The Complexity of Probabilistic Verification

COSTAS COURCOUBETIS

University of Crete, and ICS, Farth, Heraklion, Greece

AND

MIHALIS YANNAKAKIS

AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. We determine the complexity of testing whether a finite state, sequential or concurrent probabilistic program satisfies its specification expressed in linear-time temporal logic. For sequential programs, we present an algorithm that runs in time linear in the program and exponential in the specification, and also show that the problem is in PSPACE, matching the known lower bound. For concurrent programs, we show that the problem can be solved in time polynomial in the program and doubly exponential in the specification, and prove that it is complete for double exponential time. We also address these questions for specifications described by ω-automata or formulas in extended temporal logic.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Program Verification; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems–*complexity of proof procedures*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs–*mechanical verification*; G.3 [**Discrete Mathematics**]: Probability and Statistics–*probabilistic algorithms*

General Terms: Algorithms, Theory, Verification

Additional Key Words and Phrases: Automata, EXPTIME-complete, Markov chain, model checking, probabilistic algorithm, PSPACE-complete, temporal logic

1. *Introduction*

It has been realized that randomization may lead to algorithms with better complexity than deterministic ones, or even allow for the solution of problems (especially in distributed computation) that cannot be solved deterministically. Verifying properties of these algorithms is in general harder than for the deterministic ones since the effects of randomization are sometimes nonintuitive and difficult to grasp. These observations provide the motivation for the development of formal methods and tools for the verification of probabilistic

Authors' addresses: C. Courcoubetis, University of Crete, Heraklion, Greece; M. Yannakakis, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974-2070, e-mail: mihalis@research.att.com

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.
© 1995 ACM 0004-5411/95/0700-0857 $03.50

programs, and explain the flurry of activity in the last several years in the area of probabilistic verification.

In traditional program verification, one must first define the *formal specification* of the program. A simple language for specifying temporal requirements on the computations of programs is *temporal logic* introduced by Pnueli [1981]. There are two types of temporal logic: linear and branching time. In this paper, we will be concerned with linear-time temporal logic. To verify a program, one specifies the desired properties of the program by a formula in this logic. The program is correct if all its computations satisfy the formula.

In its generality, an algorithmic solution to the verification problem is hopeless. Things become more manageable for finite-state programs, in which the variables range over finite domains [Clarke et al. 1983; Queille and Sifakis 1982]. There is a number of communication and synchronization protocols that are in this category. In this case, state properties can be described by atomic propositions, and the specification can be written as a formula in *propositional* temporal logic. Checking that the program is a model of its specification can be done algorithmically.

A number of papers in recent years examine the extension of model checking to probabilistic programs. In the context of probabilistic programs, the notion of correctness needs also to become probabilistic: a program is correct if *almost all* computations satisfy the specification, that is, the specification is satisfied with probability one [Hart and Sharir 1984; Lehman and Shelah 1982]. Concurrency introduces a degree of nondeterminism, for example, due to the asynchronous execution of processes. In this case, the program is correct if it meets the specifications with probability 1 even under a worst-case scenario [Vardi 1985; Pnueli and Zuck 1986] (we define formally the model in the next section).

In this paper, we show the following results:

(1) Testing if a (finite-state) sequential probabilistic program satisfies its linear temporal logic specification can be done in time exponential in the size of the specification and linear in the size of the program; also in space polynomial in the specification and polylogarithmic in the program. The exact probability that the program satisfies the specification can be computed in time exponential in the specification and polynomial in the program.

(2) Testing if a concurrent probabilistic program satisfies its specification is hard for doubly exponential time, and can be solved in time doubly exponential in the specification and quadratic in the size of the program.

We note that as a rule, programs tend to be large, and specifications tend to be short. The previously known lower bound for Problem 1 (the sequential case) was PSPACE-hardness [Vardi 1985], and for Problem 2 (the concurrent case) was EXPTIME-hardness [Vardi and Wolper 1986]. Thus, Problem 1 is PSPACE-complete as in the nonprobabilistic case [Clarke et al. 1983]. The previous upper bound in the general case was triply exponential time for both problems (double exponential space for Problem 1) [Vardi 1985]. (This reference contained also a "shortcut" that would lead to a double exponential algorithm; however, the author found subsequently an error in this part [M. Vardi, private communication].) For a restriction of the logic, Pnueli and Zuck [1986] and Vardi and Wolper [1986] give different algorithms that run in single exponential time (for both problems). The restricted logic has the same expressive power as the standard temporal logic; the problem is that the known methods for translating from the full to the restricted logic are nonelementary.

There are two basic approaches to model checking: the tableau-based [Clarke et al. 1983; Lichtenstein and Pnueli 1985] and the automata-theoretic [Wolper et al. 1983]. For the sequential case, we use a different and very intuitive technique: We modify the probabilistic program in a step-by-step manner and at the same time simplify the temporal formula, until it becomes a simple propositional formula, at which point the verification problem becomes trivial. This technique also allows us to perform a quantitative analysis of sequential programs, that is, compute the exact probability of satisfaction.

For the concurrent case, we use the automata-theoretic approach. From a formula, one can construct an exponentially larger ω-automaton on infinite words (a so-called Buchi automaton [Buchi 1962]), which accepts the computations that satisfy the formula. Such automata are strictly more powerful than linear-time temporal logic and can also express significant extensions of it [Wolper 1983], at a higher, but still single exponential cost. From a formula f, one can construct an automaton for either f (i.e., accepting the good computations) or for $\neg f$ (accepting the bad computations) at the same cost. It is more convenient to work with the latter automaton, and solve the *probabilistic emptiness* problem: Decide whether the set of computations of the program that are accepted by the automaton has probability 0.

The difficulty of the problem for automata stems from their nondeterminism. If the automaton is deterministic, or even "almost" deterministic (in a well-defined sense, see Section 4.2), the probabilistic emptiness problem can be solved in polynomial time [Vardi and Wolper 1986]. Determinizing ω-automata is highly nontrivial. There have been several constructions and proofs over the last 20 years,[1] which tend to be very complex and increase the size by two or three exponentials. Furthermore, deterministic Buchi automata are strictly less powerful than nondeterministic, so that one has to go to a more general type of ω-automaton. We show that "almost" deterministic Buchi automata (in the above sense) are equivalent to nondeterministic, and provide a simple construction of complexity $2^{O(n)}$. Essentially the same construction has been obtained independently by Safra [1988], where he also presents a *full* determinization construction to a Rabin-type ω-automaton with complexity $2^{O(n \log n)}$ [Safra 1988]. (The "almost" deterministic Buchi automaton suffices for our purpose.)

Finally, we generalize our results to the extended temporal logics introduced in Wolper [1983] and Wolper et al. [1983]. These logics use automata as temporal connectives. They combine the expressive power of automata with the succinctness of the standard temporal logic.

The rest of this paper is organized as follows: In Section 2, we give the necessary definitions (temporal logic, automata, and the model of probabilistic programs). Section 3 examines the verification problem for (the standard) temporal logic. We give an algorithm for the sequential case, and prove the lower bound for the concurrent case. In Section 4, we study the problem for ω-automata. For sequential probabilistic problems, we show how to solve the probabilistic emptiness problem, and also how to perform quantitative analysis

[1]For example, see McNaughton [1986], Buchi [1973], Choueka [1974], Rabin [1972], and Vardi [1985].

(compute the exact probability of satisfaction). For the concurrent case, we describe the semi-determinization construction reducing the emptiness problem to the "almost" deterministic case. In Section 5, we combine and extend all the techniques to generalize the results to the Extended Temporal Logic. Finally, in Section 6, we summarize our results and reflect on the different proof techniques that we used to obtain them.

2. *Background*

2.1. TEMPORAL LOGIC. We review the basic definitions of *linear time propositional temporal logic* (PTL) as defined in Gabby et al. [1980] and Pnueli [1981]. Formulas in PTL are built from a set *Prop* of atomic propositions using Boolean connectives, the unary temporal connective X (next), and the binary temporal connective U (until). The formulas in PTL are interpreted over *computations*. Informally, a computation is an infinite sequence of states, where in each state every atomic proposition has a truth value, that is, is true or false. (In this context, finite computations are viewed as staying in their final state forever.) Formally, a computation is a function $\pi: \mathbf{N} \rightarrow 2^{Prop}$ from the set of natural numbers (denoted by a boldface $\mathbf{N}$) to the set of truth assignments for the set *Prop* of propositions; at each time instant (natural number) $i \in \mathbf{N}$ a proposition $p \in Prop$ is true in the computation π iff $p \in \pi(i)$. For a computation π and time instant $i \in \mathbf{N}$, we define the satisfaction of a formula f by π at instant i (denoted $\pi, i \models f$) inductively on the structure of f as follows:

—$\pi, i \models p$ for $p \in Prop$ iff $p \in \pi(i)$.
—$\pi, i \models \xi \& \psi$ iff $\pi, i \models \xi$ and $\pi, i \models \psi$.
—$\pi, i \models \neg \phi$ iff not $\pi, i \models \phi$.
—$\pi, i \models X\phi$ iff $\pi, i + 1 \models \phi$.
—$\pi, i \models \xi U \psi$ iff for some $j \geq i$, $\pi j \models \psi$ and for all k, $i \leq k < j$ $\pi, k \models \xi$.

We say that a computation π *satisfies* a formula ϕ, denoted $\pi \models \phi$, iff $\pi, 0 \models \phi$.

A computation π can be viewed as an infinite word over the alphabet $\Sigma = 2^{Prop}$. Each temporal logic formula ϕ viewed as an acceptor of infinite words, defines a language $L_\omega(\phi)$ over $\Sigma = 2^{Prop}$; namely, $L_\omega(\phi)$ is the set of all computations π that satisfy ϕ.

2.2. AUTOMATA ON INFINITE WORDS. Let Σ be a (finite) alphabet. The set of finite words over Σ is denoted as Σ^*, and the set of infinite words as Σ^ω. A (*transition*) *table* is a tuple $\tau = (\Sigma, S, \rho)$, where Σ is the alphabet, S is a finite set of states, and $\rho: S \times \Sigma \rightarrow 2^S$ is the transition function. A state is *deterministic* if $|\rho(s, a)| \leq 1$ for all letters $a \in \Sigma$; the table τ is deterministic if all the states are. We can think of a table τ as a directed graph, which has S as its set of nodes and whose arcs are labeled by letters of the alphabet Σ; the graph has an arc from a node s to a node t labeled by the letter $a \in \Sigma$ iff $t \in \rho(s, a)$.

A *run* of the table τ over a finite word $w = a_1 \cdots a_n$ over Σ is a sequence of states $\mathbf{s} = s_0, \ldots, s_n$ such that $s_i \in \rho(s_{i-1}, a_i)$ for $1 \leq i \leq n$. The transition function ρ can be extended to Σ^* in the standard way: if $w \in \Sigma^*$ is a finite word and s, t are two states, then $t \in \rho(s, w)$ iff there is a run over w that starts at state s and ends in state t. The transition function can be also extended as usual to sets of states: if Q is a set of states and $w \in \Sigma^*$ a finite word, then $\rho(Q, w) = \bigcup_{s \in Q} \rho(s, w)$. A run of a table τ over an infinite word $w = a_1 a_2 \cdots$ is an infinite sequence of states $\mathbf{s} = s_0, s_1, \ldots$ such that $s_i \in \rho(s_{i-1}, a_i)$ for $i \geq 1$. For an infinite run $\mathbf{s}$, the set *inf*($\mathbf{s}$) is the set of states that occur infinitely often in $\mathbf{s}$, that is, *inf*($\mathbf{s}$) $= \{v \mid |\{i : s_i = v\}| = \infty\}$.

For a table $\tau = (\Sigma, S, \rho)$, we denote by $det(\tau) = (\Sigma, 2^S, \rho')$ the deterministic table that results from applying the usual subset construction to τ; table $det(\tau)$ has one state for every subset $Q \subseteq S$ of states of τ, and its transition function ρ' maps a set Q and a letter $a \in \Sigma$ to $\{\rho(Q, a)\}$ where $\rho(Q, a) = \bigcup_{s \in Q} \rho(s, a)$. For a state s, we denote by $det(\tau, s)$ the restriction of $det(\tau)$ to the states that are reachable from $\{s\}$; that is, $det(\tau, s)$ includes only those subsets $Q \subseteq S$ for which there is a word $w \in \Sigma^*$ such that $Q = \rho(s, w)$.

An ω-automaton A consists of a table $\tau_A = (\Sigma, S, \rho)$, a set of starting states $S_0 \subseteq S$, and an acceptance condition. The automaton is deterministic if τ is deterministic and $|S_0| = 1$. In the usual automata on finite strings, acceptance of a run is determined by the final state of the run; in the case of ω-automata and infinite runs $\mathbf{s}$, acceptance is determined by its infinity set *inf*($\mathbf{s}$). There are several ways in which one can specify which infinity sets are accepting and which ones are not. The simplest one is the Buchi acceptance condition. A Buchi automaton $A = (\tau_A, S_0, F)$, $F \subseteq S$, is an ω-automaton with the following acceptance condition. Automaton A accepts an infinite word w if there is a run $\mathbf{s}$ of τ_A over w such that $\mathbf{s}$ starts with a state in S_0 and repeats some state of F infinitely often, that is, *inf*($\mathbf{s}$) $\cap F \neq \emptyset$. We define as $L_\omega(A)$ the set of infinite words accepted by the ω-automaton A. Nondeterministic Buchi automata accept exactly the ω-regular languages; deterministic Buchi automata accept a proper subset.

There is an important result relating Buchi automata and temporal logic formulas viewed as language acceptors. In Emerson and Sistla [1983] and Wolper et al. [1983], it is proved that given a PTL formula ϕ of length n over the set of propositions *Prop*, one can build a Buchi automaton A_ϕ on the alphabet $\Sigma = 2^{Prop}$, such that $L_\omega(A_\phi)$ is precisely the set $L_\omega(\phi)$, and A_ϕ has at most $2^{O(n)}$ states.

2.3. Probabilistic Programs. Our model of a sequential probabilistic program is a finite state Markov chain as in Hart and Sharir [1984], Lehman and Shelah [1982], and Vardi [1985]. A *Markov chain* $M = (X, A, \{p_{vw} | (v, w) \in A\}, \{p_0(v) | v \in X\})$ consists of a set X of *states*; a set $A \subseteq X \times X$ of *transitions* (arcs); an assignment of positive *transition probabilities* p_{vw} to all transitions (v, w), so that for each state v, the sum $\sum_{w \in X} p_{vw}$ is equal to 1 (the pairs (v, w) that are not in A are considered to have their corresponding transition probabilities p_{vw} equal to 0); an *initial probability distribution* p_0 on the states such that $\sum_{v \in X} p_0(v) = 1$. We will often view a Markov chain M as a directed graph (X, A) whose arcs are labeled by the transition probabilities. An infinite sequence of states $\mathbf{X} = X_0, X_1, \ldots$ is called a *trajectory* of the Markov chain. Typically, we will use a boldface capital letter such as $\mathbf{X}, \mathbf{Y}$ to denote a trajectory, and use the corresponding subscripted roman letter X_i, Y_i to denote the ith element of the trajectory.

A Markov chain M induces a stochastic process $\{X_n, n = 0, 1, \ldots\}$ taking values from the set X of states of M, such that the distribution at time $n = 0$

(i.e., the distribution of the random variable X_0) is given by p_0, and the distribution of X_n for any $n > 0$ conditioned on the entire previous history $X_0, \dots, X_{n-1}$ is equal to the distribution conditioned on the previous state X_{n-1} (this is the so-called *Markov property*) and is given by the transition probabilities; that is, for any $v, w \in X$, the probability that $X_n = w$ given that $X_{n-1} = v$ is p_{vw}. More formally, there is a probability space $(X^\omega, \boldsymbol{F}, P_M)$ defined on the set X^ω of trajectories. The family of measurable sets $\boldsymbol{F}$ is the Borel field generated by the *basic cylinder sets* $C(x)$, $x \in X^*$, where $C(x)$ is the set of trajectories $\mathbf{X}$ with prefix x. The measure P_M is defined on the basic cylinder sets (and can then be extended uniquely to the rest of $\boldsymbol{F}$) as follows: $P_M(C(x_0x_1 \cdots x_n)) = p_0(x_0)p_{x_0x_1} \cdots p_{x_{n-1}x_n}$. For more background on probability theory and Markov chains, we refer the reader to Breiman [1968] and Kemeny et al. [1976].

Let Σ be the alphabet of a specification. In the case of a temporal logic specification (a PTL formula) f over a set *Prop* of atomic propositions, Σ is the family 2^{Prop} of all subsets of *Prop*; in the case of a specification given by an ω-automaton, Σ is the alphabet of the automaton. Our model of a sequential probabilistic program is a pair (M, V) consisting of a Markov chain M with a finite state space X and a function V: $X \to \Sigma$ that associates with every state of the chain a letter from Σ; if Σ is 2^{Prop}, this function associates with every state the set of propositions that are true in that state. We call V the *valuation* function. If $\mathbf{X} = X_0, X_1, \dots$ is a trajectory of M, then its *valuation* $V(\mathbf{X})$ is the sequence $V(X_0), V(X_1), \dots,$ an infinite word over Σ. We will say that $\mathbf{X}$ satisfies a specification given by a PTL formula f (or automaton A) if its valuation $V(\mathbf{X})$ is in the language $L_\omega(f)$ (respectively, $L_\omega(A)$) of the specification, as defined in Subsections 2.1 and 2.2. In Vardi [1985], it is shown that for any PTL formula f and ω-automaton A, the sets of trajectories $\{\mathbf{X} \in X^\omega | V(\mathbf{X}) \in L_\omega(f)\}$ and $\{\mathbf{X} \in X^\omega | V(\mathbf{X}) \in L_\omega(A)\}$ are measurable. We will use $P_M(L_\omega(f))$ and $P_M(L_\omega(A))$ to denote respectively the measures of these two sets. That is, $P_M(L_\omega(f))$ (respectively, $P_M(L_\omega(A))$) denotes the probability that a trajectory of the Markov chain M satisfies the specification f (respectively, A).

A concurrent program is modeled by a finite state concurrent Markov chain as in Vardi [1985]. Informally, this is a Markov chain augmented with nondeterministic states. When the chain is at a randomizing state, a transition is chosen randomly according to a probability transition matrix; when the chain is at a nondeterministic state, a transition is chosen by a *scheduler*, possibly depending on the history of the chain. Formally, a *concurrent Markov chain* $M = (N, R, A, \{p_{vw} | v \in R, (v, w) \in A\}, \{p_0(v) | v \in X\})$ consists of a finite set X of states, which is partitioned into a set N of *nondeterministic* and a set R of *randomising states*; a set $A \subseteq X \times X$ of transitions (arcs); an assignment of positive probabilities to transitions coming out of randomizing states, so that the probabilities on the transitions coming out of each randomizing state $v \in R$ sum to 1; an initial probability distribution p_0 on the states, again summing to 1. A *scheduler* is a function u: $X^* \times N \to X$ such that for all $\sigma \in X^*$ and $v \in N$, the image $u(\sigma, v) = w$ has the property that $(v, w) \in A$. For each scheduler u, we can define a probability space $(X^\omega, \boldsymbol{F}, P_{M,u})$ on the set of trajectories X^ω of the concurrent Markov chain M as follows: The set $\boldsymbol{F}$ is again the Borel field generated by the basic cylinder sets $C(x)$, $x \in X^*$ (it does not depend on the scheduler). The measure $P_{M,u}$ is defined on the basic cylinder sets (and can then be extended uniquely to the rest of $\boldsymbol{F}$) as follows:

$P_{M,u}(C(x_0x_1 \cdots x_n)) = p_0(x_0) \cdot q_1 \cdots q_n$, where $q_i = p_{x_{i-1}x_i}$ if $x_{i-1} \in R$, $q_i = 1$ if $x_{i-1} \in N$ and $u(x_0x_1 \cdots x_{i-2}, x_{i-1}) = x_i$, and $q_i = 0$; otherwise, (if $x_{i-1} \in N$ and $u(x_0x_1 \cdots x_{i-2}, x_{i-1}) \neq x_i$). Clearly, if the set N of nondeterministic states is empty, then a concurrent Markov chain reduces to an ordinary Markov chain.

As above, let Σ be the alphabet of a specification. Our model of a concurrent probabilistic program is a pair (M, V) consisting of a concurrent Markov chain M with a finite state space X and a valuation function V: $X \to \Sigma$ that associates with every state of M a letter from Σ. For any scheduler u and any PTL formula f or Buchi automaton A, the sets of trajectories $\{\mathbf{X} \in X^{\omega} | V(\mathbf{X}) \in L_{\omega}(f)\}$ and $\{\mathbf{X} \in X^{\omega} | V(\mathbf{X}) \in L_{\omega}(A)\}$ are again measurable [Vardi 1985]. We use $P_{M,u}(L_{\omega}(f))$ (respectively, $P_{M,u}(L_{\omega}(A))$) to denote the probability that a trajectory of M under the scheduler u satisfies the specification f (respectively, A).

We say that a sequential probabilistic program satisfies a formula f if $P_M(L_{\omega}(f)) = 1$. A concurrent probabilistic program satisfies f if for all schedulers u, $P_{M,u}(L_{\omega}(f)) = 1$. We have the analogous definitions for automata specifications.

2.4. Graph-Theoretic Terminology. We will view automata and Markov chains as directed graphs with labeled edges. We assume familiarity with basic graph theoretic techniques and algorithms. We mention here some of the basic terms that we will be using.

A directed graph consists of a set of nodes and a set of arcs (or edges). If $u \to v$ is an arc from u to v, then we say that v is an *immediate successor* of u, and u is an *immediate predecessor* of v. If there is a path from node u to node v, then we say that u can reach v, that u is an *ancestor* of v, and that v is a *descendant* of u. A graph is *strongly connected* if every node can reach every other node.

A *strongly connected component* (or s.c.c. for short) of a graph is a maximal subgraph that is strongly connected. The nodes of a graph can be partitioned into strongly connected components. Depth-first-search is a classic algorithm for finding the strongly connected components of a graph with time complexity linear in the number of nodes and edges of the graph (see, e.g., Aho et al. [1974]). An s.c.c. is *trivial* if it consists of a single node and contains no edge; otherwise, it is *nontrivial*. An s.c.c. C is a *bottom* s.c.c. if there is no arc coming out of C, that is, the descendants of the nodes of C are themselves in C.

3. *Verifying Temporal Logic Specifications*

3.1. Sequential Probabilistic Programs. We are given a PTL formula f over a set of propositions *Prop*, and a finite state Markov chain $M = (X, A, \{p_{vw} | (v, w) \in A\}, \{p_0(v) | v \in X\})$ with a valuation function V: $X \to 2^{Prop}$ modeling a sequential probabilistic program. It is well known that for verification purposes, the exact values of the probabilities are not important. Nevertheless, it is helpful to think in quantitative terms. The algorithm transforms step-by-step the formula and the Markov chain, eliminating one by one the temporal connectives, while preserving the probability of satisfaction of the formula. There are two transformations C_U and C_X corresponding to the two temporal connectives U and X. We start by describing first C_U.

3.1.1. *Construction C_U.* Let $\phi U \psi$ be an "innermost" temporal subexpression of f; that is, a subexpression such that ϕ, ψ are composed of atomic propositions and Boolean connectives. For each state of M and atomic proposition in *Prop*, the valuation V specifies a truth value. Thus, we can evaluate ϕ and ψ on each state of M. The construction C_U produces a new Markov chain M' with valuation V' and a new formula f' as follows.

We first partition the states of the Markov chain in the three disjoint subsets $X = X^{YES} \cup X^{NO} \cup X^{?}$ according to the following rules:

(1) Assign to X^{YES} all states u satisfying ψ and to X^{NO} all states satisfying $\neg \phi$ and $\neg \psi$. Viewing the Markov chain as a graph, let H be the subgraph induced on the remaining states satisfying ϕ and $\neg \psi$.
(2) Assign to X^{NO} all states u of H such that H does not contain any path from u to a state v that has a transition $v \to w$ in M to a state w satisfying property ψ; in other words, it is impossible for u to reach a state satisfying ψ without passing first through a state that satisfies $\neg \phi$ and $\neg \psi$.
(3) Assign to X^{YES} all states u of H such that H does not contain any path from u to a state v that belongs to X^{NO} (by rule 2) or that has a transition $v \to w$ in M to a state w satisfying $\neg \phi$ and $\neg \psi$.
(4) Assign to $X^{?}$ all the remaining states of H (i.e., that were not assigned by the previous rules).

The interpretation of the above sets is that if a path starts from a state in X^{YES} (respectively, X^{NO}) then with probability one it will satisfy the formula $\phi U \psi$ (respectively, $\neg(\phi U \psi)$), and if it starts from a state in $X^{?}$ then both events have nonzero probability. The following lemma proves these properties, and also shows how one can compute the probability that $\phi U \psi$ is satisfied starting from each state u. Recall that p_{uv} denotes the probability of the transition $u \to v$.

LEMMA 3.1.1.1. *Let q_u denote the probability that $\phi U \psi$ is satisfied starting from state u. These probabilities can be computed from the following linear system of equations*:

$$\begin{aligned} q_u &= \sum_v p_{uv} q_v, && \text{if} \quad u \in X^{?}; \\ q_u &= 1, && \text{if} \quad u \in X^{YES}; \\ q_u &= 0 && \text{if} \quad u \in X^{NO}. \end{aligned}$$

This set of equations has a unique solution.

PROOF. We show first that $q_u = 0$ for all states u of X^{NO} and $q_u = 1$ for all $u \in X^{YES}$. Suppose that u was assigned to a set by rule 1. If $u \in X^{YES}$, then u satisfies ψ and therefore all trajectories starting at u satisfy $\phi U \psi$. If $u \in X^{NO}$, then u satisfies $\neg \phi$ and $\neg \psi$ and therefore all trajectories starting at u satisfy $\neg(\phi U \psi)$. Suppose that u was assigned to X^{NO} by rule (2) and consider a trajectory $\mathbf{X}$ starting at u. Either the trajectory stays forever in the subgraph H, in which case it never visits a state satisfying ψ, or it exists H at some point, in which case the first state that it visits outside H satisfies $\neg \phi$ and $\neg \psi$. In both cases, the trajectory satisfies $\neg(\phi U \psi)$.

Suppose that u was assigned to X^{YES} by rule (3) and consider a trajectory $\mathbf{X}$ starting at u. Almost surely the trajectory $\mathbf{X}$ reaches eventually a bottom strongly connected component C of the Markov chain M and visits infinitely often all the states of C. (This is a fundamental property of Markov chains that we will often use; see, for example, Kemeny et al. [1976, Propositions 4.27 and 4.28].) Suppose that $\mathbf{X}$ never exits the subgraph H. Then C must be contained in H. Since C is a bottom s.c.c. of M, no transition exits C and therefore the states of C must have been assigned to X^{NO} by rule (2), contradicting the fact that $u \in X^{YES}$ by rule (3). Thus, $\mathbf{X}$ must exit the subgraph H. Let $v \to w$ be the first transition by which $\mathbf{X}$ exits H. From rule (3), it must be the case that w satisfies ψ and therefore $\mathbf{X}$ satisfies $\phi U \psi$.

Consider now a state $u \in X^?$. Since u satisfies ϕ and $\neg \psi$, a trajectory $\mathbf{X}$ starting at u satisfies $\phi U \psi$ iff its suffix from the second state onward satisfies $\phi U \psi$. If the second state is v, then the probability of this event is q_v. Therefore, $q_u = \sum_v p_{uv} q_v$, if $u \in X^?$.

Thus, we have shown that the probabilities q_u, $u \in X$ satisfy the equations of the lemma. It remains to show that they form the unique solution. Clearly, any two solutions can only differ on $X^?$. Consider two solutions $a = (a_u)_{u \in X^?}$ and $b = (b_u)_{u \in X^?}$. Let T be the matrix of transition probabilities p_{uv} restricted to the rows and columns of $X^?$. From the linear system, we have $a - b = T \cdot (a - b)$.

Let us form a Markov chain $M^?$ whose state space consists of $X^?$ and an additional absorbing state w. The additional state w has only a transition to itself with probability 1. Every state $u \in X^?$ has the same transitions to other states $v \in X^?$ with the same probability as in the original chain M; in addition, if u has in M transitions to states outside $X^?$, then u has a transition in $M^?$ to w with probability $\sum_{v \notin X^?} p_{uv}$. The state w by itself forms one bottom s.c.c. of $M^?$. We claim that this is the only bottom s.c.c. Assume that there is another bottom s.c.c. C contained in $X^?$. Then C is also a bottom s.c.c. in the chain M, and its states should have been assigned to X^{NO} by rule (2), a contradiction. Thus, every trajectory of $M^?$ starting at any state of $X^?$ goes eventually to the absorbing state w with probability 1. Let T^k be the kth power of the matrix T. The uv entry of T^k is equal to the probability that a trajectory starting at state u is after k steps in state v. Since the trajectory goes eventually to w (and stays there) with probability 1, it follows that $\lim_{k \to \infty} T^k = 0$ (for more details, see Kemeny et al. [1976, Section 5.1, and in particular Proposition 5.3]). Since $a - b = T \cdot (a - b)$, we have also $a - b = T^k \cdot (a - b)$, and taking the limit as k tends to infinity we conclude that $a = b$. □

We describe the construction of the new chain M' and a new formula f'. The new chain M' has a larger state space X' and is defined over the new set of atomic propositions $Prop' = Prop \cup \{\xi\}$ where ξ is a new atomic proposition.

States of M'. For each $u \in X^{YES}$ there is a state (u, ξ) in M'. For each $u \in X^{NO}$ there is a state $(u, \neg \xi)$. For each $u \in X^?$, there are two states $(u, \xi), (u, \neg \xi)$ in M'. A state (u, ξ) satisfies all the atomic propositions that u satisfies including the new atomic proposition ξ; $(u, \neg \xi)$ has similar properties except that it does not satisfy ξ. The way we will define the transition probabilities in M' will give to a state (u, ξ) the following interpretation. The distribution of the trajectories of M' projected on the first state component starting from (u, ξ) will be the same as the distribution of the trajectories of M starting from u conditioned on the event that they satisfy $\phi U \psi$.

Transitions of M'. Every transition $u \to v$ in M implies one or two transitions in M'. The transition probability of $(u, \xi_1) \to (v, \xi_2)$, $\xi_i = \xi, \neg \xi, i = 1, 2$, is defined as being equal to the probability that M starting from state u transitions next to state v and from state v onward satisfies property ξ_2 conditioned on the event that in state u it satisfies property ξ_1. (ξ is the property $\phi U \psi$ and $\neg \xi$ is the property $\neg(\phi U \psi)$.) In more detail, we have the following cases.

(1) $u, v \in X^{YES} \cup X^{NO}$. In this case, there is exactly one state (u, ξ_1) with first component u and exactly one state (v, ξ_2) with first component v, where $\xi_i = \xi, \neg \xi, i = 1, 2$ depending on whether u and v are in X^{YES} or X^{NO}. We include the transition $(u, \xi_1) \to (v, \xi_2)$ in M' with probability p_{uv}.
(2) $u \in X^{YES} \cup X^{NO}, v \in X^?$. Then there is exactly one state (u, ξ_1) with first component u (where $\xi_1 = \xi$ or $\neg \xi$), and two states with first component v. We include the two transitions: $(u, \xi_1) \to (v, \xi)$ with probability $p_{uv} q_v$ and $(u, \xi_1) \to (v, \neg \xi)$ with probability $p_{uv} \bar{q}_v$, where $\bar{q}_v = 1 - q_v$.
(3) $u \in X^?$. If $v \in X^?$, we have two transitions: $(u, \xi) \to (v, \xi)$ with probability $p_{uv} q_v / q_u$ and $(u, \neg \xi) \to (v, \neg \xi)$ with probability $p_{uv} \bar{q}_v / \bar{q}_u$. If $v \in X^{YES}$, we have only the transition $(u, \xi) \to (v, \xi)$ with probability p_{uv}/q_u, and if $v \in X^{NO}$ we have only the transition $(u, \neg \xi) \to (v, \neg \xi)$ with probability $p_{uv}/\bar{q}_u$.

Initial distribution of M'. Let $p_0(u)$ be the probability of state u in the initial distribution of M. If $u \in X^{YES} \cup X^{NO}$, then in M' we define $p_0((u, \xi_1)) = p_0(u)$, where (u, ξ_1), $\xi_i = \xi$ or $\neg \xi$ is the unique state with first component u. If $u \in X^?$, then $p_0((u, \xi)) = p_0(u) q_u$ and $p_0((u, \neg \xi)) = p_0(u) \bar{q}_u$.

Let f' be the PTL formula with atomic propositions in *Prop'* obtained from f by substituting the new atomic proposition ξ in the place of $\phi U \psi$. We shall show that the probability that a trajectory of M satisfies the formula f is equal to the probability that a trajectory of M' satisfies the formula f'.

Let g be the mapping of trajectories of M' into trajectories of M by projecting on the first component of the states of M'. Let $\mathbf{X} = X_0, X_1, \ldots$ be a trajectory of the Markov chain M, let $\mathbf{X}' = X'_0, X'_1, \ldots$ be a trajectory of M', and let $\mathbf{Y} = Y_0, Y_1, \ldots$ be defined as the process $Y_i = g(X'_i)$, $i = 0, 1, \ldots$. Note that the processes $\mathbf{X}$ and $\mathbf{Y}$ both take values in X^ω. The following holds:

LEMMA 3.1.1.2. *The processes* $\mathbf{X}$ *and* $\mathbf{Y}$ *have the same distribution.*

PROOF. It suffices to show that the two distributions agree on the basic cylinder sets, that is,

$$P_M(X_0 = x_0, \ldots, X_m = x_m) = P_{M'}(Y_0 = x_0, \ldots, Y_m = x_m)$$

for every $m + 1$-sequence $(x_0, \ldots, x_m) \in X^{m+1}$ (see Breiman [1968]). We prove this as follows: Let $P_M(x_0, \ldots, x_m)$ denote $P_M(X_0 = x_0, \ldots, X_m = x_m)$ and $P_{M'}(x'_0, \ldots, x'_m)$ denote $P_{M'}(X'_0 = x'_0, \ldots, X'_m = x'_m)$, respectively. Observe that if (v, ξ_i), $\xi_i = \xi$ or $\neg \xi$, is a state of M' and u is an immediate predecessor of v in M, then (v, ξ_i) has exactly one immediate predecessor in M' with first component u: if $u \in X^{YES} \cup X^{NO}$, then there is only one state with first component u in M'; if $u \in X^?$, then the immediate predecessor of (v, ξ_i) is (u, ξ_i), that is, it agrees in the second component. Therefore, starting from a state $x'_m = (x_m, \xi_t)$ of M' and proceeding backward, we see that M' has a unique path $x'_0 \cdots x'_m$ which ends in x'_m and has projection $x_0 \cdots x_m$. From the definition of the initial distribution and the transition probabilities of M', it is easy to check that, if $\xi_t = \xi$ then $P_{M'}(x'_0, \ldots, x'_m) = P_M(x_0, \ldots, x_m)q_{x_m}$, and if $\xi_t = \neg\, \xi$, then $P_{M'}(x'_0, \ldots, x'_m) = P_M(x_0, \ldots, x_m)\bar{q}_{x_m}$.

The claim follows now easily from these observations. If $x_m \in X^{YES} \cup X^{NO}$, then there is only one prefix $x'_0 \cdots x'_m$ of trajectories of M' that projects to $x_0 \cdots x_m$, and $P_{M'}(Y_0 = x_0, \ldots, Y_m = x_m) = P_{M'}(x'_0, \ldots, x'_m) = P_M(x_0, \ldots, x_m)$. If $x_m \in X^?$, then there are two such prefixes, one ending at (x_m, ξ) and the other ending at $(x_m, \neg\, \xi)$, and again $P_{M'}(Y_0 = x_0, \ldots, Y_m = x_m) = P_M(x_0, \ldots, x_m)$. □

LEMMA 3.1.1.3. *A trajectory* $\mathbf{X}'$ *of the Markov chain* M' *satisfies with probability one the following property: at each time* $k \geq 0$, ξ *is a simple proposition satisfied in state* X'_k *iff* $X'_k, X'_{k+1}, \ldots$ *satisfies the formula* $\phi\, U \psi$. (*In other words,* $(\phi\, U \psi \equiv \xi)$ *holds at all times in all trajectories of* M' *with probability one.*)

PROOF. It is enough to show that starting at any state of the form (u, ξ) a trajectory $\mathbf{X}'$ satisfies with probability one the formula $\phi\, U \psi$, and starting at any state of the form $(u, \neg\, \xi)$ it will satisfy with probability one the negation of the above formula.

Case (a). Initial state $(u, \neg\, \xi)$, $u \in X^{NO}$. Consider the subgraph G_1 of M' induced on the states whose first component is in X^{NO}. This subgraph is identical with the subgraph of M induced on the states of X^{NO} and has the property that all transitions (in M') out of G_1 occur on states of G_1 satisfying $\neg\, \phi$ and $\neg\, \psi$. Furthermore, by construction all states in G_1 satisfy $\neg\, \psi$. If our initial state satisfies $\neg\, \phi$, $\neg\, \psi$, the formula $\neg(\phi\, U \psi)$ is satisfied trivially. If we start at a state satisfying ϕ and $\neg\, \psi$, then with probability one any trajectory $\mathbf{X}'$ of M' will either not exit G_1 and will remain in a bottom strongly connected component of G_1 whose states satisfy ϕ and $\neg\, \psi$ or it will eventually hit a state satisfying $\neg\, \phi$ and $\neg\, \psi$. In either case $\mathbf{X}'$ satisfies $\neg(\phi\, U \psi)$.

Case (b). Initial state $(u, \xi), u \in X^{YES}$. Consider the subgraph G_2 of M' induced on the states whose first component is in X^{YES}. This subgraph is identical with the subgraph of M induced on the states of X^{YES} and has the property that all transitions (in the graph of M') out of G_2 occur on states of G_2 satisfying ψ. Furthermore, by construction all states in G_2 satisfy ϕ and $\neg\, \psi$ or satisfy ψ. An important observation is that there is no bottom strongly connected component in G_2 consisting of states satisfying ϕ and $\neg\, \psi$ since by the construction C_U such a component would be part of G_1. If our initial state satisfies ψ, the formula $\phi\, U \psi$ is satisfied trivially. If we start at a state satisfying ϕ and $\neg\, \psi$, then with probability one any trajectory $\mathbf{X}'$ of M' will eventually hit a state satisfying ψ and will hence satisfy $\phi\, U \psi$.

Case (c). Initial state (u, ξ) or $(u, \neg\, \xi)$, $u \in X^?$. Consider the subgraphs G_3 and G_4 of M' induced on the states (u, ξ) and $(u, \neg\, \xi)$ respectively, where $u \in X^?$. By the construction C_U these subgraphs are identical and they do not contain any bottom strongly connected component of M' since such components would be already in G_1. Also all transitions out of G_3 are into states in G_2 and all transitions out of G_4 are into states in G_1. Consider a trajectory $\mathbf{X}'$ starting in G_3. This trajectory will with probability one eventually hit a state in G_2 with all previous states in G_3. Since all states in G_3 satisfy ϕ and $\neg\, \psi$ and starting at G_2 almost all trajectories satisfy $\phi U \psi$, it follows that $\mathbf{X}'$ also satisfies $\phi U \psi$ with probability one. For a similar reason any trajectory starting at G_4 will satisfy $\neg(\phi U \psi)$ with probability one. □

We can show now that the transformation C_U preserves the probability of satisfaction.

PROPOSITION 3.1.1.4. $P_M((L_\omega(f)) = P_{M'}(L_\omega(f'))$.

PROOF. Since $\mathbf{X}$ and $\mathbf{Y}$ have the same distribution it follows that $P_M(L_\omega(f)) = P_{M'}(L_\omega(f))$. Since $(\phi U \psi \equiv \xi)$ holds at all times in all trajectories of M' with probability one, it follows that $P_{M'}(L_\omega(f)) = P_{M'}(L_\omega(f'))$, and the proof is complete. □

We will describe now the transformation C_X for the X connective. The construction of M' proceeds in similar steps as in the case of C_U. Let $X\phi$ be an "innermost" temporal subexpression of f; that is, a subexpression such that ϕ is composed of atomic propositions and Boolean connectives. We can evaluate ϕ on each state of M. The construction C_X produces a new Markov chain M' and a new formula f' as follows:

3.1.2. *Construction C_X.* We first partition the states of the Markov chain into the three disjoint subsets $X = X^{YES} \cup X^{NO} \cup X^?$ defined as follows: X^{YES} contains all states u for which all transitions are into states satisfying ϕ, and X^{NO} contains all states for which all transitions are into states satisfying $\neg\phi$. A state in $X^?$ has transitions to states satisfying ϕ and to states satisfying $\neg\phi$. As in the case of the Until operator, the interpretation of the above sets is that if a path starts from a state in X^{YES} (respectively, X^{NO}), then with probability one it will satisfy the formula $X\phi$ (respectively, $\neg X\phi$), and if it starts from a state in $X^?$ then both events have nonzero probability.

Let p_{uv} denote the probability of the transition $u \to v$, and let q_v denote the probability that $X\phi$ is satisfied starting from state v. These probabilities can be computed from the equations $q_u = \Sigma_v p_{uv}$, where the sum ranges over all successor states v of u satisfying property ϕ if $u \in X^?$. Otherwise, $q_u = 1$ if $u \in X^{YES}$, $q_u = 0$, if $u \in X^{NO}$.

The new chain M' has a larger state space X' and is defined over the new set of atomic propositions $Prop' = Prop \cup \{\xi\}$, where ξ is a new atomic proposition, as follows:

States of M'. They are similarly defined as in the construction C_U.

The way we will define the transition probabilities in M' gives to a state (u, ξ) the following interpretation. The distribution of the trajectories of M' projected on the first state component starting from (u, ξ) will be the same as the distribution of the trajectories of M starting from u conditioned on the event that they satisfy $X\phi$.

Transitions of M'. Every transition $u \to v$ in M implies one or two transitions in M'. The transition probability of $(u, \xi_1) \to (v, \xi_2)$, $\xi_i = \xi, \neg\xi, i = 1, 2$, is defined as being equal to the probability that M, being at state u, transitions next to state v and from state v onward satisfies property ξ_2, conditioned on the event that in state u it satisfies property ξ_1. (ξ is the property $X\phi$ and $\neg\xi$

is the property $\neg X\phi$.) In more detail, we have the following cases:

(1) $u, v \in X^{YES} \cup X^{NO}$. Then M' contains a unique state (u, ξ_1) with first component u and a unique state (v, ξ_2) with first component v. We include in M' the transition $(u, \xi_1) \to (v, \xi_2)$ with probability p_{uv}.
(2) $u \in X^{YES} \cup X^{NO}, v \in X^?$. Let (u, ξ_1) be the unique state of M' with first component u. We include two transitions: $(u, \xi_1) \to (v, \xi)$ with probability $p_{uv}q_v$ and $(u, \xi_1) \to (v, \neg \xi)$ with probability $p_{uv}\bar{q}_v$, where $\bar{q}_v = 1 - q_v$.
(3) $u \in X^?$. If $v \in X^?$ and v satisfies ϕ, we have two transitions: $(u, \xi) \to (v, \xi)$ with probability $p_{uv}q_v/q_u$ and $(u, \xi) \to (v, \neg \xi)$ with probability $p_{uv}\bar{q}_v/q_u$. If $v \in X^?$ and v satisfies $\neg \phi$, we have two transitions: $(u, \neg \xi) \to (v, \xi)$ with probability $p_{uv}q_v/\bar{q}_u$ and $(u, \neg \xi) \to (v, \neg \xi)$ with probability $p_{uv}\bar{q}_v/\bar{q}_u$. If $v \in X^{YES} \cup X^{NO}$ and v satisfies ϕ, we have only the transition $(u, \xi) \to (v, \xi_2)$ with probability p_{uv}/q_u, where (v, ξ_2) is the unique state of M' with first component v, and if $v \in X^{YES} \cup X^{NO}$ and v satisfies $\neg \phi$ we have only the transition $(u, \neg \xi) \to (v, \xi_2)$ with probability $p_{uv}/\bar{q}_u$.

Initial distribution of M'. If $u \in X^{YES} \cup X^{NO}$, then the unique state (u, ξ_1) of M' with first component u gets probability $p_0((u, \xi_1)) = p_0(u)$. If $u \in X^?$, then $p_0((u, \xi)) = p_0(u)q_u$ and $p_0((u, \neg \xi)) = p_0(u)\bar{q}_u$.

Let f' be the PTL formula with atomic propositions in *Prop'* obtained from f by substituting the new atomic proposition ξ in the place of $X\phi$. Then, the claim of Proposition 3.1.1.4 holds in this case as well with the proof following the same lines as in the case of C_U.

If f has k temporal operators, we can compute $P_M(L_\omega(f))$ as follows: We apply k times the appropriate transformations C_U, C_X in order to get the sequence $f^1, M^1, \ldots, f^k, M^k$, where f^k is a simple propositional formula. Then $P_M(L_\omega(f)) = P_{M^k}(L_\omega(f^k))$, which is simply the sum of the initial probabilities in M^k over all states satisfying f^k.

In the remainder of this subsection, we will analyze the time and space complexity of computing the probability that M satisfies f, and the complexity of the simpler verification problem, that is, testing if the probability is 1. We measure the size $|f|$ of a formula f by its number of Boolean and temporal connectives. For the verification problem we measure the size $|M|$ of the Markov chain M by its number of nodes and edges. For the problem of computing the exact probability, we may either assume that the transition probabilities of M are rationals and include in the size of M the space (in bits) needed to specify the probabilities, or we may assume that the transition probabilities are real numbers and use a model of exact real arithmetic.

For verification purposes, we only need to check that the formula f^k is satisfied by all initial states of the final Markov chain M^k (states with nonzero probability in the initial distribution). It is clear from the constructions C_U and C_X that the underlying graph of the chain M' (the states and the transitions of M') depends only on the graph of M, and that the initial states of M' depend on the initial states of M. The central part in the construction of (the graph of) M' is the partition of the nodes of M into the three sets X^{YES}, X^{NO} and $X^?$. Once we have computed this partition, the rest of the construction is straightforward. We show below how to compute this partition in the case of the Until operator; the case of the Next operator is simple and is omitted.

(*Elimination of the temporal subexpression* $\phi U \psi$).

(1) Evaluate ϕ and ψ in every state of M. Assign every state that satisfies ψ to X^{YES}, and every state that satisfies $\neg\phi$ and $\neg\psi$ to X^{NO}.
(2) Let H be the subgraph induced by the states that satisfy ϕ and $\neg\psi$. Compute the strongly connected components (s.c.c.) of H.
(3) Process the s.c.c.'s of H bottom-up. For every bottom s.c.c. D of the current graph H, assign the states of D to one of the sets X^{YES}, X^{NO} or $X^{?}$ as follows, and then remove D from the graph H. (NOTE: when a s.c.c. of H is processed, the nodes of all its successors in the original graph H have been already assigned and removed from H.)
   Case 1. If there is no arc of M coming out of D (i.e., D is a bottom s.c.c. of M), or if all such arcs go to states in X^{NO}, then assign all the states of D to X^{NO}.
   Case 2. If there are some arcs of M coming out of D and all of these arcs go to states in X^{YES}, then assign the states of D to X^{YES}.
   Case 3. Otherwise, assign the states of D to $X^{?}$.

Let t be the number of connectives in the subexpression $\phi U\psi$ that is eliminated. Then the size of the new formula f' is $|f'| = |f| - t$. Clearly, we can perform Step 1 in time $O(t|M|)$. As is well known, the strongly connected components of a graph can be computed in time linear in the size (number of nodes and edges) of the graph (see, e.g., Aho et al. [1974]). Thus, Step (2) takes time $O(|M|)$, and Step (3) can be also performed easily in linear time. Therefore, we can construct the graph of the new chain M' in time $O(t|M|)$. Since the elimination of each temporal operator doubles (at most) the number of nodes and edges, the time needed to construct the graph of the final chain M^k is $O(2^{|f|}|M|)$.[2]

We analyze now the space complexity. First, we will argue that we can eliminate a temporal subexpression and construct the graph of the new chain M' from that of M using work space of $O(|f| + \log^2|M|)$ bits. Recall that we can test whether there is a path from one node to another node in a graph with m nodes in space $O(\log^2 m)$. Consider the elimination of a subexpression $\phi U\psi$ from f (the case of the Next operator is again similar and simpler). First, we have to determine the states of the new Markov chain M'. A state u of M gives rise to a state (u, ξ) of M', iff $u \notin X^{NO}$. To test whether this is the case, first we evaluate ϕ and ψ at u. If u satisfies ψ or does not satisfy ϕ, then we can deduce immediately whether u is in X^{NO} or not. If u satisfies ϕ and $\neg\psi$, then we cycle through all the states v, and for each state v we check if v satisfies ψ, and if so, test whether there is a path from u to v through states of H (that is, going through nodes that satisfy ϕ and $\neg\psi$) in space $\log^2|M|$. To test whether u gives rise to a state $(u, \neg\xi)$, that is, whether $u \notin X^{YES}$, first we evaluate again ϕ and ψ at u. If u satisfies ϕ and $\neg\psi$, then we cycle through all the states v of M, and for each state v we evaluate ϕ and ψ at v, check if $v \in X^{NO}$ (as described above), and test whether there is a path through H from u to v; this can be accomplished also using space $O(\log^2|M|)$. The arcs of M' can be generated easily in the same space as the nodes.

Using these observations, we can write a recursive algorithm that generates the states and transitions of the final chain M^k from M. Since the depth of recursion is $|f|$ and the size of the Markov chain doubles every time, the work space needed is proportional to $|f|(|f| + \log^2(2^{|f|}|M|))$, that is, $O(|f|^3 + |f|\log^2|M|)$.

[2]This time bound is under the uniform cost criterion, which is traditionally used for graph algorithms. Under the logarithmic cost criterion there is an extra logarithmic factor.

If we wish to compute the probability $P_M(L_\omega(f))$ that a trajectory of the Markov chain M satisfies the formula f, then every time we eliminate a temporal connective from the formula, we have to compute the initial distribution and the transition probabilities of the new Markov chain. This amounts to solving a linear system of equations with dimension equal to the number of states. Thus, the final Markov chain M^k (including the probabilities) can be constructed in time exponential in $|f|$ and polynomial in M. Summarizing, we have:

THEOREM 3.1.2.1. *We can test if a finite state probabilistic program M satisfies a formula f in time $O(|M|2^{|f|})$, or in space polynomial in f and polylogarithmic in M. We can compute the probability of satisfaction $P_M(L_\omega(f))$ in time exponential in f and polynomial in M.*

The algorithm can be easily extended to handle the extension of temporal logic with *past connectives* [Lichtenstein and Pnueli 1985]. There are two past connectives: the unary connective *Previous* which is analogous to *Next*, and the binary connective *Since* which is analogous to *Until*. Using the same techniques, we can define similar transformations for these connectives and derive the same complexity bounds as in Theorem 3.2.1 for the logic that includes both past and future connectives.

3.2. CONCURRENT PROBABILISTIC PROGRAMS. Given a formula f and a concurrent Markov chain M, we wish to determine whether M satisfies f, that is, whether $P_{M,u}(L_\omega(f)) = 1$ for all schedulers u. We can do this as follows: We first construct a Buchi automaton A for $\neg f$, and then test if there is a scheduler u such that $P_{M,u}(L_\omega(A)) > 0$ using the algorithm that we shall present in Section 4.2. Each of the two steps contributes one exponential. The size (number of nodes and arcs) of the automaton A is $2^{O(|f|)}$, and the probabilistic emptiness problem for an automaton A and a concurrent Markov chain M can be solved in time $|M|^2 \cdot 2^{O(|A|)}$. Therefore, the overall time complexity of the algorithm is $|M|^2 \cdot 2^{2^{O(|f|)}}$, doubly exponential in the size of the formula and quadratic in the size of the program. We shall prove that the problem requires double exponential time, $2^{2^{\Omega(n)}}$ in the total size n of the input (the formula and the Markov chain).

THEOREM 3.2.1. *Determining whether a concurrent probabilistic program satisfies a formula is complete for double exponential time.*

PROOF. The reduction is from the membership problem for exponential space-bounded alternating Turing machines [Chandra et al. 1981]. As is well known, ASPACE$(S(n)) = \bigcup_{c>0}$ DTIME$(c^{S(n)})$.

Let T be an alternating Turing machine working in space $S(n) = 2^n$. We assume without loss of generality that T has only one tape, which contains initially its input x. Recall that an alternating Turing machine has four types of states: *existential*, *universal*, *accepting*, and *rejecting*. We assume without loss of generality that the machine has two possible next moves from each existential and universal state, and it halts when it is in an accepting or rejecting state. We refer to the accepting and rejecting states as the *halting* states. Let Γ be the tape alphabet, Q the set of states, and let Δ be $\Gamma \cup (Q \times \Gamma)$; we call Δ the *extended tape alphabet*. As usual, a configuration of the machine is described by a string $w_0 w_1 \cdots w_{2^n-1}$ of length 2^n over Δ, where $w_i = \gamma \in \Gamma$ if tape cell i has symbol γ and the tape head is at another cell, and $w_i = \langle q, \gamma \rangle \in Q \times \Gamma$ if in addition the head is at cell i and the state is q. From now on, when we refer to the content of a tape cell in a configuration, we will mean the corresponding symbol of the *extended* tape alphabet Δ. The type of a configuration (universal, existential, etc.) is determined by the type of its state. A *computation* is a sequence of configurations, starting from the initial one, where each configuration follows from the previous one according to the next move relation of the machine T. We assume without loss of generality that no computation repeats a configuration; that is, all computations terminate with an accepting or rejecting configuration within time $c^{S(n)}$ for some constant c.

Computation by an alternating Turing machine can be viewed as a two-person game between an existential player E and a universal player U. A position of the game is a configuration of the machine, and the game starts with the initial configuration. Depending on the type of a configuration (existential or universal), player E or U moves choosing the next configuration according to the next move relation of T. Player E wins if the final configuration is accepting, and U wins if it is rejecting. The input x is accepted (respectively, rejected) by the machine T if player E (respectively, U) has a winning strategy starting from the initial configuration corresponding to x.

Given an alternating Turing machine T as above and an input x of length n, we shall construct a temporal formula f and a concurrent probabilistic program M, both of size $O(n)$, such that T accepts x if and only if there is a scheduler S for which $P_{M,S}(L_\omega(f)) > 0$; that is, T rejects x if and only if M satisfies the formula $\neg f$. Before going into the details, we give a brief, high-level description of the construction. Consider the verification problem as a game between an indifferent probabilistic player P who chooses the transitions out of randomizing states of the program M and a purposeful player S (the scheduler) who chooses the transitions out of the nondeterministic states. The formula f serves as the referee who checks that S follows the rules and does not cheat, and decides who wins the game. A simplified version of the program M is given below. Assuming that the scheduler S does not cheat, it constructs the indicated configurations in the nondeterministic moves. The goal of the scheduler is to have the program run forever.

```
repeat until a rejecting state of $T$ is reached
  nondeterministically construct the initial configuration of $T$
  while the state is not a halting state do
  if the state is universal then choose probabilistically the next move of $T$
    else choose nondeterministically the next move
  nondeterministically construct the next configuration of $T$
```

The scheduler constructs a configuration by specifying the contents of each cell. Figure 1 shows how a cell is specified. The graph of the figure starts with n levels with two nodes at each level; a traversal through this part corresponds to an n-bit number, the index of a cell in binary. The last level has one node for each element of the extended tape alphabet Δ. For each $\delta \in \Delta$, we have an atomic proposition, which for simplicity we denote also by δ; this proposition is false at all nodes except for the node at the last level that corresponds to δ. Consider the graph of Figure 1 with a back edge added from node v to node u.

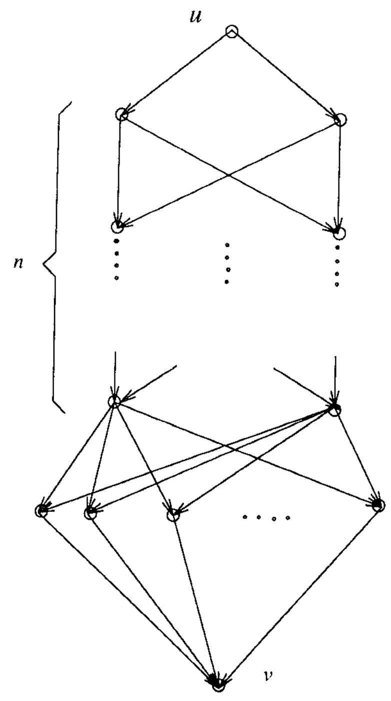

FIGURE 1

The scheduler constructs a configuration by going around this loop 2^n times, specifying in order the contents of cell 0, cell 1,..., up to cell $2^n - 1$.

In Figure 2, we show in more detail the concurrent Markov chain M. In the initial distribution, node a at the top has probability 1, and all other nodes have probability 0. Each oval in the figure is called a *block*, and is a copy of the graph of Figure 1, except for block I whose last level is slightly different, as will be explained later. We will describe now how the program works assuming that the scheduler does not cheat. It will be clear from this description which nodes are nondeterministic and which are randomizing. All transitions out of randomizing nodes have probability 1/2.

An iteration of the outer loop (closed by the arc $c \to a$) corresponds to a complete computation of the Turing machine T from the initial to a halting configuration. In block I, the scheduler constructs the initial configuration. (Thus, all nodes of block I are nondeterministic.) An iteration of the inner loop (closed by the arc $c \to b$) corresponds to a move from one configuration to the next. First, in block O, the scheduler reproduces the current configuration (in the first iteration of the inner loop, this is the initial configuration). In block C1, the probabilistic player P gives a random test to the scheduler S to verify that S has not changed the contents. In particular, P specifies randomly the index of a cell, and the scheduler is supposed to reproduce (in the last level of C1) the contents of that cell in the current configuration. Thus, in block C1, the entry node (node u of Figure 1) and the nodes of the first $n - 1$ levels are randomising, while the two nodes of the nth level are nondeterministic. If the current configuration is universal, then S goes to node m_1 which is randomising; otherwise, it goes to node m_2 which is nondeterministic. There are two possible next moves of the Turing machine out of the current configuration,

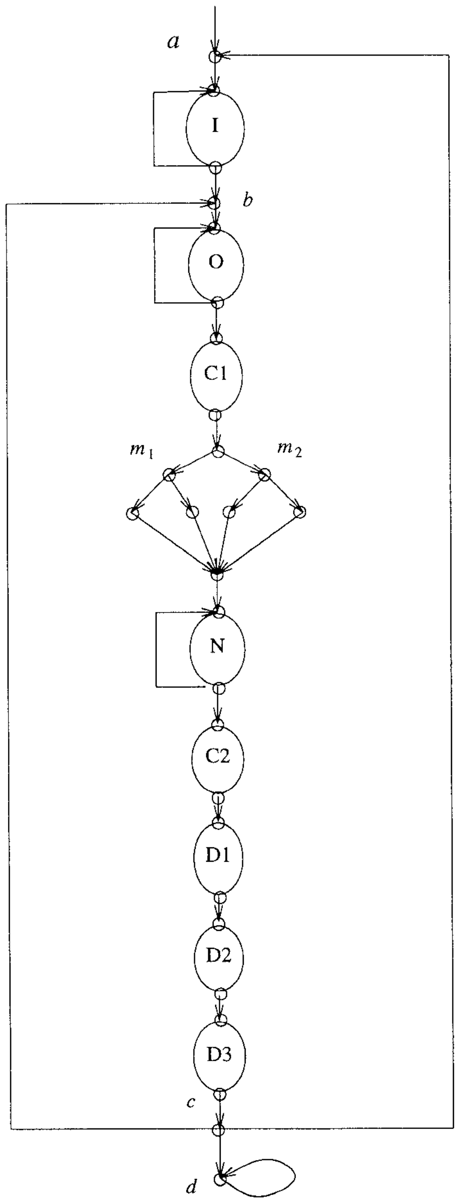

FIGURE 2

and these two possible moves correspond to the two possible transitions out of each of the nodes m_1, m_2. If the configuration is universal, then the probabilistic player P chooses the next move at node m_1, and if the configuration is existential, then the scheduler S chooses the next move at node m_2. In block N, the scheduler constructs the next configuration. In block C2, the probabilistic player P gives a random test to S to verify that the new configuration conforms to the next move relation of the Turing machine. Namely, as in the case of block C1, player P specifies randomly the index of a cell, and the scheduler is supposed to reproduce (in the last level of C2) the contents of that cell as specified in block N. In blocks D1, D2, D3, the scheduler reproduces from the old configuration (block O) the contents of the cell indexed in C2 and its two adjacent cells. Finally, if the new configuration is rejecting, then the scheduler goes to the dead state d; if it is accepting, it returns to a to start a new computation; otherwise, it returns to b.

The temporal formula f consists of several parts that check the honesty of the scheduler. (Recall that S wants to satisfy f.) There are parts of f which check that S obeys the following rules.

(1) The scheduler S constructs in each of the blocks I, O, N a configuration cell-by-cell in order starting from cell 0 up to $2^n - 1$.
(2) In every execution of block I, the scheduler constructs the initial configuration.
(3) Wherever S is supposed to reproduce the contents of a cell (in blocks C1, C2, D1, D2, D3), it does so faithfully.
(4) If the current configuration is universal, then S lets P choose the next move (i.e., S goes to m_1).
(5) If the index of the cell specified in block C2 is k, then the indexes in blocks D1, D2, D3 are $k - 1, k, k + 1$, respectively.
(6) The contents of the cell in block C2 follow correctly from those of the cells in blocks D1, D2, D3 and the next move chosen at node m_1 or m_2.
(7) If the new configuration is rejecting, then S breaks out of the outer loop moving to the dead state d, and if it is universal or existential then S returns to b.
(8) Node a is visited infinitely often.

In the description of f, we will use for simplicity in the notation also the two unary temporal operators G ("always") and F ("eventually"). These operators can be expressed using the "until" operator. The formula $F\phi$ abbreviates $trueU\phi$, and $G\phi$ abbreviates $\neg F \neg \phi$. For every node of M there is an atomic proposition that is true only at that node; for simplicity, we use the label of the node to denote also this proposition. Part (8) is easy: the formula GFa states that node a is visited infinitely often. We shall describe one by one the other parts of f now.

Part (1). We describe it for block I; there is an analogous subformula for blocks O and N. For each of the $n + 1$ levels of the block there is a proposition l_i which is true for the nodes at level i and is false for all other nodes. We have a proposition t, which is true for exactly one of the two nodes in each of the first n levels of the block and is false for the other node (as well as the other nodes of M). Every block of M has its own disjoint set of such propositions l_i and t. When traversing a block, the choice of the node at level i represents a truth value (true or false) for t that corresponds to the ith bit (1 or 0) of the index of a cell. Let u be the entry node of block I and v the exit node (refer to Figure 1).

Consider the formula $\phi_1 = G\{a \to [\bar{t}Uv]\}$, where $\bar{t}$ stands for $\neg t$. This formula states that every time the path passes through node a, then all nodes it meets until it hits v for the first time must satisfy $\bar{t}$; that is, the first time the scheduler traverses I it must choose the index $00 \cdots 0$. The formula $\phi_2 = G\{l_1 \to [(tUl_{n+1}) \equiv (\bar{u}Ub)]\}$ states that, whenever the path is at level 1, two subformulas have the same truth value. The first subformula tUl_{n+1} is true iff all bits of the index in this iteration of I are 1, and the second subformula $\bar{u}Ub$

is true iff the path meets node b before u, that is, the path exits block I after this iteration.

Finally, we have a formula ϕ_3, which says that if one iteration chooses index k, then the next iteration (if there is one) must choose index $k + 1$. The formula ϕ_3 is a conjunction of subformulas, each of which concerns one bit of the index. For each $i = 1, \dots, n$ consider the formula $G\{(l_i \& \bar{t}) \rightarrow ([(l_i \vee t) U l_{n+1}] \equiv X[\bar{l}_i U (l_i \& t)])\}$. This formula says that, if in an iteration of I the ith bit of the index is 0, then two subformulas have the same truth value. The first subformula $(l_i \vee t) U l_{n+1}$ is true iff all the subsequent bits $(i + 1, \dots, n)$ of the index are 1. The second subformula is true iff in the next iteration of I the ith bit is 1. (The use of the next operator X can be avoided altogether; that is, we could carry out the reduction using only the Until operator. However, the construction is complicated enough as it is, so we will not go into this.) In a similar way, we can construct a formula that forces the scheduler to choose the correct ith bit in the next iteration of I (if there is a next iteration) in case the ith bit in the current iteration is 1.

The conjunction of ϕ_1, ϕ_2, and ϕ_3 forces the scheduler to specify a complete configuration in block I as desired. There is an analogous formula for blocks O and N.

Part (2). As we indicated earlier, the last level of block I differs from Figure 1. This level of I has $n + 1$ nodes $w_1, \dots, w_{n+1}$. The first n nodes represent the first n symbols of the initial configuration (recall n is the length of the input x), and node w_{n+1} represents the blank symbol. Recall that we have a proposition δ for every element of the extended alphabet Δ; the proposition is true at a node w_i iff w_i represents the symbol δ. The formula for Part (2) consists of 1) a subformula ψ_1 stating that the first iteration of I visits node w_1, 2) a subformula ψ_2 which states that if an iteration of I visits w_i with $i < n + 1$, then the next iteration visits w_{i+1}, and 3) a subformula ψ_3 which states that if an iteration visits w_{n+1}, then the next iteration (if there is one) visits also w_{n+1}. These subformulas are easy to construct. For example, ψ_2 has for each $i = 1, \dots, n$ a conjunct of the form $G\{w_i \rightarrow X[\bar{l}_{n+1} U w_{i+1}]\}$.

Part (3). We construct a formula $\xi(I, C1)$ which states the following: If we compare any iteration of block I with the next execution of block C1, they either differ in one of the first n levels (i.e., in some bit of the index) or else they agree in the $n + 1$th level (i.e., in the content of the cell). Recall the propositions l_i and t for block I, and let k_i and s be the analogous propositions for block C1. Suppose that the path is at the top node u of I, starting an iteration of I. For $i = 1, \dots, n$, the ith bit of the index in this iteration is 1 if and only if the formula $g_i(\mathrm{I}) = \bar{l}_i U (l_i \& t)$ is true (at the present node u). The ith bit of the index chosen in the next execution of C1 is 1 iff the formula $g_i(C1) = \bar{k}_i U (k_i \& s)$ is true (again at the present node u). Similarly, for each $\delta \in \Delta$, the formula $g_\delta(I) = \bar{l}_{n+1} U (l_{n+1} \& \delta)$ is true iff the symbol δ is chosen in this iteration of I, and $g_\delta(C1) = \bar{k}_{n+1} U (k_{n+1} \& \delta)$ is true iff δ is chosen in C1. Let $\theta_1(I, C1) = \bigvee_{i=1,\dots,n} [g_i(I) \neq g_i(C1)]$, and $\theta_2(I, C1) = \&_{\delta \in \Delta} [g_\delta(I) \equiv g_\delta(C1)]$. The formula $\xi(I, C1)$ is $G\{u \rightarrow [\theta_1(I, C1) \vee \theta_2(I, C1)]\}$.

There are analogous formulas $\xi(O, C1)$, $\xi(O, D1)$, $\xi(O, D2)$, $\xi(O, D3)$, $\xi(N, C2)$ for the respective pairs of blocks. There is also a corresponding formula for blocks N and C1, which contains on the right hand side of the implication an additional disjunct $\bar{b} U a$ besides $\theta_1(N, C1)$ and $\theta_2(N, C1)$.

(Nodes a and b are indicated in Figure 2.) That is, when the scheduler constructs a configuration in block N, this configuration must agree with the next execution of C1 (in the relevant cell), unless the path breaks out of the inner loop visiting node a before node b, which means that the computation of the Turing machine terminates, and a new computation starts.

Part (4). Let $r_1, \ldots, r_{n+1}$ be the propositions corresponding to the levels of block O. For each universal state q and each $\delta \in \{q\} \times \Gamma$ we have a conjunct $G[(r_{n+1} \& \delta) \to (\neg(m_1 \vee m_2) U m_1)]$. We do not need an analogous subformula for an existential state because the scheduler S does not gain anything by going to state m_1 and letting the probabilistic player P choose the next move, when S could have chosen the move itself according to the rules.

Part (5). We have formulas stating that the index chosen in C2 is the same as the index chosen in the next execution of D2, the index of D1 is 1 less than the index of D2, and the index of D2 is 1 less than the index of D3. All arithmetic is $\bmod 2^n$; that is, the successor of $11 \cdots 1$ is $00 \cdots 0$. We saw in Part (1) how we can express in temporal logic the fact that one index is the successor of another.

Part (6). We have one formula for every rule in the next move relation of the Turing Machine. Such a rule has the following form: When the machine T is at an existential or universal state q, the tape head scans letter α and the first of the two alternative moves is chosen, then T prints β, moves to state p and shifts the head to the right. Suppose that the path is at node b of M starting a new iteration of the inner loop. For each block $B = C2, D1, D2, D3$ and each $\delta \in \Delta$, we can construct as in Part (3) a formula $g_\delta(B)$ which states that the next execution of block B chooses symbol δ. Similarly, we can construct a formula h which states that the next time the path reaches node m_1 or m_2 it takes the left branch (corresponding to the first of the two possible moves from the current configuration). The formula corresponding to this rule of T is the conjunction of the following three formulas:

$$G\left\{\left[b \,\&\, h \,\&\, g_{\langle q, \alpha\rangle}(D1)\right] \to \bigvee_{\gamma \in \Gamma} \left[g_\gamma(D2) \,\&\, g_{\langle p, \gamma\rangle}(C2)\right]\right\}$$

(Case 1: The tape head in cell D1),

$$G\left\{\left[b \,\&\, h \,\&\, g_{\langle q, \alpha\rangle}(D2)\right] \to g_\beta(C2)\right\}$$

(Case 2: The head in cell D2),

$$G\left\{\left[b \,\&\, h \,\&\, g_{\langle q, \alpha\rangle}(D3)\right] \to \bigvee_{\gamma \in \Gamma} \left[g_\gamma(D2) \,\&\, g_\gamma(C2)\right]\right\}$$

(Case 3: The head in cell D3).

We include analogous formulas for every rule in the next move relation of the Turing machine. In addition, there is a formula stating that if the head is not in any of the cells D1, D2, D3, then the content of the cell C2 is the same as that of D2.

Although we could have treated the boundary cells (0 and $2^n - 1$) differently, note that this is not really necessary: Using arithmetic $\bmod 2^n$ amounts to considering the tape of the machine as being circular rather than linear. Since the machine does not use space more than 2^n, it never attempts during a computation to move the head right of the rightmost cell $2^n - 1$, nor of course, left of the leftmost cell; that is, the machine behaves the same way regardless of whether the tape is linear or circular.

Part (7). Let y_i be the propositions corresponding to the levels of block N. For every $\delta \in Q \times \Gamma$, if the first component of δ is a rejecting state we include a formula $G\{[y_{n+1} \,\&\, \delta] \rightarrow [(\bar{b} \,\&\, \bar{a})\, U d]\}$, and if it is a universal or existential state we include a formula $G\{[y_{n+1} \,\&\, \delta] \rightarrow [\bar{a}\, U b]\}$.

This concludes our construction of f. Clearly, the formula f has size $O(n)$, and also the program M has $O(n)$ nodes and arcs.

We shall show that T accepts the input x (i.e., the existential player E has a winning strategy) if and only if there is a scheduler S such that $P_{M,S}(L_\omega(f)) > 0$.[3] The one direction is obvious: If T accepts x, then the scheduler plays honestly simulating the Turing machine using the winning strategy of the existential player. With this scheduler, every possible computation path of M satisfies f.

Suppose that T rejects x, that is, the universal player U has a winning strategy. We shall show that for every scheduler S, the probability of satisfaction is $P_{M,S}(L_\omega(f)) = 0$. Recall that every computation of T terminates within $t = c^{2^n}$ steps, for some constant c. Suppose that the trajectory $\mathbf{X}$ is at node a of M (a new computation is starting). We claim that no matter what the scheduler does, and independent of the past, with probability 2^{-2t} the path will either not return ever again to a or will have violated one of parts (1)–(7) of the formula f by the time it returns to a. That is, if E_1 is the event that $\mathbf{X}$ returns (at least) once to node a and satisfies parts (1)–(7) of f, then for every scheduler S the probability of this event $P_{M,S}(E_1)$ is at most $1 - 2^{-2t}$. Consequently, for any scheduler S, the probability of the event E_k that $\mathbf{X}$ returns at least k times to a and satisfies parts (1)–(7) of f is $P_{M,S}(E_k) \leq (1 - 2^{-2t})^k$, which tends to 0 as k tends to infinity. Since the trajectory has to pass through a infinitely often to satisfy f (by part (8)), it follows that any scheduler S has zero probability of satisfying f.

So it suffices to show that $P_{M,S}(E_1) \leq 1 - 2^{-2t}$ for every scheduler S. Let D be the set of trajectories $\mathbf{X} = X_0, X_1, \ldots$ starting from node a such that $\mathbf{X}$ returns to a at some instant l and this initial portion $\alpha = X_0, \ldots, X_l$ of $\mathbf{X}$ satisfies rules (1)–(7). By the construction of the formula f, if a trajectory violates one of these rules, then it does not satisfy the corresponding subformula of f. Thus, $E_1 \subseteq D$. Partition D into two sets: the set D_1 of those trajectories $\mathbf{X}$ which simulate a computation of the Turing machine during the initial portion α as we explained in the description of the concurrent Markov chain M, and the set $D_2 = D - D_1$ of the remaining trajectories of D, that is, those in which the scheduler "cheats" in some way.

Consider a trajectory $\mathbf{X} \in D_1$. Its prefix α until the first return to a simulates a computation of the Turing machine T. Since T rejects the input x, the universal player U has a winning strategy. In every iteration of the inner

[3]In fact, in this construction there is a scheduler S for which the probability of satisfaction $P_{M,S}(L_\omega(f))$ is nonzero if and only if there is a scheduler for which the probability is equal to one, but this fact is not necessary for the theorem.

loop (move from one configuration to the next), if the current configuration is universal and thus it is the turn of the universal player U to choose the next move in the Turing machine game, then the probabilistic player P gets to choose the next move in the game of the concurrent Markov chain (by rule 4) and it has probability $1/2$ of choosing the right move according to the winning strategy of the universal player U. If P chooses the winning moves of U in all the steps, then the final configuration will be rejecting and the trajectory will not return to a by rule (7). Therefore, in every trajectory $\mathbf{X} \in D_1$, there is a step in which the probabilistic player P deviates from the winning strategy. Since there are at most t moves, P has probability 2^{-t} of playing optimally in each step. Therefore, the probability of D_1 is $P_{M,S}(D_1) \leq 1 - 2^{-t}$.

Suppose that a trajectory $\mathbf{X} \in D$ has the following two properties: (i) the first execution of block O produces the initial configuration, and every execution of block O thereafter until the first return to node a produces the same configuration as the previous execution of block N, and (ii) every execution of block N produces the correct next configuration that follows from the configuration of the previous execution of block O and the move chosen at node m_1 or m_2 according to the rules of the Turing machine. Then the prefix α of $\mathbf{X}$ until the first return to a simulates a computation of the Turing machine and $\mathbf{X} \in D_1$. This follows from the construction of M and the fact that α obeys rules (1)–(7). Thus, every trajectory $\mathbf{X} \in D_2$ violates property (i) or (ii).

Suppose that $\mathbf{X} \in D_2$ violates property (i). Assume that in some execution of block O during the prefix α, some cell k is given different contents than in the previous execution of block N; the argument is similar if the first execution of O does not produce the initial configuration (and thus disagrees with block I). Suppose that the probabilistic player P chooses this index k in the next execution of block C1, an event that occurs with probability $2^{-n} > 2^{-t}$. In the last level of block C1, the scheduler chooses a symbol that disagrees either with the symbol in cell k of the previous execution of block O or with that of block N. In the first case, the subformula $\xi(O, C1)$ of Part (3) of f is violated, and in the second case, the subformula $\xi(N, C1)$ is violated. In either case, $\mathbf{X}$ does not satisfy f.

Suppose that $\mathbf{X} \in D_1$ violates property (ii). That is, in some execution of block N, some cell k of the constructed configuration is assigned the wrong contents. Again, if the probabilistic player P chooses this index k in the next execution of C2 (an event that occurs with probability 2^{-n}), it can be seen by a similar argument as in the case of property (i) that the formula f is violated. Thus, from cases (i) and (ii), we have that the probability of E_1 conditioned on D_2 is $P_{M,S}(E_1|D_2) \leq 1 - 2^{-n}$.

Since $E_1 \subseteq D_1 \cup D_2$, it follows that $P_{M,S}(E_1) \leq P_{M,S}(D_1) + P_{M,S}(E_1 \& D_2) = P_{M,S}(D_1) + P_{M,S}(E_1|D_2)P_{M,S}(D_2)$. The second term is bounded from above by $(1 - 2^{-n})(1 - P_{M,S}(D_1)) \leq (1 - P_{M,S}(D_1)) - 2^{-n}2^{-t}$. Therefore, $P_{M,S}(E_1) \leq 1 - 2^{-n}2^{-t} \leq 1 - 2^{-2t}$. As we argued earlier, it follows that $P_{M,S}(L_\omega(f)) = 0$. □

COROLLARY 3.2.2. *There is a constant c such that the problem of determining whether a concurrent Markov chain M satisfies a formula f cannot be solved in time* $O(2^{2^{cm}})$, *where* $m = |f| + |M|$.

PROOF. From the classical time hierarchy theorem (see e.g., Aho et al. [1974]), for any constant $b < 1$, there are languages L in DTIME(2^{2^n}) but not

DTIME($2^{2^{bn}}$). For such a language L, there is an alternating Turing machine T that recognizes L and uses space 2^n (recall that ASPACE($S(n)$) = $\bigcup_{c>0}$ DTIME($c^{S(n)}$)). By the proof of Theorem 3.2.1 given an input string x of length n, we can construct a formula f and a concurrent Markov chain M such that M satisfies f if and only if $x \in L$. Both f and M have size $O(n)$, that is, $m = |f| + |M| \leq d \cdot n$ for some constant d. Since L is not in DTIME($2^{2^{bn}}$), it follows that we cannot determine whether M satisfies f in time $O(2^{2^{cm}})$ for $c = b/d$. Note that this applies to RAM's as well as Turing machines, since they can simulate each other with polynomial overhead. □

4. *Verifying Automata Specifications*

4.1. Sequential Probabilistic Programs. We have a specification given by a Buchi automaton $A = (\tau_A, s_0, F)$ over the alphabet Σ, and a (sequential) Markov chain M with state space X and valuation function V: $X \to \Sigma$. We wish to determine whether M satisfies the specification, and more generally, we wish to compute the probability of satisfaction $P_M(L_\omega(A))$. A first observation is that we can simplify the problem by allowing with no loss of generality that from now on $\Sigma = X$. This holds since we can always replace A with the automaton A' with transition function $\rho'_A(s, x) = \rho_A(s, V(x))$ where ρ_A is the transition function of A. For any trajectory $\mathbf{X}$ of M, the infinite word $V(\mathbf{X})$ is accepted by the automaton A with alphabet Σ iff $\mathbf{X}$ is accepted by the automaton A' with alphabet X. Thus, $P_M(L_\omega(A)) = P_M(L_\omega(A'))$. We will also assume with no loss of generality that A has a transition from every state on each letter of X (by adding transitions to a new dead state, if necessary), and that A has a unique initial state s_0.

We outline the basic ideas of our approach before we go into the technical details. We shall construct from M another Markov chain M_A that is "coupled" with M, in the sense that there is a one-to-one correspondence between their trajectories and we can view both chains as being defined over the same probability space. The chain M_A is a refinement of M which at every point in time keeps tracks of both the state of M and all possible states that the automaton A can be in, given the past history up to that point. A trajectory of M_A hits with probability one a bottom strongly connected component of M_A. We classify the bottom s.c.c. of M_A into *accepting* and *rejecting*, and show that the following property holds: A trajectory that hits an accepting bottom s.c.c. is accepted by the automaton A with probability one; a trajectory that hits a rejecting bottom s.c.c. is rejected by A with probability one. Therefore, $P_M(L_\omega(A)) = 1$ iff all bottom s.c.c. are accepting, and $P_M(L_\omega(A)) = 0$ iff they are all rejecting. More generally, $P_M(L_\omega(A))$ is equal to the probability that a trajectory of M_A hits an accepting bottom s.c.c., a quantity that can be easily computed using standard techniques from the theory of Markov chains. The main technical difficulty is in classifying the bottom s.c.c. of M_A and proving the above property. This involves a combinatorial analysis of the interaction between Markov chains and Buchi automata. We proceed now with the technical development.

We view the Markov chain M also as an automaton over the alphabet X which generates a language $L_\omega(M)$ of infinite words. A transition of M from state v to a state x is labeled with the letter x (the head of the transition); that is, the transition function ρ_M of M from state v on letter $x \in X$ is as follows:

$\rho_M(v, x) = \{x\}$ if M has a transition from v to x, and is $\emptyset$, otherwise. Note that M is a deterministic automaton. Each trajectory of M generates an infinite word over the alphabet X which is identical to the trajectory, except that the first state of the trajectory is missing. For this reason, we first add a new initial state x_0 to the Markov chain, and add a transition from x_0 to every state v with the probability of the transition equal to the probability of v in the (old) initial distribution (of course, if this probability is 0, then the transition $x_0 \to v$ is not included). Thus, from now on, we assume that the Markov chain M has a unique initial state x_0. We denote by $L_\omega(M)$ the set of (infinite) words generated by M, and we use $P_M(L_\omega(A))$ to denote the probability that a word generated by M is accepted by A.

We can take now the product of M and A as two automata over the same alphabet X. If S is the set of states of A, then the set of states of the product transition table $\tau_{M \times A}$ is $X \times S$. On letter $x \in X$, a state (v, s) of the product has transitions to all states (x, s') such that $s' \in \rho_A(s, x)$ if M has a transition from v to x, and has not transitions otherwise. Note that all transitions of $\tau_{M \times A}$ coming into a state with first component x are transitions on letter x. Therefore, if we apply the subset construction on $\tau_{M \times A}$, in the resulting table every state that has an incoming transition, say on letter x, must consist of pairs with the same first component, namely x. In particular, this property holds for every state of $\det(\tau_{M \times A}, (x, s))$ for any state (x, s) of $\tau_{M \times A}$. In the following, we will omit from the subset construction all states that do not have this property (i.e., contain pairs with different first components), since they have no incoming transitions and play no role, and we will use $\det(\tau_{M \times A})$ to denote this reduced table. It is easy to see that this is isomorphic to the table obtained by first applying the subset construction to A and then forming its product with M. If $x \in X$ is a state of M and $Q \subseteq S$ a subset of states of A, we will use the shorthand (x, Q) to denote the set of pairs $\{(x, s) | s \in Q\}$.

We can establish a correspondence between the runs of $\tau_{M \times A}$ or $\det(\tau_{M \times A})$ and the paths of M as follows: Every (finite or infinite) run of $\tau_{M \times A}$ or $\det(\tau_{M \times A})$ *corresponds* to a (unique) path in M obtained by projecting every state of the run on its first component. Conversely, for every path of M starting at any state x and for any state (x, s) of $\tau_{M \times A}$ or state (x, Q) of $\det(\tau_{M \times A})$ with first component x, there is at least one corresponding run of $\tau_{M \times A}$ starting from (x, s), and there is exactly one corresponding run of $\det(\tau_{M \times A})$ starting from (x, Q); this follows from the definitions and the fact that the automaton A has a transition from every state on every letter of X.

Given a pair $x_{init} \in X$, $s_{init} \in S$ of states, we can construct a Markov chain M' over the graph $\det(\tau_{M \times A}, (x_{init}, s_{init}))$; that is, the states and transitions of M' are those of the transition table $\det(\tau_{M \times A}, (x_{init}, s_{init}))$, and the initial state of M' is (x_{init}, s_{init}). The transition probabilities of M' are defined as follows: As we noted, every state (x, Q) of M' is a set of pairs that have the same first component x. Furthermore, for every transition $x \to x'$ of M and every state (x, Q) of M' with first component x, the chain M' has exactly one transition from (x, Q) to another state (x', Q') with first component x'. We associate with this transition of M' the same probability as that of the transition $x \to x'$ of M. Let $\mathbf{Y} = Y_0, Y_1, \ldots$ be the process of trajectories of M'. If we consider M and M' starting from initial states x_{init} and $\{(x_{init}, s_{init})\}$ respectively, we can regard them as being constructed on the same probability space so that for each trajectory $\mathbf{X} = x_{init}, X_1, X_2, \ldots$ of M there is a unique corresponding trajectory $\mathbf{Y}$ of M'; $\mathbf{Y}$ is simply the run of M' over the word $X_1 X_2 \cdots$ starting from $\{(x_{init}, s_{init})\}$. Note also that for each $\mathbf{Y}$ the corresponding $\mathbf{X}$ is the projection of $\mathbf{Y}$; X_k is the first component of the pairs of states in Y_k (they all have the same first component). By this construction a point in our probability space corresponds to a pair $(\mathbf{X}, \mathbf{Y})$ of coupled trajectories of M and M' and the construction depends on the initial states x_{init}, s_{init}. The Markov chain M_A that we mentioned earlier in the outline of our approach is the chain M' with $x_{init} = x_0$ and $s_{init} = s_0$.

We consider $\tau_{M \times A}$ as a Buchi automaton with initial state (x_0, s_0) and with set of accepting states $X \times F$, where F is the set of accepting states of A. Observe that a trajectory $\mathbf{X} = x_0 x_1 x_2 \cdots$ of M starting from the initial state x_0 corresponds to (is the projection of) some accepting run of $\tau_{M \times A}$ starting from state (x_0, s_0) if and only if the automaton A accepts the infinite word $x_1 x_2 \cdots$ generated by the trajectory $\mathbf{X}$. Thus, the automaton $\tau_{M \times A}$ accepts the intersection $L = L_\omega(M) \cap L_\omega(A)$. Clearly, the probability $P_M(L_\omega(A))$ that we want to compute is equal to the probability $P_M(L)$.

With every accepting state (x, f) of $\tau_{M \times A}$ we can associate a subset $L(x, f)$ of L: The set of words for which $\tau_{M \times A}$ has a run that repeats (x, f) infinitely often. For some accepting states, the associated set $L(x, f)$ has probability 0 in M, while for some other accepting states it may have nonzero probability. Clearly, it is only the latter accepting states that "contribute" to $P_M(L)$. We shall give a syntactic characterization of the latter states; as we shall show, they are those states of $\tau_{M \times A}$ that satisfy the following definition. Later in this subsection, we will give other equivalent characterizations.

Definition 4.1.1. A state (x, s), $x \in X$, $s \in S$ is *recurrent* if the graph of $\det(\tau_{M \times A}, (x, s))$ has a bottom strongly connected component C containing a state y for which $(x, s) \in y$. We associate with a recurrent state (x, s) a word $\gamma \in X^*$ that takes $\det(\tau_{M \times A}, (x, s))$ from $\{(x, s)\}$ to a state of the s.c.c. C.

For a recurrent state (x, s) there are in general an infinite number of words γ that satisfy the above condition; we just pick one of them arbitrarily.

For a pair of states $x_{rep} \in X$, $s_{rep} \in S$, we define the event A as the event that a trajectory of the Markov chain M starting at x_{rep} has a corresponding run of $\tau_{M \times A}$ that starts in (x_{rep}, s_{rep}) and repeats (x_{rep}, s_{rep}) infinitely often.

LEMMA 4.1.2. *Assume that (x, s) is recurrent and the Markov chain M starts in state x. Let γ be the input word associated with the recurrent state (x, s) as in Definition* 4.1.1. *Then the probability of the event A with $x_{rep} = x$, $s_{rep} = s$, conditioned on the event that M performs first the transitions in γ, is equal to one.*

PROOF. We will provide a procedure that constructs with probability one for each trajectory $\mathbf{X}$ of the chain that has as a prefix $x \cdot \gamma$ a corresponding run of $\tau_{M \times A}$ that starts in (x, s) and repeats (x, s) infinitely often. The procedure will partition $\mathbf{X}$ into an infinite number of segments, $\mathbf{X} = x\alpha_1 x\alpha_2 \cdots$ so that each α_i has γ as a prefix and there is a run of $\tau_{M \times A}$ corresponding to $\mathbf{X}$ that starts from (x, s) and visits (x, s) at the end of each segment; that is, there is a run of $\tau_{M \times A}$ over each word $\alpha_i x$ for all i that starts and ends in (x, s).

Consider the Markov chains M and M' constructed on the same probability space with initial states $x_{init} = x$, $s_{init} = s$. Let $\mathbf{X} = X_0, X_1, \ldots$ be a trajectory of the Markov chain M such that $X_0 = x$, $X_1 \cdots X_m = \gamma$ and let $\mathbf{Y} = Y_0, Y_1, \ldots$ be the corresponding trajectory of M' starting from $Y_0 = (x, s)$. Our procedure follows **Y** until the first time n_1 at which both of the following conditions are met: (a) $(x, s) \in Y_{n_1}$ and (b) $X_{n_1+1} \cdots X_{n_1+m} = \gamma$. We claim that n_1 is almost surely finite. Since (x, s) is recurrent, the table $\det(\tau_{M\times A}, (x, s))$ has a bottom strongly connected component C that contains a state y such that $(x, s) \in y$. By the definition of M', C is also a bottom s.c.c. of M'. By the choice of γ, the run of $\det(\tau_{M\times A}, (x, s))$ (and of M') starting from (x, s) over the word γ ends in a state of C. Thus, $Y_m \in C$. Since C is a bottom s.c.c. of M' it follows by ergodicity that **Y** will visit infinitely often all states in C with probability one, and hence the particular state y of C will also appear infinitely often with probability one in **Y**.

Let $\gamma = x_1 \cdots x_m$ and let $p(\gamma)$ be the probability that M starting from state x performs the transitions of γ; that is, $p(\gamma)$ is the product of the probabilities of the transitions $x \to x_1, \dots, x_{m-1} \to x_m$. Assume that $Y_k = y$ for some $k > 0$. Then $X_k = x$, and by the Markov property

$$P_M(X_{k+1} = x_1, \dots, X_{k+m} = x_m | X_k = x) = P_M(X_1 = x_1, \dots, X_m = x_m | X_0 = x)$$
$$= p(\gamma) > 0.$$

Since **Y** visits y infinitely often with probability one, it follows that the complement of the event $\{Y_k = y, X_{k+1} = x_1, \dots, X_{k+m} = x_m\}$ has zero probability and hence n_1 is finite with probability one.

We can construct now the first segment of **X**: we let $\alpha_1 = X_1, \dots, X_{n_1-1}$. Since $Y_{n_1} = y$ and $(x, s) \in y$, there is a run of $\tau_{M\times A}$ with projection $x\alpha_1 x$ that starts and ends in state (x, s). The remainder of the trajectory **X** from X_{n_1} onward has again prefix $x\gamma$.

Since we have a Markov process, we can repeat the same step on the remainder of the trajectory **X** and use the same arguments to show that the procedure will construct almost surely the second segment $\alpha_2 = X_{n_1+1}, \dots, X_{n_2-1}$ of the trajectory with the same properties; there is a run of $\tau_{M\times A}$ with projection $x\alpha_2 x$ that starts and ends in state (x, s) and the remainder of the trajectory from X_{n_2} onward has again prefix $x\gamma$. In general, if E_k denotes the event that the procedure succeeds in constructing k segments with the above properties, then the probability of E_{k+1} conditioned on E_k is 1, by the same arguments. It follows by induction that for all k the probability of E_k is 1. Hence, the probability of $\bigcap_k E_k$ is also 1, which means that the procedure will almost surely be able to partition a trajectory **X** with prefix $x\gamma$ into an infinite number of segments, and thus construct a corresponding run of $\tau_{M\times A}$ that repeats (x, s) infinitely often. □

For a pair of initial states $x_{init} \in X$, $s_{init} \in S$, and a pair of repetition states $x_{rep} \in X$, $s_{rep} \in S$, we define the event ***B*** as the event that an infinite trajectory of the Markov chain M starting at x_{init} has a corresponding run of $\tau_{M\times A}$ that starts in (x_{init}, s_{init}) and repeats (x_{rep}, s_{rep}) infinitely often.

LEMMA 4.1.3. *Assume that (x, s) is not recurrent. Then, for any pair of initial states $x_{init} \in X$, $s_{init} \in S$, the probability of **B** with $x_{rep} = x$, $s_{rep} = s$, is equal to zero.*

PROOF. We show first that if $(x_{init}, s_{init}) = (x, s)$ then $P(\boldsymbol{B}) = P(A) = 0$. Let M and M' be defined as before with initial states x and (x, s), respectively. Then, with probability one, all trajectories **Y** of M' will be eventually absorbed in some bottom strongly connected component of $\det(\tau_{M\times A}, (x, s))$ and since (x, s) is not recurrent, no state of this component contains (x, s). From this it follows that, with probability one, any run of $\tau_{M\times A}$ hits (x, s) only a finite number of times.

We prove now that $P(\boldsymbol{B}) = 0$ for arbitrary (x_{init}, s_{init}). Consider M and M' with initial states x_{init} and (x_{init}, s_{init}), respectively. We define the stopping times t_n, $n = 1, 2, \dots$ such that $t_n(\mathbf{Y})$ denotes the time at which (x, s) appeared for the nth time in an element of the sequence $\mathbf{Y}$. Consider the sequence of events $\boldsymbol{H}_n$, $n = 1, 2, \dots$, defined as follows: $\boldsymbol{H}_n = \{\mathbf{Y} | t_n(\mathbf{Y}) < \infty$ and $X_{t_n} X_{t_n+1} \cdots$ has an accepting run in the automaton $(\tau_{M\times A}, \{(x, s)\}, \{(x, s)\})\}$. Then, by using the Markov property $P(\boldsymbol{H}_n) = P(\{t_n(\mathbf{Y}) < \infty\}) P(\boldsymbol{A})$, where $\boldsymbol{A}$ corresponds to $x_{rep} = x$, $s_{rep} = s$, and since we already proved that $P(\boldsymbol{A}) = 0$, it follows that $P(\boldsymbol{H}_n)$ is equal to zero. Since $B \subseteq \bigcup_{n=1}^{\infty} \boldsymbol{H}_n$, it follows that $P(\boldsymbol{B}) = 0$. □

We can solve now the probabilistic emptiness problem.

PROPOSITION 4.1.4. *There is nonzero probability $P_M(L_\omega(A))$ that the Markov chain M generates a word accepted by the automaton A if and only if the initial state (x_0, s_0) of the automaton $\tau_{M\times A}$ can reach an accepting recurrent state (i.e., state (x, f) with $f \in F$).*

PROOF. If the initial state (x_0, s_0) of $\tau_{M\times A}$ cannot reach an accepting recurrent state, then the probability $P_M(L_\omega(A))$ is zero by Lemma 4.1.3. Conversely, suppose that the initial state (x_0, s_0) can reach an accepting recurrent state (x, f). Let β be a word that takes the automaton $\tau_{M\times A}$ from the initial state to (x, f) and let γ be a word associated with the recurrent state (x, f) as in Definition 4.1.1, that is, γ is a word that takes $\det(\tau_{M\times A}, (x, f))$ from state (x, f) to a state of a bottom strongly connected component that contains some state y such that $(x, f) \in y$. There is nonzero probability that the Markov chain starts by performing first the transitions of the word $\beta\gamma$. Conditioned on this event, by Lemma 4.1.2, the trajectory of M has almost surely a corresponding run in $\tau_{M\times A}$ that repeats (x, f) infinitely often, and thus A accepts the word that is generated by the trajectory. The proposition follows. □

We can also compute the exact probability $P_M(L_\omega(A))$. Let M_A be the chain M' with $x_{init} = x_0$, $s_{init} = s_0$; that is, M_A is the Markov chain defined by the transition table $\det(\tau_{M\times A}, (x_0, s_0))$ by associating the corresponding transition probabilities of M and having initial state (x_0, s_0). We call a bottom strongly connected component of M_A *accepting* if it has a state that contains some accepting recurrent state (x, f), $f \in F$; otherwise, we say the component is *rejecting*.

PROPOSITION 4.1.5. *$P_M(L_\omega(A))$ is equal to the probability that a trajectory of M_A hits an accepting bottom strongly connected component.*

PROOF. Consider M and M_A constructed on the same probability space as before. Any trajectory $\mathbf{Y}$ of M_A will almost surely be eventually absorbed in some bottom strongly connected component D of M_A. We shall show the following two statements which together imply the proposition:

(1) If the bottom s.c.c. D is accepting, then almost surely the automaton A accepts the word generated by the trajectory $\mathbf{X}$ of M that is coupled with $\mathbf{Y}$.

(2) If the bottom s.c.c. D is rejecting, then almost surely the automaton A rejects the word generated by the trajectory $\mathbf{X}$ of M that is coupled with $\mathbf{Y}$.

We prove statement (1). Consider a pair $\mathbf{X}, \mathbf{Y}$ such that $\mathbf{Y}$ hits a bottom strongly connected component D containing a state y such that y contains some accepting recurrent state $(x, f), f \in F$. For this (x, f), let γ be the word defined as in Definition 4.1.1. Then $P(X_{n+1} \cdots X_{n+|\gamma|} = \gamma | Y_n = y) > 0$ and by ergodicity $Y_n = y$ infinitely often with probability one. Let σ be the first time that $Y_\sigma = y$ and $X_{\sigma+1} \cdots X_{\sigma+|\gamma|} = \gamma$. By the above observation it follows that given that $\mathbf{Y}$ hits D, σ is almost surely finite. Consider the word $X_0, \ldots, X_\sigma$. Clearly, for this word, there is a run of A that starts in s_0 and ends in f. Now since $X_\sigma = x, X_{\sigma+1} \cdots X_{\sigma+|\gamma|} = \gamma$, and (x, f) is recurrent, by Lemma 4.1.2 it follows that almost surely $X_\sigma, X_{\sigma+1}, X_{\sigma+2}, \ldots$ has a corresponding run of $\tau_{M \times A}$ that starts from (x, f) and repeats (x, f) infinitely often. The above run of $\tau_{M \times A}$ can be mapped into a run of A that starts from f and repeats f infinitely often by projecting on the second component of the states of $\tau_{M \times A}$. This completes the first part of the proof.

We show statement (2). Consider a pair $\mathbf{X}, \mathbf{Y}$ such that $\mathbf{Y}$ hits a rejecting bottom strongly connected component D. We argue that the probability that $\mathbf{X}$ corresponds to an accepting run of $\tau_{M \times A}$ is 0. Suppose that $\tau_{M \times A}$ has a corresponding run starting from (x_0, s_0) that repeats some state (x, f), $x \in X$, $f \in F$, infinitely often. By the definition of the subset construction, at every point in time the state of the run of $\tau_{M \times A}$ is contained in the state of $\mathbf{Y}$ at that point. Since $\mathbf{Y}$ is absorbed in the bottom s.c.c. D, there must exist a state y of D that contains (x, f). Since D is rejecting, the state (x, f) is not recurrent. By Lemma 4.1.3, the probability that $\mathbf{X}$ has a corresponding run in $\tau_{M \times A}$ starting from (x_0, s_0) that repeats (x, f) infinitely often is 0. This completes the second part of the proof, and the proposition follows. □

Proposition 4.1.5 suggests the following algorithm for computing the probability $P_M(L_\omega(A))$:

(1) Construct the product table $\tau_{M \times A}$.
(2) Compute the set of recurrent states (x, f) of $\tau_{M \times A}$ that are accepting (i.e., $f \in F$).
(3) Construct the Markov chain M_A.
(4) Compute the probability $P_M(L_\omega(A))$ as in the statement of Proposition 4.1.5.

If we only want to solve the probabilistic emptiness problem (i.e., test whether $P_M(L_\omega(A)) = 0$), we do not need steps (3) and (4), but only have to test whether (x_0, s_0) can reach an accepting recurrent state in $\tau_{M \times A}$. If we want to test whether $P_M(L_\omega(A)) = 1$, then we do not need to compute the exact transition probabilities for M_A or to perform step (4). We only have to compute the underlying graph of M_A (i.e., the table $\det(\tau_{M \times A}, (x_0, s_0))$), and check whether all bottom strongly connected components of M_A contain an accepting recurrent state.

We analyze now the time and space complexity of this algorithm. We measure the sizes $|A|$ and $|M|$ of the automaton A and the Markov chain M by their numbers of nodes and edges. If we want to compute the exact probability, then we include in the size of M the lengths in binary of the transition probabilities as in Section 3.1. Except for Step (2), all the other steps are straightforward. The product transition table $\tau_{M\times A}$ has size at most $|M||A|$, and can be constructed in time proportional to its size. If we let n be the number of states of A, and m and e be the number of states and arcs of M, then the deterministic table $\det(\tau_{M\times A})$ has at most $m2^n$ states and $e2^n$ arcs, and can be constructed by standard methods. Step (4) involves solving a linear system of equations of dimension equal to the number of states of M_A.

Step (2) can be simplified by the following facts, which we prove below: (a) In every strongly connected component (s.c.c.) D of $\tau_{M\times A}$, either all states are recurrent or none are, and (b) to determine whether the states of D are recurrent, we may ignore the rest of $\tau_{M\times A}$ and determinize only D. Consider a state (x,s) of D and the deterministic transition table $\det(D,(x,s))$. Every state of this table is a set of pairs which have the same state of M as their first component. We say that such a state y with first component u is *fully specified* if it has a transition on each letter $v\in X$ for which M has a transition $u\to v$. Note that although every state of $\det(\tau_{M\times A},(x,s))$ is fully specified, this may not be the case for some states of $\det(D,(x,s))$: a state y of the latter table will not be fully specified if no member of y has a transition on letter v to any state of D (i.e., all its transitions in $\tau_{M\times A}$ on letter v go to states outside D).

LEMMA 4.1.6. *Let D be a strongly connected component of $\tau_{M\times A}$ and let (x,s) be a state of D. The following are equivalent*:

(1) *State (x,s) is recurrent.*
(2) *The deterministic transition table $\det(D,(x,s))$ has a bottom strongly connected component all of whose states are fully specified.*
(3) *M contains a finite path γ starting from state x such that any path $\gamma\delta$ of M that extends γ has a corresponding run within D starting from state (x,s).*
(4) *All states of D are recurrent.*

PROOF

(1) implies (2). Applying the subset construction on D starting from (x,s) is the same as applying it on $\tau_{M\times A}$, except that we ignore states that do not belong to D. It follows that the table $\det(D,(x,s))$ can be obtained from the table $\det(\tau_{M\times A},(x,s))$ by first removing all states that do not contain any element of D and then restricting the rest of the states to elements of D. From Definition 4.1.1, the table $\det(\tau_{M\times A},(x,s))$ has a bottom s.c.c. C containing a state y such that $(x,s)\in y$. Observe that if a state z can reach state y in the table $\det(\tau_{M\times A})$, then every member of y, and in particular (x,s), can be reached by some member of z in $\tau_{M\times A}$. Since every member of $\det(\tau_{M\times A},(x,s))$ can be reached from (x,s), we conclude that all members of the s.c.c. C of $\det(\tau_{M\times A},(x,s))$ (as well as all their ancestors) have a nonempty intersection with D, and therefore have corresponding states in the table $\det(D,(x,s))$. These states that correspond to the members of C form a bottom strongly connected component of $\det(D,(x,s))$; all states of this bottom s.c.c. are fully specified, since the states of C are fully specified in $\det(\tau_{M\times A},(x,s))$.

(2) implies (3). Consider a run of $\det(D,(x,s))$ from $\{(x,s)\}$ to a state of the bottom s.c.c. that satisfies condition (2), and let γ be the corresponding path of M. It is not hard to see that γ satisfies condition (3), since all states of the bottom s.c.c. are fully specified.

(3) implies (1). Consider the run of det($\tau_{M\times A}$, (x, s)) corresponding to γ and starting from $\{(x, s)\}$, and extend the run until it reaches a bottom s.c.c.; let $\gamma\delta$ be the path corresponding to the extended run, let z be the last state of the run, and C the bottom s.c.c. that contains it. From condition (3), z contains some state, say (x', s') from D. Let ζ be a string that takes $\tau_{M\times A}$ from state (x', s') to state (x, s), and let y be the state of det($\tau_{M\times A}$, (x, s)) reached from z on input ζ. Clearly, y contains (x, s) and belongs to the same bottom s.c.c. C of det($\tau_{M\times A}$, (x, s)).

(4) is equivalent to (1). Clearly, (4) implies (1). To show the converse, suppose that γ is such that condition (3) is satisfied for the state (x, s). Let (x',s') be any other state of D, and let β be a path of M that corresponds to a run of $\tau_{M\times A}$ from (x', s') to (x, s). Then any path of M that extends the path $\beta\gamma$ has a corresponding run in $\tau_{M\times A}$ which starts at (x', s') and stays within D. That is, (x', s') satisfies also condition (3), and thus is also recurrent. □

Thus, we can perform Step (2) of the algorithm as follows: First, we partition $\tau_{M\times A}$ into strongly connected components. Then, we take every s.c.c. D that contains an accepting state, pick an arbitrary state (x, s) of D (only one state), compute the table det(D, (x, s)) and test whether it satisfies condition (2) of Lemma 4.1.6. An easy calculation shows that the sum of the sizes of the tables det(D, (x, s)) (over all s.c.c.'s D of $\tau_{M\times A}$) is at most $e2^n$, where e is the number of arcs of M and n is the number of nodes of A. To see this, consider the following function which maps every transition of a table that we compute to a pair consisting of an arc of M and a subset of states of A. If det(D, (x, s)) is one of the computed tables, and (v, Q) a state of the table that has a transition on letter $w \in X$, then map this transition to the pair consisting of the arc $v \to w$ of M and the subset Q of states of A. If two transitions were mapped to the same pair $(v \to w, Q)$, then the transitions must belong to different tables det(D, (x, s)) and det(D', (x', s')) because the tables are deterministic, and furthermore the components D and D' must be distinct because we only compute (at most) one table for a component. This implies that the states (v, q), $q \in Q$, belong to two different strongly connected components of $\tau_{M\times A}$, which is contradiction. Therefore, the function we defined is one-to-one, and hence the sum of the sizes of the computed tables det(D, (x, s)) is at most $e2^n$.

It is not too hard to see also that we can determine whether a state (x, s) is recurrent using work space of $O((|A| + \log|M|)^2)$ bits. To do this, cycle over all possible states y of det($\tau_{M\times A}$) that contain (x, s), and for each one of them: (1) test whether there is a path from $\{(x, s)\}$ to y, and if so, (2) test whether y is in a bottom s.c.c. of det($\tau_{M\times A}$, (x, s)). To check (2), cycle over all possible states z of det($\tau_{M\times A}$), and for each one of them, test whether there is a path from y to z and from z to y; state y is not in a bottom s.c.c. iff for some z there is a path from y to z but not from z to y. Similarly, we can solve using the same amount of work space the probabilistic emptiness and universality problems, that is, determine whether $P_M(L_\omega(A)) = 0$ or 1.

Summarizing, we have shown:

THEOREM 4.1.7. *We can compute the probability $P_M(L_\omega(A))$ that the Markov chain M generates a word accepted by the automaton A in time exponential in $|A|$ and polynomial in $|M|$. Furthermore, we can determine if this probability is* 1 (*or* 0, *or in between*) *in time $O(|M|2^{O(|A|)})$, or in space polynomial in $|A|$ and polylogarithmic in $|M|$.*

Vardi [1985] has shown that the probabilistic emptiness problem is PSPACE-hard, and the same is true of the probabilistic universality problem. Thus, the problems are PSPACE-complete.

4.2. CONCURRENT PROBABILISTIC PROGRAMS. For this subsection, we assume that the specification is given in terms of a Buchi automaton A that accepts the undesirable computations. Thus, we are given a concurrent Markov chain M, and we want to solve the probabilistic emptiness problem: decide if $P_{M,u}(L_\omega(A)) = 0$ for all schedulers u.

This problem is considerably easier computationally, if the automaton is deterministic or even "almost" deterministic in the following sense. A Buchi automaton is *deterministic in the limit* if all the accepting states and their descendants are deterministic states. Thus, once a run goes through an accepting state, it continues deterministically. As shown in Vardi [1985] and Vardi and Wolper [1986], the probabilistic emptiness problem for such automata with respect to concurrent Markov chains can be solved in polynomial time. We will solve in this subsection the probabilistic emptiness problem by reducing it to this special case, that is, we will construct from a given Buchi automaton A with n states an equivalent Buchi automaton B with $2^{O(n)}$ states that is deterministic in the limit. Using this construction, the probabilistic emptiness problem for a concurrent Markov chain M and a Buchi automaton with n states can be solved in time $O(|M|^2 \cdot 2^{O(n)})$. As mentioned in the Introduction, essentially the same construction was obtained independently by Safra [1988]. Safra gave furthermore a *full* determinization construction which from a Buchi automaton with n states constructs a deterministic Rabin-type ω-automaton with $2^{O(n \log n)}$ states [Safra 1988]. This construction (combined with results of Vardi [1985]) yields an algorithm for the probabilistic emptiness problem with somewhat worse complexity, proportional to $2^{O(n \log n)}$ instead of $2^{O(n)}$. The "almost" deterministic construction suffices for our purpose.

First, we will prove an important structural property of Buchi automata. Let $\tau_A = (X, S, \rho_A)$ be a transition table. Recall that given a set of states $y \in 2^S$ and a word $w \in X^*$, the notation $\rho_A(y, w)$ is used to denote the set $\bigcup\{\rho_A(s, w) | s \in y\}$.

LEMMA 4.2.1. *Let $A = (\tau_A, \{s_0\}, F)$ be a Buchi automaton. An infinite word w is accepted by A if and only if w can be written as $w = w_0 w_1 w_2 \cdots$, where w_n, $n = 0, 1, 2, \ldots,$ are finite words with the following property: there exists a state $f \in F$ and a set $Q \in 2^S$ containing f such that*

(a) $f \in \rho_A(\{s_0\}, w_0)$ *and*
(b) $\rho_A(\{f\}, w_i) = \rho(Q, w_i) = Q$ *for all* $i = 1, 2, \ldots$.

PROOF. Clearly if w satisfies the above conditions, it is accepted by A. We prove now that if w is accepted by A, then (a) and (b) must hold.

Since w is accepted there is a corresponding run of A repeating some state $f \in F$ infinitely often. Let $t_1, t_2, \ldots,$ be the times that the run repeats state f and let $w(i)$ denote the ith letter in the word w. At each t_i, $i = 1, 2, \ldots,$ we start a new copy of $\det(\tau_A, f)$ to produce a run over $w(t_i + 1)w(t_i + 2) \cdots$ starting in state $\{f\}$. Let $d_{t_i}(t)$ be the state of such a run at time $t \geq t_i$ (if $\overline{w} = w(t_i + 1) \cdots w(t)$, then $d_{t_i}(t) = \rho_A(\{f\}, \overline{w})$). It is easy to observe that if $i < j$, then for all $t > t_i, t_j$, we get that $d_{t_j}(t) \subseteq d_{t_i}(t)$. Also, if $d_{t_i}(k) = d_{t_j}(k)$ for some k, i, j, then $d_{t_i}(t) = d_{t_j}(t)$ for all $t > k$.

We define now the sequence of *strictly ordered* sets $d_{\hat{t}_1}(t), d_{\hat{t}_2}(t), \ldots$, where $\hat{t}_i$, $i = 1, 2, \ldots$, is a subsequence of t_i, $i = 1, 2, \ldots$, as follows. Let $\hat{t}_1 = t_1$ and define $\hat{t}_{i+1} = \min\{t_j | t_j > \hat{t}_i, d_t(t) \subset d_{\hat{t}_i}(t)\ \forall t > t_j\}$ if the above set is nonempty, else $\hat{t}_{i+1}$ is not defined. Since $0 < \hat{t}_1 < \cdots < \hat{t}_k < t$ implies that $d_{\hat{t}_k}(t) \subset \cdots \subset d_{\hat{t}_1}(t)$, it follows that there can be most $|S|$ strictly ordered sets and hence there are finitely many $\hat{t}_i$'s. Let $\hat{t}_m$ be the largest element in this set. This $\hat{t}_m$ has the property that $\hat{t}_m \in \{t_1, t_2, \ldots\}$ and for all $t_i > \hat{t}_m$ there is some sufficiently large $k > t_i$ for which $d_{t_i}(t) = d_{\hat{t}_m}(t)$, $\forall t > k$.

Consider the infinite sequence $d_{\hat{t}_m}(t_i)$, $t_i \geq \hat{t}_m$. Since its elements are from a finite set it follows that there is a state $Q \in 2^S$ that appears infinitely often in the sequence $d_{\hat{t}_m}(t_i)$. Also since by definition f appears in the run of A at the times t_i, $i = 1, 2, \cdots$, it follows that $f \in d_{\hat{t}_m}(t_i)$, $\forall t_i > \hat{t}_m$, and hence $f \in Q$. Define the sequence $\tau_0 = \min\{t_i | t_i > \hat{t}_m, d_{\hat{t}_m}(t_i) = Q\}$, $\tau_{i+1} = \min\{t_i | t_i > \tau_i, d_{\hat{t}_m}(t_i) = d_{\tau_i}(t_i) = Q\}$, $i = 0, 1, \ldots$. Since Q appears infinitely often in $d_{\hat{t}_m}(t_i)$, $i = 1, 2, \ldots$, and all runs $d_{t_i}(t), t_i > \hat{t}_m$, will eventually coincide with $d_{\hat{t}_m}(t)$, it follows that τ_i, $i = 0, 1, \ldots$, are all finite. By defining $w_0 = w(1) \cdots w(\tau_0)$, and $w_{i+1} = w(\tau_i + 1) \cdots w(\tau_{i+1})$, $i = 0, 1, \ldots$, the proof of the lemma follows. □

From a Buchi automaton $A = (\tau_A, s_0, F)$ we construct a new Buchi automaton B that is deterministic in the limit and accepts the same language. The automaton B consists of two parts C and D. The first part C is just A itself with all the states nonaccepting and the same initial state s_0, which is also the initial state of B. (We could take also C to be the result of applying the subset construction on A.) The second part D has states corresponding to pairs (P, Q) of sets of states of A. For each $f \in F$, there is an ε transition from state f (in C) to state $(\{f\}, \{f\})$ of D. (If desired, ε transitions to a state can be replaced in the usual way by transitions to the appropriate successors.) The accepting states of automaton B are the states of the form (P, P), that is, with equal first and second components. (We could also take as accepting only those states (P, P) where $P \cap F \neq \emptyset$.) The transitions from a state (P, Q) are as follows. The second component of a pair follows the usual subset construction; that is, on letter a the state (P, Q) goes to a state (P', Q') where $Q' = \rho_A(Q, a) = \bigcup\{\rho_A(s, a) | s \in Q\}$. There are two cases for the first component: if the state is not accepting (i.e., $P \neq Q$), then $P' = \rho_A(P, a) \cup \rho_A(Q \cap F, a)$; if the state is accepting (i.e., $P = Q$) then $P' = \rho_A(Q \cap F, a)$ (we define $\rho_A(\emptyset, a) = \emptyset$ for all a).

PROPOSITION 4.2.2. *The Buchi automata A and B accept the same language.*

PROOF. We prove first that if a word is accepted by B it is also accepted by A. Consider such an infinite word $w = w(0)w(1) \cdots$. We will show that A has a run over w which is accepting.

Since w is accepted by B, there is an accepting run of B over w repeating states of the form (P, Q), $P = Q$, infinitely often. Assume that this occurs at the times $t_1, t_2, \ldots$, where t_1 corresponds to the first time the run starts visiting states of the automaton D. We denote by $(P(t), Q(t))$ the state of the run at time $t \geq t_1$. Consider the time interval between t_k and t_{k+1} for some $k \geq 1$. It is easy to show that the construction of D implies the following property for all $t = t_k + 1, \ldots, t_{k+1}$: for any state $s \in P(t)$, there exists a run of A over the word $w(t_k) \cdots w(t-1)$ starting from some state in $Q(t_k)$ and ending in state s, such that the run visits some state in F. This property can be shown by an induction on t (starting from $t_k + 1$) using the definition of the transition function ρ_A. For $t = t_k + 1$, since $P(t_k) = Q(t_k)$, we have $P(t) = \rho_A(Q(t_k) \cap F, w(t_k))$ and the property holds. For the inductive step, a state s of $P(t)$ either belongs to $\rho_A(P(t-1), w(t-1))$ or to $\rho_A(Q(t-1) \cap F, w(t-1))$. In the first case, the claim follows from the induction hypothesis. In the second case, there is a run of A over the word $w(t_k) \cdots w(t-2)$ from some state of $Q(t_k)$ to a state $f \in Q(t-1) \cap F$ and f has a transition to s on letter $w(t-1)$, proving the claim. Consequently, for any state s in $Q(t_{k+1})$, $k \geq 1$, there exists a run of A over $w(t_k) \cdots w(t_{k+1} - 1)$ starting from some state in $Q(t_k)$ and ending in state s, which contains some state in F.

We define an infinite tree T as follows: At level 0, the tree has a single node (root) labeled s_0. At level k, the tree has $|Q(t_k)|$ nodes; each node being labeled with a distinct state in $Q(t_k)$. With each node at level $k+1, k \geq 0$, we associate a parent node from level k as follows: Level 1 has a single node whose parent is the root node. For $k > 0$, let s be the label of the child node at level $k+1$. We define its parent to be any node in level k whose label s' has the property that there exists a run of A over $w(t_k) \cdots w(t_{k+1} - 1)$ starting from state s' in $Q(t_k)$ and ending in state s in $Q(t_{k+1})$, which goes through some state of F (if more than one such nodes exist at level k, we pick one among them). By the previous remark, a node has always a parent; hence, the tree T is well defined. One can observe that any path of the tree T starting from the root defines a run of the automaton A over w with the property that if the path has length greater than k, then the run visits at least $k-1$ times states in F (if the path is finite and has length k it defines a run of A over $w(0) \cdots w(t_k - 1)$). Since T is infinite and has finite branching (bounded by $|S|$), it follows by König's lemma that it has an infinite path. By the previous observation, the run corresponding to this path visits states in F an infinite number of times and hence is an infinite run of A over w which is accepting.

It remains to prove that if a word is accepted by A it is also accepted by B. Consider such an infinite word w accepted by A and use Lemma 4.2.1 to write w as $w_0 w_1 \cdots$ and to define the $f_w \in F$ and $Q_w \in 2^S$ associated with the above w. Let $t_1, t_2, \ldots$ be the times corresponding to the beginning of the words $w_1, w_2, \ldots$ in w. We will construct now an accepting run of B over w. The first piece of the run corresponds to the word w_0 and ends in the state $(\{f_w\}, \{f_w\})$ of D. Such a run clearly exists since by virtue of Lemma 4.2.1, $f_w \in \rho_A(s_0, w_0)$. Since D is deterministic, it remains to show that the run of D over $w_1 w_2 \cdots$ will visit states of the form (P, Q), $P = Q$, infinitely often. Assume that this is not the case. Then there exists a time k after which no such state is visited by the run. Let t_m be the smallest element of the sequence $t_1, t_2, \ldots$ as defined above for which $k < t_m$. We will show that the run will visit an accepting state at some time t such that $t_m < t \leq t_{m+1}$, and hence get a contradiction.

Assume that for all times $t, t_m \leq t < t_{m+1}$, the state $(P(t), Q(t))$ of the run is such that $P(t) \neq Q(t)$. We show that this implies $P(t_{m+1}) = Q(t_{m+1})$ as follows. We observe that $Q(t_m) = Q(t_{m+1}) = Q_w$, $f_w \in Q(t_m)$, and $Q_w = \rho_A(\{f_w\}, w_m)$ by virtue of Lemma 4.2.1. The above observation together with the definition of the automaton D and the assumption that $P(t) \neq Q(t)$, $t_m \leq t < t_{m+1}$, imply that $Q_w \subseteq P(t_{m+1})$. But as we noted before $Q(t_{m+1}) = Q_w$, and by the construction of D we have $P(t_{m+1}) \subseteq Q(t_{m+1})$. This implies $P(t_{m+1}) = Q_w = Q(t_{m+1})$ and the proof is completed. □

We will describe briefly now for completeness how one can solve the probabilistic emptiness problem for a Buchi automaton B that is deterministic in the limit and a concurrent Markov chain M [Vardi 1985; Vardi and Wolper 1986]; see also Pnueli and Zuck [1986]. As in the previous subsection, we will assume, without loss of generality, that the alphabet of the automaton B is X (the state space of M), that the automaton has a transition from every state on every letter of X, that it has initial state s_0 and set of accepting states F, and that the Markov chain has a unique initial state x_0. We view again M as an automaton that generates words of X and use $P_{M,u}(L_\omega(B))$ to denote the probability that the concurrent Markov chain M with scheduler u generates a word accepted by B. Form the product transition table $\tau_{M\times B}$, which we view again as a Buchi automaton with initial state (x_0, s_0) and set of accepting states $X \times F$. Note that $\tau_{M\times B}$ is also deterministic in the limit. We consider the states of $\tau_{M\times B}$ as being partitioned into nondeterministic and randomizing states according to their first component (a state of M). The algorithm is based on the following characterization:

PROPOSITION 4.2.3. *Let B be a Buchi automaton that is deterministic in the limit and let M be a concurrent Markov chain as above. There is a scheduler u such that $P_{M,u}(L_\omega(B)) > 0$ if and only if the initial state (x_0, s_0) of $\tau_{M\times B}$ can reach an accepting state (x, f) (i.e., with $f \in F$) and $\tau_{M\times B}$ has a subset D of states containing (x, f) which satisfies the following three conditions*:

(1) *the subgraph of $\tau_{M\times B}$ induced by D (denoted $\tau_{M\times B}[D]$) is strongly connected,*
(2) *the subgraph $\tau_{M\times B}[D]$ is nontrivial, that is, does not consist of a single node without any edge,*
(3) *the table $\tau_{M\times B}$ does not have any transition out of D corresponding to a probabilistic transition of M, that is, if (y, t) is a state of D and y is a randomizing state of M, then all the immediate successors of (y, t) in $\tau_{M\times B}$ are also in D.*

PROOF. Assume that there is an accepting state (x, f) of $\tau_{M\times B}$ and a subset D containing (x, f) that satisfies the above three conditions. We will describe a scheduler u such that $P_{M,u}(L_\omega(B)) > 0$. Let β be a word that takes $\tau_{M\times B}$ from the initial state (x_0, s_0) to (x, f), and let T be a directed spanning tree of the subgraph $\tau_{M\times B}[D]$ rooted at (x, f) with all arcs directed toward the root. Since $\tau_{M\times B}[D]$ is strongly connected there exists such a tree. The scheduler u operates as follows: At the beginning it tries to form the word β; that is, if the trajectory is at a nondeterministic state and the prefix so far agrees with the prefix of β, then u chooses the transition according to the next letter of β. With nonzero probability the scheduler u will succeed in this initial part to form β, which means that there is a corresponding run of $\tau_{M\times B}$ that arrives at (x, f). Assume from now on that this is the case and let us concentrate on the remainder of the trajectory, which we denote by $\tilde{\mathbf{X}} = \tilde{X}_0, \tilde{X}_1, \ldots$. Let $\tilde{\mathbf{Y}} = \tilde{Y}_0, \tilde{Y}_1, \ldots$ be the corresponding run of $\tau_{M\times B}$ starting from (x, f); the run is unique because $\tau_{M\times B}$ is deterministic in the limit.

The scheduler u follows $\tilde{\mathbf{Y}}$ and acts in the following way, which ensures that the run stays always within the subset D. Suppose that $\tilde{\mathbf{Y}}$ is at a nondeterministic state (y, q) of D. If $(y, q) \neq (x, f)$, then the scheduler u chooses the transition from (y, q) to its parent in T, and if $(y, q) = (x, f)$, then u chooses a transition to some arbitrary state of D. By condition (3), the run stays in D.

Let us say that step $i = 0, 1, \ldots$ of the run is successful if the arc $(\tilde{Y}_i, \tilde{Y}_{i+1})$ is in the tree T, or if $\tilde{Y}_i$ is the root (x, f) and $\tilde{Y}_{i+1}$ is any node of D. Clearly, if at some point there are n consecutive successful steps, then the run goes at least once through state (x, f) during this time. At every point in time, there is nonzero probability that the step will be successful, and indeed that the next n steps will be successful. It is not hard to show then by standard arguments that with probability one, there will be n consecutive successful steps, and in fact this will happen infinitely often. (If E_k, $k \geq 1$, denotes the event that there are k segments, each consisting of n or more consecutive successful steps, then for every k the conditional probability of E_{k+1} given E_k is 1; therefore, $P_{M,u}(E_k) = 1$ for all k, and hence $P_{M,u}(\bigcap_k E_k) = 1$.) Consequently, with probability 1, the run $\tilde{\mathbf{Y}}$ goes infinitely often through (x, f). It follows that $P_{M,u}(L_\omega(B)) > 0$.

Suppose now that $P_{M,u}(L_\omega(B)) > 0$ for some scheduler u. If (x, f) is an accepting state of $\tau_{M\times B}$, D is a subset of states containing (x, f), and $\beta \in X^*$ is a string that takes $\tau_{M\times B}$ from (x_0, s_0) to (x, f), we let $H((x, f), D, \beta)$ be the event that a trajectory $\mathbf{X}$ of M has prefix β and corresponds to a run of $\tau_{M\times B}$ that is at state (x, f) after β and has infinity set equal to D. Clearly, the union of these events contains all the accepted trajectories. If all these events had probability 0, then the same would be true of their (countable) union. Since $P_{M,u}(L_\omega(B)) > 0$, there exist (x, f), D and β such that $P_{M,u}(H((x, f), D, \beta)) > 0$. Let $\tilde{M}$ be the concurrent Markov chain that is identical to M except that it starts at state x, and let $\tilde{u}$ be the scheduler that maps every string $\sigma \in X^*$ and nondeterministic state y to $\tilde{u}(\sigma, y) = u(\beta\sigma, y)$. Let K be the set of trajectories $\tilde{\mathbf{X}}$ of $\tilde{M}$ whose corresponding run $\tilde{\mathbf{Y}}$ in $\tau_{M\times B}$ starting from (x, f) has infinity set D. Then $P_{M,u}(H((x, f), D, \beta)) = P_{M,u}(\beta \text{ is a prefix of } \mathbf{X}) \cdot P_{\tilde{M},\tilde{u}}(K)$, and thus, $P_{\tilde{M},\tilde{u}}(K) > 0$. By definition, D contains (x, f), and since D is the infinity set of a run, the subgraph $\tau_{M\times B}[D]$ is strongly connected and nontrivial (thus satisfies conditions 1 and 2). Suppose that $\tau_{M\times B}$ has an arc from a randomizing state (y, t) of D to a state (z, q) outside D. This arc corresponds to a transition $y \to z$ of $\tilde{M}$. Every trajectory $\tilde{\mathbf{X}}$ of K corresponds to a run $\tilde{\mathbf{Y}}$ of $\tau_{M\times B}$ that visits infinitely often (y, t). Every time that $\tilde{\mathbf{Y}}$ is in state (y, t) there is nonzero probability that $\tilde{M}$ will follow the transition $y \to z$, thereby forcing the run to exit D. Since $\tilde{\mathbf{Y}}$ visits (y, t) infinitely often, the probability that this will occur at some point is 1, which implies $P_{\tilde{M},\tilde{u}}(K) = 0$, contradicting our assumption. It follows that D satisfies also condition 3. □

We can test for the conditions of this characterization by the following algorithm.

(1) Construct the table $\tau_{M\times B}$ and let G be the subgraph that is reachable from the initial state (x_0, s_0).

(2) Repeat the following steps until either the current graph G becomes empty or the algorithm halts with success.

  (2a) Find the strongly connected components of the current graph G. Remove from G all the arcs that connect different s.c.c.'s, and remove all the s.c.c.'s that are trivial or do not contain any accepting state.

  (2b) Let Q be the set of nodes (y, t) of G such that y is a randomizing state of M and an arc out of (y, t) was removed in step 2a. If Q does not intersect some (remaining) s.c.c. of G, then halt with success; otherwise, remove from G the nodes of Q.

It is easy to see that the algorithm halts with success if and only if there is an accepting state (x, f) and a set D that contains it and satisfies conditions (1)–(3). Every execution of the loop can be implemented to run in linear time in the size of $\tau_{M \times B}$, and the number of iterations is certainly no greater than the number of nodes of $\tau_{M \times B}$. If we start with an arbitrary Buchi automaton A with n states, then we can construct the automaton B that is deterministic in the limit and has size $2^{O(n)}$. If the Markov chain M has m nodes and e arcs, then $\tau_{M \times B}$ has no more than $m2^{O(n)}$ nodes and $e2^{O(n)}$ arcs. Thus, the complexity of the algorithm is at most $O(me2^{O(n)})$.

The results of Section 3.2 imply that solving the probabilistic emptiness problem for a (general) Buchi automaton and a concurrent Markov chain requires exponential time. We know that we can construct from a PTL formula f an exponentially larger Buchi automaton A for $\neg f$. By Corollary 3.2.2 determining whether a concurrent Markov chain satisfies f requires double exponential time in $|f| + |M|$. Thus, the probabilistic emptiness problem for M and the automaton A requires single exponential time, more precisely it requires time $2^{\Omega((|A|+|M|)^d)}$ for some constant d that depends on the constant c of Corollary 3.2.2 and the translation from a PTL formula to an automaton. It is possible to give also a direct reduction from ASPACE$(n) = \bigcup_c$ DTIME(c^n) showing that the lower bounds holds for $d = 1$, that is, the problem requires in fact time $2^{\Omega(|A|+|M|)}$. Specifically, given an alternating Turing machine T that uses linear space and an input x of length n, we can construct a Buchi automaton A and a concurrent Markov chain M such that T accepts x if and only if there is a scheduler u such that $P_{M,u}(L_\omega(A)) > 0$, and both A and M have size $O(n)$. This fact implies as in Corollary 3.2.2 that there is a constant c such that the probabilistic emptiness problem cannot be solved in time $O(2^{c(|A|+|M|)})$. We do not include the reduction, to keep the length of the paper within reasonable bounds. The construction is similar to that of Theorem 3.2.1; it is somewhat simpler, though it requires some care to ensure that A and M have linear size. Summarizing, we have:

THEOREM 4.2.4. *The probabilistic emptiness problem for a Buchi automaton A and a concurrent Markov chain M can be solved in time $O(|M|^2 \cdot 2^{O(|A|)})$. Furthermore, the problem requires exponential time in the total input size $|A| + |M|$.*

Remark. It is possible to perform a more refined analysis of concurrent Markov chains, similar to that of the simpler case of sequential chains. This analysis leads to an alternative algorithm of roughly the same complexity, and it can be further used for quantitative analysis; this direction is pursued in Courcoubetis and Yannakakis [1990] in the context of designing optimization strategies for Markov decision processes.

We could have used the transformation of this subsection to a Buchi automaton that is deterministic in the limit in order to solve the probabilistic emptiness problem also in the case of sequential programs (ordinary Markov chains). The probabilistic emptiness problem for a (ordinary) Markov chain M and a Buchi automaton B that is deterministic in the limit can be solved in time $O(|M| \cdot |B|)$ or in polylogarithmic space [Vardi 1985]. (Observe, that in this case the part of the automaton $\tau_{M \times B}$ that is reachable from an accepting state is deterministic, and therefore, the condition of Proposition 4.1.4 simplifies to the following condition: $P_M(L_\omega(B)) > 0$ if and only if the initial state (x_0, s_0) of $\tau_{M \times B}$ can reach a bottom strongly connected component that contains an accepting state.) This approach would give an alternative algorithm for the probabilistic emptiness problem of Section 4.1, also of complexity $O(|M|2)^{O(|A|)}$ (with somewhat larger constant in the exponent).

The analysis of the previous subsection will play a central role in the next section where we examine the Extended Temporal Logic. We note that the transformation of Proposition 4.2.2 does not help with the problem of computing the probability $P_M(L_\omega(A))$ or of just testing whether this probability is 1 (the probabilistic universality problem): It can be shown that the probabilistic universality problem for a (ordinary) Markov chain and a Buchi automaton that is deterministic in the limit is PSPACE-hard. Further, in the case of concurrent Markov chains, the probabilistic universality problem for a Buchi automaton that is deterministic in the limit (i.e., testing whether $P_{M,u}(A) = 1$ for all schedulers u) requires exponential time.

5. *Extended Temporal Logics*

In Wolper [1983], it was shown that standard temporal logic cannot express certain properties that can be expressed by automata. For this reason, it was extended using automata as temporal connectives. Three such logics were defined in Wolper et al. [1983]. Formulas in these logics are built from a finite set *Prop* of atomic propositions using Boolean connectives and temporal connectives corresponding to automata. The most general and powerful of the logics uses Buchi automata. Let $A = (\tau_A, s_1, F)$ be a Buchi automaton over the alphabet $\Sigma = \{a_1, \dots, a_l\}$. If $f_1, \dots, f_l$ are formulas of the extended temporal logic (ETL), then $A(f_1, \dots, f_l)$ is also a formula of ETL. Satisfaction of this formula by a computation π at time instant i is defined as follows. $\pi, i \models A(f_1, \dots, f_l)$ iff there is an infinite word $w = a_{j_0} a_{j_1} \cdots$ accepted by A, such that $\pi, i + k \models f_{j_k}$ for all $k \geq 0$.

It was shown in Sistla et al. [1987] that from a formula f of the extended temporal logic, one can build a Buchi automaton of size $2^{O(|f|^2)}$ that accepts the same language, where $|f|$ includes the sizes of the automata appearing in the formula. (In the case of the two other simpler extensions, the size of f appears linearly in the exponent [Wolper et al. 1983].) Thus, by Theorem 4.2.4, we can test if a concurrent probabilistic program satisfies an ETL formula in time quadratic in the size of the program and doubly exponential in the square of the formula. In the case of sequential probabilistic programs, we could follow a similar approach and use Theorem 4.1.7. However, the algorithm we would thus obtain would be doubly exponential. As we shall see, we can do much better. In the rest of this section, we will combine the techniques of Sections 3.1 and 4.1, and with some more work show the following result.

THEOREM 5.1. *We can test if a finite state sequential probabilistic program M satisfies a formula f of the extended temporal logic in time $O(|M|2^{O(|f|)})$, or in space polynomial in f and polylogarithmic in M. We can compute the probability $P_M(L_\omega(f))$ of satisfaction in time exponential in f and polynomial in M.*

Let f be a formula of ETL over a set *Prop* of propositions, M a (sequential) Markov chain with a finite set of states X and V: $X \to 2^{Prop}$ a valuation. We wish to compute the probability $P_M(L_\omega(f))$ that a trajectory of M satisfies the formula f. We use a technique similar to that of Section 3.1 for PTL. We take an innermost subexpression $A(f_1, \dots, f_l)$ of f, replace it with a new atomic proposition ξ to obtain a new formula f' with one less temporal operator, and construct a new Markov chain M' such that $P_M(L_\omega(f)) = P_{M'}(L_\omega(f'))$. For verification purposes, that is, to tell whether this probability is 0, 1, or inbetween, the transition probabilities of M and M' are not important; only the underlying graphs are important. However, we will use the transition probabilities to prove correctness of our method.

Since $A(f_1, \ldots, f_l)$ is an innermost temporal subexpression of f, the f_i's are composed of atomic propositions and Boolean connectives, and can be evaluated at all the nodes of M. As in Section 4.1, we will regard A in the following as being an automaton over the alphabet X (the set of states of the Markov chain M) rather than alphabet Σ; for $x \in X$, there is a transition from state s to state s' on letter x iff the automaton with alphabet Σ has a transition from s to s' on some $a_i \in \Sigma$ such that x satisfies f_i. By this definition, a trajectory $\mathbf{X} = X_0 X_1 \cdots$ of M satisfies the formula $A(f_1, \ldots, f_l)$ iff the automaton A (with alphabet X) accepts the infinite word $\mathbf{X}$. We assume without loss of generality that A has a transition on every letter $x \in X$ for every state.

We will not add here a new initial state to the chain M. As in Section 4.1, we define the product transition table $\tau_{M\times A}$ and the deterministic table $\det(\tau_{M\times A})$. Recall that every state of $\det(\tau_{M\times A})$ is a set consisting of pairs all of which have the same node of M as their first component. Also, every run of $\tau_{M\times A}$ (or $\det(\tau_{M\times A})$) projected on the first component of the states gives a trajectory of M; we say that the run corresponds to this trajectory of M. A run of $\tau_{M\times A}$ is *accepting* if it repeats infinitely often some state (x, f) with $f \in F$. Note that, if $\mathbf{X} = x_0 x_1 x_2 \cdots$ is a trajectory of M and s is a state of A, the automaton A has an accepting run over the infinite word $x_1 x_2 \cdots$ starting from state s iff $\tau_{M\times A}$ has an accepting run corresponding to $\mathbf{X}$ starting from state (x_0, s).

Let $s_1, \ldots, s_n$ be the states of A, and let $\mathbf{X} = x_0 x_1 x_2 \cdots$ be a trajectory of M starting at an arbitrary node x_0. We define the *type* t of $\mathbf{X}$ to be a Boolean (0–1) vector $t_1 \cdots t_n$ of length n, where for each i, $t_i = 1$ iff $\tau_{M\times A}$ has an accepting run starting at (x_0, s_i) which corresponds to $\mathbf{X}$; equivalently, the automaton A with s_i as its initial state accepts the infinite word $x_1 x_2 \cdots$. It is easy to see that, if we know the type, say r of the suffix $x_1 x_2 \cdots$, then we can uniquely determine the type t of $x_0 x_1 x_2 \cdots$, as follows: For each i, the automaton A accepts $x_1 x_2 \cdots$ starting from state s_i iff there is a $s_j \in \rho_A(s_i, x_1)$ such that A accepts $x_2 x_3 \cdots$ starting from state s_j. That is, for all $i = 1, \ldots, n$, the following equation holds:

$$t_i = \bigvee \{ r_j | s_j \in \rho_A(s_i, x_1) \}. \qquad (*)$$

We construct a graph G, as follows: The nodes of G are all pairs (x, t) where x is a node of M and t is a Boolean n-vector. We include an arc $(x_0, t) \to (x_1, r)$ between two nodes of G iff t and r satisfy the condition $(*)$ above and M has an arc $x_0 \to x_1$. Note that every node (x_1, r) of G has exactly one immediate predecessor in G with first component x_0 for every immediate predecessor x_0 of x_1 in M; the reason is that t is determined uniquely in $(*)$ from x_1 and r.

By the construction of G, every finite or infinite path of G projected on the first component of its nodes, is a path of M; we let g denote this projection mapping from paths of G to paths of M. Conversely, every path in M is the projection of at least one path in G. To see this, let $\mathbf{X} = x_0 x_1 x_2 \cdots$ be an infinite path of M (any finite path can be extended to an infinite one). For each j, let t^j be the type of the suffix $x_j x_{j+1} \cdots$. By the construction of G, for each j it contains an arc $(x_j, t^j) \to (x_{j+1}, t^{j+1})$. Therefore, the path $\mathbf{X}$ of M is the projection of the path $(x_0, t^0)(x_1, t^1) \cdots$; we call this path of G, the *augmented path* corresponding to $\mathbf{X}$.

For a node (u, t) of G, let $P_M(u, t)$ be the probability that a trajectory of M starting at u has type t. We say that (u, t) is *probable* if $P_M(u, t) > 0$; otherwise, it is *improbable*. Consider a trajectory $\mathbf{X} = X_0, X_1, X_2, \ldots$ with $X_0 = u$, $X_1 = v$. If $\mathbf{X}$ has type t then the suffix $X_1, X_2, \ldots$ has type r with the property that G contains the arc $(u, t) \to (v, r)$; conversely, if the suffix $X_1, X_2, \ldots$ has type r and the graph G contains the arc $(u, t) \to (v, r)$, then $\mathbf{X}$ has type t. Therefore, $P_M(u, t) = \sum_{(u,v) \to (v,r)} p_{uv} P_M(v, r)$, where p_{uv} is the probability of the transition $u \to v$ in M and the sum extends over all arcs $(u, v) \to (v, r)$ out of node (u, t) in G. If G contains an arc $(u, t) \to (v, r)$ and (v, r) is probable, then $P_M(u, t) \geq p_{uv} P_M(v, r) > 0$ and thus (u, t) is also probable. Therefore, if a node of G is probable, then so are all its ancestors. This implies in particular that for every strongly connected component C of G, either all nodes of C are probable or none is.

The new Markov chain M' is defined as follows: The underlying graph of M' is the subgraph of G induced on the probable nodes. The probability of a transition $(u, t) \to (v, r)$ in M' is defined to be $p_{uv} P_M(v, r)/P_M(u, t)$. Note that for every node (u, t) of M', the sum of the probabilities of all the transitions out of (u, t) is equal to 1 because

$$P_M(u, t) = \sum_{(u,v) \to (v,r)} p_{uv} P_M(v, r).$$

If $p_0(u)$ is the probability of u in the initial distribution of M, then the probability of (u, t) in the initial distribution of M' is $p_0(u, t) = p_0(u) P_M(u, t)$. Finally, the set *Prop'* of propositions for M' consists of the old set *Prop* of propositions for M and a new proposition ξ. A node (u, t) of M' satisfies a proposition in *Prop* iff u does, and satisfies the new proposition ξ iff there is a j such that $t_j = 1$ and $s_j \in \rho_A(s_1, u)$. (Recall, s_1 is the initial state of A.) Obviously, a trajectory $\mathbf{X}'$ of M' starting at (u, t) satisfies the formula $A(f_1, \ldots, f_l)$ iff its projection $g(\mathbf{X}')$ satisfies it. Furthermore, it is easy to see that by our definitions, if $g(\mathbf{X}')$ has type t, then $\mathbf{X}'$ satisfies $A(f_1, \ldots, f_l)$ iff ξ is true at node (u, t). As we will see later, the projection of almost all trajectories of M' starting at node (u, t) have type t.

Let $\mathbf{X} = X_0, X_1, \ldots,$ be a trajectory of M, let $\mathbf{X}' = X'_0, X'_1, \ldots,$ be a trajectory of M', and let $\mathbf{Y} = Y_0, Y_1 \cdots$ be its projection $g(\mathbf{X}')$ (if $X'_i = (u, t)$, then $Y_i = u$).

LEMMA 5.2. *The processes* $\mathbf{X}$ *and* $\mathbf{Y}$ *have the same distribution.*

The proof of the lemma is similar to that of Lemma 3.1.1.2 and is omitted. We still have to show how we can determine which nodes of G are probable (are in M'), and to prove the analogue of Lemma 3.1.1.3 for this setting.

As in Section 4.1, we say that a bottom s.c.c. of $\det(\tau_{M \times A})$ is *accepting* if it has some state which contains an accepting recurrent state of $\tau_{M \times A}$, and it is *rejecting* otherwise. We know from Section 4.1 that for any node $x \in X$ and state s of A, a trajectory $\mathbf{X}$ of M starting at node x has the following properties with probability one: (i) the corresponding run of $\det(\tau_{M \times A})$ starting at $\{(x, s)\}$ hits eventually a bottom strongly connected component, and (ii) this bottom s.c.c. is accepting if and only if the table $\tau_{M \times A}$ has a corresponding accepting run starting from (x, s). We say that a trajectory $\mathbf{X}$ of M is *typical* if (1) $\mathbf{X}$ satisfies these properties, and (2) all pairs of the augmented path of G corresponding to $\mathbf{X}$ are probable (and thus are in M'). It follows easily from the Markov property that for any node x of M, almost all trajectories starting at x are typical. Note that by condition (2), every typical trajectory of M is the projection of some trajectory of M', namely of the augmented path corresponding to $\mathbf{X}$.

The following lemma gives a necessary (but not sufficient) condition for a pair (u, t) to be probable.

LEMMA 5.3. *If a pair (u, t) is probable, then it satisfies the following conditions*:

(*a*) *For each i such that $t_i = 1$, the state (u, s_i) of $\tau_{M \times A}$ can reach some accepting recurrent state.*

(*b*) *For every state y that is in an accepting bottom s.c.c. of $det(\tau_{M \times A})$ and has first component u, there is an i such that $(u, s_i) \in y$ and $t_i = 1$.*

PROOF. Condition (a) follows immediately from Lemma 4.1.3. Suppose that a state y in an accepting bottom s.c.c. C of $\det(\tau_{M \times A})$ violates condition (b). Let z be a state of C which contains some recurrent state (x, f) with $f \in F$, and let $\gamma \in X^*$ be the finite word as in Definition 4.1.1 for this recurrent state; recall that by Lemma 4.1.2 almost all trajectories of M which start from state x and have prefix γ have a corresponding accepting run in $\tau_{M \times A}$ starting from (x, f). Consider a trajectory $\mathbf{X} = u, X_1, X_2, \dots$ of M starting from u and the corresponding run of $\det(\tau_{M \times A})$ starting from state y. As in the proof of Proposition 4.1.5 with probability 1, there is a finite time σ such that the run of $\det(\tau_{M \times A})$ is in state z at time σ, and the remainder of $\mathbf{X}$ has prefix γ. Thus, for some $(u, s_i) \in y$, there is a run of $\tau_{M \times A}$ from (u, s_i) to (x, f) corresponding to the prefix $u, X_1, \dots, X_\sigma$ of $\mathbf{X}$. Since the following substring of $\mathbf{X}$ is γ, with probability 1 this run can be extended to an accepting run of $\tau_{M \times A}$. Therefore, for almost all trajectories $\mathbf{X}$ of M starting at u, the type t of $\mathbf{X}$ has $t_i = 1$ for at least some i such that $(u, s_i) \in y$. □

LEMMA 5.4. *Let α be a finite path of M which corresponds to a run of $det(\tau_{M \times A})$ that starts at state (u, s_i) and ends in a bottom s.c.c. C. If M′ contains a path that starts at (u, t) and has projection α, then $t_i = 1$ if C is an accepting s.c.c., and $t_i = 0$ if C is rejecting.*

PROOF. Let (v, q) be the last node of a path of M' that starts at (u, t) and has projection α, and let y be the last node of the run of $\det(\tau_{M \times A})$ corresponding to α which starts at state $\{(u, s_i)\}$. Suppose that C (the s.c.c. of y) is a rejecting bottom s.c.c. Since C is rejecting, no element of y can reach in $\tau_{M \times A}$ an accepting recurrent state. Since (v, q) is a node of M', and thus is probable, from Lemma 5.3(a) we know that $q_j = 0$ for all j such that $(v, s_j) \in y$. From eq. $(*)$ in the construction of G, it follows by an easy induction on the length of α that

$$t_i = \bigvee \{q_j | (v, s_j) \in y\}. \qquad (**)$$

Thus, $t_i = 0$.

Suppose that C is an accepting bottom s.c.c. Since (v, q) is probable, we know from Lemma 5.3(b) that $q_j = 1$ for some j such that $(v, s_j) \in y$. It follows from the eq. $(**)$ that $t_i = 1$. □

We can prove now the analogue of Lemma 3.1.1.3 for automata connectives.

LEMMA 5.5. *A trajectory* $\mathbf{X}' = (X_0, t^0), (X_1, t^1), \ldots$ *of* M' *is with probability* 1 *the augmented trajectory of its projection* $\mathbf{X} = X_0, X_1, \ldots$ *in* M. *That is, for all* $k \geq 0$, *the type of* $X_k X_{k+1} \cdots$ *is* t^k, *and* $A(f_1, \ldots, f_l) \equiv \xi$ *holds at all times in* $\mathbf{X}'$.

PROOF. It suffices to show that for any node (u, t) of M', the type of the projection $g(\mathbf{X}')$ of almost all trajectories $\mathbf{X}'$ starting at (u, t) is t. Let us say that a trajectory of M' is *typical* if its projection, a trajectory in M, is typical. We know that for any node u of M, the probability (in M) that a trajectory starting at u is typical is 1. It follows from Lemma 5.2 that the same is true of M': for any node (u, t) of M', the probability (in M' now) that a trajectory starting at (u, t) is typical is 1. We shall show that for any typical trajectory of M' starting at (u, t), its projection has type t.

Consider such a typical trajectory $\mathbf{X}'$, and let $\mathbf{X} = g(\mathbf{X}')$ be its projection and r be the type of $\mathbf{X}$. Since $\mathbf{X}'$ is typical, for each $i = 1, \ldots, n$, the run of $\det(\tau_{M \times A})$ that corresponds to $\mathbf{X}$ starting from $\{(u, s_i)\}$ ends in a bottom s.c.c. C which is rejecting or accepting according as $r_i = 0$ or 1. Let α be a (finite) prefix of $\mathbf{X}$ such that after α the run is in a state of C. Since α is the projection of a path of M' starting at (u, t), we know from Lemma 5.4 that $t_i = 0$ or 1 according as C is rejecting or accepting. That is, $t = r$, the type of $\mathbf{X}$. □

PROPOSITION 5.6. $P_M(L_\omega(f)) = P_{M'}(L_\omega(f'))$.

PROOF. A trajectory $\mathbf{X}'$ of M' satisfies the formula f if and only if its projection $g(\mathbf{X}')$ satisfies it. Thus, Lemma 5.2 implies $P_M(L_\omega(f)) = P_{M'}(L_\omega(f))$. From Lemma 5.5, with probability 1, a trajectory of M' satisfies the formula f if and only if it satisfies the formula f', and thus $P_{M'}(L_\omega(f)) = P_{M'}(L_\omega(f'))$. The proposition follows. □

It remains to show how we can determine which nodes of G are probable and which are not. To test whether a pair (u, t) is probable, the obvious way would be to construct an automaton B_t that accepts the infinite words of type t and solve the probabilistic emptiness problem for the Markov chain M and the automaton B_t as in Section 4.1. The automaton B_t can be constructed as the product of n automata, where for each $i = 1, \ldots, n$, if $t_i = 1$, then the ith component of the product is the automaton A_i obtained from A by letting its initial state be s_i, and if $t_i = 0$, the ith component is an automaton accepting the complement of the language of A_i. This method involves complementation and the final algorithm would be double exponential. A second method follows from the following fact: the pair (u, t) is probable if and only if there is a finite word w such that for each i, the run of $\det(\tau_{M \times A})$ on w starting from state $\{(u, s_i)\}$ ends in an accepting bottom s.c.c. if $t_i = 1$, and in a rejecting bottom s.c.c. if $t_i = 0$. This condition can be tested by constructing the product of the automata $\det(\tau_{M \times A}, (u, s_1)), \ldots, \det(\tau_{M \times A}, (u, s_n))$, and solving a reachability problem in the product. This method avoids complementation but it would still require forming an n-fold product. The resulting algorithm would be singly exponential, but the size of A would appear quadratically rather than linearly in the exponent. We shall describe now another method for determining the probable pairs that meets the bounds of Theorem 5.1.

From our earlier discussion, we know that if a node of G is probable, then so are all of its ancestors. First, we shall prove that the same property holds for the nodes that satisfy the necessary condition of Lemma 5.3.

LEMMA 5.7. *If a node of G satisfies the condition of Lemma* 5.3, *then so do all of its ancestors.*

PROOF. Let $(u,t) \to (v,r)$ be an arc of G and suppose that (v,r) satisfies the condition of Lemma 5.3. We will prove that (u,t) satisfies it also. To show part (a) of the condition, suppose that $t_i = 1$. The existence of the arc $(u,t) \to (v,r)$ in G implies that M has a transition $u \to v$ and that eq. $(*)$ holds with v in place of x_1 there. By eq. $(*)$, there is an index j such that $r_j = 1$ and $s_j \in \rho_A(s_i, v)$. Therefore, $\tau_{M\times A}$ has an arc from state (u, s_i) to state (v, s_j). Since (v,r) satisfies the condition of Lemma 5.3 and $r_j = 1$, the state (v, s_j) of $\tau_{M\times A}$ can reach some accepting recurrent state, and therefore, the same is true of (u, s_i).

To show that (u,t) satisfies part (b) of the condition, consider a state y in an accepting bottom s.c.c. C of $\det(\tau_{M\times A})$ and suppose the first component of y is u. Since M has a transition $u \to v$, the state y has a transition in $\det(\tau_{M\times A})$ to a (unique) state z with first component v. State z is the same bottom s.c.c. C, and therefore z contains an element (v, s_j) such that $r_j = 1$. Because of the transition $y \to z$ in $\det(\tau_{M\times A})$, the state y must contain an element (u, s_i) such that $s_j \in \rho_A(s_i, v)$. From eq. $(*)$, we have $t_i = 1$, and therefore (u,t) satisfies part (b) of the condition of Lemma 5.3. □

Consider a strongly connected component C of G. From the construction of G, all nodes of M that are first components of the nodes of C, belong to (form a subset of) the same s.c.c., say K of M. (A path from one node (u,t) to another node (v,r) of C projects to a path from u to v.) Thus, there is a (many-to-one) correspondence between the s.c.c.'s of G and those of M, where a s.c.c. of G corresponds to a unique s.c.c. of M, and a s.c.c. of M corresponds to one or more s.c.c.'s of G. We know that either all nodes of C are probable or none is. Thus, every s.c.c. of the chain M' is also a s.c.c. of G. In order to determine the chain M', it suffices to determine its bottom s.c.c.'s; then M' is a subgraph of G induced by these s.c.c.'s and their ancestors. The following Proposition characterizes the bottom s.c.c.'s of M'.

PROPOSITION 5.8. *For every bottom s.c.c. K of M there is a unique highest s.c.c. of G, denoted as $h(K)$, which corresponds to K and whose nodes satisfy the conditions of Lemma* 5.3; *that is, any other s.c.c. of G that corresponds to K and satisfies the condition of Lemma* 5.3 *is a descendant of $h(K)$. A s.c.c. of G is a bottom s.c.c. of M' if and only if it is $h(K)$ for some bottom s.c.c. K of M.*

We will need two lemmas to prove the proposition.

LEMMA 5.9. *If C is a bottom s.c.c. of M', then it corresponds to a bottom s.c.c. K of M.*

PROOF. Suppose that C is a bottom s.c.c. of M' but the corresponding s.c.c. K is not a bottom s.c.c. of M. A trajectory of M starting at any node u of K is absorbed with probability 1 in a bottom s.c.c. of M. No such trajectory is the projection of a trajectory in M' starting at a node (u, t) of C, contradicting Lemma 5.2 and the fact that (u, t) is probable. □

LEMMA 5.10. *Let C be a s.c.c. of G with the corresponding s.c.c. K of M. Then the following are equivalent.*

(1) *If $u \to v$ is an arc of K, then every node (v, r) of C with first component v has an immediate predecessor (u, t) in C with first component u.*
(2) *Every finite path in K is the projection of a path in C.*
(3) *No other s.c.c. of G corresponding to K is an ancestor of C.*

PROOF. We will show first the equivalence of conditions (1) and (2). Reverse all the arcs in K and C, and let K', C' be the reversed strong components. Clearly, (2) is equivalent to the following condition: every finite path in K' is the projection of a path in C'.

Assume that (1) holds to show (2). Condition (1) states that for every node (v, r) of C' and every arc $v \to u$ of K', the node (v, r) has an arc in C' to a node with first component u. It follows that for every node (v, r) of C' and every finite path $p = v \to v_1 \cdots \to v_m$ of K', there is a path of C' starting at (v, r) whose projection is p. This fact implies condition (2), unless there is a node w of K' that does not have any corresponding node in C' with first component w. To see that this is not the case, take any node (v, r) of C', and a path p from v to w in K' (it exists because K' is strongly connected); there is a corresponding path from (v, r) in C' ending at some node (w, t).

Suppose now that condition (2) holds but C does not satisfy condition (1); that is, some node (v, r) of C' has no immediate successor in C' corresponding to a successor u of v in K'. If p is a path in K', then let $H(p)$ be the set of all the paths in C' whose projection is p; the set of final nodes of these paths is denoted $h(p)$. We shall construct a path p such that $h(p) = \varnothing$ and hence $H(p) = \varnothing$; this will contradict (2). The path p has the form $vup_1vup_2vup_3 \cdots$, where each p_i is the string of intermediate nodes in some path from node u to node v. Let l be the number of nodes of C' with first component v. Since node (v, r) does not have an arc in C' to a node with first component u, and every other node with first component u has at most one such arc, $|h(vu)| \le l - 1$.

Define p_1 as follows: Let (u, t) be any node of $h(vu)$, and let up_1v be the projection of any path from (u, t) to (v, r) (it exists because C' is strongly connected). We claim that $|h(vup_1vu)| \le |h(vu)| - 1$. To show this, define the following mapping ψ from $h(vup_1vu)$ to $h(uv)$. For every member (u, q) of $h(vup_1vu)$ pick a (arbitrary) path of C' with projection vup_1vu and define $\psi(u, q)$ to be the second node of this path of C'. Clearly, $\psi(u, q)$ is in $h(vu)$. By our definition, there is a path of C' from $\psi(u, q)$ to (u, q) with projection up_1vu. Since there can be at most one path in C' that starts at $\psi(u, q)$ and has projection up_1vu, it follows that the mapping ψ is one-to-one. However, it is not onto the path of C' starting at (u, t) with projection up_1v ends in (v, r), which does not have an arc to a node with first component u. Thus, (u, t) is not in the image set of ψ. It follows that $|h(vup_1vu)| \le |h(vu)| - 1 \le l - 2$.

We can define inductively the strings $p_2, p_3, \ldots$ in an analogous way so that $|h(vup_1vu \cdots p_kvu)| \le l - k - 1$. Assume that we have defined $p_1, \ldots, p_k$. If the set $h(vup_1vu \cdots p_kvu)$ is not empty, we let (u, t') be any node in it, we take a path in C' from (u, t') to (v, r) and let $up_{k+1}v$ be its projection. As in the basis case, we can define a mapping ψ' from $h(vu \cdots p_{k+1}vu)$ to $h(vu \cdots p_kvu)$ which is one-to-one but not onto. Therefore, $|h(vu \cdots p_{k+1}vu)| \le |h(vu \cdots$

$p_k vu)| - 1 \leq l - k - 2$. Thus, in at most l steps, we have a path p such that $h(p) = \emptyset$.

The equivalence of (1) and (3) follows from the fact that for every arc $u \to v$ of K, every node (v, r) of C has exactly one immediate predecessor (u, t) in G with first component u. Suppose that the s.c.c. C violates condition (1). That is, for some arc $u \to v$ of K, there is a node (v, r) of C whose immediate predecessor (u, t) is not in C; then it must be a higher s.c.c. D of G. Since $u \in K$, the s.c.c. D also corresponds to K and thus (3) does not hold for C. Conversely, suppose that C violates condition (3); that is, G has a s.c.c. D that corresponds to K and is an ancestor of C. Take a path from any node (w, q) of D to C and let $(u, t) \to (v, r)$ be the arc of the path that enters C. Since w and v belong to the s.c.c. K of M, and M has a path from w to u and then to v, node u also belongs to the same s.c.c. K. Since (v, r) has a unique immediate predecessor in G with first component u, and that predecessor (u, t) is not in C, it follows that condition (1) does not hold for C. □

We are ready now to prove Proposition 5.8.

Proof of Proposition 5.8. Let K be a bottom s.c.c. of M, and let $\theta(K)$ be the set of s.c.c.'s of G that correspond to K and whose nodes satisfy the condition of Lemma 5.3. Clearly, $\theta(K)$ is nonempty; in particular, all probable nodes with first component from K belong to s.c.c.'s in $\theta(K)$. Let C be any member of $\theta(K)$ that has no ancestor in $\theta(K)$. By Lemma 5.7, no ancestor of C in G violates the condition of Lemma 5.3. Therefore, C has no proper ancestor corresponding to K; that is, C satisfies condition (3) of Lemma 5.10, and therefore also (2). We shall argue that C contains all the probable pairs (u, r) whose first component u is in K.

Let (u, r) be a probable pair with $u \in K$, and let $\mathbf{X}$ be a typical trajectory in M of type r starting at u. Since $\mathbf{X}$ is typical, for each $i = 1, \dots, n$, the run of $\det(\tau_{M \times A})$ that starts at $\{(u, s_i)\}$ and corresponds to $\mathbf{X}$ enters after some finite time a bottom s.c.c. which is accepting or rejecting depending on r_i. Let α be a prefix of $\mathbf{X}$ such all these runs for all i have entered bottom s.c.c.'s after α. Since condition (2) holds, there is a path in C with projection α. Let (u, t) be the first node of this path and (v, q) be the last node. By our assumptions, node (v, q) satisfies the condition of Lemma 5.3. Using the same arguments as in the proof of Lemma 5.4, we can show that $t = r$. Therefore, the probable pair (u, r) is in C.

Thus, the s.c.c. C contains all probable nodes with first component in K. This has several implications. First, it implies that our chosen s.c.c. C is unique, and thus every other member of $\theta(K)$ is a descendant of this s.c.c., which we denote as $h(K)$. Second, since this s.c.c. contains all the probable pairs with first component in K, it is the only s.c.c. of the chain M' that corresponds to the s.c.c. K of M. If $h(K)$ was not a bottom s.c.c. of M', then any descendant would have to correspond also to K (because K is a bottom s.c.c. of M). It follows that $h(K)$ is a bottom s.c.c. of M'.

From Lemma 5.9, every bottom s.c.c. of M' must correspond to a bottom s.c.c. K of M, and from the above arguments it must actually be $h(K)$. □

Summarizing our discussion, we can construct the graph of the new Markov chain M' by the following algorithm:

(1) Build the tables $\tau_{M \times A}$ and $\det(\tau_{M \times A})$, and find the recurrent states of $\tau_{M \times A}$.
(2) Construct the graph G.

(3) Remove from G the nodes that violate the condition of Lemma 5.3.
(4) For each bottom s.c.c. K of M find the highest corresponding s.c.c. $h(K)$ in G; by Proposition 5.8, there is a unique such s.c.c. which contains all the probable nodes (u, t) with $u \in K$. The graph of the new Markov chain M' is the subgraph of G induced on these s.c.c.'s and their ancestors.

Step (1) can be performed as in Section 4.1 in time $O(|M|2^{O(|A|)})$. If M has m nodes and e arcs, and A has n states, then the graph G has $m2^n$ nodes and $e2^n$ arcs, and can be certainly constructed in time $O(e2^{O(n)})$. It is not too hard also to implement Steps (3) and (4) with the same time complexity. Thus, the new Markov chain M' has at most $m2^n$ nodes and $e2^n$ arcs, and its graph can be constructed in time $O(e2^{O(n)})$.

Suppose that the given ETL formula f has k temporal connectives represented by k automata with $n_1, \ldots, n_k$ states and let $n = \sum_1^k n_i$. If we iterate our transformation eliminating one by one all the temporal connectives, we will finally construct a simple propositional formula f^k (with no temporal connectives) and a Markov chain M^k with at most $m2^n$ nodes and $e2^n$ arcs. For purposes of qualitative verification (determining if the probability that M satisfies f is 0 or 1), the exact values of the transition probabilities are not important; f is satisfied with probability 0 (respectively, 1) iff no initial state (respectively, all initial states) of M^k satisfies the formula f^k. We can construct the graph of the final Markov chain M^k, and thus perform the qualitative verification, in time $O(|M|2^{O(|A|)})$.

By similar arguments as in the case of PTL (Section 3.1), it can be seen that we can construct the graph of the Markov chain M' that results after eliminating one temporal connective using a transducer that works in space polynomial in $|A|$ and polylogarithmic in $|M|$. It follows then that the same holds for the final chain M^k.

If we wish to determine the exact probability of satisfaction $P_M(L_\omega(f))$, we have to compute the transition probabilities of M', that is, we need to compute the probabilities $P_M(u, t)$ for all nodes (u, t) of M'. These probabilities form a solution to the following linear system of equations in variables $q(u, t)$ (one variable for each node of M'):

(1) For every node (u, t) of $M', q(u, t) = \sum_{(v, r)} p_{uv} q(v, r)$, where the sum ranges over all immediate successors (v, r) of (u, t) in M';
(2) for every node u of M, $\sum_t q(u, t) = 1$, where the sum ranges over all t such that (u, t) is in M'.

Clearly, the system contains more equations than variables, and thus some of the equations are redundant. The probabilities $P_M(u, t)$ satisfy obviously this linear system. It suffices to show that the system has a unique solution.

PROPOSITION 5.11. *The linear system of the eqs.* (1) *and* (2) *has a unique solution.*

PROOF. Let us write eq. (1) as $q = R \cdot q$, where q is the vector of variables, and R is a matrix whose rows and columns are indexed by the nodes of M' (which correspond to the variables) with $R[(u, t), (v, r)] = p_{uv}$ if M' has an arc from (u, t) to (v, r), and $R[(u, t), (v, r)] = 0$, otherwise. If α is a finite path of M, we denote by $p(\alpha)$ the product of the probabilities of the transitions of the path, and call this number the *probability* of α. If α' is a finite path of M' which corresponds to (i.e., has projection) path α of M, we let $p(\alpha') = p(\alpha)$.

and call it the probability of α'. Consider now the jth power R^j of the matrix R. From the definitions, $R^j[(u,t),(v,r)] = \Sigma\{p(\alpha')|\alpha'$ is a path in M' of length j from (u,t) to $(v,r)\}$. As we have already noted, no two paths from (u,t) to (v,r) have the same projection. Therefore, $R^j[(u,t),(v,r)] = \Sigma\{p(\alpha)|\alpha$ is a path in M of length j which is the projection of a path from (u,t) to (v,r) in $M'\}$.

Let α be a finite path of M starting at some node u. We say that α is *determined* if for each $i = 1,\ldots,n$, the run of $\det(\tau_{M\times A})$ starting at $\{(u,s_i)\}$ ends in a bottom s.c.c., say C_i. If α is a determined path, then we define its *type* to be the vector t, where $t_i = 0$ or 1 according as C_i is rejecting or accepting. Let $d_j(u,t,v) = \Sigma\{p(\alpha)|\alpha$ is a path of length j in M from u to v which is determined and has type $t\}$. Let $d_j(u,t) = \Sigma_v d_j(u,t,v)$, let $d_j(u) = \Sigma_t d_j(u,t)$, and $\epsilon_j(u) = 1 - d_j(u)$. Note that $\epsilon_j(u)$ is the probability that a path of M of length j starting at u is not determined, that is, for some i, the corresponding run of $\det(\tau_{M\times A})$ from state $\{(u,s_i)\}$ has not reached a bottom s.c.c. Thus, we know that as $j \to \infty$, we have $\epsilon_j(u) \to 0$.

Suppose that a path α of M of length j from some node u to some node v is determined of type t; that is, α contributes to $d_j(u,t,v)$. From Lemma 5.4, we know that for any type $t' \neq t$, there is no path in M' that starts at node (u,t') and has projection α; thus α does not contribute to $R^j[(u,t'),(v,r)]$ for any r. Furthermore, we know that for every node (v,r) of M' there is a path ending at (v,r) with projection α; therefore, this path must start at (u,t), and thus α contributes to $R^j[(u,t),(v,r)]$ for all r. We conclude that for any r, we have $d_j(u,t,v) \leq R^j[(u,t),(v,r)]$.

Furthermore, consider a path α of M that contributes to $R^j[(u,t),(v,r)]$, that is, α has length j and is the projection of a path from (u,t) to (v,r) in M'. If α is determined, then, by Lemma 5.4, its type must be t, and thus α contributes to $d_j(u,t,v)$. The set of paths of M of length j starting at u that are not determined has probability $\epsilon_j(u)$. Therefore, $R^j[(u,t),(v,r)] \leq d_j(u,t,v) + \epsilon_j(u)$. Thus, the difference $R^j[(u,t),(v,r)] - d_j(u,t,v)$ lies between 0 and $\epsilon_j(u)$, and therefore it tends to 0 as $j \to \infty$.

Note that if a path α of M is determined of type t and thus contributes to $d_j(u, t)$, then all its extensions also contribute to it. Therefore, $d_j(u, t)$ is monotonically increasing as a function of j. Since it is bounded from above, it has a limit, and let $d_\infty(u, t)$ be this limit. If q is any solution to the system (1) and (2), then for any j the solution satisfies the equation $q = R^j q$, that is, $q(u, t) = \Sigma_{(v,r)} R^j[(u, t), (v, r)] \cdot q(v, r) = \Sigma_{(v,r)} \{R^j[(u, t), (v, r)] - d_j(u,t,v)\} \cdot q(v,r) + \Sigma_{(v,r)} d_j(u,t,v) \cdot q(v, r)$. The second term on the right-hand side is equal to $\Sigma_v d_j(u, t, v)\Sigma_r q(v, r)$ that because of eq. (2) is equal to $\Sigma_v d_j(u, t, v) = d_j(u, t)$. Thus, as j tends to ∞, the first term of the right-hand side tends to 0 and the second term tends to $d_\infty(u, t)$. It follows that $q(u, t) = d_\infty(u, t)$. □

Therefore, we can compute the transition probabilities of the Markov chain M' by solving the linear system of eqs. (1) and (2). The rest of the arguments to finish the proof of Theorem 5.1 are similar to the case of PTL studied in Section 3.1.

Remark 5.12. The conditions of Lemma 5.10 are closely related to the recurrence condition of Section 4.1. As we showed there, in a strongly connected component C of $\tau_{M\times A}$ either all nodes are recurrent or none is. It can be shown that the first case occurs if and only if C corresponds to a bottom s.c.c. K of M, and every path in K is the projection of a path in C (condition (2) of Lemma 5.10). If $\tau_{M \times A}$ has the same property as our graph G here, namely that every state (x, s) has one immediate predecessor with first component u for every immediate predecessor u of x in M, then we can efficiently determine what states are recurrent without having to apply the subset construction (just use condition (1) of Lemma 5.10). If A is the Buchi automaton constructed from a PTL formula, then $\tau_{M \times A}$ has indeed this property. This leads to a different algorithm for verifying if a sequential probabilistic program M satisfies a PTL formula f, which runs also in time linear in $|M|$ and exponential in $|f|$, though the exponent is somewhat larger. However, the algorithm can be simplified to match the complexity of the one in Section 3.1. We will not describe here this alternative algorithm for PTL.

6. *Conclusions*

We addressed the problem of verification of probabilistic finite state programs with respect to specifications expressed in linear time temporal logic or by ω-automata. We studied two types of probabilistic programs: sequential programs, modeled by ordinary Markov chains, and concurrent programs, modeled by concurrent Markov chains. We considered three types of specifications: Propositional linear temporal logic (PTL), ω-automata, and extended temporal logic (ETL). We shall summarize our results and discuss the proof techniques.

In the case of sequential programs, we showed that the verification problem can be solved with the same complexity for all three types of specification: linear time in the size of the program and exponential in the size of the specification. We showed also that the problem can be solved in PSPACE, matching the known lower bound. Furthermore, we presented algorithms for performing quantitative analysis, that is, computing the exact probability that a sequential program satisfies the specification, instead of only determining whether the probability is equal to 1. For all three types of specification, the quantitative problem can be solved in time polynomial in the program and exponential in the specification.

In the case of concurrent programs, the complexity depends on the type of the specification. For ω-automata the complexity of our algorithm is quadratic in the program and exponential in the specification. For the other two types of specification, PTL and ETL, the complexity is quadratic in the program and doubly exponential in the specification. We proved also lower bound results showing that double exponential time is required in the case of PTL (and hence also ETL) specifications, and exponential time is required in the case of ω-automata specifications.

In order to achieve essentially optimal bounds in the different cases, we had to use a variety of techniques. In the case of concurrent programs, we used the automata-theoretic approach for all three types of specification. Converting from a formula in linear or extended temporal logic to an ω-automaton increases the size by one exponential. However, since the lower bound for the logics is one exponential higher than the bound for automata, we do not lose our chances for optimality by doing this conversion and working only with automata. It was known that if the automaton is deterministic in the limit then the probabilistic emptiness problem can be solved in polynomial time. We solved the problem for general automata in exponential time by reducing it to this special case: we presented a semi-determinization construction which converts a general Buchi automaton to an exponentially larger automaton that is deterministic in the limit.

It would have been nice if we could reduce all three types of specification to the automata case for sequential programs, as we did for the concurrent programs. Unfortunately, this is not possible. The reason is that the time complexity for specifications in PTL (or ETL) is the same as for ω-automata: single exponential. This means that we cannot achieve optimality for PTL by simply reducing it to the automata case and invoking a general algorithm for automata, because we would lose an extra exponential in the conversion from a PTL formula to an automaton. One possibility would be to try to convert directly from a PTL formula to an exponentially larger automaton that is deterministic in the limit, and then use a polynomial algorithm for this automaton. Most likely, converting from a PTL formula to an automaton that is deterministic in the limit requires a double exponential blowup in size. Proving this is outside the scope of this paper, but we note that by Corollary 3.2.2, the *time* required for such a direct conversion is provably double exponential: If it was possible to do the direct conversion in less than double exponential time, then we could use it also in the concurrent case to solve the verification problem for PTL specifications in less than double exponential time, which we proved is impossible.

In the case of sequential programs, we needed different techniques. Our basic approach in all cases was to construct from the given Markov chain (program) and the given specification a new Markov chain that combines the two refining the original chain, such that the verification problem can be solved easily on the new chain. The particular techniques of this construction and the supporting analysis depend on the type of the specification. In the case of PTL formulas, we transformed step by step the formula and the Markov chain eliminating the temporal connectives one by one, until at the end we had a simple propositional formula for which the verification problem is trivial. In the case of automata specifications, the construction of the new chain was rather simple; it involved forming the product of the original chain and the automaton and applying the subset construction. Our algorithm was based on a combinatorial analysis of the interaction between Markov chains and ω-automata which led to a characterization of the states of the new chain that guarantee (almost surely) acceptance. In the automata case, we could have also used the semi-determinization transformation to solve the probabilistic emptiness problem. As we remarked in Section 4 however, the semi-determinization does not help for the universality problem, which is critical for the extension to ETL.

Extended temporal logic encompasses and extends in succinctness and expressive power both PTL and ω-automata. As in the case of PTL, our algorithm transformed step by step the given Markov chain and ETL formula eliminating one by one the temporal (automata) connectives. Performing one such elimination step involved a deeper combinatorial analysis of the interactions between Markov chains and automata.

ACKNOWLEDGMENT. We thank Moshe Vardi for helpful discussions, and for pointing out a simplification in the construction of Section 4.2. We thank the referees for suggestions that helped improve the presentation of this material.

REFERENCES

AHO, A. V., HOPCROFT, J., AND ULLMAN, J. D. 1974. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass.

BREIMAN, L. 1968. *Probability*, Addison-Wesley, Reading, Mass.

BUCHI, J. R. 1962. On a decision method in restricted second order arithmetic. In *Proceedings of the International Congress on Logic. Method and Philosophical Science*. pp. 1–12.

BUCHI, J. R. 1973. The monadic second-order theory of ω_1. In *Decidable Theories II*. Lecture Notes in Mathematics, vol. 328. Springer-Verlag, New York, pp. 1–128.

CHANDRA, A. K., KOZEN, D. C., AND STOCKMEYER, L. J. 1981. Alternation. *J. ACM 28*, 1 (Jan.) pp. 114–133.

CHOUEKA, Y. 1974. Theories of automata on ω-tapes: A simplified approach. *J. Comput. Syst. Sci. 8*, 117–141.

CLARKE, E. M., EMERSON, E. A., AND SISTLA, A. P. 1983. Automatic verification of finite-state concurrent systems using temporal logic specifications: A practical approach. In *Proceedings of the 10th Annual ACM Symposium on Principles of Programming Languages* (Austin, Tex., Jan. 24–26). ACM, New York, pp. 117–126.

COURCOUBETIS, C., AND YANNAKAKIS, M. 1990. Markov decision processes and regular events. In *Proc. of the 17th Intl. Colloquium on Automata, Languages and Programming*. pp. 336–349.

EMERSON, E. A., AND HALPERN, J. Y. 1983. "Sometimes" and "Not Never" revisited: On branching vs. linear time. In *Proceedings of 10th Annual ACM Symposium on Principles of Programming Languages* (Austin, Tex., Jan. 24–26). ACM, New York, pp. 127–140.

EMERSON, E. A., AND LEI, C. L. 1985. Modalities for model checking: Branching time strikes back. In *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages* (New Orleans, La., Jan. 14–16). ACM, New York, pp. 84–96.

EMERSON, E. A., AND SISTLA, A. P. 1983. A triple exponential decision procedure for CTL*. In *CMU Workshop on Logic of Programs*. Carnegie-Mellon Univ., Pittsburgh, Pa.

FRANCEZ, N., AND RODEH, M. 1980. A distributed data type implemented by a probabilistic communication scheme. In *Proceedings of 21st IEEE Symposium on Foundations of Computer Science*. IEEE, New York, pp. 373–379.

GABBY, D., PNUELI, A., SHELAH, S., AND STAVI, J. 1980. On the temporal analysis of fairness. In *Proceedings of 7th Annual ACM Symposium on Principles of Programming Languages* (Las Vegas, Nev., Jan. 28–30). ACM, New York, pp. 163–173.

HART, S., AND SHARIR, M. 1984. Probabilistic temporal logic for finite and bounded models. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing* (Washington, D.C., Apr. 30–May 2). ACM, New York, pp. 1–13.

HART, S., SHARIR, M., AND PNUELI, A. 1983. Termination of probabilistic concurrent programs. *ACM Trans. Prog. Lang. Syst. 5*, 3 (July), 356–380.

KEMENY, J. G., SNELL, J. L., AND KNAPP, A. W. 1976. *Denumerable Markov Chains*. Springer-Verlag, New York.

LEHMAN, D., AND RABIN, M. O. 1981. On the advantage of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of the 8th Annual ACM Symposium on Principles of Programming Languages* (Williamsburg, Va., Jan. 26–28). ACM, New York, pp. 133–138.

LEHMAN, D., AND SHELAH, S. 1982. Reasoning with time and chance. *Inf. Control 53*, 165–198.

LICHTENSTEIN, O., AND PNUELI, A. 1985. Checking that finite-state concurrent programs satisfy their linear specification. In *Proceedings of the 12th Annual ACM Symposium on Principles of Programming Languages* (New Orleans, La., Jan. 14–16). ACM, New York, pp. 97–107.

MCNAUGHTON, R. 1966. Testing and generating infinite sequences by a finite automaton. *Inf. Contr. 9*, 521–530.

PNUELI, A. 1981. The temporal logic of concurrent programs. *Theoret. Comput. Sci. 13*, 45–60.

PNUELI, A. 1983. On the extremely fair treatment of probabilistic algorithms. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing* (Boston, Mass., Apr. 25–27). ACM, New York, pp. 278–290.

PNUELI, A., AND ZUCK, L. 1986. Probabilistic verification by tableaux. In *Proceedings of the 1st Symposium on Logic in Computer Science*. IEEE, New York.

QUEILLE, J. P., AND SIFAKIS, J. 1982. Fairness and related properties in transition systems. Research Report #292. IMAG, Grenoble, Switzerland.

RABIN, M. O. 1972. Automata on infinite objects and Church's problem. In *Proceedings of the Regional AMS Conference Series in Mathematics*, vol. 13. AMS, Providence, R.I., pp. 1–22.

SAFRA, S. 1988. On the complexity of ω-automata. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*. IEEE, New York, pp. 319–327.

SISTLA, A. P., AND CLARKE, E. M. 1985. The complexity of propositional linear temporal logics. *J. ACM 32*, 3 (July), 733–749.

SISTLA, A. P., VARDI, M. Y., AND WOLPER, P. 1987. The complimentation problem for Buchi automata with application to temporal logic. *Theoret. Comput. Sci. 49*, 217–237.

VARDI, M. 1985. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of 26th IEEE Symposium on Foundations of Computer Science*. IEEE, New York, pp. 327–338.

VARDI, M., AND WOLPER, P. 1986. An automata-theoretic approach to automatic program verification. In *Proceedings of the 1st Symposium on Logic in Computer Science*. IEEE, New York.

WOLPER, P., VARDI, M. Y., AND SISTLA, A. P. 1983. Reasoning about infinite computation paths. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*. IEEE, New York, pp. 185–194.

WOLPER, P. 1983. Temporal logic can be more expressive. *Inf. Control 56*, 72–99.

RECEIVED NOVEMBER 1989; REVISED DECEMBER 1994; ACCEPTED APRIL 1995