Probabilistic Linear-Time Model Checking: an Overview of The Automata-Theoretic Approach

Moshe Y. Vardi[⋆]

Rice University, Department of Computer Science, Houston, TX 77005-1892, USA

Abstract. We describe the automata-theoretic approach to the algorithmic verification of probabilistic finite-state systems with respect to linear-time properties. The basic idea underlying this approach is that for any linear temporal formula we can construct an automaton that accepts precisely the computations that satisfy the formula. This enables the reduction of probabilistic model checking to ergodic analysis of Markov chains.

1 Introduction

Temporal logics, which are modal logics geared towards the description of the temporal ordering of events, have been adopted as a powerful tool for specifying and verifying concurrent systems [Pnu81]. One of the most significant developments in this area is the discovery of algorithmic methods for verifying temporal logic properties of *finite-state* systems [CE81,QS81,LP85,CES86]. This derives its significance both from the fact that many synchronization and communication protocols can be modeled as finite-state programs, as well as from the great ease of use of fully algorithmic methods. Finite-state programs can be modeled by transition systems where each state has a bounded description, and hence can be characterized by a fixed number of Boolean atomic propositions. This means that a finite-state program can be viewed as a finite *propositional Kripke structure* and that its properties can be specified using *propositional* temporal logic. Thus, to verify the correctness of the program with respect to a desired behavior, one only has to check that the program, modeled as a finite Kripke structure, satisfies (is a model of) the propositional temporal logic formula that specifies that behavior. Hence the name *model checking* for the verification methods derived from this viewpoint. Surveys can be found in [CG87,CGL93,Wol89].

For linear temporal logics, a close and fruitful connection with the theory of automata over infinite words has been developed [VW86,VW94,Var96]. The basic idea is to associate with each linear temporal logic formula a finite automaton over infinite words that accepts exactly all the computations that satisfy the formula. This enables the reduction of various decision problems, such as satisfiability and model-checking, to known automata-theoretic problems, yielding clean and asymptotically optimal algorithms. Furthermore, these reductions are very helpful for implementing temporal-logic

[⋆] Supported in part by NSF grants CCR-9628400 and CCR-9700061, and by a grant from the Intel Corporation. URL: `http://www.cs.rice.edu/~vardi`.

based verification methods, and are the key to techniques such as *on-the-fly* verification [CVWY92] that help coping with the "state-explosion" problem.

In view of the attractiveness of the model-checking approach, one would like to extend its applicability as much as possible. In particular, we would like to extend the model-checking approach to deal with *probabilistic* systems, since the introduction of probabilistic randomization into protocols has been shown to be extremely useful (see, for example, [LR81]). For probabilistic systems the situation becomes more complex. In such systems there is a probability measure defined on the set of computations. The notion of correctness now becomes probabilistic: we study here the notion that the program is correct if the probability that a computation satisfies the specification is one. (For investigations of quantitative notions of correctness, see, for example, [HJ94].) Even in this setting, the automata-theoretic approach is very useful, as was shown in [Var85,VW86,CY90,CY95]. In this paper we provide an overview of the automata-theoretic approach to probabilistic model checking of linear temporal properties. We show how it provides essentially optimal algorithms in most cases, and highlight one case where it does not.

2 Automata Theory

A *nondeterministic automaton* words is a tuple $A = (\Sigma, S, S_0, \rho, \alpha)$, where

- Σ is a finite alphabet,
- S is a finite set of states,
- $S_0 \subseteq S$ is a set of initial states,
- $\rho : S \times \Sigma \to 2^S$ is a (nondeterministic) transition function, and
- α is an *acceptance condition* (will be defined precisely later).

A state $s \in S$ is *deterministic* if $|\rho(s,a)| = 1$ for all $a \in \Sigma$. The automaton A is said to be *semi-deterministic* if all states are deterministic. If in addition $|S_0| = 1$, then A is said to be *deterministic*.

A *run* of A over a infinite word $w = a_1 a_2 \ldots$, is a sequence $s_0, s_1, \ldots$, where $s_0 \in S_0$ and $s_i \in \rho(s_{i-1}, a_i)$, for all $i \geq 1$. Acceptance is defined in terms of *limits*. The limit of a run $r = s_0, s_1, \ldots$ is the set $\lim(r) = \{s \mid s = s_i \text{ infinitely often}\}$. A *Büchi* acceptance condition is a set $F \subseteq S$ of *accepting states*. A set $T \subseteq S$ is *accepting* if $T \cap F \neq \emptyset$. Note that when $F = S$, all sets are accepting; we call such an accepting condition a *trivial* acceptance condition. A *Rabin* condition is a subset G of $2^S \times 2^S$, i.e., it is a collection of pairs of sets of states, written $[(L_1, U_1), \ldots, (L_k, U_k)]$. A set T is accepting if for some i we have that $T \cap L_i \neq \emptyset$ and $T \cap U_i = \emptyset$. A *Streett* condition is also a subset G of $2^S \times 2^S$, written $[(L_1, U_1), \ldots, (L_k, U_k)]$. A set T is accepting if for for all i we have that if $T \cap L_i \neq \emptyset$, then $T \cap U_i \neq \emptyset$ (thus, Streett acceptance is dual to Rabin acceptance). Note that Büchi acceptance is a special case of both Rabin acceptance (the pair $\langle F, \emptyset \rangle$), and Streett acceptance (the pair $\langle S, F \rangle$). A Büchi (resp., Rabin, Streett) automaton is an automaton on infinite words with Büchi (resp., Rabin, Streett) acceptance condition. Büchi, Rabin, and Streett conditions can all be viewed as *fairness* conditions, which are limit conditions on runs [Fra86]. A run r is accepting if $\lim(r)$ is accepting. An infinite word w is *accepted* by A if there is an accepting run of A over w. The set of infinite words accepted by A, called the *language of* A, is denoted $L(A)$.

It is known that Rabin and Streett automata are not more expressive than Büchi automata. More precisely, if A is a Rabin or Streett automaton, then there is a Büchi automaton A^b such that $L(A) = L(A^b)$ [Tho90]. When A is a Rabin automaton, the translation to a Büchi automaton is quadratic, but when A is a Streett automaton, the translation is exponential [SV89].

An automaton A is *nonempty* if $L(A) \neq \emptyset$. One of the most fundamental algorithmic issues in automata theory is testing whether a given automaton is nonempty. The *nonemptiness problem* for automata is to decide, given an automaton A, whether A is nonempty.

Proposition 1. [CES86] *The nonemptiness problem for Büchi automata is decidable in linear time.*

Proof: Let $A = (\Sigma, S, S_0, \rho, F)$ be the given automaton. Consider the directed graph $G_A = (S, E_A)$, where $E_A = \{\langle s,t\rangle \,|\, t \in \rho(s,a), a \in \Sigma\}$. It can be shown that A is nonempty iff G_A has a nontrivial maximal strongly connected component that is reachable from S_0 and intersects F nontrivially. As a depth-first-search algorithm can construct a decomposition of the graph into strongly connected components in linear time [CLR90], the claim follows. ▮

Since there is a quadratic translation of Rabin automata to Büchi automata, Proposition 1 yields a quadratic algorithm for nonemptiness of Rabin automata. For Streett automata, a direct algorithm is needed.

Proposition 2. [Eme85] *The nonemptiness problem for Streett automata is decidable in polynomial time.*

Proof: Let $A = (\Sigma, S, S_0, \rho, G)$ be the given automaton. Again we start by decomposing G_A into maximal strongly connected components reachable from S_0. We then iterate the following operation

> For a component Q and a pair $\langle L, U\rangle \in G$, if $Q \cap L \neq \emptyset$ and $Q \cap U = \emptyset$, then delete the states in L from G_A and recompute the decomposition into maximal strongly connected components.

Since each iteration deletes states from G_A, the above operations can be applied only $|S|$ times. $L(A)$ is nonempty iff the final decomposition contains a nontrivial component. ▮

As we shall see, closure under Boolean operations plays in important role in the application to verification. We first consider closure under intersection.

Proposition 3. [Cho74] *Let A_1, A_2 be Büchi automata. Then there is a Büchi automaton A such that $L(A) = L(A_1) \cap L(A_2)$.*

Proof: We show the construction explicitly for the case that A_1 has a trivial acceptance condition. Let $A_1 = (\Sigma, S_1, S_1^0, \rho_1, S_1)$ and $A_2 = (\Sigma, S_2, S_2^0, \rho_2, F_2)$. Let $A = (\Sigma, S, S^0, \rho, F)$, where $S = S_1 \times S_2$, $S^0 = S_1^0 \times S_2^0$, $F = S_1 \times F_2$, and $(s', t') \in \rho((s,t), a)$ if $s' \in \rho_1(s,a)$ and $t' \in \rho_2(t,a)$. ▮

We denote the intersection automaton by $A_1 \cap A_2$.

Büchi automata are also closed under complementation.

Proposition 4. [Büc62,Kla91,KV97b] *Let A be a Büchi automaton over an alphabet Σ. Then there is a Büchi automaton A^c such that $L(A^c) = \Sigma^\omega - L(A)$.*

If A has n states, then A^c has $n^{O(n)}$ states, which is known to be tight [Mic88].

Unlike automata on finite words, Büchi automata are not closed under determinization, i.e., nondeterministic Büchi automata are more expressive than deterministic Büchi automata [Tho90]. In contrast, Rabin and Streett automata are closed under determinization. In fact, they are also closed under co-determinization.

Proposition 5. [McN66,Saf89] *Let A be a Büchi automaton. There are deterministic Rabin (resp., Streett) automata A_d and A^{cd} such that $L(A^d) = L(A)$ and $L(A^{cd}) = \Sigma^\omega - L(A)$*

If A has n states, then A_d and A^{cd} have $n^{O(n)}$ states and $O(n)$ pairs.

3 Verification

We focus here on *finite-state systems*, i.e., systems in which the variables range over finite domains. The significance of this class follows from the fact that a significant number of the communication and synchronization protocols studied in the literature are in essence finite-state systems [Liu89,Rud87].

A finite-state system over a set $Prop$ of atomic propositions is a structure of the form $M = (W, w_0, R, V)$, where W is a finite set of states, $w_0 \in W$ is the initial state, $R \subseteq W^2$ is a total transition relation, and $V : W \rightarrow 2^{Prop}$ assigns truth values to propositions in a set $Prop$ for each state in W. The intuition is that W describes all the states that the program could be in (where a state includes the content of the memory, registers, buffers, location counter, etc.), R describes all the possible transitions between states (allowing for nondeterminism), and V relates the states to the propositions (e.g., it tells us in what states the proposition request is true). The assumption that R is total (i.e., that every state has a child) is for technical convenience. We can view a terminated execution as repeating forever its last state.

Let $\mathbf{u}$ be an infinite sequence $u_0, u_1, \ldots$ of states in W such that $u_0 = w_0$, and $u_i R u_{i+1}$ for all $i \geq 0$. Then the sequence $V(u_0), V(u_1) \ldots$ is a *computation* of M. If we take Σ to be 2^{Prop}, then a computation of M is a word in Σ^ω. We denote the set of computations of M by $L(M)$. A *specification* for M is a language $\sigma \subseteq \Sigma^\omega$. M *satisfies* σ if every computation of M is in σ.

A finite-state system $M = (W, w_0, R, V)$ can be viewed as a Büchi automaton $A_M = (\Sigma, W, \{w_0\}, \rho, W)$, where $\Sigma = 2^{Prop}$ and $s' \in \rho(s, a)$ iff $(s, s') \in R$ and $a = V(s)$. As this automaton has a set of accepting states equal to the whole set of states, every infinite run of the automaton is accepting; that is, $L(A_M) = L(M)$. If σ is expressed in terms of a *specification automaton* A_σ, then M satisfies σ iff $L(M) \subseteq L(A_\sigma)$. This is called the *language-containment* approach to verification [Kur94]. Note that $L(M) \subseteq L(A_\sigma)$ iff $L(M) \cap (\Sigma^\omega - L(A_\sigma)) = \emptyset$ iff $L(M) \cap L(A_\sigma^c)) = \emptyset$ iff $L(A_M \cap A_\sigma^c) = \emptyset$, where A_σ^c is an automaton that complement A_σ and $A_M \cap A_\sigma^c$ is the intersection of A_M and A_σ^c. Thus, verification is ultimately reducible to the emptiness problem. Using Propositions 1 and 4, we get:

Theorem 6. *Checking whether a Büchi automaton A_σ with n states is satisfied by a finite-state program M can be done in time $O(||M|| \cdot n^{O(n)})$.*

We note that a time upper bound that is polynomial in the size of the program and exponential in the size of the specification is considered here to be reasonable, since the specification is usually rather short [LP85]. It is unlikely that the exponential bound can be improved, as the problem is PSPACE-complete [Wol83].

4 Temporal Logic and Automata

Formulas of *linear time propositional temporal logic* (LPTL) are built from a set $Prop$ of atomic propositions and are closed under the application of Boolean connectives, the unary temporal connective X (next), and the binary temporal connective U (until) [Pnu77,GPSS80]. LPTL is interpreted over *computations*. A computation is a function $\pi : \omega \rightarrow 2^{Prop}$, which assigns truth values to the elements of $Prop$ at each time instant (natural number). For a computation π and a point $i \in \omega$, we have that:

- $\pi, i \models p$ for $p \in Prop$ iff $p \in \pi(i)$.
- $\pi, i \models \xi \wedge \psi$ iff $\pi, i \models \xi$ and $\pi, i \models \psi$.
- $\pi, i \models \neg\varphi$ iff not $\pi, i \models \varphi$
- $\pi, i \models X\varphi$ iff $\pi, i+1 \models \varphi$.
- $\pi, i \models \xi U \psi$ 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 π *satisfies* a formula φ, denoted $\pi \models \varphi$, iff $\pi, 0 \models \varphi$. We say that a finite-state system $M = (W, w_0, R, V)$ satisfies φ if $\pi \models \varphi$ for every computation π of M.

As we observed computations can also be viewed as infinite words over the alphabet 2^{Prop}. The following theorem establishes the correspondence between LPTL and Büchi automata.

Proposition 7. [VW94] *Given an LPTL formula φ, one can build a Büchi automaton $A_\varphi = (\Sigma, S, S_0, \rho, F)$, where $\Sigma = 2^{Prop}$ and $|S| \leq 2^{O(|\varphi|)}$, such that $L(A_\varphi)$ is exactly the set of computations satisfying the formula φ.*

It follows that $M \models \varphi$ iff $L(M) \subseteq L(A_\varphi)$ iff $L(M) \cap (\Sigma^\omega - L(A_\varphi)) = \emptyset$ iff $L(M) \cap L(A^c_\varphi)) = \emptyset$ iff $L(A_M \cap A^c_\varphi) = \emptyset$. Note that rather than construct first A_φ and then A^c_φ, involving a doubly exponential blow-up, we can construct directly $A_{\neg\varphi}$, as, clearly, $L(A_{\neg\varphi}) = L(A^c_\varphi)$. By Proposition 7, the number of states of $A_{\neg\varphi}$ is $2^{O(|\varphi|)}$. Consequently, the automaton $A_M \cap A_{\neg\varphi}$ has $|W| \cdot 2^{O(|\varphi|)}$ states. Using Proposition 1, we get:

Theorem 8. *Checking whether a formula φ is satisfied by a finite-state program M can be done in time $O(||M|| \cdot 2^{O(|\varphi|)})$,*

It is unlikely that the exponential bound can be improved, as the problem is PSPACE-complete [SC85]. Note that regardless of whether the specification is expressed using a Büchi automaton or an LPTL formula, the complexity is linear in the size of the system and exponential in the size of the specifiation.

5 Probabilistic Systems

We model probabilistic systems by (finite-state) *Markov chains*. The basic intuition is that transition between states is governed by some probability distribution. A Markov chain $M = (W, P, P_0)$ consists of a finite *state space* W; a *transition probability function* $P : W^2 \rightarrow [0, 1]$, such that $Sum_{v \in W} P(u, v) = 1$ for all $u \in W$; and an *initial probability distribution* $P_0 : W \rightarrow [0, 1]$, such that $Sum_{u \in W} P_0(u) = 1$.

As in the standard theory of Markov processes (see [KS60,KSK76]), we define a probability space called the *sequence space* $\Psi_M = (\Omega, \Delta, \mu)$, where $\Omega = W^\omega$ is the set of all infinite sequences of states, Δ is a Borel field generated by the *basic cylindric sets*

$$\Delta(w_0, w_1, \ldots, w_n) = \{\mathbf{w} \in \Omega \mid \mathbf{w} = w_0, w_1, \ldots, w_n, \ldots\},$$

and μ is a probability distribution defined by

$$\mu(\Delta(w_0, w_1, \ldots, w_n)) =$$

$$P_0(w_0) \cdot P(w_0, w_1) \cdot P(w_1, w_n) \cdot \ldots P(w_{n-1}, w_n).$$

Consider a language $\sigma \subseteq W^\omega$, viewed as specification, that is measurable wrt the sequence space Ψ_M. We say that M *almost surely satisfies* σ if $\mu(\sigma) = 1$, that is, if "almost" all computations of M are in σ.

Suppose that σ is expressed in terms of a deterministic automaton $A = (W, S, s_0, \rho, G)$. It can be shown that in this case σ is measurable [Var85]. We define the product chain $M \times A = (W \times S, P', P'_0)$, where $P'_0(\langle w, s_0\rangle) = P_0(w)$ and $P'_0(\langle w, s\rangle) = 0$ if $s \neq s_0$, and $P'(\langle u, s\rangle, \langle v, t\rangle) = P(u, v)$ if $t = \rho(s, u)$ and $P'(\langle u, s\rangle, \langle v, t\rangle) = 0$ if $t \neq \rho(s, u)$. If G is the acceptance condition $[(L_1, U_1), \ldots, (L_k, U_k)]$ (Rabin or Streett), then take $W \times G$ to be $[(W \times L_1, W \times U_1), \ldots, (W \times L_k, W \times U_k)]$. Let $sat(W \times G)$ be the set of sequences that satisfy the acceptance condition $W \times G$, i.e., whose limit is accepting.

Lemma 9. $\mu_M(L(A)) > 0$ *iff* $\mu_{M\times A}(sat(W \times G) > 0$

Lemma 9 enables us to focus on the behavior of the Markov chain in the limit. Let $M = (W, P, P_0)$ be a Markov chain and let $\alpha \subseteq 2^W \times 2^W$ be a Streett acceptance condition. Consider the graph $G_M = (W, W_0, E_M)$, where $W_0 = \{w \mid P_0(w) > 0\}$ and $E_M = \{\langle u, v\rangle \mid P(u, v) > 0\}$. An *ergodic set* in G_M is a terminal maximal strongly connected component that is reachable from W_0. Such a set is closed with respect to all positive transitions of M.

Lemma 10. $\mu_M(sat(\alpha)) > 0$ *iff some ergodic set of* G_M *is accepting wrt* α.

Note that in this analysis the exact transition probabilities are irrelevant. All that counts is whether a transition probability is 0 or not.

Ergodic analysis can be also viewed from an automata-theoretic perspective. Define the Streett automaton $A_M = (\{a\}, W, W_0, \rho_P, \alpha \cup \beta)$, where $\{a\}$ is a singleton alphabet, $\rho_P(u, a) = \{v \mid P(u, v) > 0\}$, and $\beta = \{(\{u\}, \{v\}) \mid P(u, v) > 0\}$.

Lemma 11. $\mu_M(sat(\alpha)) > 0$ *iff* $L(A_M) \neq \emptyset$.

The intuition behind the lemma is that probabilistic behavior is essentially a "fair" behavior.

We can now consider the algorithmic aspect of verifying probabilistic systems. Suppose that we are given a Markov chain M and a nondeterministic automaton A_σ. We want to check whether M almost surely satisfies σ. We construct the co-deterministic Streett automaton A_σ^{cd}, per Proposition 5. We know that M almost surely satisfies $\mathcal{A}_\sigma$ iff $\mu_M(L(A_\sigma^{cd})) = 0$. Thus, we can form the product chain $M \times A_\sigma^{cd}$ and then apply Lemma 11.

Theorem 12. *Checking whether a Büchi automaton* A_σ *with* n *states is almost surely satisfied by a Markov chain* M *can be done in time polynomial in* $||M||$ *and exponential in* $||A_\sigma||$.

It is unlikely that the exponential bound can be improved, as the problem is PSPACE-complete [Var85].

Suppose now that we have an assignment $V : W \rightarrow 2^{Prop}$ and the specification is given by an LPTL formula over $Prop$. Let $L(\varphi)$ consists of all sequences $\mathbf{w} \in \Omega$ such that $V(\mathbf{w})$ satisfies φ. It can be shown that $L(\varphi)$ is measurable [Var85]. We want to verify that M almost surely satisfies φ, i.e., M almost surely satisfies $L(\varphi)$. We can apply Proposition 7 and construct A_φ and then proceed with the algorithm of Theorem 12. The resulting algorithm is polynomial in the size of M, but is doubly exponential in the size of φ. Courcoubetis and Yannakakis [CY95] showed how almost-sure satisfaction of LPTL formulas over Markov chains can be checked in time that is exponential in the size of the specification. The algorithm does not use the translation of LPTL formulas to Büchi automata. We will return to this point in the concluding discussion.

6 Reactive Probabilistic Programs

So far we viewed systems as Markov chains. This model assumes that all the transitions of a system are probabilistic. This is adequate for *closed* systems, whose transitions are internally driven. In *reactive* systems, which interact with their external environments [HP85], some transitions are inherently nondeterministic.

A (finite-state) *reactive Markov chain* $M = (W, N, P, P_0)$ is a Markov chain (W, P, P_0) with a set $N \subseteq W$ of *nondeterministic states* (the states in $W - N$ are the *probabilistic states*). The idea is that $W - N$ is the set of states where a probabilistic transition has to be made and N is the set of states where a nondeterministic transition has to be made. If $u \in N$, then we interpret $P(u, v)$ to mean that there is a possible transition from u to v if and only if $P(u, v) > 0$.

This model, originally named *concurrent Markov chain*, was proposed in [Var85], based on earlier ideas in [HSP83]. It turns out, however, that is simply another guise of *Markov decision processes* [Der70].

To define the sequence space of a reactive Markov chain it is convenient to imagine a *scheduler*, which makes all the nondeterministic choices. A scheduler for a reactive Markov chain $M = (W, N, P, P_0)$ is a function $\tau : W^*N \rightarrow W$, i.e., a function that assigns a state to each sequence of states that end with a nondeterministic state, such that $\tau(w_0, \ldots, w_n) = w$ only if $P(w_n, w) > 0$.

Let $M = (W, N, P, P_0)$ be a reactive Markov chain. A scheduler τ for M gives rise to an infinite Markov chain $M^\tau = (W^+, P^\tau, P_0^\tau)$, where

- $P_0^\tau(w) = P_0(w)$ for all $w \in W$ and $P^\tau(x) = 0$ for all $x \in W^+ - W$,
- $P^\tau : W^+ \times W^+ \rightarrow [0, 1]$ is defined as follows (where x and y are arbitrary members of W^+):
  - $P^\tau(xu, xuv) = P(u, v)$ if $u \in W - N$,
  - $P^\tau(xu, xuv) = 1$, if $u \in N$ and $\tau(xu) = v$, and
  - $P^\tau(x, y) = 0$, otherwise.

Intuitively, M^τ describes the behavior of the system under the scheduler τ.

Consider a specification $\sigma \subseteq W^\omega$. We need to "lift" σ to the level of M^τ. To that end, we define a mapping $V : W^+ \rightarrow W$ by $V(u_1 u_2 \ldots u_n) = u_n$. Now take $L(\sigma) \subseteq (W^+)^\omega$ as the set of sequences $\mathbf{u} \in (W^+)^\omega$ such that $V(\mathbf{u}) \in \sigma$. We now say that M almost surely satisfy σ if $\mu_{M^\tau}(L(\sigma)) = 1$ for all schedulers τ. The intution is that σ almost surely holds regardless of the environment decisions. Dually, we can ask whether the exists a scheduler τ such that $\mu_{M^\tau}(L(W^\omega - \sigma)) > 0$.

Suppose that σ is given in terms of a deterministic Streett automaton A with acceptance condition G. Then, as in Lemma 9, almost sure satisfaction of A can be reduced to checking a limit property.

Lemma 13. *There exists a scheduler τ such that $\mu_{M^\tau}(L(A)) > 0$ iff there exists a scheduler τ such that $\mu_{(M\times A)^\tau}(sat(W \times G) > 0$*

As we did with Markov chains, we use ergodic analysis to check limit properties. Let $M = (W, N, P, P_0)$ be a reactive Markov chain and let $\alpha \subseteq 2^W \times 2^W$ be a Streett acceptance condition. Consider the graph $G_M = (W, W_0, E_M)$, where $W_0 = \{w \mid P_0(w) > 0\}$ and $E_M = \{\langle u, v\rangle \mid P(u, v) > 0\}$. An *ergodic set* in G_M is a strongly connected component of G_M that is reachable from W_0 and is closed under the positive probabilistic transitions of M, i.e., under $E_M \cap ((W - N) \times W)$.

Lemma 14. *There exists a scheduler τ such that $\mu_{M^\tau}(sat(\alpha)) > 0$ iff some ergodic set of G_M is accepting wrt α.*

Again, the ergodic analysis can be also viewed from an automata-theoretic perspective. Define the Streett automaton $A_M = (\{a\}, W, W_0, \rho_P, \alpha \cup \beta)$, where $\{a\}$ is a singleton alphabet, $\rho_P(u, a) = \{v \mid P(u, v) > 0\}$, and $\beta = \{(\{u\}, \{v\}) \mid u \in W - N$ and $P(u, v) > 0\}$.

Lemma 15. *There exists a scheduler τ such that $\mu_{M^\tau}(sat(\alpha)) > 0$ iff $L(A_M) \neq \emptyset$.*

To check that a reactive Markov chain almost surely satisfies a Büchi automaton A_σ or an LPTP formula φ we proceed as we did with Markov chains. We first construct a deterministic Streett automaton for the complementary specification. The cost is exponential for Büchi automata and doubly exponential for LPTL formuas. We then take the product with the chain and perform ergodic analysis. The resulting complexity is polynomial in the size of the chain, exponential in the size of A_σ and doubly exponential in the size of φ. A 2EXPTIME lower bound in [CY95] shows that the latter bound is asymptotically optimal.

We note that the automata-theoretic framework lends itself easily to the incorporation of fairness. The intuition underlying fairness is that we want to restruct attention to only "well-behaved" schedulers [BK97]. For example, we may want to assume that if a transition is enabled infinitely often then it is taken infinitely often. Streett conditions enable us to express rather general fairness properties [Fra86]. We say that a scheduler τ is *fair* wrt a fairness condition G if $\mu_{M^\tau}(sat(G)) = 1$. We can now modify the definition of almost-sure satisfaction by quantifying universally only over fair schedulers. It turns out that the algorithmic modification required to handle fairness is rather straightforward; all we have to do is to add the Streett condition describing the fairness condition to the Streett condition used in Lemmas 14 and 15.

7 Concluding Remarks

Over the last 15 years, the automata-theoretic approach to verification has proven itself to be a rather powerful paradigm. Essentially, almost all decision problems related to specification and verification of finite-state systems can be expressed and solved using automata-theoretic tools. See [VW86,EJ91,VW94,BVW94,CY95,KV97a] for numerous examples. The result in Section 5 is a rare exception. As we saw there, using the automata-theoretic approach we obtained an algorithm for checking almost-sure satisfaction of LPTL formulas over Markov chains whose complexity is doubly exponential in the size of the input formula. A non-automata-theoretic algorithm, whose complexity is exponentially lower (i.e., a single exponential) is described in [CY95]. Can the improved algorithm be given an automata-theoretic account? We believe that the answer is positive and suspect that such account can be given in terms of “weak alternating automata” [Var95,KV97b].

References

[BK97] C. Baier and M.Z. Kwiatowska. Automatic verification of liveness properties of randomized systems. In *ACM Symp. on Principles of Distributed Systems (PODC)*, 1997.

[Büc62] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Internat. Congr. Logic, Method and Philos. Sci. 1960*, pages 1–12, Stanford, 1962. Stanford University Press.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, editor, *Computer Aided Verification, Proc. 6th Int. Conference*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155, Stanford, June 1994. Springer-Verlag, Berlin.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.

[CG87] E.M. Clarke and O. Grumberg. Avoiding the state explosion problem in temporal logic model-checking algorithms. In *Proc. 6th ACM Symposium on Principles of Distributed Computing*, pages 294–303, Vancouver, British Columbia, August 1987.

[CGL93] E.M. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Decade of Concurrency – Reflections and Perspectives (Proceedings of REX School)*, volume 803 of *Lecture Notes in Computer Science*, pages 124–175. Springer-Verlag, 1993.

[Cho74] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[CVWY92] C. Courcoubetis, M.Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275–288, 1992.

[CY90] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proc. 17th Int. Coll. on Automata Languages and Programming*, volume 443, pages 336–349, Coventry, July 1990. Lecture Notes in Computer Science, Springer-Verlag.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42:857–907, 1995.

[Der70] C. Derman. *Finite-State Markovian Decision Processes*. Academic Press, New York, 1970.

[EJ91] E.A. Emerson and C. Jutla. Tree automata, Mu-calculus and determinacy. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 368–377, San Juan, October 1991.

[Eme85] E.A. Emerson. Automata, tableaux, and temporal logics. In *Proc. Workshop on Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 79–87. Springer-Verlag, 1985.

[Fra86] N. Francez. *Fairness*. Texts and Monographs in Computer Science. Springer-Verlag, 1986.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symposium on Principles of Programming Languages*, pages 163–173, January 1980.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, 6:512–535, 1994.

[HP85] D. Harel and A. Pnueli. On the development of reactive systems. In K. Apt, editor, *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Summer Institutes*, pages 477–498. Springer-Verlag, 1985.

[HSP83] S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Trans. on Programming Languages*, 5:356–380, 1983.

[Kla91] N. Klarlund. Progress measures for complementation of ω-automata with applications to temporal logic. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 358–367, San Juan, October 1991.

[KS60] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Van Nostrad, Princeton, 1960.

[KSK76] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. Springer-Verlag, New York, 1976.

[Kur94] R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.

[KV97a] O. Kupferman and M.Y. Vardi. Synthesis with incomplete informatio. In *2nd International Conference on Temporal Logic*, pages 91–106, Manchester, July 1997. Kluwer Academic Publishers.

[KV97b] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *Proc. 5th Israeli Symposium on Theory of Computing and Systems*, pages 147–158. IEEE Computer Society Press, 1997.

[Liu89] M.T. Liu. Protocol engineering. *Advances in Computing*, 29:79–195, 1989.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th ACM Symposium on Principles of Programming Languages*, pages 97–107, New Orleans, January 1985.

[LR81] D. Lehman and M. O. Rabin. On the advantage of free choice: A fully symmetric and fully distributed solution to the dining philosophers problem. In *Proc. 8th ACM Symposium on Principles of Programming Languages*, pages 133–138, 1981.

[McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[Mic88] M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[Pnu81] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[QS81] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th International Symp. on Programming*, volume 137, pages 337–351. Springer-Verlag, Lecture Notes in Computer Science, 1981.

[Rud87] H. Rudin. Network protocols and tools to help produce them. *Annual Review of Computer Science*, 2:291–316, 1987.

[Saf89] S. Safra. *Complexity of automata on infinite objects*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1989.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *Journal ACM*, 32:733–749, 1985.

[SV89] S. Safra and M.Y. Vardi. On ω-automata and temporal logic. In *Proc. 21st ACM Symposium on Theory of Computing*, pages 127–137, Seattle, May 1989.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 327–338, Portland, October 1985.

[Var95] M.Y. Vardi. Alternating automata and program verification. In *Computer Science Today –Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science*, pages 471–485. Springer-Verlag, Berlin, 1995.

[Var96] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, Berlin, 1996.

[VW86] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. First Symposium on Logic in Computer Science*, pages 322–331, Cambridge, June 1986.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

[Wol89] P. Wolper. On the relation of programs and computations to models of temporal logic. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proc. Temporal Logic in Specification*, volume 398, pages 75–123. Lecture Notes in Computer Science, Springer-Verlag, 1989.