Monitoring Temporal Properties of Stochastic Systems⋆

A. Prasad Sistla and Abhigna R. Srinivas

Abstract. We present highly accurate deterministic and probabilistic methods for monitoring temporal properties of stochastic systems. The deterministic algorithms employ timeouts that are set dynamically to achieve desired accuracy. The probabilistic algorithms employ coin tossing and can give highly accurate monitors when the system behavior is not known.

1 Introduction

Designing and developing correct concurrent systems is a challenging process. An alternative to developing such components is to use an existing off-the-shelf component even if it does not exactly satisfy the user requirements. In our earlier works [7, 8, 15], we proposed an approach for customizing such components to user requirements. The approach proposed there uses a run time monitor that continuously monitors the executions of the component and reports any violations of the desired property. In this paper, we develop methods that make such monitors highly accurate using the information about the underlying system. We also develop alternate highly accurate randomized algorithms for such systems.

We assume that the correctness of the system is specified by a LTL formula or by an automaton on infinite strings. The monitoring problem consists of observing the computation of the system in real-time and reporting a violation if it does not satisfy the formula. This is complicated by the fact that the computations are non-terminating, i.e., infinite and property violation needs to be detected by observing only the prefixes of computations. It is well known that the only properties that can be monitored exactly are the safety properties [1, 14, 4]. However, an arbitrary property can still be monitored by erring sometimes. That is, not reporting a violation when there is one (called *false acceptances*), or reporting a violation when there is none (*false rejections*).

Two different techniques have been employed, in the literature, to monitor a general property f. In the first approach [2], call it a *liberal* approach, one monitors for the violations of the safety part of the property f (note that any property f is equivalent to the conjunction of a safety and a liveness property [1].). In this case, the monitor exhibits false acceptances, but not false rejections. In the case of false acceptances, the computation violates the liveness part of f. An alternate approach, called *conservative* approach [7, 8, 15], is to take a safety property h that implies f and to monitor for its violations. This method allows

⋆ This research is partly supported by the NSF grants CCF-0742686 and CCR-0205365

false rejections, but not false acceptances. The number of false rejections can be reduced by taking h to be as weak as possible, although in general there is no weakest such property h [7, 8, 15]. Note that one can also have monitors that exhibit false rejections as well as acceptances.

We model the system being monitored as a stochastic system and define two different accuracy measures for monitors of stochastic systems— *acceptance* accuracy and *rejection* accuracy. The *acceptance accuracy* is defined to be the probability that a good computation of the system ,i.e., one satisfying the formula f, is accepted by the monitoring algorithm. The *rejection accuracy* is defined to be the probability that a bad computation, i.e., one not satisfying the formula f, is rejected by the algorithm. Ideally, both the accuracies should have value one. Note that the acceptance accuracy of a liberal algorithm is always one while the rejection accuracy of a conservative algorithm is always one.

In this paper, we consider systems modeled by Hidden Markov Chains (HMC) [3], which are Markov chains that have outputs associated with each state; only the outputs are observable but not the states. HMCs are widely used for modeling systems whose states can not be fully observed. We present a conservative monitor for such systems whose acceptance accuracy can be made as close to one as possible. This algorithm dynamically resets the timeouts to achieve the desired accuracy. We show that the timeouts can be reset so that their values increase only linearly with each reset and such that the desired accuracy is achieved. Both the accuracies of the monitor have value one when the states of the HMC are fully visible.

The above methods for achieving accuracy assume that the underlying Markov Chain defining the system is known. In this paper we propose probabilistic algorithms for monitoring systems when their possible behaviors are not known in advance. Actually, we use the standard automata based approach [14] for monitoring the safety part of the property and the probabilistic approach for monitoring the liveness part. A *probabilistic monitor* for a property is a randomized algorithm (i.e., one that use random choices) that rejects with probability one every computation that does not satisfy the property. A *strong* probabilistic monitor for a property is a probabilistic monitor that accepts, with non-zero probability, every computation that satisfies the property. We say that a property is strongly monitorable if there is a strong monitor for it. For example, a strong monitor for the liveness property $\Diamond P$ rejects with some probability p after each input until P is satisfied.

We show that the class of strongly monitorable properties is exactly the class of properties that can be recognized by deterministic, possibly infinite state, Buchi automata. This implies that that the property $\Box\Diamond P$ is not strongly monitorable. Although, $\Box\Diamond P$ is not strongly monitorable, we present a probabilistic monitor for this property that satisfies some nice properties. We present such probabilistic monitors for the LTL formula $\Box\Diamond P \rightarrow \Box\Diamond Q$ and conjunctions of such formulas. These algorithms together with traditional deterministic algorithms for safety properties give us probabilistic monitors for deterministic Buchi automata as well as deterministic Streett automata.

We also give *hybrid* algorithms that combine the probabilistic methods with the conservative approaches of [7, 8, 15], that employ counters, to obtain highly accurate monitors. For example, the accuracy of a conservative monitor (for $\Diamond P$) employing a counter of size k can be improved by a factor of 2^k by combining with randomized techniques. We present experimental results demonstrating the high accuracy of the hybrid algorithms.

In summary, the following are the main contributions of the paper:

- A conservative deterministic monitor employing counters, for monitoring properties specified by deterministic Buchi automata, for systems modeled as HMCs. These monitors can be designed to achieve a desired accuracy and have accuracy 1 when the HMC is fully visible.
- A number of probabilistic monitoring techniques for liveness properties.
- Introduction of strong monitors and the result showing that the class of languages having strong monitors is exactly the class of languages whose complement is recognized by possibly infinite state Buchi automata.
- Highly accurate hybrid techniques that combine deterministic and probabilistic methods.

The paper is organized as follows. Section 2 contains definitions. Section 3 presents a highly accurate deterministic monitor when system behavior is known. Section 4 describes probabilistic and hybrid methods. Section 5 describes experimental results. Section 6 has concluding remarks and comparison to related work. Appendix which contains proofs of lemmas is in section 7.

2 Definitions and Notation

Sequences. Let S be a finite set. Let $\sigma = s_0, s_1, \ldots$ be a possibly infinite sequence over S. The length of σ, denoted as $|\sigma|$, is defined to be the number of elements in σ if σ is finite, and ω otherwise. For any $i \geq 0$, $\sigma[0, i]$ denotes the prefix of σ up to s_i. If α_1 is a finite sequence and α_2 is a either a finite or a ω-sequence then $\alpha_1\alpha_2$ denotes the concatenation of the two sequences in that order. We let S^*, S^ω denote the set of finite sequences and the set of infinite sequences over S. If $C \subseteq S^\omega$ and $\alpha \in S^*$ then αC denotes the set $\{\alpha\beta\ : \beta \in C\}$.

Automata and LTL. A *Büchi automaton* (NBA for short) $\mathcal{A}$ on infinite strings is a quintuple $(Q, \Sigma, \delta, q^0, F)$ where Q is a possibly infinite set of states; Σ is a finite alphabet of symbols; $\delta\colon Q \times \Sigma \to 2^Q$ is a transition function; $q_0 \in Q$ is an initial state; $F \subseteq Q$ is a set of accepting/final automaton states. The *generalized transition function* $\delta^*\colon Q \times \Sigma^* \to 2^Q$ is defined in the usual way, i.e., for every state q, $\delta^*(q, \epsilon) = \{q\}$, and for any $\sigma \in \Sigma^*$ and $a \in \Sigma$, $\delta^*(q, \sigma a) = \cup_{q' \in \delta^*(q,\sigma)} \delta(q', a)$. If for every $(q, a) \in Q \times \Sigma$, $|\delta(q, a)| = 1$, then $\mathcal{A}$ is called a *deterministic* Büchi automaton (or DBA for short). Let $\sigma :\ a_1, \ldots$ be an infinite sequence over Σ. A *run* r *of* $\mathcal{A}$ on σ is an infinite sequence $r_0, r_1, \ldots$ over Q such that $r_0 = q_0$ and for every $i > 0$, $r_i \in \delta(r_{i-1}, a_i)$. The run r of a Büchi automaton is *accepting* if there exists an infinite set I of indices such that, for each $i \in I$, $q_i \in F$. The automaton $\mathcal{A}$ *accepts* the ω-string σ if it has an accepting run over σ (for the case of DBAs, the automaton has a single run over σ). The *language accepted by* $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of ω-strings that $\mathcal{A}$ accepts. A language L' is called *ω-regular* if it is accepted by some Büchi automaton. A Streett automaton is like a Buchi automaton excepting that its accepting condition is given by a set of pairs of states the form (U, V) where $U, V \subseteq Q$. In this case, a run $r = r_0, ..., r_i, ...$ is accepting if for every pair (U, V) in the accepting condition, if the set $\{i : r_i \in U\}$ is infinite then the set $\{i : r_i \in V\}$ is also infinite. A finite state Buchi or Streett automaton is one that has only a finite number of states. LTL is the standard Propositional Linear Temporal Logic proposed by Pnueli. It uses the temporal operators $\Box$ (always),$\Diamond$ (some time), $\mathcal{U}$ (until), $\bigcirc$ (next-time) together with standard boolean connectives including $\rightarrow$(implication).

Hidden Markov Chains. We assume that the reader is familiar with basic probability theory and random variables and Markov chains. We consider stochastic systems given as Markov Chains [9] and monitor their computations for satisfaction of a given property specified by an automaton or a temporal formula. A Markov chain $G = (S, R, \phi)$ is a triple satisfying the following: S is a set of countable states; $R \subseteq S \times S$ is a total binary relation (i.e., for every $s \in S$, there exists some $t \in S$ such that $(s, t) \in R$); and $\phi : R \rightarrow (0, 1]$ is a probability function such that for each $s \in S$, $\sum_{(s,t)\in R} \phi((s, t)) = 1$. Note that, for every $(s, t) \in R$, $\phi((s, t))$ is non-zero. Intuitively, if at any time the system is in a state $s \in S$, then in one step, it goes to some state t such that $(s, t) \in R$ with probability $\phi((s, t))$. A finite path p of G is a sequence $s_0, s_1, ..., s_n$ of states such that $(s_i, s_{i+1}) \in R$ for $0 \leq i < n$. We extend the probability function to such paths, by defining $\phi(p) = \prod_{0\leq i<n} \phi((s_i, s_{i+1}))$.

We assume that there is a finite set $\mathcal{P}$ of atomic propositions that represent conditions on system states. Let Σ denote $2^{\mathcal{P}}$, the power set of $\mathcal{P}$. Each member of Σ denotes the set of atomic propositions that are true in a state of the system. From here onwards, we assume that Σ is the input alphabet of the property automata that we consider. If the property is given by a temporal formula then the atomic propositions appearing in the formula are drawn from $\mathcal{P}$. For any $C \subseteq \Sigma^\omega$, let $\bar{C}$ denote the set $\Sigma^\omega - C$. For an atomic proposition $P \in \Sigma$, when used in a sequence, P represents the set of elements of Σ that contain P; similarly $\neg P$ represents the set of elements that do not contain P.

A Hidden Markov Chain (HMC) [3] $H = (G, O, r_0)$ is a triple where $G = (S, R, \phi)$ is a Markov chain, $O : S \rightarrow \Sigma$ is the output function and $r_0 \in S$ is the initial state. Intuitively, for any $s \in S$, $O(s)$ is the output generated in state s and is the set of atomic propositions true in s; this output is generated when ever a transition entering state s is taken. The generated symbols become inputs to the monitor. H is called Hidden Markov chain because, one only observes the outputs generated in each state but not the actual state. We extend the output function O to paths of G as follows. For any finite path $p = s_0, s_1, ..., s_n$ in G, $O(p) = O(s_0), O(s_1), ..., O(s_n)$. The probability distribution on the single step state transition of G induces a probability distribution on the sets of sequences of outputs generated. To define these distributions formally, let $\mathcal{E}$ be the smallest class of subsets of Σ^ω satisfying the following properties: for every $\alpha \in \Sigma^*$, $\alpha\Sigma^\omega \in \mathcal{E}$; $\mathcal{E}$ is closed under countable union (i.e., if $C_0, ...C_i, ...$ is a finite or infinite sequence of elements in $\mathcal{E}$, then $\bigcup_{i\geq 0} C_i$ is also in $\mathcal{E}$); it is closed under complementation (i.e., for every $C \in \mathcal{E}$, $\bar{C}$ is also in $\mathcal{E}$). The elements of $\mathcal{E}$ are called *measurable* subsets of Σ^ω. It is not difficult to see that $\mathcal{E}$ is also closed under countable intersections. It can be shown that, for any automaton $\mathcal{A}$ with input alphabet Σ, $L(\mathcal{A})$ is measurable.

Now, for any system state $r \in S$, we define a probability function $\mathcal{F}_r$ defined on $\mathcal{E}$ as follows. Intuitively, for any $C \in \mathcal{E}$, $\mathcal{F}_r(C)$ denotes the probability that an output sequence generated from the system state r, is in C. $\mathcal{F}_r$ is the unique probability measure satisfying all the probability axioms [9], such that for every $\alpha \in \Sigma^*$ and $C = \alpha\Sigma^\omega$, $\mathcal{F}_r(C)$ is the sum of $\phi(p)$, for all finite paths p of G starting from the state r such that $O(p) = \alpha$. For the HMC chain given in figure 1 and for $\alpha = (\{Q\}, \{Q\}, \{Q\})$, there are two paths p, i.e., s_0, s_0, s_0 and s_0, s_0, s_2, such that $O(p) = \alpha$ and hence $\mathcal{F}_{s_0}(C) = \frac{2}{9}$.

Let $D \in \mathcal{E}$ be such that $\mathcal{F}_r(D) \neq 0$. We let $\mathcal{F}_{r|D}$ denote the conditional probability function given D; formally, for any $C \in \mathcal{E}$, $\mathcal{F}_{r|D}(C) = \frac{\mathcal{F}_r(C\cap D)}{\mathcal{F}_r(D)}$. For any LTL formula g, we let $\mathcal{F}_{r|g}$ denote the conditional distribution $\mathcal{F}_{r|D}$ where D is the set of input sequences that satisfy g. For any $\alpha \in \Sigma^*$ and $C = \alpha\Sigma^\omega$, we let $\mathcal{F}_r(\alpha)$ denote the probability $\mathcal{F}_r(C)$ and $\mathcal{F}_{r|\alpha}$ denote the conditional probability function $\mathcal{F}_{r|C}$. For a set $C \subseteq \Sigma^*$, we let $\mathcal{F}_r(C)$ denote $\mathcal{F}_r(C\Sigma^\omega)$.

Example 1: Consider the HMC S_1 given in figure 1. Here the set of atomic propositions $\mathcal{P} = \{P, Q\}$. It should be easy to see that $\mathcal{F}_{s_0}(\Diamond P) = \frac{1}{2}$.

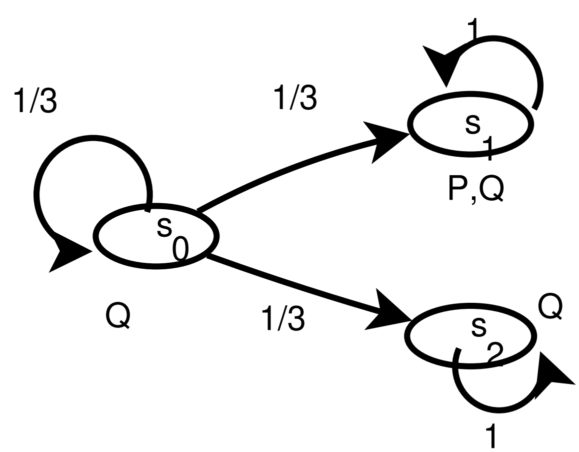

Fig. 1. System S_1

Deterministic Monitors. A monitor $M : \Sigma^* \rightarrow \{0,1\}$ is a computable function with the property that, for any $\alpha \in \Sigma^*$, if $M(\alpha) = 0$ then $M(\alpha\beta) = 0$ for every $\beta \in \Sigma^*$. For an $\alpha \in \Sigma^*$, we say that M rejects α, if $M(\alpha) = 0$, otherwise we say M accepts α. Thus if M rejects α then it rejects all its extensions. For an infinite sequence $\sigma \in \Sigma^\omega$, we say that M rejects σ iff there exists a prefix α of σ that is rejected by M; we say M accepts σ if it does not reject it. Let $L(M)$ denote the set of infinite sequences accepted by M. It is not difficult to see that $L(M)$ is a safety property and is measurable. The *acceptance accuracy* of M for $\mathcal{A}$ with respect to the HMC H is defined to be the probability $\mathcal{F}_{r_0|L(\mathcal{A})}(L(M))$ where r_0 is the initial state of H. Intuitively, it is the conditional probability that a sequence generated by the system is accepted by M, given that it is in $L(\mathcal{A})$. Roughly speaking, it is the fraction of the sequences in $L(\mathcal{A})$, generated from r_0, that are accepted by M. Let C, D be the complements of $L(\mathcal{A})$ and $L(M)$ respectively, i.e., $C = \Sigma^\omega - L(\mathcal{A})$ and $D = \Sigma^\omega - L(M)$. Then the *rejection accuracy* of M for $\mathcal{A}$ (also for $L(\mathcal{A})$) with respect to H is defined to be the probability $\mathcal{F}_{r_0|C}(D)$. This is the probability that a sequence generated by the system is rejected by M, given that it is not in $L(\mathcal{A})$. It is easy to see that this rejection accuracy is equal to $(1 - \mathit{acceptance\ accuracy\ of\ M\ for\ C\ with\ respect\ to\ H})$.

We say that M is a *conservative monitor* for a language $L' \subseteq \Sigma^\omega$, if $L(M) \subseteq L'$, i.e., it rejects every sequence not in L'. We say that M is a conservative monitor for an automaton $\mathcal{A}$ (resp., for a LTL formula ϕ) if it is a conservative monitor for $L(\mathcal{A})$ (resp., for C where C is the set of sequences that satisfy ϕ). Note that the rejection accuracy of a conservative monitor is 1.

Example 2: Consider the following conservative monitor M_1 for the LTL formula $\Diamond P$. It accepts all finite sequences of length ≤ 2. It accepts a finite sequence of length greater than two only if it has a P in the first three symbols. Clearly, $L(M_1) = \{(\neg P)^i P\beta : i \leq 2, \beta \in \Sigma^\omega\}$. Now consider the system HMC of Example 1. The first input produced by S_1 from state s_0 is $\neg P$, the probability that either the second or the third symbol is a P is $\frac{4}{9}$. From this, it should be easy to see that $F_{s_0|\Diamond P}(L(M_1)) = \frac{8}{9}$. Hence the acceptance accuracy of M_1 for $\Diamond P$ with respect to the system S_1 is $\frac{8}{9}$.

Probabilistic Monitors. We also define probabilistic monitors. A *probabilistic monitor* $M : \Sigma^* \rightarrow [0,1]$ is a function that associates a probability $M(\alpha)$ with each $\alpha \in \Sigma^*$ such that for every $\alpha, \beta \in \Sigma^*$, $M(\alpha\beta) \leq M(\alpha)$. Intuitively, $M(\alpha)$ denotes the probability that α is accepted by M. We extend M to infinite sequences as follows. For any $\sigma \in \Sigma^\omega$, $M(\sigma) = \lim_{i\rightarrow\infty} M(\sigma[0,i])$. $M(\sigma)$ represents the probability of acceptance of σ by M. We say that M is a probabilistic monitor for a language L' if $M(\sigma) = 0$ for all $\sigma \in \Sigma^\omega - L'$. That is every sequence not in L' is rejected with probability 1. Although we defined monitors as functions, many times monitors are given by algorithms (deterministic or probabilistic) that take inputs and reject some input sequences. With each such algorithm there is an implicitly defined unique monitor function.

The definition of acceptance accuracy for probabilistic monitors is a little more involved. Let M be a probabilistic monitor for an automaton $\mathcal{A}$ with input alphabet Σ. Let $\mathcal{F}_{r_0|L(\mathcal{A})}$ be the conditional probability distribution function

on the set of input sequences generated by the system. For any $n > 0$, let $Y_n = \sum_{\alpha \in \Sigma^n} (\mathcal{F}_{r_0|L(\mathcal{A})}(\alpha) M(\alpha))$; note Σ^n is the set of all sequences of length n. Because of the monotonicity of the function M, it is easy to see that $Y_n \geq Y_{n+1}$ for all $n > 0$. We define the acceptance accuracy of M for $\mathcal{A}$ with respect to the given system to be $\lim_{n \to \infty} Y_n$. The rejection accuracy of M for $\mathcal{A}$ with respect to the HMC H, is defined to be the $\lim_{n \to \infty} Z_n$ where Z_n is obtained from the expression for Y_n by replacing $L(\mathcal{A})$ by its complement and $M(\alpha)$ by $1 - M(\alpha)$.
Example 3: Consider the following probabilistic monitor M_2 for $\Diamond P$. M_2 looks at the current symbol. If it is a P it accepts and it will accept all subsequent inputs. If the current symbol is $\neg P$ and it has not seen a P since the beginning, then it rejects with probability $\frac{1}{3}$ and accepts with probability $\frac{2}{3}$. Formally, $M_2((\neg P)^{n-1} P \beta) = (\frac{2}{3})^{n-1}$ for all $n \geq 1$ and for all $\beta \in \Sigma^*$. Now consider the HMC S_1 given in Example 1. It can be shown by simple probabilistic analysis that $Y_n = 2 \sum_1^{n-1} (\frac{2}{3})^i (\frac{1}{3})^i$. From this, we see that the accuracy of M_2 for $\Diamond P$ with respect to system S_1 , which is $\lim_{n \to \infty} Y_n$, is $\frac{4}{7}$. In general if the rejection probability at each step used is p, it can be shown that the acceptance accuracy of M_2 for $\Diamond P$ with respect to S_1 is $2 \frac{1-p}{2+p}$. Thus the acceptance accuracy can be increased arbitrarily close to 1 by decreasing p.

A useful function. Let $\mathbf{Z}_{\geq 2}$ denote the set of integers greater than or equal to two. Let $c \geq 2$ be an integer. Consider a monitor which reads an infinite sequence of inputs. After the i^{th} input, it rejects with probability $\frac{1}{c^i}$ and stops, otherwise it continues. The probability that it never rejects is given by the value $G(c)$ defined as follows. For any integer $c \geq 2$, $G(c) = \prod_{i=1}^{\infty} (1 - \frac{1}{c^i})$. More precisely, $G(c) = \lim_{n \to \infty} \prod_{i=1}^{n} (1 - \frac{1}{c^i})$. Observe that each term in the product increases with i and goes to 1 as $i \to \infty$. Lemma 1 gives bounds on $G(c)$.

Lemma 1. *For every* $c \geq 2$, $\exp\left(\frac{-(2 \cdot c^3 + 3 \cdot c^2 + c + 1)}{2 \cdot c^2 \cdot (c^2 - 1)}\right) \leq G(c) \leq (1 - \frac{1}{c})$.

Using the above bound, it can be shown that $G(2) > 0.26$, $G(3) > 0.55$, $G(4) > 0.68$, etc. From the above bound, we see that $\lim_{c \to \infty} G(c) = 1$.

3 Accurate Deterministic Monitors

In this section we give methods for designing highly accurate deterministic monitors that monitor the executions of a system, given as a HMC, against a property specified by a deterministic finite state Buchi automaton. Here we assume that we know the initial system state and the output sequence generated by the system, but we can not observe the system state.

Let $H = (G, O, r_0)$ be a HMC where $G = (S, R, \phi)$ is a finite Markov chain and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be the given deterministic finite state Buchi automaton. For any $q \in Q$, let $\mathcal{A}_q$ be the automaton obtained by changing the initial state of $\mathcal{A}$ to q. Let $(s, q) \in S \times Q$. We call each such pair as a product state. Note that, for any such (s, q), $\mathcal{F}_s(L(\mathcal{A}_q))$ is the probability that an infinite output sequence generated from the system state s is accepted by the automaton $\mathcal{A}$ when started in the state q. We say that (s, q) is an *accepting* product state if $\mathcal{F}_s(L(\mathcal{A}_q)) = 1$.

We say that it is a *rejecting* product state if $\mathcal{F}_s(L(\mathcal{A}_q)) = 0$. Later we show how the accepting and rejecting product states can be determined.

Our monitoring algorithm works as follows. As the monitor gets inputs from the system, it simulates the automaton $\mathcal{A}$ on the input sequence using the variable a_state. It also keeps a set s_states which is the set of states the system can be in. y If at any time all states in $s\_states \times \{a\_state\}$ are accepting product states then it accepts. If all these states are rejecting product states then it rejects. Otherwise, it continues. In addition, it also maintains a counter, denoted by *counter* variable, which is initialized to some value. In each iteration, *counter* is decremented as long as the automaton state is not an accepting state. If *counter* is zero before an accepting automaton state is reached then, it rejects. When ever an accepting automaton state is reached then this counter is reset to a new value using the function f. The algorithm for the monitor is given below. The variable x denotes the current input symbol and the variable i records the number of times the counter has been reset. Here r_0 is the initial state of the system.

$a\_state := q_0;\ s\_states := \{r_0\};$
$i := 1;\ counter := f(s\_states, i, a\_state);$
$x := O(r_0);$
Loop forever
  $a\_state := \delta(a\_state, x);$
  $x := get\_nextinput();$
  $s\_states := \{s' : (s, s') \in R,\ s \in s\_states,\ O(s') = x\};$
  If every state in $s\_states \times \{a\_state\}$ is an accepting product state
    accept();
  If every state in $s\_states \times \{a\_state\}$ is a rejecting product state
    reject();
  $counter := counter - 1;$
  If $counter = 0$ and $a\_state \notin F$ reject();
  If $a\_state \in F$
    $\{i := i + 1;\ counter := f(s\_states, i, a\_state)\}$

When ever the monitor rejects (or accepts) then it immediately stops; in this case, it is assumed that it rejects (or accepts) all future inputs.

It is easy to see that the monitor rejects any input sequence that is not in $L(\mathcal{A})$ since after certain point, a_state is never in F. The accuracy of the monitor is highly dependent on the function f used in resetting the counter. One possibility, as indicated in [7, 8], is to reset it to a constant k. In this case, it can be shown that the accuracy of the resulting monitor is going to be zero many times. The following theorem shows that by increasing the reset value of *counter* linearly with i, we can achieve a desired accuracy.

Theorem 1. *For any given rational y such that $0 \leq y < 1$, there exists a constant a such that if $f(X, i, q) = a \cdot i$, for every $X \subseteq S$ and $q \in Q$, then the* *accuracy of the above monitor is at least y. Further more such a constant a is computable in time polynomial in the sizes of $H, \mathcal{A}$.*

Proof Sketch: Let y be as given in the statement of the theorem and c be an integer such that $G(c) \geq y$ and $c \geq 2$. We show that there exists an efficiently computable constant a such that by setting $f(X, i, q) = a \cdot i$, the probability that an accepting state of the automaton is not reached with in $f(X, i, q)$ inputs is at most $\frac{1}{c^i}$; that is this probability decreases exponentially with i.

For an automaton state $q \in Q$ and $\alpha \in \Sigma^*$, we say that α is a *minimal acceptable* sequence for q if $|\alpha| > 0$, $\delta^*(q, \alpha) \in F$ and for every proper prefix α' of α, $\delta^*(q, \alpha') \notin F$; that is α is a minimal finite sequence accepted from the automaton state q. For an integer $k > 0$, for any $q \in Q$, let $MINSEQ(q, k)$ denote $\{\alpha : \alpha \text{ is a minimal acceptable sequence for } q \text{ and } |\alpha| \leq k\}$. It is easy to see that no sequence in $MINSEQ(q, k)$ is a proper prefix of another sequence in it.

We say that a product state (s, q) is *good* if $\mathcal{F}_s(L(\mathcal{A}_q)) > 0$. For any good product state (s, q), integer $k > 0$, define $h(s, q, k) = \mathcal{F}_{s|L(\mathcal{A}_q)}(MINSEQ(q, k))$. Intuitively, $h(s, q, k)$ is the conditional probability that an infinite output sequence $\beta \in L(\mathcal{A}_q)$ generated from s has a prefix in $MINSEQ(q, k)$; i.e., starting from q an accepting automaton state is reached with in k outputs. Note that for a good product state (s, q), $h(s, q, k)$ monotonically increases with k and goes to 1 in the limit.

Now, the theorem follows from the following two lemmas whose proof is given in the appendix. $\square$

Lemma 2. *1. There exists a constant a that is efficiently computable from $G, \mathcal{A}$ and c such that for every good product state (s, q) and every $i > 0$, $h(s, q, a \cdot i) \geq (1 - \frac{1}{c^i})$.*
2. The set of accepting states and rejecting product states can be computed in time linear in the sizes of H and $\mathcal{A}$.

Lemma 3. *For any X, q, if $f(X, i, q) = a \cdot i$ where a is as given in lemma 2, then the accuracy of the above monitor is at least y.*

For an interesting subclass of HMCs, called *fully visible* HMCs, we can obtain a much simpler monitoring algorithm. We say that the HMC $H = (G, O, r_0)$ where $G = (S, R, \phi)$, is *fully visible* if O is a one-one function, i.e. for any two distinct $s, s' \in S$, $O(s) \neq O(s')$. This means that one can uniquely determine the current state of the system by its output. In this case, the set *s_states* is always a singleton set. If at any time, s denotes the unique state in the set *s_states*, then from the property of Markov chains, it can be shown that eventually (s, a_state) is either an accepting or a rejecting product state. Thus for fully visible HMCs, we simplify the loop body of the monitoring algorithm by simply checking whether $s\_states \times \{a\_state\}$ contains either an accepting or a rejecting state; thus accepting or rejecting respectively.

Lemma 4. *For a fully visible HMC, the simplified algorithm is correct and its acceptance and rejection accuracies are both equal to one.*

Example 4: Now consider the example of a HMC given in Figure 2. In this example, initially the system is in state s which is non-critical region (labeled by the proposition N). From s, it may loop there or go to s' or go to state t which is the trying region where the user sends a request for the resource, labeled by T. From here, the system may loop, or go to state v denoting that the resource server crashed or go to state w where the resource is granted. Note both states t, v are labeled by T. Thus we can not tell whether the system is in state t or in v. In state s', it can loop or go to u' where it requests for the resource. In u', it may loop or go to w'. In state w' the resource is allocated. Note that the resource server does not crash when requested from u'. This is because, here, a more reliable server is employed. Now our monitoring algorithm can be used to monitor for the desired property $g = \Box(T \rightarrow \Diamond C)$. Using the approach given in the proof of lemma 2, we get $f(X, i, q)$ to be $a \cdot i$ where $a = \max\{2, \lceil\frac{\ln c}{\ln 3}\rceil\}$ and c is an integer such that $G(c) \geq y$.

The appendix gives a method for computing a value of a in theorem 1. We can modify the above monitor by maintaining a probability function ψ such that for each state $s \in s\_states$, $\psi(s)$ is the probability that the system is in state s given the sequence of outputs generated thus far. Using such ψ, we can obtain a smaller constant a that gives smaller reset counter value.

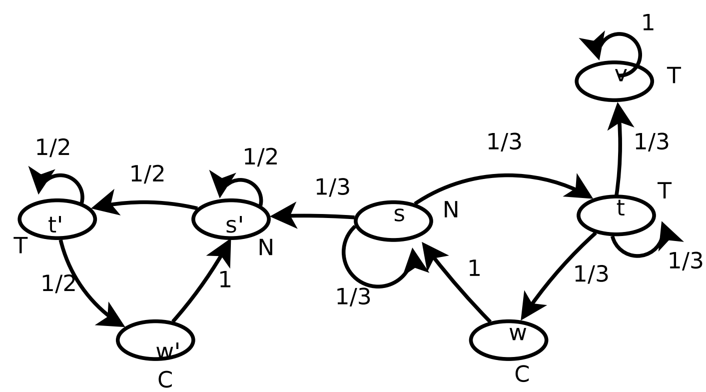

Fig. 2. Resource Acquisition

4 Probabilistic Monitors

In the algorithm given in the previous section, we need to know the HMC defining the system. In this section we consider probabilistic monitors that can be used to monitor any system for which the underlying HMC is not known.

Recall that if M is a probabilistic monitor for a language L' then for every $\sigma \notin L'$, $M(\sigma) = 0$. (Note that the rejection accuracy of a probabilistic monitor for a property with respect to HMC is one.) We say that M is a *strong monitor* for L' if M is a probabilistic monitor for L', and in addition, for every $\sigma \in L'$, $M(\sigma) > 0$.

Existence of Strong Monitors.

The following lemma shows that the class of languages that are strongly monitorable is exactly the class of languages whose complement is recognized by deterministic Buchi automata.

Lemma 5. *There is a strong monitor for a language L iff $\bar{L}$ is recognized by a possibly infinite state deterministic Buchi automaton.*

The above lemma has the following implications. Since the sets of sequences satisfying $\neg\Diamond P$, $\neg\Diamond\Box P$ are recognizable by deterministic Buchi automata, it follows that there are strong monitors for $\Diamond P$ and $\Diamond\Box P$. On the contrary, it is well known that the set of sequences satisfying $\neg\Box\Diamond P$ is not recognizable by any finite or infinite state deterministic Buchi automaton. Hence we have the following lemma.

Lemma 6. *There are no strong monitors for the LTL formulas $\Box\Diamond P$ and $\Box\Diamond P \rightarrow \Box\Diamond Q$.*

We have already given in section 2, Example 3, a probabilistic monitor M_3 for $\Diamond P$. This monitor acts as follows. In the beginning, after it sees each input symbol, if the symbol is not P then it rejects with probability p. This behavior continues until the first P is seen. Once it sees a P, it does not reject any more from then onwards. Suppose that the first P occurs as the i^{th} input symbol, then the probability that this sequence is accepted by M is $(1-p)^i$. Clearly, M is a strong monitor for $\Diamond P$.

Monitors with Desirable Properties.

The above monitor for $\Diamond P$ can be modified to obtain a monitor for $\Box\Diamond P$ which acts like M_3 until the first P and repeats this algorithm immediately after each subsequent P until the another P is seen. Unfortunately, the resulting monitor rejects any sequence that has infinite occurrences of $\neg P$ even if it satisfies $\Box\Diamond P$. This problem can be mitigated to a large extent by reducing the probability of rejection in geometric progression after each occurrence of a P.

Monitor for $\Box\Diamond P$. We construct a simple probabilistic monitor M_5 for the LTL formula $\Box\Diamond P$. Let $c \geq 2$ be any integer. M_5 rejects with probability $\frac{1}{c}$ after each input until the first P is seen. After the occurrence of the first P and before the occurrence of the second P, it rejects with probability $\frac{1}{c^2}$ after each input symbol. In general, in the period between the occurrence of the i^{th} P and the $(i+1)$st P, after each input symbol, it rejects with probability $\frac{1}{c^{i+1}}$. It is easy to see that M_5 rejects every input sequence that does not satisfy $\Box\Diamond P$ with probability one since P never occurs after some point, and after this point, M_5 rejects continuously with the same probability. Now consider any sequence σ that satisfies $\Box\Diamond P$. We say that such a sequence is *well behaving*, if the distance between any two successive Ps is bounded. For such a sequence σ, let $bound(\sigma)$

be the smallest value of m such that the first P occurs with in the first m input symbols and the distance between any successive Ps is bounded by m.

Lemma 7. *The monitor M_5 accepts every well behaving sequence with non-zero probability.*

It is to be noted that no deterministic monitor can accept all well behaving sequences that satisfy $\Box\Diamond P$, even those that use dynamically increasing counters. This is because for any well behaving sequence σ, the value of $bound(\sigma)$ is not known in advance. It is fairly straightforward to obtain a similar monitoring algorithm for any deterministic Buchi automaton.

Monitor for $\Box\Diamond P \rightarrow \Box\Diamond Q$. Now we give the description of probabilistic monitor M_7 for the LTL formula $\Box\Diamond P \rightarrow \Box\Diamond Q$. Let $c \geq 2$ be an integer. M_7 behaves as follows. Through out its operation, whenever it sees a symbol which is a symbol other than P it continues. Whenever it sees a P, M_7 rejects with probability $\frac{1}{c^{i+1}}$ where i is the number of occurrences of Q before the current input. This scheme can be implemented by keeping a variable p initialized to $\frac{1}{c}$. When ever a P occurs M_7 rejects with probability p. When ever a Q occurs p is set to $\frac{p}{c}$. It is not difficult to see that M_7 is a probabilistic monitor for $\Box\Diamond P \rightarrow \Box\Diamond Q$. A sequence not satisfying the formula, has infinitely many Ps and only finitely many Qs. If the number of Qs is m, then M_7 rejects infinitely often with the same probability which is $\frac{1}{c^{m+1}}$. Hence the input will be rejected with probability one. Any sequence that has only a finite number of Ps will be accepted with non-zero probability. Consider a sequence that satisfies the formula and that has infinitely many Ps. It also has infinitely many Qs. We say that such a sequence σ is well behaving if the number of Ps between successive Qs is bounded. Using the same reasoning as before, we can conclude that M_7 accepts every well behaving sequence with non-zero probability.

We can extend the above monitor for formulas of the form $\bigwedge_{i=1}^{m}(\Box\Diamond P_i \rightarrow \Box\Diamond Q_i)$. For each $i = 1, ..., m$, we build a monitor $M_7^{(i)}$ that monitors the formula $\Box\Diamond P_i \rightarrow \Box\Diamond Q_i$. They all coordinate so that if any of them rejects then all of them reject and stop. It is easy to see that the resulting monitor is a probabilistic monitor for the main formula. If a sequence satisfying the formula is well behaving with respect to each of the conjuncts of the formula then such a sequence will be accepted with non-zero probability. It is fairly straightforward to see how we can construct monitor for a deterministic Street automaton using the above approach by simulating the automaton on the input string and by monitoring the run of the automaton for the acceptance condition using the above monitor.

Hybrid Algorithms

The accuracy of the monitoring algorithms given earlier for $\Diamond P$, $\Box\Diamond P$ and $\Box\Diamond P \rightarrow \Box\Diamond Q$ can be improved substantially by combining with the deterministic counter based algorithms.

We illustrate this by constructing such a monitor for the formula $\Diamond P$. (The construction can be adopted easily for $\Box\Diamond P$ and $\Box\Diamond P \rightarrow \Box\Diamond Q$.) A deterministic monitor D for $\Diamond P$, uses counter k, and accepts an input if a P occurs in the first k inputs. Otherwise it rejects. We can also combine the above deterministic algorithm with the probabilistic monitor M_3 to get a hybrid algorithm H that works as follows: if a P occurs in the first k inputs then it never rejects; if a P does not occur in the first k symbols then after reading the k^{th} symbol, it rejects with probability p; it repeats this process for ever; that is, if a P occurs in the next k inputs then it does not reject any more, otherwise it rejects with probability p after the second k input symbols, and so on.

We first compare the above three simple algorithms with respect to their acceptance accuracies on uniformly distributed inputs. For $n > 0$, let H_n be a HMC that generates a sequence of outputs such that the first P occurs with equal probability (i.e., with probability $\frac{1}{n}$) in any of the n outputs starting with the second output. The appendix gives the formal definition of H_n.

Lemma 8. *Let $\rho_{prob,n}$ denote the ratio of the acceptance accuracy of the probabilistic algorithm to that of the deterministic algorithm for the property $\Diamond P$ with respect to the HMC H_n. Similarly let $\rho_{hybrid,n}$ denote the ratio of the acceptance accuracy of the hybrid algorithm to that of the deterministic algorithm. Then, in the limit as $n \to \infty$, $\rho_{hybrid,n} \to \frac{1}{p}$ and $\rho_{prob,n} \to \frac{1-p}{(k-1)p}$.*

The above lemma shows that for large values of n, the hybrid algorithm is more accurate than the deterministic algorithm for all values of p, while the probabilistic algorithm is more accurate than the deterministic algorithm for values of $p \leq \frac{1}{k}$.

Generating low probabilities. We see that both the probabilistic and hybrid algorithms can be made more accurate by decreasing p, the probability of rejection. In case of the hybrid algorithm, p can be made as low as $\frac{1}{2^k}$, where k is the counter value, by using a coin tossing scheme as follows. In each step, i.e., after reading each input symbol, the algorithm tosses an unbiased coin. If no P occurs in the first k inputs and all the previous k coin tosses turned "head" then it rejects, otherwise it continues; note that the probability, that all k coin tosses result in "head", is $\frac{1}{2^k}$. (If we take $k = 100$ then this probability is less than 10^{-30}). This process is repeated. In this case, the hybrid algorithm is more accurate than the deterministic one by a factor of 2^k. Note that here k serves a dual purpose— as a counter and also for generating low probabilities.

5 Experimental Results

We conducted some simple experiments to evaluate the acceptance accuracy of the different algorithms. We evaluated the algorithms for monitoring $\Diamond P$ and $\Box\Diamond P$. We generated strings of length 10^6. Each symbol is from $\{P, \neg P\}$. We used two different types of inputs. The first type of inputs is where the distance between successive P symbols is uniformly distributed between 1 and 1000. The second type of input is where this distance is normally distributed with different means (μ) and standard deviations (σ). We considered three different combinations of μ, σ; these are $(200, 20)$, $(300, 30)$ and $(500, 50)$. We evaluated the deterministic, hybrid and probabilistic methods. For deterministic and hybrid algorithms we used counters of length 200,300 and 500. For probabilistic algorithms we used probabilities of $\frac{1}{100}$, $\frac{1}{500}$ and $\frac{1}{1000}$.

For the property $\Box\Diamond P$, for the probabilistic method, we started with an initial probability of rejection and decreased it by a factor of $\frac{1}{2}$ in each successive period. For the hybrid algorithm, we used a biased coin where the probability of getting "heads" is given by this probability. Thus if this probability is $\frac{1}{100}$ then the probability of rejection in the hybrid algorithm is as low as $(\frac{1}{100})^{200}$ if a counter of length 200 is chosen.

For $\Box\Diamond P$, for uniform distribution, all algorithms excepting the hybrid algorithm rejected every input (all false rejections). This is because all of them have some probability of rejection before the next P and this gets amplified with successive Ps. For the hybrid algorithms,the probability of rejection is so low that none of the inputs strings is rejected, i.e., all are accepted.

For normal distribution, the deterministic algorithm with counter 200 rejected all inputs; with counter 300 accepted all inputs where $\mu = 200$ and $\sigma = 20$ but rejected all others and so on. The hybrid algorithm accepted all while the probabilistic algorithm rejected all.

This shows that the hybrid algorithm is best for high acceptance accuracy for $\Box\Diamond P$. We carried out similar experiments for $\Diamond P$ for uniform and normal distributions. As predicted, the hybrid algorithm fared well always and has the best acceptance accuracy. The deterministic algorithm fared well for normal distribution if $\mu + 3\sigma < k$.

6 Conclusions

In this paper, we have proposed algorithms for accurate monitoring of liveness properties. These algorithms combined with traditional algorithms for safety properties can be used for monitoring any ω-regular property. We presented a simple deterministic method that can be tuned to appropriate accuracy for a system given by Hidden Markov Chain. We proposed randomized algorithms when system behavior is not known. Among these algorithms the hybrid algorithms are extremely accurate.

As has been indicated in the introduction, our earlier works [7, 8, 15] gave different techniques for synthesizing conservative deterministic monitors for monitoring temporal properties. Deterministic liberal monitoring algorithms have been proposed in [2]. Runtime monitoring has also been used for interface synthesis in [10] where the interactions between the module and the interface are considered as a two person game.

In [5, 6] Larsen et.al. propose a method which, given a context specification and an overall specification, derive a temporal safety property characterizing the set of all implementations which, together with the given context, satisfy the overall specification. There has been much work in the literature on monitoring violations of safety properties in distributed systems. In these works, the safety property is typically explicitly specified by the user. A method for monitoring and checking quantitative and probabilistic properties of real-time systems has been given in [13]. These works take specifications in a probabilistic temporal logic (called CSL) and monitors for its satisfaction. The probabilities are deduced from the repeated occurrence of events in a computation.

None of the above works employ accuracy measures for monitors and none of them use randomization for monitoring liveness properties as we do. Our techniques are entirely new and have been experimentally validated.

Model checking probabilistic systems models as Markov chains was considered in the works of [16, 11, 12]. There they also construct a product of the Markov chain and the automata/tableaux associated with the LTL formula. While they concentrate on verification, we concentrate on the corresponding monitoring problem. Further more, we assume that during the computation the state of the system is not fully visible.

Acknowledgments: We are thankful to Amir Pnueli for his insight suggestions and comments, and to Yosef Lischitz for initial help on experiments.

References

1. B. Alpern and F. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, 1985.
2. M. Amorium and G. Rosu. Efficient monitoring of omega-languages. In *Proceedings of the International Conference on Computer Aided Verification (CAV'05), Volume 3576 of LNCS, pages 364-378, Springer*, 2005.
3. O. Cappe, E. Moulines, and T. Riden. *Inferencing in Hidden Markov Models*. Springer, 2005.
4. O. Kupferman and M. Vardi. Model checking safety properties. In *Proceedings of the International Conference on Computer Aided Verification*, 1999.
5. K. Larsen. Ideal specification formalisms = expressivity + compositionality + decidability + testability + ... In *Invited Lecture at CONCUR 1990, LNCS 458*, 1990.
6. K. Larsen. The expressive power of implicit specifications. In *ICALP 1991, LNCS 510*, 1991.
7. T. Margaria, A. Sistla, B. Steffen, and L. D. Zuck. Taming interface specifications. In *CONCUR2005*, pages 548–561, 2005.
8. T. Margaria, A. Sistla, B. Steffen, and L. D. Zuck. Taming interface specifications. In *www.cs.uic.edu/š̃istla*, 2005.
9. A. Papoulis and S. U. Pillai. *Probability, Random Variables and Stochastic Processes*. McGrawHill, NewYork, 2002.
10. A. Pnueli, A. Zaks, and L. D. Zuck. Monitoring interfaces for faults. In *Proceedings of the* 5^{th} *Workshop on Runtime Verification (RV'05)*, 2005. To appear in a special issue of ENTCS.
11. A. Pnueli and L. Zuck. Probabilistic verification by tableux. In *In Proceedings of First IEEE Symposium on Logic in Computer Science*, pages 322–331, 1986.
12. A. Pnueli and L. Zuck. Probabilistic verification. *Information and Computation*, 103:1–29, 1993.
13. U. Sammapun, I. Lee, and O. Sokolsky. Rt-mac:runtime monitoring and checking of quantitative and probabilistic properties. In *Proc. of 11th IEEE International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA 2005)*, pages 147–153, 2005.

14. A. P. Sistla. On characterization of safety and liveness properties in temporal logic. In *Proceedings of the ACM Symposium on Principle of Distributed Computing*, 1985.
15. A. P. Sistla, M. Zhou, and L. Zuck. Monitoring off-the-shelf components. In *Proceedings of the International Conference on Verification, Model Checking and Abstract Interpretation*, 2006.
16. M. Vardi. Automatic verification of probabilistic concurrent systems. In *26th annual Symposium on Foundations of Computer Science*, pages 327–338. IEEE Computer Society Press, 1985.

7 Appendix

In this section we give the proofs of the different lemmas.

Lemma 1: For every integer $c \geq 2$, $\exp\left(\frac{-(2\cdot c^3+3\cdot c^2+c+1)}{2\cdot c^2\cdot(c^2-1)}\right) \leq G(c) \leq (1-\frac{1}{c})$.

Proof Sketch: Throughout we assume c is an integer such that $c \geq 2$. We have $G(c) = \prod_{i=1}^{\infty}(1-\frac{1}{c^i})$. Let $F(c) = \frac{1}{G(c)}$. Taking logarithms on both sides, we have

$$\ln(F(c)) \;=\; \Sigma_{i\geq 1}\ln\left(\frac{1}{(1-\frac{1}{c^i})}\right) \tag{1}$$

Using the following identity for $0 \leq x < 1$ (obtained using Taylor's series expansion),

$$\ln\left(\frac{1}{(1-x)}\right) \;=\; \Sigma_{j\geq 1}\frac{x^j}{j} \tag{2}$$

we get

$$\ln(F(c)) \;=\; \Sigma_{i\geq 1}\Sigma_{j\geq 1}\frac{1}{j\cdot c^{i\cdot j}} \tag{3}$$

After interchanging the summations, we get

$$\ln(F(c)) \;=\; \Sigma_{j\geq 1}\frac{1}{j}\Sigma_{i\geq 1}\frac{1}{(c^j)^i} \tag{4}$$

After further simplification, we get

$$\ln(F(c)) \;=\; \Sigma_{j\geq 1}\frac{1}{j}\cdot\frac{1}{c^j-1} \tag{5}$$

The above sum can be written as

$$\ln(F(c)) \;=\; \frac{1}{c-1}+\frac{1}{2\cdot(c^2-1)}+\Sigma_{j\geq 3}\frac{1}{j}\cdot\frac{1}{c^j-1} \tag{6}$$

Now using the fact that, for $j \geq 3$, $j\cdot(c^j-1) \geq 2\cdot c^j$, we get

$$\ln(F(c)) \;\leq\; \frac{1}{c-1}+\frac{1}{2\cdot(c^2-1)}+\frac{1}{2}\Sigma_{j\geq 3}\frac{1}{c^j} \tag{7}$$

Substituting the equality $\Sigma_{j\geq 3}\frac{1}{c^j} = \frac{1}{(c^3-c^2)}$, and simplifying, we get

$$\ln(F(c)) \;\leq\; \frac{(2\cdot c^3+3\cdot c^2+c+1)}{2\cdot c^2\cdot(c^2-1)} \tag{8}$$

We get the desired lower bound of the lemma by taking exponentiation and by observing that $G(c) = \frac{1}{F(c)}$. The upper bound for $G(c)$ follows from the definition of $G(c)$.

Lemma 2

1. There exists a constant a that is efficiently computable from $G, \mathcal{A}$ and c such that for every good product state (s,q) and every $i > 0$, $h(s,q,a\cdot i) \geq (1-\frac{1}{c^i})$.

2. The set of accepting states and rejecting product states can be computed in time linear in the sizes of H and $\mathcal{A}$.

Proof Sketch: First we prove the second part of the lemma. First we recall some properties [9] of finite state Markov chains. Let $G' = (S', R', \phi')$ be such a Markov chain. The Strongly Connected Component (SCC) of G' is a strongly connected component of the directed graph (S', R'), i.e., a maximal subset C of states such that there is path connecting every pair of states in C that passes only through states in C. A SCC is called a *terminal* SCC if none of the nodes in it has an edge connecting it to a node out side of C. It is well known[9] that when the system starts executing, with probability one, it eventually reaches some terminal SCC C of G' and visits every state in C infinitely often.

Now, to prove the lemma we construct a product of the HMC H and the automaton $\mathcal{A}$. Recall that $H = (G, O, r_0)$ is the HMC where $G = (S, R, \phi)$ is a finite Markov chain and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is the given deterministic Buchi automaton. We define a Markov chain, $G' = (S', R', \phi')$, obtained by taking the product of H and the automaton $\mathcal{A}$. Roughly speaking, G' simulates the automaton $\mathcal{A}$ on H. Formally, $S' = S \times Q$, $R' = \{((s, q), (s', q')) : (s, s') \in R,\ \delta(q, O(s)) = \{q'\}\}$, and $\phi'((s, q), (s', q')) = \phi(s, s')$. It is not difficult to see that, for any state $u \in S'$, the sum of $\phi((u, u'))$ for all u' such that $(u, u') \in R'$ is one. Hence G' is a Markov chain. We say that a SCC C is an *accepting* SCC if it is a terminal SCC and there exists a state $(s, q) \in C$ such that $q \in F$. Consider a product state $u \in S'$. From the following claim, we see that the problems of checking if u is a good product state, if it is an accepting product state or rejecting product states are decidable in time linear in the size of G' and hence linear in the products of the sizes of H and $\mathcal{A}$. The claim is proved easily.
Claim:

- u is a good product state iff some accepting SCC is reachable from u.
- u is an *accepting* product state iff every terminal SCC reachable from u is an accepting SCC.
- u is a *rejecting* state iff every terminal SCC reachable from u is non-accepting.
  □

In order to prove the first part of the lemma we define another Markov chain $G'' = (S'', R'', \phi'')$ as follows. S'' is the set of all good product states; $R'' = \{(u, v) : u, v \in S'', (u, v) \in R'\}$, i.e., R'' is the restriction of R' to states in S'';for $(u, v) \in R''$ where $u = (s, q)$, $v = (s', q')$, $\phi''(u, v) = \frac{\phi'(u,v) \times \mathcal{F}_{s'}(L(\mathcal{A}_{q'}))}{\mathcal{F}_s(L(\mathcal{A}_q))}$. A *final* product state is a state of the form (s, q) where $q \in F$. It is not difficult to show that G'' is a Markov chain. It is easy to see that, for each $s \in S$, $q \in Q$, $\mathcal{F}_s(L(\mathcal{A}_q))$ is exactly the probability that an accepting SCC is reached in G' from the state (s, q). Using standard results from Markov chains [9], it is easy to see that $\mathcal{F}_s(L(\mathcal{A}_q))$, for each $s \in S$ and $q \in Q$, can be computed in time polynomial in the size of G' (actually in time cubic in the cardinality of S'). From this, we see that G'' can be computed in time polynomial in the size of G' and hence in the sizes of G and $\mathcal{A}$.

For any $u \in S''$, it can be shown that the the probability that an arbitrary infinite path starting from u visits a final product state infinitely often is one. For any $u \in S''$, define l_u to be the length of the shortest path in G'' from u to a final product state such that the path contains at least one transition; also, let p_u be the product of the probabilities of all the transitions on this path. Now it should be easy to see that the probability that an arbitrary finite path from u of length at least l_u contains a final product state, is at least p_u. Let $l = \max\{l_u : u \in S''\}$ and $p = \min\{p_u : u \in S''\}$. Note that $p > 0$ and l is bounded. It should be easy to see that the probability that any finite path of length l in G'' contains at least one final product state is at least p. Equivalently, the probability that any such path does not contain a final product state is at most $(1-p)$. Now consider any path of length $m \cdot l$ from a node $u = (s, q)$ for some $m > 0$. The probability that this path does not contain a final product state is at most $(1-p)^m$.

It can easily be shown that, for any $u = (s, q)$ in S'' and for any $k > 0$, $h(s, q, k)$ is exactly the probability that a path of length k from node u in G'' contains a final node. From this we see that $h(s, q, l \cdot m) \geq 1 - (1-p)^m$. Now, choose an m such that $1-(1-p)^m \geq 1 - \frac{1}{c^i}$, that is $(1-p)^m \leq \frac{1}{c^i}$. Let $d = \frac{1}{1-p}$. If $m \geq \lceil \frac{\ln c}{\ln d} \rceil \cdot i$ then $h(s, q, l \cdot m) \geq 1 - \frac{1}{c^i}$. Now, let $a = l \cdot \lceil \frac{\ln c}{\ln d} \rceil$. ¿From the above analysis we see that, for any $i > 0$, $h(s, q, a \cdot i) \geq 1 - \frac{1}{c^i}$. The above value of a is the required constant and it can be efficiently computed since G'', and hence p, can be computed in time polynomial in the sum of the sizes of G and $\mathcal{A}$. □

Lemma 3: For any X, q, if $f(X, i, q) = a \cdot i$ where a is as given in the lemma 2, then the accuracy of the above monitor is at least y.

Proof Sketch: Consider any $\alpha \in L(\mathcal{A})$ which is generated from the initial system state r_0. The probability that this sequence is not rejected by the monitor before the first accepting automaton state is reached is $\geq (1 - \frac{1}{c})$, i.e., not rejected when $i = 1$. The probability that it is not rejected when the program variable $i \leq 2$ is $\geq (1 - \frac{1}{c})(1 - \frac{1}{c^2})$. In general, the probability that it is not rejected when $i \leq k$ is $\geq (1 - \frac{1}{c})...(1 - \frac{1}{c^k})$. Hence the probability that α is never rejected is $\Pi_{k \geq 1}(1 - \frac{1}{c^k})$ which is $G(c)$. The lemma follows from the fact that $G(c) \geq y$. □

Lemma 4: For a fully visible HMC, the simplified algorithm is correct and its acceptance and rejection accuracies are both equal to one.

Proof Sketch: Consider the simplified algorithm given at the end of section 3. Now consider the product state (s, a_state) where s is the unique state in s_states. Consider the sequence of these product states after each input. It is easy to see that this is a path in the Markov chain G'. Hence eventually it will be in terminal SCC with probability. If this terminal SCC is accepting then the product state will be accepting. If it is rejecting then the product state will be rejecting. Since every SCC is accepting or rejecting, eventually the product state will be either accepting or rejecting. The correctness follows from this. □

Lemma 5: There is a strong monitor for a language L iff $\bar{L}$ is recognized by a deterministic Buchi automaton.

Proof: First we prove that if $\bar{L}$ is recognized by some deterministic Buchi automaton then there is a strong monitor for L. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be the automaton that accepts $\bar{L}$. The monitor runs the automaton $\mathcal{A}$ on the input string and rejects with probability $\frac{1}{2}$ when ever an accepting automaton state is reached. Formally we define the monitor function M as follows. For any $\alpha = \alpha_0, ... \in \Sigma^*$ of length m, let $q_0, ..., q_m$ be the finite run of $\mathcal{A}$ on α, i.e., q_0 is the initial state, $q_{i+1} = \delta(q_i, \alpha_i)$ for $0 \leq i < m$; let $f(\alpha)$ denote the number of distinct values of i, $0 \leq i \leq m$ such that $q_i \in F$. We define $M(\alpha) = (\frac{1}{2})^{f(\alpha)}$. It should be easy to see that for any infinite α accepted by $\mathcal{A}$,i.e., for any $\alpha \in \bar{L}$, $M(\alpha) = 0$ and for any infinite α rejected by $\mathcal{A}$, i.e., $\alpha \in L$, $M(\alpha) > 0$.

Now, we prove the lemma in the other direction. Assume that there is a strong monitor M for L. Now, we construct a Buchi automaton $\mathcal{A}$ that accepts $\bar{L}$. The states of $\mathcal{A}$ are elements of Σ^*. Formally, $\mathcal{A} = (\Sigma^*, \Sigma, \delta, \epsilon, F)$ where $\delta(\alpha, a) = \alpha a$, ϵ is the empty string and F is the set of αa, where $\alpha \in \Sigma^*$ and $a \in \Sigma$, such that, for some $i > 0$, $M(\alpha) \geq (\frac{1}{2})^i$ and $M(\alpha a) < (\frac{1}{2})^i$. Since M is a monotonically non-increasing function, it is easily seen that an infinite sequence β is accepted by $\mathcal{A}$ iff for each $i > 0$, there exists a prefix $\beta^[0, i]$ of β such that $M(\beta[0, i]) < (\frac{1}{2})^i$; thus β is accepted by $\mathcal{A}$ iff $M(\beta) = 0$. $\square$

Lemma 8: In the limit as $n \to \infty$, $\rho_{hybrid,n} \to \frac{1}{p}$ and $\rho_{prob,n} \to \frac{1-p}{(k-1)p}$.

Proof Sketch: First we define the HMC H_n. $H_n = (G_n, O_n, s_0)$ where $G_n = (S_n, R_n, \phi_n)$ and S_n, R_n, ϕ_n, O_n are as given below. S_n has $n+1$ nodes $s_0, ..., s_n$ connected in a chain, and in addition from each s_i there is an edge to s_n including the self loop in s_n. The node s_n is the only node that outputs P. The probability of the transition (s_i, s_n) is $\frac{i+1}{n}$ and that of the transition (s_i, s_{i+1}) is $1 - \frac{i+1}{n}$. The formal definition is given below. $S_n = \{s_i : 0 \leq i \leq n\}$; $R_n = \{(s_i, s_{i+1}) : 0 \leq i < n-1\} \cup \{(s_i, s_n) : 0 \leq i \leq n\}$; $O(s_n) = \{P\}$, and for $0 \leq i < n$, $O(s_i) = \emptyset$; for $0 \leq i < n-1$ $\phi_n((s_i, s_{i+1})) = 1 - \frac{i+1}{n}$, for $0 \leq i \leq n-1$, $\phi_n((s_i, s_n)) = \frac{i+1}{n}$ and $\phi_n((s_n, s_n)) = 1$. It is not difficult to see that among the sequences of outputs generated by H_n, the first P occurs with equal probability in the n outputs starting with the second output.

Recall that $\rho_{prob,n}$ denotes the ratio of the accuracy of the probabilistic algorithm to that of the deterministic algorithm for the property $\Diamond P$ with respect to the HMC H_n. Similarly $\rho_{hybrid,n}$ denotes the ratio of the accuracy of the hybrid algorithm to that of the deterministic algorithm.

Observe that the accuracy of the deterministic algorithm for the property $\Diamond P$ with respect to H_n is $\frac{k-1}{n}$. The accuracy of the probabilistic algorithm is easily seen to be equal to $\sum_{1 \leq i \leq n} \frac{(1-p)^i}{n}$ which equals to $\frac{(1-p)(1-(1-p)^n)}{np}$. Now it should be easy to see that $\rho_{prob,n}$ is $\frac{(1-p)(1-(1-p)^n)}{(k-1)p}$. It should now be easy to see that the limit as $n \to \infty$, $\rho_{prob,n} \to \frac{1-p}{(k-1)p}$.

The accuracy of the hybrid algorithm is equal to $\frac{k-1}{n} + \sum_{1 \leq i \leq d-1} (\frac{k(1-p)^i}{n}) + \frac{n \mod k}{n}(1-p)^d$ where $d = \lfloor \frac{n}{k} \rfloor$. After simplification, we get the accuracy to be $\frac{k}{n} \frac{1-(1-p)^d}{p} + \frac{n \mod k}{n}(1-p)^d - \frac{1}{n}$. ¿From this, it is easy to see that in the limit as $n \to \infty$, it is the case that $d \to \infty$ and $\rho_{hybrid,n} \to \frac{1}{p}$. $\square$

Lemma 7: The monitor M_5 accepts every well behaving sequence with non-zero probability.

Proof: Let σ be a well behaving sequence and $m = bound(\sigma)$. For each $i > 0$, let α_i be the prefix of σ up to the i^{th} P in σ. The probability that α_1 is accepted by M_5 is at least $(1 - \frac{1}{c})^m$. By a simple induction on i, it can be shown that, for all $i > 0$, the probability that α_i is accepted by M_5 is at least u_i where $u_i = \prod_{j=1}^{i}(1 - \frac{1}{c^j})^m$. Clearly, for each $i > 0$, $u_{i+1} < u_i$. Let $u = \lim_{i\to\infty} u_i$. The probability that M_5 accepts σ is at least u. It is not difficult to see that $u = (G(c))^m$ where $G(c)$ is the function defined at the end of section 2. We see that $u > 0$ since $G(c) > 0$. $\square$