Model-checking for Probabilistic Real-time Systems

(Appeared in the Proc. of the 18th ICALP, LNCS 510, 1991)

Rajeev Alur — AT&T Bell Labs
Costas Courcoubetis — University of Crete and FORTH
David L. Dill — Stanford University

Abstract

Model-checking is a method of verifying concurrent systems in which a state-graph model of the system behavior is compared with a temporal logic formula. This paper extends model-checking to *stochastic real-time* systems, whose behavior depends on probabilistic choice and quantitative time. The specification language is TCTL, a branching-time temporal logic for expressing real-time properties. We interpret the formulas of the logic over generalized semi-Markov processes. Our model can express constraints like "the delay between the request and the response is distributed uniformly between 2 to 4 seconds".

We present an algorithm that combines model-checking for real-time non-probabilistic systems with model-checking for finite-state discrete-time Markov chains. 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.

1 Introduction

As digital systems become smaller and cheaper, their use to control and interact with physical processes will inevitably increase. Because of this, it is important to find methods for reasoning about computational models that are physically realistic. Physical systems work in *real time* and behave *probabilistically*, so we need formal methods for analyzing real-time stochastic systems.

One of the most successful techniques for *automatic verification* of finite-state systems has been *model-checking*: a property is given as a formula of a propositional temporal logic and automatically compared with a state-transition graph representing the actual behavior of the system. One of the advantages of this method is its efficiency: model-checking is linear in the product of the size of the structure and the size of the formula, when the logic is the branching-time temporal logic CTL (computation tree logic) ([CES86]). CTL model-checking has been used for proving the correctness of concurrent systems such as circuits and communication protocols.

In the last few years, model-checking has been extended to models of *real-time* and *probabilistic* systems. Model-checking algorithms have been devised for several different models of real time ([AH89], [HJ89], [ACD90], [Le90]). Of these models, we prefer those that consider time to be a *dense linear order*, because they are more realistic (although admittedly more difficult to analyze). In a previous paper, we presented a method for model-checking systems in dense time ([ACD90]): timing was added to a finite state-transition graph through a finite set of real-valued clocks which keep track of the various delays in the system. Timing was added to CTL by allowing subscripts to the temporal operators. For example, $\exists\Diamond_{<3}\,p$ means that some p-state is reachable within 3 time units.

In the simplest type of probabilistic model-checking, the state-transition graph is converted into a finite-state Markov chain by placing probabilities on the transitions ([LS82], [Va85], [PZ86], [CY88]). The path quantifiers in the logic, which previously meant "for all paths" and "there exists a path" are reinterpreted to mean "with probability one" and "with positive probability," respectively. These methods can easily be extended to real-time when time is considered to be discrete (each passing time unit results in a state transition) ([HJ89]), but the dense-time case is much more challenging.

We add probabilities to our model of timed state-transition graphs 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], [W80]). The process has uncountably many states, so we define a projection process over a finite state-space. Unfortunately the projection process is not Markovian. In general, the properties of such processes are hard to analyze. However, we show that it can be used to check TCTL-properties.

The paper is organized as follows: In section 2, we briefly review the model-checking for discrete Markov chains against CTL-specifications. The process model and the logic TCTL is defined in section 3. Section 4 develops the model-checking algorithm. In the concluding section, we mention directions in which this work can be extended.

2 Model-checking for discrete-time Markov chains

Various researchers have studied the problem of verification of finite-state probabilistic programs ([LS82], [Va85], [PZ86], [CY88]).

The program is modeled by a labeled Markov chain over a finite state space. Let AP be a set of atomic propositions. A labeled Markov chain is then specified by giving (1) a finite set S of states, (2) a labeling function $\mu : \mathrm{S} \mapsto 2^{\mathrm{AP}}$ giving the assignment of truth values to atomic propositions in each state, and (3) a probability transition matrix π. For every pair of states, $\pi(s, s')$ gives the probability of the transition from s to s'. This gives rise to a Markov process $X(n)$ over the state space S. The probability structure also gives rise to a probability measure on the sequence space S^{ω}. We will use ρ to denote a sample run of the system.

The correctness of the system is usually specified by ω-automata or temporal logics. In this paper, we will use the branching-time logic CTL ([EC82]) as the specification formalism. Let us first review the syntax of CTL. Each atomic proposition is a CTL-formula. Any boolean combination of CTL-formulas is a CTL-formula. If ϕ is a path formula then $\exists\phi$ is a CTL-formula. The path formulas are defined by the clause: if ϕ_1 and ϕ_2 are CTL-formulas then $\Box\phi$ and $\phi_1 \mathcal{U} \phi_2$ are path formulas.

The universal quantifier is defined as:

$$\forall \phi_1 \mathcal{U} \phi_2 \;=\; \neg[\exists\Box\neg\phi_2 \;\vee (\neg\phi_2)\,\mathcal{U}(\neg\phi_1 \wedge \neg\phi_2)]$$

Some of the commonly used abbreviations include $\exists\Diamond\phi$ for $\exists\,\mathbf{true}\,\mathcal{U}\,\phi$, $\forall\Diamond\phi$ for $\neg\exists\Box\neg\phi$, and $\forall\Box\phi$ for $\neg\exists\Diamond\neg\phi$. For instance, $\exists\Diamond p$ means that with positive probability some p-state is visited. $\forall\Box p$ means that p stays invariant with probability 1.

The semantics of the logic is defined with respect to labeled Markov chains. The satisfaction relation is defined inductively. We say that a run ρ satisfies a path formula $\phi_1 \mathcal{U} \phi_2$ iff $\rho(n) \models \phi_2$ for some n, and $\rho(n') \models \phi_1$ for all $n' < n$. A run ρ satisfies $\Box\phi$ iff $\rho(n) \models \phi$ for all n. The existential quantifier means "with positive probability". $s \models \exists\phi$ iff the set of runs starting at s and satisfying the path formula ϕ has positive probability. The universal quantifier means "with probability 1".

As it turns out, the satisfaction relation does not depend on the actual values in the probability transition matrix π, but only on which of its entries are non-zero.

The model-checking question is the following: given a labeled Markov chain with an initial state s_0, and a CTL-formula ϕ, whether or not $s_0 \models \phi$. The problem can be reduced to that of (non-probabilistic) model-checking for CTL with fairness constraints. First we construct a Kripke structure with S as the set of vertices. There is an edge between s and s' iff $\pi(s, s')$ is positive. Given a formula ϕ, the model-checking algorithm labels the vertices recursively with the subformulas of ϕ (or with their negation). A vertex s is labeled with $\exists\, \phi_1 \mathcal{U}\, \phi_2$ iff there is a path (in the usual graph-theoretic sense) from s to a vertex s' labeled with ϕ_2 visiting only the vertices labeled with ϕ_1. This is because the Markov process traverses any finite path in the Kripke structure with positive probability. Handling the formulas of the form $\exists \Box \psi$ is trickier. For a run ρ let $\mathit{inf}(\rho)$ denote the set of states repeating infinitely often. A set of vertices V forms a *bottom strongly connected component* (BSCC) iff (1) it is a strongly connected component, and (2) for every $v \in V$, all the successors of v are also in V. The following lemma is crucial for model-checking:

> *For any $V \subseteq$ S, the set of runs ρ starting in V such that $\mathit{inf}(\rho) = V$, has positive measure iff V is a bottom strongly connected component.*

Hence, we label a vertex s with $\exists \Box\, \psi$ iff there is a path from s leading to a BSCC such that all the vertices along this path and in the BSCC are labeled with ψ.

Probabilistic model-checking is no harder than the non-probabilistic case as far as the algorithmic complexity is concerned. It can be solved in PTIME.

3 Introducing Real-time

We want to introduce the notion of time explicitly in the 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 $\mathcal{F}$". We assume that all the system delays are distributed within known constant bounds and have finite interval support. That is, we consider distributions that satisfy the following constraint:

> *There exist 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.*

Note that this also includes discrete probability functions which choose the values from a finite set of integers. To keep the presentation simple, we assume that for every density function f there exists a single interval I such that for every $t \in \mathbf{R}$, 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 1 interval.

To model concurrent delays we need to introduce several clocks running concurrently. Our notation is borrowed from the literature on generalized semi-Markov processes ([W80], [Sh87]).

A (finite-state) *real-time probabilistic process* has the following components:

- A finite set S of states.
- A labeling function $\mu : \mathrm{S} \mapsto 2^{\mathrm{AP}}$ giving the assignment of truth values to atomic propositions in each state.
- Associated with each state s is a finite set of events E(s). E(s) gives the set of events that are scheduled to occur when the process is in state s. We will denote the set of all events $(\cup_{s \in \mathrm{S}}\, \mathrm{E}(s))$ by E.

- A probabilistic transition function $next : \mathrm{S} \times P^+(\mathrm{E}) \times \mathrm{S} \mapsto [0, 1]$. ($P^+(\mathrm{E})$ denotes the set of all nonempty subsets of E.) $next(s, \alpha, s')$ is the probability that the next state is s' when the event set α occurs in state s. For every state s and event set α, we require $(\Sigma_{s' \in \mathrm{S}}\, next(s, \alpha, s')) = 1$.
- The functions $l : \mathrm{E} \mapsto \mathbf{N}$, and $u : \mathrm{E} \mapsto \mathbf{N}$, giving the lower and upper bounds on the times at which events get scheduled. We assume that for every event e, $l_e \leq u_e$. An event e is called a *fixed-delay* event if $l_e = u_e$, and a *variable-delay* event, otherwise.
- Associated with each event e is a probability density function f_e such that, for $t \in \mathbf{R}$, $\mathrm{f}_e(t)$ is positive iff $l_e \leq t \leq u_e$.

The process starts in some state s. 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. All the clocks count down at the same rate as the real-time. A 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. We assume that for each state s, the events in $\mathrm{E}(s)$ are ordered. A generalized state is then represented by $\langle s, c\rangle$, where c is a vector of real numbers of length $|\mathrm{E}(s)|$. The vector c should satisfy: $0 < c(e) \leq u_e$, for each event e. We use Σ to denote the space of all generalized states.

For a clock vector c, let $min(c)$ be the event set such that $c(e) \geq c(e')$, for each $e \in \mathrm{E}(s)$, $e' \in min(c)$. When the system is in the generalized state $\langle s, c\rangle$, $min(c)$ comprises of the events scheduled to occur at the earliest time. Let $tmin(c) = c(e)$ for $e \in min(c)$. A state transition will be triggered at time $tmin(c)$ because of the occurrence of events in $min(c)$. The process moves to some state s' such that $next(s, min(c), s')$ 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 $old(s, c, s') = \mathrm{E}(s') \cap [\mathrm{E}(s) - min(c)]$ is the set of events that were scheduled previously, and (2) the set $new(s, c, s') = \mathrm{E}(s') - old(s, c, s')$ is the set of events that are newly scheduled. The clock readings for the events in the new state are determined as follows: for each newly scheduled event e, the value of the corresponding clock is chosen independently from the interval $[l_e, u_e]$ using the associated probability distribution. The clocks corresponding to the events in $old(s, c, s')$ keep counting down without being affected by the state-transition.

The single-step probability transition function can be computed as follows: For each $e \in new(s, c, s')$, let I_e be an open interval of the real-line. Let β be the set of generalized states of the form $\langle s', c'\rangle$ where for each old event e, $c'(e)$ equals $c(e) - tmin(c)$, and for each newly scheduled event e, $c'(e) \in I_e$. The probability that the generalized state at time $tmin(c)$ belongs to the set β equals

$$next(s, min(c), s') \cdot \Pi_{e \in new(s,c,s')} \int_{I_e} \mathrm{f}_e.$$

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 state-space Σ. A sample path ρ of the process is a map from $\mathbf{R}$ to Σ. 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. Note that, with probability 1, the process makes a finite number of transitions in a bounded interval of time. Hence it is fully specified by an ω-sequence over Σ which records the generalized states at all the transition points. The single-step probability transition function gives a probability measure over the space of all sample paths.

To write real-time specifications, we need to extend the syntax of the logic also. One way to extend the syntax of a temporal logic for writing down real-time constraints, is to allow subscripts on the temporal operators to limit their scope in time ([Ko89], [EMSS89], [AH89]). We will use the logic TCTL ([ACD90]) as a specification language. In this logic we can write $\exists\Diamond_{<5}\, p$ to say that p becomes true within 5 time units along some computation path. As before, we now interpret $\exists$ as "with positive probability".

The formulas of TCTL are inductively defined along with the path formulas:

- Each atomic proposition p is a TCTL-formula.
- If ϕ_1 and ϕ_2 are TCTL-formulas then so are $\neg\phi_1$ and $(\phi_1 \wedge \phi_2)$.
- If ϕ is a path formula then $\exists\phi$ is a TCTL-formula.
- If ϕ_1 and ϕ_2 are TCTL-formulas then $\Box_{\sim d}\phi_1$ and $(\phi_1 \mathcal{U}_{\sim d}\phi_2)$ are path-formulas, where $d = 0, 1, 2, \ldots$ and $\sim$ stands for one of the binary relations $<$, $\leq$, $=$, $\geq$, or $>$.

The formula $\exists\,\phi$ means that there is a positive probability that the sample paths satisfy ϕ. A sample path ρ satisfies $(\phi_1 \mathcal{U}_{<d}\phi_2)$ iff till some time value t less than d all the states along ρ satisfy ϕ_1, and $\rho(t)$ satisfies ϕ_2.

The other logical connectives can be defined as usual. Throughout the paper we will use $\sim$ to denote one of the binary relations $<$, $\leq$, $=$, $\geq$, or $>$. The universal formulas are defined using the following definition:

$$\forall(\phi_1 \mathcal{U}_{\sim d}\phi_2) \;=\; \neg[\,\exists\Box_{\sim d}\neg\phi_2 \;\vee\; \exists((\neg\phi_2)\,\mathcal{U}_{\sim d}(\neg\phi_1 \wedge \neg\phi_2))\,]$$

The quantifier $\forall$ means "with probability 1". We also define abbreviations $\exists\Diamond_{\sim d}\phi$ for $\exists(\mathbf{true}\,\mathcal{U}_{\sim d}\phi)$, $\forall\Diamond_{\sim d}\phi$ for $\neg\exists\Box_{\sim d}\neg\phi$, and $\forall\Box_{\sim d}\phi$ for $\neg\exists\Diamond_{\sim d}\neg\phi$. For instance, $\forall\Box_{\leq 3}\, p$ means that p stays invariant till time 3 with probability 1. A typical response property that "every p-state is guaranteed (with probability 1) to be followed by some q-state within time 2," is specified by the following formula:

$$\forall\Box[\,p \;\rightarrow\; \forall\Diamond_{\leq 2}\, q\,].$$

The unrestricted temporal operators correspond to TCTL operators subscripted with ≥ 0. For example, $\exists\Diamond\phi$ corresponds to $\exists\Diamond_{\geq 0}\phi$.

The semantics of TCTL is defined with respect to real-time probabilistic processes. We define below what it means for a TCTL-formula to be true in a generalized state of this process, and for a path formula to be true along a sample path. The satisfaction relation is defined inductively as follows:

$\langle s, c\rangle \models p$ iff $p \in \mu(s)$.
$\langle s, c\rangle \models \neg\phi$ iff $\langle s, c\rangle \not\models \phi$.
$\langle s, c\rangle \models (\phi_1 \wedge \phi_2)$ iff $\langle s, c\rangle \models \phi_1$ and $\langle s, c\rangle \models \phi_2$.
$\langle s, c\rangle \models \exists\phi$ iff the set of sample paths starting at $\langle s, c\rangle$ and satisfying ϕ has positive probability.
$\rho \models \Box_{\sim d}\phi$ iff $\rho(t) \models \phi$ for all $t \sim d$.
$\rho \models (\phi_1 \mathcal{U}_{\sim d}\phi_2)$ iff $\rho(t) \models \phi_2$ for some $t \sim d$, and $\rho(t') \models \phi_1$ for all $t' < t$.

The model-checking question can now be phrased as follows:

Given a TCTL-formula ϕ, and a probabilistic real-time process with a specified initial generalized state $\langle s_0, c_0\rangle$, whether or not $\langle s_0, c_0\rangle \models \phi$.

4 Model-checking

In this section, we show that the truth of a formula can be checked by exploring a finite graph derived from the original system. Our algorithm is the "obvious" probabilistic extension of the existing algorithm for model-checking in non-probabilistic dense real-time case ([ACD90]). However, the proof of correctness is non-trivial.

4.1 The Projection Process

The first step in this construction is to group the uncountably many generalized states into a finite number of equivalence classes.

Informally, we say that $\langle s, c\rangle$ is equivalent to $\langle s, c'\rangle$ iff c and c' agree on the integral parts of all the clock values and the integral parts of the differences between clock values. Let us make this precise. Let $\lfloor t\rfloor$ denote the greatest integer smaller than 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.

Formally, define $\langle s, c\rangle \equiv \langle s, c'\rangle$ iff

(1) for each $e \in \mathrm{E}(s)$, $c(e)$ and $c'(e)$ agree on the integral parts, and
(2) for each pair e and e' in $\mathrm{E}(s)$, $[c(e) - c(e')]$ and $[c'(e) - c'(e')]$ agree on the integral parts.

We use $[s, c]$ to denote the equivalence class to which $\langle s, c\rangle$ belongs. We will call such a class a *projected state*. Let Σ_0 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:

$$|\Sigma_0| < \Sigma_{s\in \mathrm{S}}[\,\mathrm{E}(s)! \cdot \Pi_{s\in \mathrm{E}(s)}\, u_e\,].$$

Let $X_p(t)$ be the process over the state space Σ_0 obtained from $X(t)$ by projecting each 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.

The first step of the model-checking algorithm is to define a graph G whose vertex set is Σ_0 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 $X(t)$ or because of the change in the equivalence class of clock values due to the passage of time. For example, consider a system in state s with $\mathrm{E}(s) = \{x, y\}$. Suppose the clock vector is $[0.8, 1.2]$. The projected state is the set $\{\langle s, c\rangle : c(x) \in (0,1) \wedge c(y) \in (1,2) \wedge c(y) < c(x) + 1\}$, and it will be denoted as $[s, 0 < x < 1 < y < (x+1)]$. The projected state does not change for next 0.2 time units. However, at $t = 0.2$, the clock vector is $[0.6, 1.0]$, which belongs to a different equivalence class. This projected state is only transient: at time $0.2 + \epsilon$, for any small positive ϵ, the generalized state belongs to a different equivalence class described by $[s, 0 < x < y < 1]$. The projected state changes again at time 1.2 because the clock for x reaches 0, and a state-transition is triggered.

For a clock vector c, let $\mathit{minf}(c)$ denote the set of events with minimum fractional part. Let us denote this minimum fractional part by $\mathit{tminf}(c)$. The projected state changes at time $\mathit{tminf}(c)$. We call a projected state $[s, c]$ *transient* iff $\mathit{tminf}(c) = 0$.

Now we define an edge relation over the vertex set Σ_0 to get a graph G. Consider a projected state $[s, c]$. First we consider the case when it is a transient state. Let $\epsilon > 0$ be such that for each

$0 < \epsilon' < \epsilon$, $\langle s, c-\epsilon'\rangle$ and $\langle s, c-\epsilon\rangle$ are equivalent. There is exactly one edge out of $[s, c]$ to $[s, c-\epsilon]$. For example, the vertex $[s, 0 < x < 1 = y]$ has a single edge to $[s, 0 < x < y < 1]$.

Now consider the case when $[s, c]$ is a nontransient state with $minf(c) \neq min(c)$. In this case, as time progresses, the projected state of the system changes when the clocks in $minf(c)$ touch an integer boundary. There is a single transition coming out of $[s, c]$ to the transient state $[s, c - tminf(c)]$. For example, $[s, 0 < x < 1 < y < (x+1)]$ has an edge to $[s, 0 < x < 1 = y]$.

Finally let us consider the case where $[s, c]$ is a nontransient state with $minf(c) = min(c)$. In this case we have an edge to every vertex $[s', c']$ satisfying (1) $next(s, min(c), s')$ is positive, (2) for every $e \in new(s, c, s')$ with $l_e < u_e$, $c'(e) \in (l_e, u_e)$, $c'(e)$ is not an integer, and for any e', $c'(e) - c'(e')$ is not an integer, (3) for every $e \in new(s, c, s')$ with $l_e = u_e$, $c'(e) = l_e$, and (4) for every $e \in old(s, c, s')$, $c'(e) = c(e) - tmin(c)$. For example, consider the vertex $[s, 0 < x < y < 1]$. The change in the projected state is triggered by the occurrence of x. Suppose $next(s, x, s')$ is positive, and $\mathrm{E}(s') = \{y, z\}$ with $l_z = 1$ and $u_z = 2$. There are two possible successors corresponding to the state-transition from s to s' on x: $[s', 0 < y < 1 < z < (y+1)]$ and $[s', 0 < y < 1 < z < 2, (z-y) > 1]$.

We will provide now a sequence of lemmas which relate properties of $X(t)$ to the region graph G. 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 Σ_0, 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 $t_0, t_1, \ldots$ 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}. The following lemma asserts that the process X_p faithfully follows the finite paths in G.

Lemma [1]. Let $\gamma = \gamma_0, \ldots \gamma_n$ be a finite sequence of projected states, and assume that the process starts in some $\langle s, c\rangle \in \gamma_0$. The set of sample paths ρ such that γ is a prefix of $\gamma(\rho_p)$, has positive measure iff there is an edge from γ_i to γ_{i+1} in G, for each i.

Proof. The definition of G ensures the following two facts: (1) If there is an edge from v to v' in G, and if the process is started in some $\langle s, c\rangle \in v$, then there is a positive probability that the next projected state will be v'. (2) If there is no edge from v to v' in G, and if the current projected state is v, then the probability of the next projected state being v' is zero. The lemma follows from these two observations. ∎

Now we need to consider infinite paths in G. For a sample path ρ_p of the projected process, let $inf(\rho_p)$ denote the set of projected states appearing infinitely often along $\gamma(\rho_p)$. 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 algorithm for model-checking for discrete Markov chains (section 2) relies 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. Consider a projected state $v = [s, 0 < x < y < z < 1]$. Consider an edge to $v' = [s', 0 < y < x < z < 1]$ (this corresponds to a state transition triggered by the event x; it gets rescheduled in state s'). If the current generalized state is $\langle s, c\rangle \in v$, the probability of the next projected state being v' is proportional to the difference $[c(z) - c(y)]$. Assume that the projection process visits v infinitely many times. If the sequence of the value of $[c(z) - c(y)]$ upon entry to v converges to 0, then the probability of transitioning from v to v' also converges to 0. This prevents us from concluding that X_p visits v' also infinitely often. The following lemma shows that this case cannot happen.

Lemma [2]. Let $V \subseteq \Sigma_0$ be a set of projected states, and assume that the process starts in some $\langle s, c\rangle \in v$, for some $v \in V$. The set of sample paths ρ such that $inf(\rho_p) = V$ has positive measure iff V is a bottom strongly connected component of G.

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

Now let us consider the other direction. In the following argument only the fractional parts of the differences between the clock values play an important role, so henceforth consider all references to clock values to be modulo 1.

We say that a visit of X_p to a projected state v is δ-separated iff if two clocks differ then they differ by at least δ. Consider an edge from v to v' in G. First we observe the following: 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 being 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 $n = 3|\mathrm{E}|$. 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 for a variable-delay event is chosen, there exists an interval I_j such that if x lies in I_j then it is at least $(1/n)$-separated from all the old clocks. Hence, for each variable-delay event e, there is a positive probability p_e that the clock for e is chosen at least $(1/n)$-separated from all the other clocks. Let p_m be the minimum of all these values. Note that p_m depends only upon the distributions f_e and the separation $(1/n)$. Let $t = \max\{u_e : e \in \mathrm{E}\}$. Consider the process in the generalized state $\langle s, c\rangle$. Suppose the clocks in c are $\delta(c)$-separated. Let δ_0 be the maximum of $\delta(c)$ and $(1/n)$. Notice that for the clocks corresponding to fixed-delay events, there is no choice in setting the value. However, whenever such a clock is set, the probability of it being δ_0-separated from all the other clocks is at least p_m. By time t all the current clocks run down to 0. For each event e, let P_e denote the property that at time t either e is not scheduled, or that the last time e was scheduled before t, its clock was chosen at least δ_0 away from the other clocks. Independent of the other choices, the probability of P_e is bounded below by p_m. Hence, the probability of all properties P_e holding is at least $p_m^{|\mathrm{E}|}$. Thus, the probability that at time t all the clocks are separated by δ_0 is at least $p_m^{|\mathrm{E}|}$ (note that this probability does not depend on the initial state $\langle s, c\rangle$) — (‡).

Let V be a strongly connected component such that the probability of the process staying in V and visiting all states in V infinitely often is positive. By (‡), the process X_p makes infinitely many δ_0-separated visits to some vertex in V. Since V is finite, it follows from (†), that the process traverses each edge starting in V infinitely often with probability 1. Hence, V must be a BSCC. ∎

4.2 Model-checking of untimed formulas

First we consider only the "untimed" fragment of TCTL; that is, consider the formulas where all temporal operators have the subscript "≥ 0". Given such a formula ϕ we label the vertices of G with subformulas of ϕ using the following rules starting with the subformulas of length 1, then with those of length 2, and so on.

- Label $[s, c]$ with an atomic proposition p iff $p \in \mu(s)$.
- Label v with $\neg\psi$ iff v is not labeled with ψ, label it with $\phi_1 \wedge \phi_2$ iff it is labeled with both ϕ_1 and ϕ_2.
- Label v with $\exists(\phi_1 \mathcal{U} \phi_2)$ iff there exists a finite path $\gamma = \gamma_0, \ldots \gamma_n$ in G starting at v such that (1) all the intermediate vertices $\gamma_0, \ldots \gamma_{n-1}$ are labeled with ϕ_1, (2) the

last vertex γ_n is labeled with ϕ_2, and (3) either γ_n is a transient state or it is labeled with ϕ_1.

- Label v with $\exists\Box\,\psi$ iff there a finite path γ starting at v and ending at vertex in a BSCC V of G such that all vertices on γ and all vertices in V are labeled with ψ.

The correctness of the algorithm is stated by the following lemma:

Lemma [3]. The above algorithm labels a vertex $[s, c]$ with ψ iff the process started in $\langle s, c\rangle$ satisfies ϕ.

Proof. The proof is by induction on the structure of the formula. The cases when ψ is an atomic formula, or a boolean combination of formulas, are trivial. The correctness for the case $\psi = \exists(\phi_1\,\mathcal{U}\,\phi_2)$ follows from the definitions and the lemma 1. The correctness for the case $\psi = \exists\Box\,\psi'$ uses lemma 2. ∎

4.3 Model-checking of timed formulas

It is now fairly straightforward to handle the case of formulas with timing, such as $\exists(\phi_1\,\mathcal{U}_{\sim d}\phi_2)$, using the ideas in [ACD90]. The basic method is to add a clock to the system to keep track of the time that has elapsed since model-checking of the formula began.

Let us consider a formula of the form $\psi = \exists(\phi_1\,\mathcal{U}_{<d}\phi_2)$ and a generalized state $\langle s, c\rangle$. To check the formula, we introduce an additional clock x^* set to d. Consider this augmented process X^* which starts in $\langle s, c\rangle$. States of this new process are labeled with an additional proposition $p_{<d}$ which is true iff the clock x^* is still active. The definition of the satisfaction of formulas is as before. To check the truth of the formula ψ we check the truth of the "untimed" formula $\exists[\phi_1\,\mathcal{U}\,(\phi_2 \wedge p_{\sim d})]$ for the new process. To analyze this process we extend the equivalence relation to handle the additional clock. This corresponds to further refining of the projected states. In the remaining section we formalize these ideas.

Given a TCTL-formula ϕ, let $d(\phi)$ be $d+1$ where d is the largest constant appearing as a subscript in ϕ. Let Σ^* comprise of generalized states of the form $\langle s, c\rangle$ where the clock vector c, in addition to the clocks associated with the events in $\mathrm{E}(s)$, may assign a value to x^*. If c assigns some value to x^*, we say that x^* is *active*, and in that case, $c(x^*) \leq d(\phi)$. The equivalence relation $\equiv$ is extended to Σ^*. The definition is same as in Section 4.1 (we consider x^*, whenever active, to be in the set of currently scheduled events). The set of projected states is Σ_0^*.

The region graph G^* is defined as before over the vertex set Σ_0^*. While defining the edge relation, whenever the clock x^* runs down to 0, it just becomes inactive without any state-transition.

For each subscript $\sim d$ appearing in ϕ, we introduce a new proposition $p_{\sim d}$. Label a vertex $[s, c]$ with active x^* with $p_{\sim d}$ iff $[d(\phi) - c(x^*)] \sim d$. If x^* is not active then label $[s, c]$ with $p_{\sim d}$ iff $d(\phi) \sim d$.

Now we present the model-checking algorithm. As before, given ϕ we label the vertices of G with subformulas of ϕ starting with the subformulas of length 1, then with those of length 2, and so on. The labeling of G gives a labeling for the vertices of G^*: a vertex $[s, c]$ of G^* gets the labels of $[s, c']$ in G, where c' is c without the assignment for x^*. The rules for labeling are same for atomic propositions, and for boolean combinations of formulas. The rules for timed formulas are given below.

- Label $[s, c]$ with $\exists(\phi_1\,\mathcal{U}_{\sim d}\phi_2)$ iff there exists a finite path $\gamma = \gamma_0, \ldots \gamma_n$ in G^* starting at $[s, c^*]$ such that (1) c^* is an extension of c which assigns $d(\phi)$ to x^*, (2) all the intermediate vertices $\gamma_0, \ldots \gamma_{n-1}$ are labeled with ϕ_1, (3) the last vertex γ_n is labeled with ϕ_2 and with $p_{\sim d}$, and (4) either γ_n is a transient state or it is labeled with ϕ_1.

- Label $[s, c]$ with $\exists\Box_{\sim d}\,\psi$ iff G^* has a finite path γ starting at $[s, c^*]$ and leading to a BSCC V of G^* such that all vertices on γ and all vertices in V are labeled either with ψ or with $\neg p_{\sim d}$.

Lemma [4]. The labeling of vertices with the timed formulas is correct: a vertex $[s, c]$ is labeled with ψ iff the process started in $\langle s, c\rangle$ satisfies ϕ.

Proof. The proof is by induction on the structure of the formula. The cases when ψ is an atomic formula, or a boolean combination of formulas, are trivial. Let us consider the case $\psi = \exists(\phi_1\,\mathcal{U}_{\sim d}\phi_2)$, and a projected state $[s, c]$.

Let X^* be the augmented process which starts in the generalized state $\langle s, c^*\rangle$. The clock vector c^* is extension of c which assigns $d(\phi)$ to x^*. the additional clock x^* does not interfere with the state-transition behavior of X^*. Along a sample path of X^* the additional proposition $p_{\sim d}$ is true at time t iff $t \sim d$. The process X satisfies $[\phi_1\,\mathcal{U}_{\sim d}\,\phi_2]$ with positive probability iff the process X^* satisfies $[\phi_1\,\mathcal{U}\,(\phi_2 \wedge p_{\sim d})]$ with positive probability.

The set of projected states for X^* is a subset of Σ_0^*. The edges of G^* capture the behavior of X^* correctly. As in Lemma 1, we can prove that X^* follows only the paths in G^*. Now the model-checking problem is same as in the untimed case.

The case $\psi = \exists\Box_{\sim d}\,\psi'$ is similar. ■

4.4 Complexity Analysis

Now we proceed to analyze the complexity of the model-checking algorithm.

Let $\delta(\mathrm{E})$ denote the maximum of $|\mathrm{E}(s)|$ over $s \in \mathrm{S}$. Thus $\delta(\mathrm{E})$ is the maximum number of events scheduled at the same time. Let $\delta(\mathrm{f})$ denote the maximum of $\Pi_{e\in \mathrm{E}(s)} u_e$ over $s \in \mathrm{S}$. The number of projected states in G^* is then $O[|S| \cdot d(\phi) \cdot \delta(\mathrm{f}) \cdot 2^{\delta(\mathrm{E})}]$.

Let $\delta(next)$ denote the number of distinct pairs of states s and s' for which $next(s, \alpha, s')$ is positive for some α. An inspection of the construction of the region graph shows that the number of edges in G^* is $O[(|S| + \delta(next)) \cdot d(\phi) \cdot \delta(\mathrm{f}) \cdot 2^{\delta(\mathrm{E})}]$.

The algorithm essentially checks CTL-formulas in the region graph G^*. From the complexity of CTL model-checking ([CES86]), it follows that the time complexity of the algorithm is linear in the size of G^* and the size of ϕ. Thus the complexity is linear in the total number of states in S, but it is exponential in the number of simultaneously scheduled events, and is proportional to the upper bounds in the probability distributions. However, as in case of discrete-time probabilistic verification, the algorithm does not use the actual values given by the transition function *next*, and the form of the distributions f_e. Thus the description of the system need only specify (1) for each s, s', and α, whether $next(s, \alpha, s')$ is positive, and (2) for each e, the values of l_e and u_e.

Even though the size of the region graph is exponential in the length of the input description, it need not be stored explicitly. As in [ACD90], a nondeterministic version of the labeling procedure can be implemented so as to use space only linear in the length of the input. Also the proof of PSPACE-hardness for TCTL model-checking ([ACD90]) shows, after definitional modifications, that the probabilistic verification problem is PSPACE-hard. The following theorem follows:

Theorem [**Model-checking**]. The probabilistic verification problem for TCTL is PSPACE-complete. ■

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.

Observe that the verification problems are typically PSPACE, even in the absence of real-time and probabilities. In case of CTL model-checking, the number of states in the global state-transition graph is exponential in the number of components in the system. Like CTL-based verification, our algorithm complexity is also proportional to the number of global states. The exponential in $\delta(\mathrm{E})$ is probably not an additional problem, since $\delta(\mathrm{E})$ is only linear in the number of independent components in the system. The additional blow-up is by the actual values of the upper bounds u_e. However, this blow-up by constants is observed in all other formalisms for real-time; even in absence of probabilities, and even with the simplifying assumption of discrete-time. We feel hopeful that the techniques emerging from the ongoing research on how to cope with the state-explosion problem ([BCMDH90], [God90]) will be useful in implementing the algorithm presented here.

We have used the branching-time logic TCTL as a specification language. Recently, various other formalisms have been proposed to specify correctness of the real-time systems ([AD90], [AH89]). We can use the ideas of this paper to develop an algorithm to verify the specifications given as deterministic timed automata ([AD90]). However, the problem of checking properties specified using nondeterministic timed automata or linear real-time logics, yet remains to be solved.

Acknowledgements: We thank Gerald Shedler for pointing us to the existing literature on modeling systems as generalized semi-Markov processes. This prompted us to reformulate our process model from an earlier draft of this paper. We also thank Joseph Halpern and Moshe Vardi for useful discussions.

References

[ACD90] R. Alur, C. Courcoubetis, D. Dill, "Model-checking for real-time systems," 5th LICS, 1990.

[AD90] R. Alur, D.L. Dill, "Automata for modeling real-time systems," 17th ICALP, In *Lecture Notes in Computer Science*, **443**, 1990.

[AH89] R. Alur, T.A. Henzinger, "A really temporal logic," 29th FOCS, 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," 5th LICS, 1990.

[CES86] E.M. Clarke, E.A. Emerson, A.P. Sistla, " Automatic verification of finite-state concurrent systems using temporal logic specifications," ACM *TOPLAS* **8(2)**, 1986.

[CY88] C. Courcoubetis, M. Yannakakis, "Verifying temporal properties of finite-state probabilistic programs," 29th FOCS, 1988.

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

[Go90] P. Godefroid, "Using partial orders to improve automatic verification methods," In *Proceedings of Workshop on Computer Aided Verification*, Rutgers, 1990.

[HJ89] H. Hansson, B. Jonsson, "A Framework for Reasoning about Time and Reliability," Real Time Systems Symposium, 1989.

[Ko89] R. Koymans, "Specifying message passing and time-critical systems with temporal logic," Ph.D. Thesis, Eindhoven Univ. of Tech., 1989.

[Le90] H.R. Lewis, "A logic of concrete time intervals," 5th LICS, 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," 1st LICS, 1986.

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

[W80] W. Whitt, "Continuity of generalized semi-Markov processes," *Math. Oper. Res.* **5**,1980.

[Va85] M. Vardi, "Automatic verification of probabilistic concurrent finite-state programs," 26th STOC, 1985.