.

36. R. Segala: Modelling and Verification of Randomized Distributed Real-Time Systems, Ph.D. Thesis. Massachusetts Institute of Technology, 1995.

37. R. van Glabbeek, S. Smolka, B. Steffen, C. Tofts: Reactive, Generative, and Stratified Models for Probabilistic Processes, Proc. LICS, pp 130-141, 1990.

38. M. Vardi: Automatic Verification of Probabilistic Concurrent Finite-State Programs, Proc. FOCS, pp 327-338, 1985.

39. S. Yuen, R. Cleaveland, Z. Dayar, S. Smolka: Fully Abstract Characterizations of Testing Preorders for Probabilistic Processes, Proc. CONCUR, LNCS 836, pp 497-512, 1994.