Verifying Automata Specifications of Probabilistic Real-time Systems

Rajeev Alur
AT&T Bell Laboratories
Murray Hill, NJ 07974, USA.

Costas Courcoubetis [1]
Computer Science Department, University of Crete, and
Institute of Computer Science, FORTH, Heraklion, Crete 71110, Greece.

David Dill [2]
Department of Computer Science
Stanford University, Stanford, CA 94305, USA.

Abstract. We present a model-checking algorithm for a system presented as a generalized semi-Markov process and a specification given as a deterministic timed automaton. This leads to a method for automatic verification of timing properties of finite-state probabilistic real-time systems.

Keywords: Real-time, Probabilistic systems, Automatic verification.

1 Introduction

There is increasing awareness that unexpected behavior from interacting processes can cause serious problems. This observation applies not only to programs and digital systems, but also to physical processes, such as robots, automobiles, manufacturing processes, and so on. Indeed, as digital systems become smaller and cheaper, their use to control and interact with physical processes will inevitably increase.

Formal verification of these systems seeks mathematical methods for reasoning about their behavior. Automatic formal verification is particularly promising, because it requires far less labor than the manual techniques. There has been a great deal of research on automatic verification methods for concurrent systems whose behavior is *timing independent*. However, physical processes work in *real-time* — their correctness depends crucially on the actual timing of events; furthermore, physical processes are *probabilistic*. Automatic verification techniques have been developed for several different models of real-time

[1]Supported by ESPRIT BRA project SPEC.
[2]Supported by the NSF under grant MIP-8858807.

[AH89, Ko90, ACD90, AD90, Le90], and for discrete Markov chains [LS82, Va85, PZ86, CY88, HJ89]. We aim to develop automatic verification techniques that can deal with probabilistic real-time processes.

We add probabilities to our model of state-transition graphs with timing constraints by associating fixed distributions with the delays. This extension makes our processes *generalized semi-markov processes*, which have been studied by researchers in the field of stochastic modeling [Sh87, Wh80]. In previous work [ACD91], we developed an algorithm for checking whether such a system satisfied a formula in TCTL [ACD90], an extension of the branching-time temporal logic CTL [EC82] that allowed the inclusion of explicit constant upper and lower time bounds in formulas. Here, we describe a different model-checking algorithm that uses *deterministic timed automata* [AD90] to specify properties to be checked. These automata can express certain types of linear behavior that cannot be described using TCTL. An example of such a requirement is the "convergent bounded response property": events a and b occur in alternation and *eventually always* b occurs within 2 seconds of the previous a" (note that this property allows an arbitrary delay between an a and the next b for any finite time, as long as the bounded response eventually holds). Furthermore, in [ACD91] we required all delays to have continuous distributions with bounded interval support. Now, in addition, we also allow discrete probability functions (this allows fixed delays) and exponential distributions.

The model-checking problem is to determine whether, with probability 1, every trajectory of the probabilistic real-time system is accepted by the timed automaton specification. We present an algorithm that combines model-checking for real-time non-probabilistic systems [AD90] with model-checking for finite-state discrete-time Markov chains [Va85, CY88]. The correctness of the algorithm is not obvious, because it analyzes the projection of a Markov process onto a finite state space. The projection process is not Markov, so our most significant result is that the model-checking algorithm works.

The paper is organized as follows: The model for probabilistic real-time system is described in section 2. Section 3 considers the specification language of timed automata. Section 4 develops the model-checking algorithm and proves its correctness. The concluding section discusses some directions in which this work can be extended.

2 The process model

In probabilistic verification, a program is usually modeled by a Markov chain over a finite state space. We want to introduce the notion of time in this model of probabilistic systems. A typical property we want to model is the following: "the delay between the request and the response is distributed according to a given probability distribution". To this end, whenever the request event happens the response event will be scheduled with an associated clock. The value of the clock will be some negative number chosen in accordance with the desired probability distribution. The reading of the clock changes with the elapsing of time, and the response event gets triggered when the clock reaches 0. We assume that all the system delays have probability distributions of one of the following two forms:

- Exponential distributions.

- Distributions with finite interval support; that is, distributions that satisfy the following constraint: there exist bounded intervals $I_1, I_2, \ldots I_n$ with integer boundaries such that the probability density function is positive in each I_i and zero everywhere else.

The latter category includes a large number of distributions such as bounded uniform distributions, and discrete probability functions which choose the values from a finite set of integers. To keep the presentation simple, we assume that for every (nonexponential) density function f there exists a single bounded interval I such that for every $t \in \mathbf{R}$, $\mathrm{f}(t)$ is positive iff $t \in I$. The model-checking algorithm needs only a trivial modification when the distributions have support of more than one interval.

To model concurrent delays we need to introduce several clocks running concurrently. Our notation is based upon the literature on generalized semi-Markov processes [Wh80, Sh87].

A (finite-state) *real-time probabilistic process* $\mathcal{M}$ has the following components:

- *States:* A finite set S of states.

- *Events:* Associated with each state s is a finite set of events $\mathrm{E}(s)$. $\mathrm{E}(s)$ gives the set of events that are scheduled to occur when the process is in state s. Let E be the set of all events $(\cup_{s \in \mathrm{S}} \mathrm{E}(s))$, $\Sigma(s)$ be the set of all nonempty subsets of $\mathrm{E}(s)$, and Σ be the set of all nonempty subsets of E.

- *Initial distribution:* An initial probability distribution function f_0 such that $\Sigma_{s \in \mathrm{S}} \mathrm{f}_0(s)$ equals 1.

- *Transitions:* A probabilistic transition function $next : \mathrm{S} \times \Sigma \times \mathrm{S} \mapsto [0,1]$. $next(s, a, s')$ is the probability that the next state is s' when the event set $a \in \Sigma(s)$ occurs in state s. For every state s and every $a \in \Sigma(s)$, we require that the sum $\Sigma_{s' \in \mathrm{S}}\, next(s, a, s')$ equals 1.

- *Delay distributions:* The set of events is partitioned into two sets: E_e are the events with exponential distribution, and E_b are the events with bounded distribution. The functions $l : \mathrm{E}_b \mapsto \mathbf{N}$, and $u : \mathrm{E}_b \mapsto \mathbf{N}$, giving the lower and upper bounds on the times at which bounded events get scheduled. A bounded event x is called a *fixed-delay* event if $l_x = u_x$, and a *variable-delay* event if $l_x < u_x$. Associated with each variable-delay event x is a probability density function $\mathrm{f}_x(r)$ such that $\int_{l_x}^{u_x} \mathrm{f}_x(r)dr$ equals 1.

The process starts in some state s such that $\mathrm{f}_0(s)$ positive. The set of initially scheduled events is $\mathrm{E}(s)$. A (real-valued) clock is associated with each event, and it shows the time remaining till the occurrence of the event. The initial values for the clocks are chosen as follows: for each event x, a random value d_x is chosen independently using the associated probability distribution, and the clock is set to the negative value $-d_x$. Thus all the clock values are negative, and all the clocks count up at the same rate as the real-time [3]. A

[3] In an equivalent, and more conventional, formulation the clocks get set to positive values, and count down triggering the events on reaching 0. Our formulation was motivated by the existing definitions for timed automata which have clocks counting up.

state-transition is triggered when the clocks associated with some events become 0. A *generalized state* of the system is described by giving its state and a vector giving the readings of clocks corresponding to all the currently scheduled events. A generalized state is then represented by $\langle s, \nu \rangle$, where ν is a vector of real numbers of length $|\mathrm{E}(s)|$ assigning values to the clocks. Note that, in our presentation, we use the same symbol to denote an event and the clock associated with it.

If the current generalized state is $\langle s, \nu \rangle$ then after a delay of ϵ such that $\nu(x) + \epsilon < 0$ for all $x \in \mathrm{E}(s)$, the generalized state is $\langle s, \nu + \epsilon \rangle$. A state transition will be triggered when some element of the clock vector becomes 0. Note that two (or more) clocks can reach 0 value simultaneously; this is because we allow fixed-delay events. Let a be the set of events that trigger the state-transition. The process moves to some state s' such that $next(s, a, s')$ is positive. With this transition some events get descheduled, whereas some events get newly scheduled. We partition the set $\mathrm{E}(s')$ into two sets:

1. the set $old(s, a, s') = \mathrm{E}(s') \cap [\mathrm{E}(s) - a]$ is the set of events that were scheduled previously, and

2. the set $new(s, a, s') = \mathrm{E}(s') - old(s, a, s')$ is the set of events that are newly scheduled.

The clock readings in the new state are determined as follows: The clocks in $old(s, a, s')$ keep counting without being affected by the state-transition. For each newly scheduled event x, a random value d_x is chosen independently using the associated probability distribution, and the clock is set to the negative value $-d_x$. Notice that for a fixed-delay event x, this value is predetermined; for a variable-delay event x, the value d_x lies in the interval (l_x, u_x); and for an exponential event x, this value can be arbitrary.

The generalized state has enough information to decide the future behavior of the process, hence, what we have defined is a continuous-time Markov process $X(t)$ over the space of all generalized states. As far as the correctness of the system is concerned, we are mainly interested in the sequence of events the process participates in. Note that, with probability 1, the process makes a finite number of transitions in a bounded interval of time. Hence a particular behavior of the process can be described by an ω-sequence of event sets paired with their occurrence times. Such a sequence is called a *timed word*. To be precise, a timed word $\rho = (\sigma, \tau)$ consists of

- an infinite word σ over the alphabet Σ of event sets, and

- an infinite, strictly increasing sequence τ of positive real numbers satisfying the *progress* property that "for every $t \in \mathbf{R}$ there is some τ_j greater than t".

The probabilistic structure of the process gives a probability measure on the space of all timed words.

3 Timed automata

The correctness of the system is usually specified by ω-automata or temporal logics. In this paper we use deterministic timed automata [AD90] as a specification formalism. The formalism of timed automata is a generalization of finite-state machines over infinite strings (see [Th90] for an exposition to ω-automata). While ω-automata accept infinite words, timed automata are additionally constrained by timing requirements and accept timed words (see the article *The Theory of Timed Automata* in this issue for a detailed discussion).

A timed automaton operates with finite control — a finite set of locations and a finite set of real-valued clocks. All clocks proceed at the same rate and measure the amount of time that has elapsed since they were started (or reset). Each transition of the automaton has an associated constraint on the values of the clocks: the transition may be taken only if the current values of the clocks satisfy this constraint. A transition, in addition to changing the location of the automaton, may reset some of the clocks.

Formally, a timed automaton $\mathcal{A}$ consists of

- a finite set of locations Q,
- a start location $q_{init} \in$ Q,
- a finite set of clocks C, and
- a finite set of transitions $\mathcal{E}$. An edge $\langle q, q', a, \lambda, \delta \rangle$ represents a transition from location q to location q' on input symbol $a \in \Sigma$. The set $\lambda \subseteq$ C gives the clocks to be reset with this transition, and δ is the associated clock constraint. A clock constraint is a boolean combination of atomic conditions of the form $x \leq c$ and $c \leq x$, where x is a clock and c is an integer constant.

Given a time word (σ, τ), the timed automaton $\mathcal{A}$ starts in its start location q_{init} at time 0 with all its clocks initialized to 0. As time advances, the values of all clocks change reflecting the elapsed time. At time τ_i, $\mathcal{A}$ changes location from q_{i-1} to q_i using some transition of the form $\langle q_{i-1}, q_i, \sigma_i, \lambda_i, \delta_i \rangle$ reading the input σ_i, provided the current values of clocks satisfy δ_i. With this transition the clocks in λ_i get reset to 0, and thus start counting time with respect to it. This behavior is captured by defining *runs* of timed automata. A run records the location and the values of all the clocks at the transition points. Specifically, a run r, denoted by $(\overline{q}, \overline{\nu})$, of a timed automaton $\mathcal{A}$ over a timed word (σ, τ) is an infinite sequence of the form

$$r:\ \langle q_0, \nu_0 \rangle \ \xrightarrow[\tau_1]{\sigma_1}\ \langle q_1, \nu_1 \rangle \ \xrightarrow[\tau_2]{\sigma_2}\ \langle q_2, \nu_2 \rangle \ \xrightarrow[\tau_3]{\sigma_3}\ \cdots$$

where each $q_i \in$ Q and ν_i is an assignment from C to $\mathbf{R}$. In addition, the following conditions are satisfied:

- *Initiation:* $q_0 = q_{init}$, and $\nu_0(x) = 0$ for all $x \in$ C.
- *Consecution:* for all $i \geq 1$, there is an edge in $\mathcal{E}$ of the form $\langle q_{i-1}, q_i, \sigma_i, \lambda_i, \delta_i \rangle$ such that the clock vector $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfies δ_i, and $\nu_i(x) = 0$ if $x \in \lambda_i$, and $\nu_i(x) = \nu_{i-1}(x) + \tau_i - \tau_{i-1}$ otherwise.

As an example of a timed automaton, consider the automaton of Figure 1. The automaton starts in location q_0, and loops between the locations q_0, q_1, q_2, and q_3. It has two clocks x and y. The clock x gets set to 0 each time it moves from q_0 to q_1 reading a. The check $(x < 1)$? associated with the c-transition from q_2 to q_3 ensures that c happens

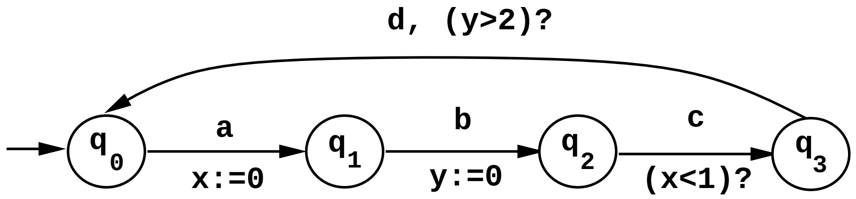

Figure 1: Timed automaton

within time 1 of the preceding a. A similar mechanism of resetting another independent clock y while reading b and checking its value while reading d, ensures that the delay between b and the following d is always at least 2. Thus the automaton has a run over (σ, τ) iff (1) $\sigma = (abcd)^\omega$, and (2) for each $j \geq 0$, $\tau_{4j+3} < \tau_{4j+1} + 1$, and $\tau_{4j+4} > \tau_{4j+2} + 2$.

We can couple acceptance criteria with timed automata, and use them to define languages of timed words. A *timed Muller automaton* $\mathcal{A}$ is a timed automaton with a Muller acceptance family $\mathcal{F} \subseteq 2^{\mathrm{Q}}$. A run $r = (\overline{q}, \overline{\nu})$ of the automaton $\mathcal{A}$ over a timed word (σ, τ) is called an *accepting run* iff the set of locations appearing infinitely often along r is in $\mathcal{F}$. The language $L(\mathcal{A})$ of timed words accepted by $\mathcal{A}$ is defined to be the set of timed words $\{(\sigma, \tau) \mid \mathcal{A}$ has an accepting run over $(\sigma, \tau)\}$.

When used as a specification formalism, the language of a timed automaton describes the desired property of timed words. A real-time probabilistic process $\mathcal{M}$ satisfies a specification presented as a timed automaton $\mathcal{A}$ iff the probability that $\mathcal{A}$ accepts the timed word generated by $\mathcal{M}$ equals 1. In other words, the set $L(\mathcal{A})$ has probability 1 according to the probability measure induced by $\mathcal{M}$ on the space of timed words. The probabilistic verification problem is to decide whether or not $\mathcal{M}$ satisfies $\mathcal{A}$. In general, the specification automaton $\mathcal{A}$ can be nondeterministic, and we donot know how to solve the probabilistic verification problem. However, the problem is solvable for a special class of timed automata, namely, *deterministic* automata.

For the conventional finite-state automata, the property of determinism requires that, for each location, given the next input symbol, the next location is uniquely determined. We want a similar criterion for determinism for the timed automata: given a location along with the clock values, and the next input symbol along with its time of occurrence, the location and the clock values after the next transition should be uniquely determined. So we allow multiple transitions starting at the same location with the same label, but require their clock constraints to be mutually exclusive so that at any time only one of these transitions is enabled. Formally, a timed automaton $\mathcal{A}$ is called *deterministic* iff for all $q \in \mathrm{Q}$, for all $a \in \Sigma$, for every pair of edges of the form $\langle q, -, a, -, \delta_1 \rangle$ and $\langle q, -, a, -, \delta_2 \rangle$, the clock constraints δ_1 and δ_2 are mutually exclusive (i.e., $\delta_1 \wedge \delta_2$ is unsatisfiable). The property of determinism ensures that a deterministic timed automaton has at most one run over a given timed word.

Deterministic timed Muller automata (DTMA) can specify many interesting properties of real-time systems. As an example, consider the automaton of Figure 2. The automaton starts in location q_0, and uses a single clock x. The clock is reset every time the automaton

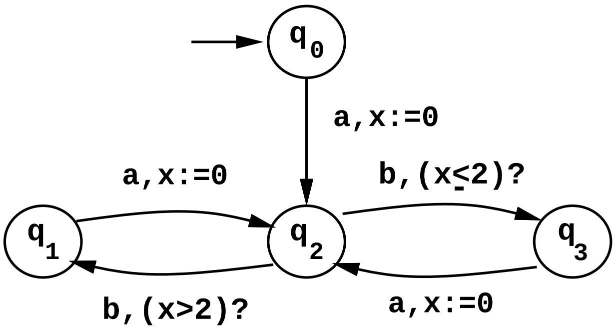

Figure 2: Deterministic timed Muller automaton

moves to location q_2 reading the symbol a. The location q_2 has two mutually exclusive outgoing transitions on b. The automaton moves to location q_3 if b appears within 2 time units after the previous a, and to location q_1 otherwise. The Muller acceptance condition is given by $\{\{q_2, q_3\}\}$. So any accepting run should loop between locations q_1 and q_2 only finitely many times, and between locations q_2 and q_3 infinitely many times. Interpreting the symbol b as a response to a request denoted by the symbol a, the automaton models the requirement of *convergent response time*: the response time should "eventually" be always less than 2 time units. The language of the automaton can be described as

$$L_{crt} \;=\; \{((ab)^\omega, \tau) \,|\, \exists i.\, \forall j \geq i.\, (\tau_{2j} < \tau_{2j-1} + 2)\}.$$

4 Verification algorithm

In this section, we show how to solve the probabilistic verification problem for specifications given as deterministic timed automata.

Let $\mathcal{M}$ be a probabilistic real-time process, and let $\mathcal{A}$ be a deterministic timed Muller automaton. We assume that $\mathcal{A}$ has precisely one run over any timed word. In general, a DTMA may not satisfy this property: it may not have any run over some timed words. However, this property can be attained by a simple transformation. First we add a dummy location q to the automaton. From each location q', for each symbol a, we add an a-labeled edge from q' to q. The clock constraint for this edge is the negation of the disjunction of the clock constraints of all the a-labeled edges starting at q'. We leave the acceptance condition unchanged. This construction preserves determinism as well as the set of accepted timed words. The new automaton has the property that for each location q' and each input symbol a, the disjunction of the clock constraints of the a-labeled edges starting at q' is a valid formula.

Before we present the algorithm, let us define an extension $\mathcal{M}^*$ of the process $\mathcal{M}$. The state of $\mathcal{M}^*$ records, in addition to the state of $\mathcal{M}$, the location and the clock values of the automaton $\mathcal{A}$ also. The $\mathcal{A}$-component of the state of $\mathcal{M}^*$ is updated so as to simulate the behavior of $\mathcal{A}$ while reading the timed word generated by $\mathcal{M}$. Observe that since $\mathcal{A}$ is deterministic, this update is completely determined by the choices made by $\mathcal{M}$. A generalized state of $\mathcal{M}^*$ is of the form $\langle s, q, \nu\rangle$, where s is a state of $\mathcal{M}$, q is a location of $\mathcal{A}$, and ν is a vector of real numbers of length $|\mathrm{E}(s) \cup \mathrm{C}|$. The interpretation ν gives values for the clocks corresponding to the scheduled events of $\mathcal{M}$ and the clocks of $\mathcal{A}$. For a state s of $\mathcal{M}$, the set of all active clocks is $\mathrm{E}^*(s) = \mathrm{E}(s) \cup \mathrm{C}$; the set of all nonexponential clocks is $\mathrm{E}_b^*(s) = \mathrm{E}_b(s) \cup \mathrm{C}$; All the clocks count the elapsed time, and thereby increasing the values in ν. The state of $\mathcal{M}$ changes when some of the clocks reach 0; the location of $\mathcal{A}$ is also updated accordingly. Let $a \in \Sigma(s)$ be the set of events causing the transition, and let $\langle q, q', a, \lambda, \delta \rangle$ be the unique edge of $\mathcal{A}$ such that the clock vector at the time of the state-transition satisfies δ. The state of $\mathcal{M}^*$ after the transition is $\langle s', q', \nu' \rangle$ such that $next(s, a, s')$ is positive. For each newly scheduled event x the value of $\nu'(x)$ is chosen according to its probability distribution, each clock y in λ is reset to 0, and all the remaining clocks keep counting without being affected by the state-transition.

This gives us a continuous-time Markov process $X^*(t)$ over the space of all generalized states of $\mathcal{M}^*$. A sample path ρ of the process is a map from $\mathbf{R}$ to the set of its generalized states. It can be constructed as described above by choosing the next state and specific values for the clocks corresponding to the newly scheduled events at all the transition points. The probability functions of $\mathcal{M}$ give a probability measure over the space of all sample paths.

The next step in this construction is to group the uncountably many generalized states of $\mathcal{M}^*$ into a finite number of equivalence classes. Informally, $\langle s, q, \nu \rangle$ is equivalent to $\langle s, q, \nu' \rangle$ iff ν and ν' agree on the integral parts of all the clock values and the integral parts of the differences between clock values. A *projected state* of $\mathcal{M}^*$ corresponds to an equivalence class of its generalized states. It keeps track of the integer parts of the clock values and the ordering of the fractional parts. The integer parts are needed to determine which clock constraints are currently satisfied, and which clocks have reached the value 0. The ordering of the fractional parts is needed to update the integral parts of different clocks in a consistent manner. This information is not required in two cases. For an exponential event x we need not keep track of its actual value. This is due to the "memoryless" property of the exponential distributions. Secondly, for a clock $y \in \mathrm{C}$, if y is never compared with a constant greater than d in the clock constraints of $\mathcal{A}$ then the value of y, once it crosses d, is of no importance. This observation is used to keep the number of projected states finite.

Let us make this precise. Let $\lfloor t \rfloor$ denote the greatest integer not exceeding t, for $t \in \mathbf{R}$. Given two real numbers t and t', we say that they *agree on the integral parts* iff (1) $\lfloor t \rfloor = \lfloor t' \rfloor$, and (2) either both t and t' are integers, or neither is an integer. For a clock $x \in \mathrm{C}$, let d_x be the largest constant d such that either $x \leq d$ or $d \leq x$ appears in some clock constraint in the transitions of $\mathcal{A}$. For a clock vector ν, a clock $x \in \mathrm{C}$ is *relevant* if $\nu(x) \leq d_x$.

Formally, define $\langle s, q, \nu \rangle \sim \langle s, q, \nu' \rangle$ iff

1. for each $x \in \mathrm{E}_b^*(s)$, either $\nu(x)$ and $\nu'(x)$ agree on the integral parts or x is not relevant for both ν and ν', and

2. for every pair x and x' in $\mathrm{E}_b^*(s)$ and relevant for ν, $[\nu(x) - \nu(x')]$ and $[\nu'(x) - \nu'(x')]$ agree on the integral parts.

We use $[s, q, \nu]$ to denote the equivalence class to which $\langle s, q, \nu \rangle$ belongs. We will call such a class a *projected state*. As an example, consider a system in state s with

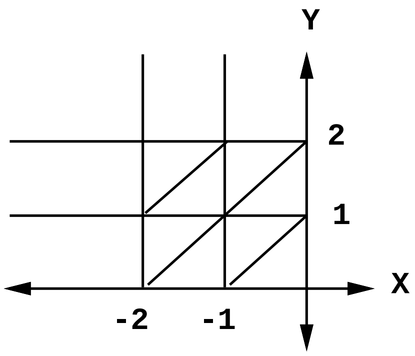

Figure 3: Equivalence classes

$\mathrm{E}(s) = \{x\}$. Let $u_x = 2$. Suppose the automaton is in location q, and it has a single clock y with $d_y = 2$. The equivalence relation on the clock vectors is depicted in Figure 3. For instance, the clock vector $[x = -0.8, y = 0.6]$ belongs to the projected state

$$\{\langle s, q, \nu\rangle : \nu(x) \in (-1, 0) \,\wedge\, \nu(y) \in (0, 1) \,\wedge\, \nu(y) > \nu(x) + 1\}.$$

We will denote it by $[s, q, -1 < x < 0, x + 1 < y < 1]$.

Let V denote the set of all projected states. There are only a finite number of projected states; the following bound can be proved by a simple combinatorial argument:

$$|\mathrm{V}| \; < \; |Q| \,\cdot\, \Pi_{x \in \mathrm{C}}\, d_x \,\cdot\, \Sigma_{s \in \mathrm{S}}[\,|\mathrm{E}_b(s) \cup \mathrm{C}|! \cdot \Pi_{x \in \mathrm{E}_b(s)}\, u_x\,].$$

Thus the number of projected states is linear in the number of states of $\mathcal{M}$, linear in the number of locations of $\mathcal{A}$, exponential in the number of simultaneously scheduled events in $\mathcal{M}$, exponential in the number of clocks of $\mathcal{A}$, and proportional to the constants involved. Each projected state can be represented in length linear in the size of the input.

Let $X_p(t)$ be the process over the state space V obtained from $\mathcal{M}^*$ by projecting each generalized state onto its equivalence class. Note that though $X_p(t)$ has a finite state space, it is not Markovian: the projected state does not have enough information to determine the future behavior of the system completely. Let ρ_p be a sample path of the projection process $X_p(t)$. Note that ρ_p is piecewise constant and right-continuous. It can be characterized by an ω-sequence over V, denoted by $\gamma(\rho_p)$, which records all the projected states appearing along ρ_p. In other words, we can find a sequence of times $t_0, t_1, t_2 \ldots$ and a sequence of projected states $\gamma_0\gamma_1\gamma_2\ldots$ such that for each i, $\rho_p(t_i) = \gamma_i$ and for any $t_i < t < t_{i+1}$, $\rho_p(t)$ is either γ_i or γ_{i+1}. For a sample path ρ_p of the projected process, let $\mathit{inf}(\rho_p)$ denote the set of projected states appearing infinitely often along $\gamma(\rho_p)$, and let $\mathit{inf}_Q(\rho_p)$ denote the projection of $\mathit{inf}(\rho_p)$ on Q. The following lemma relates the ergodic behavior of $\mathcal{M}^*$ to the probabilistic verification problem; its proof follows from the definitions.

Lemma [**1**]. The process $\mathcal{M}$ satisfies the specification $\mathcal{A}$ iff for every $\mathrm{Q}' \subseteq \mathrm{Q}$, the set of sample paths ρ of $\mathcal{M}^*$ such that $\mathit{inf}_Q(\rho_p) = Q'$ has positive probability only if $\mathrm{Q}' \in \mathcal{F}$. ∎

The first step of the model-checking algorithm is to define a graph G whose vertex set is V such that $X_p(t)$ moves along the paths in G. The state of $X_p(t)$ may change either because of a state transition of $\mathcal{M}$ or because of the change in the equivalence class of clock values due to the passage of time. For example, consider the system of Figure 3, and suppose the current state is $\langle s, q, [-0.8, 0.6] \rangle$. The projected state does not change for next 0.4 time units. However, at $t = 0.4$, the clock vector is $[-0.4, 1.0]$, which belongs to a different equivalence class. This projected state is only transient: at time $0.4 + \epsilon$, for any small positive ϵ, the generalized state belongs to a different equivalence class described by $[s, q, -1 < x < 0, 1 < y < x + 2]$. The projected state changes again at time 0.8 because the clock for x reaches 0, and a state-transition is triggered. Notice that the projected states that the process visits, are obtained by drawing a line from the point $[-0.8, 0.6]$ in the diagonally upward direction (parallel to the line $x = y$).

We call a projected state $[s, q, \nu]$ *transient* iff some clock x relevant to ν has zero fractional part. For such a state, the generalized state after a time delay of ϵ, however small, belongs to a different equivalence class. In Figure 3, the equivalence classes that lie on horizontal or vertical lines correspond to transient states.

The set of initial vertices V_0 consists of projected states of the form $[s, q_{init}, \nu]$ such that

(1) $\mathrm{f}_0(s)$ is positive, and
(2) for every fixed-delay event $x \in \mathrm{E}_b(s)$, $\nu(x) = -l_x$, and
(3) for every variable-delay event $x \in \mathrm{E}_b(s)$, $\nu(x) \in (l_x, u_x)$, and
(4) for every exponential event $x \in \mathrm{E}_e(s)$, $-\infty < \nu(x) < 0$, and
(5) if x is a variable-delay event or an exponential event, $\nu(x)$ is not an integer and for any x', $\nu(x') - \nu(x)$ is not an integer, and
(6) for every clock $x \in \mathrm{C}$, $\nu(x) = 0$.

Now we define an edge relation over the vertex set V to get a graph G. There are four different types of edges:

1. Consider a transient projected state $[s, q, \nu]$, and suppose for every $x \in \mathrm{E}(s)$, $\nu(x) \neq 0$. In this case, the change in the projected state is due to the passage of time. Let $\epsilon > 0$ be such that for each $0 < \epsilon' < \epsilon$, $[s, q, \nu + \epsilon']$ and $[s, q, \nu + \epsilon]$ are equivalent. There is exactly one edge out of $[s, q, \nu]$ to $[s, q, \nu + \epsilon]$. For example, the vertex $[s, q, -1 < x < 0, y = 1]$ has a single edge to $[s, q, -1 < x < 0, 1 < y < x + 2]$.

2. Consider a transient vertex $[s, q, \nu]$ such that the set a consisting of events $x \in \mathrm{E}(s)$ with $\nu(x) = 0$, is nonempty. In this case, the event set a triggers a state-transition of $\mathcal{M}$. Let $\langle q, q', a, \lambda, \delta \rangle$ be the unique edge such that ν satisfies δ. We have an edge to every vertex $[s', q', \nu']$ satisfying

   (1) $\mathit{next}(s, a, s')$ is positive,
   (2) for every $x \in \lambda$, $\nu'(x) = 0$,
   (3) for every $x \in \mathit{old}(s, e, s')$ or $x \in \mathrm{C} - \lambda$, $\nu'(x) = \nu(x)$, and
   (4) for every $x \in \mathit{new}(s, a, s')$,

      (a) if x is a variable-delay event then $\nu'(x) \in (l_x, u_x)$, and
      (b) if x is an exponential event then $-\infty < \nu'(x) < 0$, and
      (c) if x is a fixed-delay event then $\nu'(x) = -l_x$, and

(d) if x is a variable-delay or an exponential event, $\nu'(x)$ is not an integer and for any x', $\nu'(x') - \nu'(x)$ is not an integer,

For example, consider the vertex $[s, q, x = 0 < y < 1]$. Consider the change in the projected state triggered by the occurrence of x. Suppose $next(s, \{x\}, s')$ is positive, and $\mathrm{E}(s') = \{z\}$ with $l_z = 1$ and $u_z = 2$. Suppose $\mathcal{A}$ has a transition from $\langle q, q', \{x\}, \emptyset, true \rangle$. There are two possible edges corresponding to the state-transition from s to s' on x: $[s', q', -2 < z < -1, 0 < y < z + 2]$ and $[s', q', 0 < y < 1, -2 < z < y - 2]$, depending upon whether the fractional part of z is less than or greater than that of y. Note that the condition $4(d)$ disallows the choice $z + 2 = y$.

3. Consider the case when $[s, q, \nu]$ is a nontransient state, and suppose there is some nonexponential clock relevant to ν. As time progresses, some of the clocks in ν reach integer values changing the projected state. Let a be the set of clocks x relevant for ν such that for every relevant x', the fractional part of x is at least as much as the fractional part of x'. There is an edge corresponding to the clocks in a reaching an integral value; the new projected state is $[s, q, \nu']$ where $\nu'(x) = \lfloor \nu(x) \rfloor + 1$ if $x \in a$, and $\nu'(x) = \nu(x)$ otherwise. For instance, the projected state $[s, q, -2 < x < -1, x + 2 < y < 1]$ has an edge to $[s, q, -2 < x < -1, y = 1]$ corresponding to the event of the clock y reaching the value 1.

4. In addition, for a nontransient state $[s, q, \nu]$, for every exponential event $x \in \mathrm{E}_e(s)$, there is an edge to the projected state $[s, q, \nu']$ where $\nu'(x) = 0$ and $\nu'(x') = \nu(x')$ for $x' \neq x$. For instance, suppose in our example there was an additional exponential event z scheduled in state s, then the projected state $[s, q, -1 < x < 0, x + 1 < y < 1, -\infty < z < 0]$ has an edge to $[s, q, -1 < x < 0, x + 1 < y < 1, z = 0]$. This captures the fact that z may happen at any instant.

We will now relate properties of $\mathcal{M}^*$ to the graph G. The following lemma asserts that the process X_p faithfully follows the finite paths in G.

Lemma [2]. For every vertex $v \in V$, the set of sample paths ρ such that v appears on ρ_p is positive iff there is a path in G from some initial vertex $v_0 \in V_0$ to v.

Proof. The definition of G ensures the following: If there is no edge from u to u' in G, and if the current projected state is u, then the probability of the next projected state being u' is zero. The "only if" direction follows from this observation.

For the other direction, it suffices to show that for every edge from u to u' in G, if the process $X^*(t)$ is in some generalized state in u, then there is a positive probability that the next projected state will be u'. This follows from the definitions if the edge from u to u' is according to the clause 1 or 2. Suppose the edge from u to u' is according to the clause 3. The state u is a nontransient state, and consider the process $X^*(t)$ in some generalized state $\langle s, q, \nu \rangle$ in u. Let a be the set of nonexponential clocks reaching the integral values corresponding to this edge. Let ϵ be the time at which the clocks in a reach integral values according to ν. If all the exponential events have clock values smaller than $-\epsilon$ in ν, which has positive probability by the definition of the exponential distribution, then the next projected state is u'. The argument for the edges according to the clause 4 is similar. ∎

Now we need to relate the set of Q-states that X_p visits infinitely often to the strongly connected components of G. Given a set V' of projected states, we need to find a necessary and sufficient condition on V' so that the probability of $inf(\rho_p) = V'$ is positive. The algorithms for probabilistic model-checking for discrete Markov chains [CY88, Va85] rely on the following "probabilistic" property: if the process visits a vertex v infinitely often then each successor v' of v is also visited infinitely often. This property follows trivially from the fact that the probability of a transition from v to v', for discrete Markov chains, is a fixed positive number. We wish to prove a similar property for the projection process. But unlike the discrete case, $X_p(t)$ is not Markovian. Furthermore, the probability of a transition along an edge in G is not bounded from below, in general. Let us see how this could pose difficulties. Suppose the automaton has two clocks y and z, and there is one event x which has uniform distribution with $l_x = 0$ and $u_x = 1$. Consider a projected state $u = [s, q, x = 0 < y < z < 1]$. Consider an edge to $u' = [s', q', 0 < y < z < 1, y - 1 < x < z - 1]$. This corresponds to a state transition triggered by the event x; it gets rescheduled in state s', the new clock value is between -1 and 0, and its fractional part lies between those of y and z. If the current generalized state is $\langle s, q, \nu \rangle \in u$, the probability of the next projected state being u' is proportional to the difference $[\nu(z) - \nu(y)]$. Assume that the projection process visits u infinitely many times. If the sequence of the value of $[\nu(z) - \nu(y)]$ upon entry to u converges to 0, then the probability of transitioning from u to u' also converges to 0. This prevents us from concluding that X_p visits u' also infinitely often. The following lemma shows that this case cannot happen.

A set $W \subseteq V$ is a *bottom strongly connected component* (BSCC) of G iff (1) it is a strongly connected component, and (2) for every $v \in W$, if there is an edge from v to v' then v' is also in W.

Lemma [3]. Let $W \subseteq$ V be a set of projected states, and assume that the process starts in some $\langle s, q, \nu \rangle \in v$, for some $v \in W$. The set of sample paths ρ such that $inf(\rho_p) = W$ has positive measure iff W is a bottom strongly connected component of G.

Proof. By the previous lemma the process follows only the edges in G. Hence, if W is a BSCC, if started in V, the process stays in W with probability 1.

Now let us consider the other direction. To demonstrate better the ideas in the proof we start with a simpler case first, and then we consider the more general cases.

Case 1: We start with the case in which there are no clocks in the timed automaton and hence all the clocks correspond to events. We also assume that for each event x, x is a variable-delay event. We need first a definition which is key to our proof. We say that a visit of X_p to a projected state v is δ-separated iff upon entry all the fractional parts of the clock values differ by at least δ. Consider an edge from v to v' in G. First we observe the following fact.

Fact 1: For any sufficiently small $\delta > 0$, there is a positive ε and a positive δ' such that for any δ-separated visit to v, the probability of the next projected state to be a δ'-separated visit to v', is bounded below by ε.

Consequently, if X_p has infinitely many δ-separated visits to v, then with probability 1 it must have infinitely many δ'-separated visits to v'.

Now we identify another property of the process. Let $n = 3|\mathrm{E}^*|$, and $\delta_0 = 1/n$. Divide the interval $(0, 1)$ into n disjoint intervals $I_1, I_2, \ldots I_n$, each of length δ_0. Whenever a new clock value x is chosen, there exists an interval I_j such that if the fractional part of x lies in I_j then it is at least δ_0-separated from all the old clocks. Hence, for each event

x, each time it gets scheduled, there is a positive probability p_x that the clock for x is chosen at least δ_0-separated from all the other clocks. Note that p_x depends only upon the distribution f_x and the separation δ_0. Let $T = \max\{u_x : x \in \mathrm{E}^*\}$. We will show the following.

Fact 2: Consider the process in some arbitrary state $\langle s, q, \nu\rangle$. Then the probability that after time T the clocks of the process will be δ_0-separated is bounded below by some constant.

We prove this as follows. By time T all the active clocks in $\langle s, q, \nu\rangle$ will have reached 0 at least once. For each event x, let P_x denote the property that at time T either x is not scheduled, or that the last time x was scheduled before T, its clock was chosen at least δ_0-separated from the other clocks. One can easily see that independently of the other choices, the probability of P_x is bounded below by p_x since it corresponds to the probability that the fractional part of x will be chosen in some "free" interval I_j (containing no other clock). Let p_m be the smallest such constant for all events $x \in \mathrm{E}^*$. From the above discussion it follows that the probability of all properties P_x holding at time T is at least ${p_m}^{|\mathrm{E}^*(s)|} > {p_m}^{|\mathrm{E}^*|} > 0$.

We now complete the proof of the lemma as follows. Let $W \subset V$ be a subset such that the probability that the process stays in W and visits exactly all the states in W infinitely often, is positive. That is, the set of sample paths ρ such that ρ_p has only vertices from W and $\mathit{inf}(\rho_p) = W$, has positive measure. One can easily see that Fact 2 implies that conditioned on this event, each trajectory of the process X_p makes almost surely infinitely many δ_0-separated visits to at least one vertex u in W (possibly different for each trajectory). Now Fact 1 implies that all successors of u will appear on this trajectory with probability one infinitely often and being δ'-separated. By induction, all the vertices reachable from u must appear infinitely often with probability one. This implies that W must be a bottom strongly connected component.

Case 2: We now allow the timed automaton to have its own clocks. The proof will follow basically the same argument as the proof of Case 1. Now while defining δ-separation for a clock vector ν we require that all the scheduled events and all the relevant and distinct clocks of $\mathcal{A}$ have fractional parts δ-apart. Thus we allow the possibility of two clocks of the automaton to have the same value; that is, two automaton clocks x and y with $\nu(x) = \nu(y)$ are considered as a single clock for what concerns δ-separation; and we donot worry about the clocks not relevant to ν. We need to modify slightly Fact 2 as follows.

Fact 2′: If some generalized state $\langle s, q, \nu\rangle$ appears infinitely often in a trajectory then, with probability 1, it will infinitely often be the case that $\langle s, q, \nu\rangle$ is δ_0-separated (on the same trajectory).

We prove this as follows. Observe that every relevant clock x in ν has been reset by the expiration of some event $y(x) \in \mathrm{E}$ (possibly different each time ν appears in the trajectory). Hence the value of x is a continuation of the value of $y(x)$ (think of $y(x)$ as being "incarnated" into x). This implies that the fractional part of x will also be δ_0-separated from the same events and clocks from which the fractional part of $y(x)$ was δ_0-separated, when $y(x)$ was scheduled. Since $\langle s, q, \nu\rangle$ appears infinitely often there must exist a subsequence of times t_i, $i = 1, 2, ...$, at which $\langle s, q, \nu\rangle$ appears, such that during the interval from t_i to t_{i+1} all active events in ν and all events resetting relevant clocks in ν have been scheduled at least once. This is true if one chooses the above time sequence such that $t_{i+1} - t_i > T + d$ where d is the maximum constant d_x for all clocks x. Then ν

will be δ_0-separated at time t_{i+1} if for each event which is either in ν or resets a relevant clock in ν, at the last time it was scheduled it was δ_0-separated from the rest of the clocks. By using a similar argument as in the proof of Case 1, the probability of this happening is bounded below by ${p_m}^{|\mathrm{E}^*|} > 0$ independently of the history up to time t_{i+1}. The proof of Case 2 of the lemma can now be completed as in Case 1.

Case 3: Now we consider the case when the events are of both fixed-delay and variable-delay types. The reader should notice that if there are no fixed-delay events, then the graph G contains no reachable vertices with ν containing events with equal fractional parts since the probability of such transitions is zero (by the definition of the distribution function for the variable-delay events).

The idea in the proof is the following. All the events and clocks which have equal fractional parts must have some common "ancestor" event which is of the variable-delay type and has scheduled them by sequences of fixed-delay events (it is defined as the "initial" event expiring at time 0 if such an ancestor does not exist). An ancestor event of a singleton group is the event itself if this event is a variable-delay event, or the first variable-delay event by tracing back the "family" history of the event; a similar definition holds for clocks. Let us group the events and relevant clocks of a vertex $\langle s, q, \nu\rangle$ in groups where the events and clocks in each group have equal fractional parts and assume that the ancestor event of each group is a variable-delay event. Then the probability that when this vertex appears the fractional parts of the groups will be δ_0-separated is bounded from below by the probability that all ancestor events be δ_0-separated when scheduled, which is bounded below by ${p_m}^{|\mathrm{E}^*|}$.

In order to follow the steps of the proof in the previous case, we would like to show that if $\langle s, q, \nu\rangle$ appears infinitely often, then there are infinitely many time values $t_1, t_2, \ldots$ such that the probability that $\langle s, q, \nu\rangle$ appears at each time t_i with ν δ_0-separated, is bounded from below by some constant, independently of the history of the system. Unfortunately, this does not hold for the following reason. Two events could always be separated by less than δ, for any δ, in $\langle s, q, \nu\rangle$ with positive probability. Consider the following example: at time zero two variable delay events x_1 and x_2 with uniform distribution on $(0, 1)$ are scheduled. When each x_i expires, it schedules an event x_i' with fixed delay of 1; the events x_i' reschedule themselves when they expire. One can easily see that the separation between the x_i's will always be the same, equal to the initial separation of x_1 and x_2 which can be arbitrarily small. This example suggests the following idea. If some of the events in $\langle s, q, \nu\rangle$ eventually "synchronize", they will always have some constant separation from then on. If the minimum of the separation over all such events (after all synchronizations have occurred) is δ', we can show that with probability one, after all "synchronizations" have occurred, $\langle s, q, \nu\rangle$ will appear infinitely often being δ_0'-separated, where δ_0' is the minimum of δ_0 and δ'. Then we can use similar steps as in the previous cases to prove the lemma.

For completeness we add some steps of the proof. Consider a trajectory which repeats $\langle s, q, \nu\rangle$ infinitely often. Let $x_1, \ldots, x_n$ denote the ancestor events in ν (possibly different each time $\langle s, q, \nu\rangle$ appears). Let E_s be the ancestor events which eventually remain the same after some finite time t, and let δ' be the minimum of the separation of the fractional parts of the events in E_s. From the definition of E_s it follows that after some finite time $\langle s, q, \nu\rangle$ will appear infinitely often with (a) its events in E_s being δ' separated, and (b) the remaining of the events in ν, which do not have ancestors in E_s, will have ancestor events which are scheduled infinitely often; denote this set of events by E_c. Observe that we can find a sequence $t < t_1 < t_2 \ldots$ such that the trajectory is in state $\langle s, q, \nu \rangle$ at each time t_i, and all the ancestors of the events in E_c are scheduled at least once between each t_i and t_{i+1}. By repeating the steps in the previous proofs, it follows that at the end of the each interval the probability that the events in E_c are δ_0-separated is bounded below by a constant, hence with probability one, $\langle s, q, \nu \rangle$ will appear infinitely often being δ_0'-separated. Since for each trajectory which repeats $\langle s, q, \nu \rangle$ infinitely often such a $\delta_0' > 0$ exists, by repeating the arguments we used in the previous cases the proof follows.

Case 4: Same as in Case 3, with the additional events with exponential distributions. The proof is an easy extension of the proof of Case 3 as follows. Consider the state of the system projected on the bounded events. Then with probability one if $\langle s, q, \nu \rangle$ appears infinitely often, it will also appear infinitely often with its bounded events and relevant clocks being δ_0'-separated for some positive δ_0'. One can easily see that each time $\langle s, q, \nu \rangle$ appears and is δ_0'-separated, then all events are at least δ_0' away from 0. By the memoryless property of the exponential distributions, the probability that each exponential event lies in the interval $(-\delta_0', 0)$ is bounded from below. This implies that infinitely often when $\langle s, q, \nu \rangle$ is visited, a transition corresponding to the expiration of an exponential clock will be taken. ▪

Now putting together all the lemmas we get the following:

Theorem [**1**]. The process $\mathcal{M}$ satisfies its specification $\mathcal{A}$ iff for every bottom strongly connected component W of G such that W is reachable from some initial vertex in V_0, the projection of V' on Q belongs to $\mathcal{F}$. ▪

This gives a decision procedure for model-checking. After constructing the graph G, we only need to compute its bottom strongly connected components that are accessible from the initial vertices, and check if the set of Q-states corresponding to each such component is an accepting set for $\mathcal{A}$. The size of the graph G is exponential in the size of the input (particularly, the number of events and clocks, and the length of the constants involved). However, the graph need not be constructed explicitly. Since each vertex can be represented in length linear in the size of the input, the following theorem follows:

Theorem [**2**]. The verification problem for a finite-state probabilistic real-time process $\mathcal{M}$ and a deterministic timed automaton specification $\mathcal{A}$ can be solved in PSPACE. ▪

5 Conclusions

In this paper, we have presented a way of reasoning about finite-state real-time systems. The process model is quite general, and accounts for both the real-time and the probabilistic aspects of the system behavior.

In this presentation, we have required all distributions to be either bounded, discrete, or exponential. One can easily extend the proofs in the paper for arbitrary unbounded distributions with increasing hazard rate, or mixtures of arbitrary distributions one of which has increasing hazard rate. We conjecture that the algorithm is correct even for more general continuous distributions, however, proving the correctness may be nontrivial.

The other question of interest is whether the basic technique of the algorithm, which involves constructing the projection process, can be used to check other types of specifications. We do not yet know how to check properties specified using nondeterministic timed automata or using linear real-time logics.

The algorithm outlined here is expensive. To make it more practical, some heuristic techniques need to be devised. Observe that the verification problems are typically PSPACE, even in the absence of real-time and probabilities. We feel hopeful that the techniques emerging from the ongoing research on how to cope with the state-explosion problem [BCMDH90], [GW90] will be useful in implementing the algorithm presented here.

Acknowledgements: We thank Gerald Shedler for pointing us to the existing literature on modeling systems as generalized semi-Markov processes. We also thank Joseph Halpern and Moshe Vardi for useful discussions.

References

[ACD90] R. Alur, C. Courcoubetis, D.L. Dill, "Model-checking for real-time systems," In *Proceedings of the Fifth IEEE Symposium on Logic in Computer Science*, pages 414-425, 1990.

[ACD91] R. Alur, C. Courcoubetis, D.L. Dill, "Model-checking for probabilistic real-time systems," In *Automata, Languages, and Programming: Proceedings of the 18th ICALP*, Lecture Notes in Computer Science 510, 1991.

[AD90] R. Alur, D.L. Dill, "Automata for modeling real-time systems," In *Automata, Languages, and Programming: Proceedings of the 17th ICALP*, Lecture Notes in Computer Science 443, pages 322-335, 1990.

[AH89] R. Alur, T.A. Henzinger, "A really temporal logic," In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 164-169, 1989.

[BCMDH90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, L.J. Hwang, "Symbolic model-checking: 10^{20} states and beyond," In *Proceedings of the Fifth IEEE Symposium on Logic in Computer Science*, pages 428-439, 1990.

[CY88] C. Courcoubetis, M. Yannakakis, "Verifying temporal properties of finite-state probabilistic programs," In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, pages 338-345, 1988.

[EC82] E.A. Emerson, E.M. Clarke, "Using branching-time temporal logic to synthesize synchronization skeletons," *Science of Computer Programming* **2**, pages 241-266, 1982.

[GW91] P. Godefroid, P. Wolper, "A partial approach to model-checking," In *Proceedings of the Sixth IEEE Symposium on Logic in Computer Science*, pages 406-415, 1991.

[HJ89] H. Hansson, B. Jonsson, "A framework for reasoning about time and reliability," In *Proceedings of the Tenth IEEE Real-Time Systems Symposium*, pages 102-111, 1989.

[Ko90] R. Koymans, "Specifying real-time properties with Metric Temporal Logic," *Journal of Real-Time Systems*, **2**, pages 255-299, 1990.

[Le90] H.R. Lewis, "A logic of concrete time intervals," In *Proceedings of the Fifth IEEE Symposium on Logic in Computer Science*, pages 380-389, 1990.

[LS82] D. Lehman, S. Shelah, "Reasoning with time and chance," *Information and Control* **53**, 1982.

[PZ86] A. Pnueli, L. Zuck, "Probabilistic verification by tableaux," In *Proceedings of the First IEEE Symposium on Logic in Computer Science*, 1986.

[Sh87] G.S. Shedler, *Regeneration and Networks of Queues*, Springer-Verlag, 1987.

[Th90] W. Thomas, "Automata on infinite objects," *Handbook of Theoretical Computer Science,* volume B, pages 133-191, 1990.

[Wh80] W. Whitt, "Continuity of generalized semi-Markov processes," *Math. Oper. Res.* **5**,1980.

[Va85] M. Vardi, "Automatic verification of probabilistic concurrent finite-state programs," In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 327-338, 1985.