

Monitoring Temporal Properties of Systems

A. Prasad Sistla

University of Illinois at Chicago

earlier part joint work with Min Zhou and Lenore Zuck, later part with
Abhigna Srinivas

Outline of the talk

- Motivation

Outline of the talk

- Motivation
- Extracting Safety Properties

Outline of the talk

- Motivation
- Extracting Safety Properties
- Monitoring Stochastic Systems

Outline of the talk

- Motivation
- Extracting Safety Properties
- Monitoring Stochastic Systems
- Probabilistic Algorithms

Motivation

Given:

- An *off-the-shelf component* C with specifications Φ_C ;
- A *goal* (or user) specifications Φ_G

How can C be used to guarantee Φ_G ?

Motivation

Given:

- An *off-the-shelf component* C with specifications Φ_C ;
- A *goal* (or user) specifications Φ_G

How can C be used to guarantee Φ_G ?

- If $(\Phi_C \rightarrow \Phi_G)$ then the problem is trivial
- But C is often under-specified (or simply mismatches the goal)

Motivation

Given:

- An *off-the-shelf component* C with specifications Φ_C ;
- A *goal* (or user) specifications Φ_G

How can C be used to guarantee Φ_G ?

- If $(\Phi_C \rightarrow \Phi_G)$ then the problem is trivial
- But C is often under-specified (or simply mismatches the goal)

Solution

- Construct a *monitor* M such that $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$

Solution

- Construct a *monitor* M such that $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$
- I.e., M *monitors* the executions of C for *violations* of Φ_M

Solution

- Construct a *monitor* M such that $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$
- I.e., M *monitors* the executions of C for *violations* of Φ_M
- Requirements from M to make this feasible:

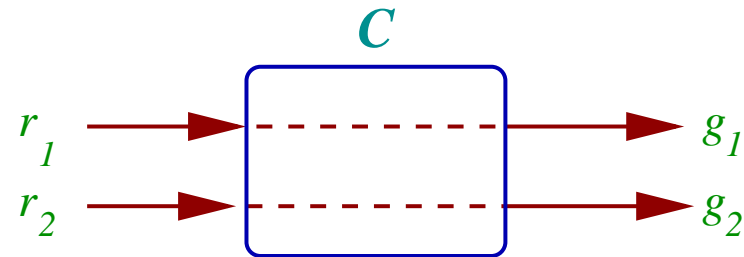
Solution

- Construct a *monitor* M such that $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$
- I.e., M *monitors* the executions of C for *violations* of Φ_M
- Requirements from M to make this feasible:
 - Φ_M is *safety*

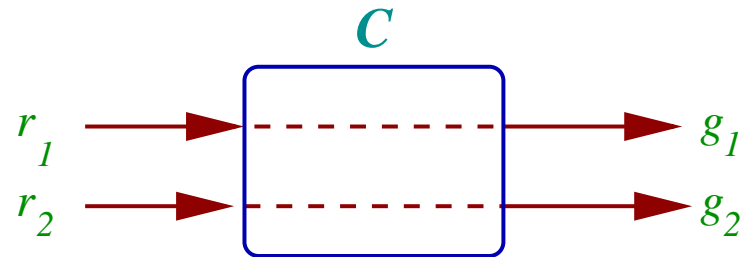
Solution

- Construct a *monitor* M such that $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$
- I.e., M *monitors* the executions of C for *violations* of Φ_M
- Requirements from M to make this feasible:
 - Φ_M is *safety*
 - $(\Phi_C \wedge \Phi_M) \rightarrow \Phi_G$, i.e., $\Phi_M \rightarrow (\neg\Phi_C \vee \Phi_G)$

Example: Permission Manager



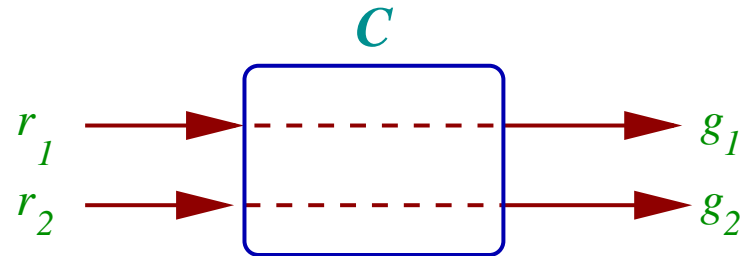
Example: Permission Manager



● Φ_G : priority for r_1

$$\square(r_1 \rightarrow (\neg g_2 \text{ until } g_1))$$

Example: Permission Manager



- Φ_G : priority for r_1

$$\Box(r_1 \rightarrow (\neg g_2 \text{ until } g_1))$$

- Φ_M : $\Box(r_1 \rightarrow (\neg g_2 \text{ unless } g_1))$

A Simple Solution

Given a non-det. Buchi automaton (NBA) $\mathcal{A} = (Q, \delta, F, q_0)$ and a positive integer k , construct a NBA \mathcal{B}_k such that:

- $L(\mathcal{B}_k)$ is a safety
- $L(\mathcal{B}_k) \subseteq L(\mathcal{A})$
- $L(\mathcal{B}_k) \subseteq L(\mathcal{B}_{k+1})$

How to construct \mathcal{B}_k ??

A Simple Solution

Given a non-det. Buchi automaton (NBA) $\mathcal{A} = (Q, \delta, F, q_0)$ and a positive integer k , construct a NBA \mathcal{B}_k such that:

- $L(\mathcal{B}_k)$ is a safety
- $L(\mathcal{B}_k) \subseteq L(\mathcal{A})$
- $L(\mathcal{B}_k) \subseteq L(\mathcal{B}_{k+1})$

How to construct \mathcal{B}_k ??

- Compose \mathcal{A} with a modulo- k counter, initialized to $k - 1$;
- Transitions from *non-accepting* states *decrement* the counter
- For transitions from *accepting* states *replenish* the counter

Generalization

Problem with \mathcal{B}_k : For monitoring $\square\diamond P$, none of them accept $P, \neg P, P, \neg P, \neg P, P, \dots$

Intuition: Let the replenished value of counter depend on the history!

Bounded Automata.

- a *state* is of the form (r, q, i) where
 - r – a “history” variable
 - $q \in Q_A$
 - i – counter, possibly ∞ (denoting rejection)
- Transitions from non-accepting states decrease the counter or set it to ∞ (where it remains)

Soundness and Completeness

Let \mathcal{A} be a NBA.

- For every bounded automaton \mathcal{N} over \mathcal{A} , $L(\mathcal{N})$ is a safety property that is in $L(\mathcal{A})$
- If \mathcal{A} is *deterministic*: every safety subset of $L(\mathcal{A})$ is accepted by some bounded automaton over \mathcal{A}

Monitoring Stochastic Systems

A *Hidden Markov Chain (HMC)* is a pair (G, O) where

- $G = (S, R, \phi)$ is a Markov chain;

Monitoring Stochastic Systems

A *Hidden Markov Chain (HMC)* is a pair (G, O) where

- $G = (S, R, \phi)$ is a Markov chain;
- $O : S \rightarrow \Sigma$ is an output function

Monitoring Stochastic Systems

A *Hidden Markov Chain (HMC)* is a pair (G, O) where

- $G = (S, R, \phi)$ is a Markov chain;
- $O : S \rightarrow \Sigma$ is an output function
- $\Sigma = 2^{\mathcal{P}}$, \mathcal{P} set of atomic propositions

Monitoring Stochastic Systems

A *Hidden Markov Chain (HMC)* is a pair (G, O) where

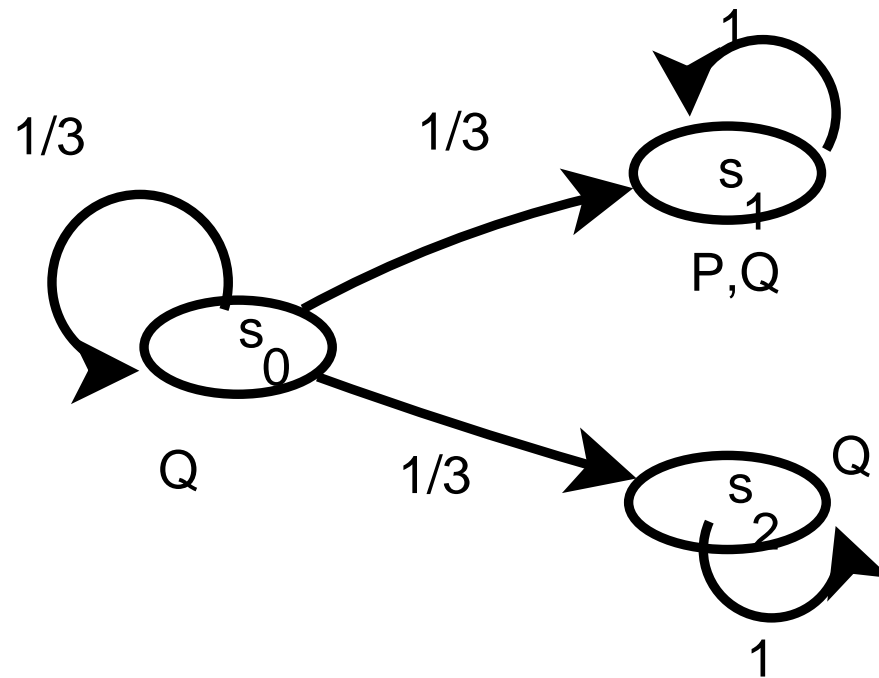
- $G = (S, R, \phi)$ is a Markov chain;
- $O : S \rightarrow \Sigma$ is an output function
- $\Sigma = 2^{\mathcal{P}}$, \mathcal{P} set of atomic propositions
- Define \mathcal{E} — the class of measurable subsets of Σ^ω — as the smallest set so that
 - For every $\alpha \in \Sigma^*$, $\alpha\Sigma^\omega \in \mathcal{E}$.
 - Closed under complementation and countable union.

Monitoring Stochastic Systems

A *Hidden Markov Chain (HMC)* is a pair (G, O) where

- $G = (S, R, \phi)$ is a Markov chain;
- $O : S \rightarrow \Sigma$ is an output function
- $\Sigma = 2^{\mathcal{P}}$, \mathcal{P} set of atomic propositions
- Define \mathcal{E} — the class of measurable subsets of Σ^ω — as the smallest set so that
 - For every $\alpha \in \Sigma^*$, $\alpha\Sigma^\omega \in \mathcal{E}$.
 - Closed under complementation and countable union.

Example



For any state s , \mathcal{F}_s defines a probability measure on \mathcal{E} .

$$\mathcal{F}_{s_0}(\diamond P) = \frac{1}{2}.$$

Accuracy of a Monitor

- The system is given by a HMC \mathcal{H} which is known.

Accuracy of a Monitor

- The system is given by a HMC \mathcal{H} which is known.
- Outputs of \mathcal{H} are observable but not the state

Accuracy of a Monitor

- The system is given by a HMC \mathcal{H} which is known.
- Outputs of \mathcal{H} are observable but not the state
- Given a deterministic Buchi automaton \mathcal{A}

Accuracy of a Monitor

- The system is given by a HMC \mathcal{H} which is known.
- Outputs of \mathcal{H} are observable but not the state
- Given a deterministic Buchi automaton \mathcal{A}
- Construct a monitor \mathcal{M} so that
 - $L(\mathcal{M}) \subseteq L(\mathcal{A})$.
 - $L(\mathcal{M})$ is a safety property.

Accuracy of a Monitor

- The system is given by a HMC \mathcal{H} which is known.
- Outputs of \mathcal{H} are observable but not the state
- Given a deterministic Buchi automaton \mathcal{A}
- Construct a monitor \mathcal{M} so that
 - $L(\mathcal{M}) \subseteq L(\mathcal{A})$.
 - $L(\mathcal{M})$ is a safety property.
- **Accuracy** of \mathcal{M} is the conditional probability $\mathcal{F}_{s_0}(L(\mathcal{M}) \mid L(\mathcal{A}))$ — s_0 initial system state.

Monitoring Algorithm

- Preprocessing

Monitoring Algorithm

- Preprocessing
 - Compute Markov chain G' — the product of G and \mathcal{A} .

Monitoring Algorithm

- Preprocessing
 - Compute Markov chain G' — the product of G and \mathcal{A} .
 - A state (s, q) in G' is **good** if $\mathcal{F}_s(L(\mathcal{A}_q)) = 1$ and **bad** if $\mathcal{F}_s(L(\mathcal{A}_q))$ is 0. \mathcal{A}_q same as \mathcal{A} with starting state q .

Monitoring Algorithm

- Preprocessing
 - Compute Markov chain G' — the product of G and \mathcal{A} .
 - A state (s, q) in G' is **good** if $\mathcal{F}_s(L(\mathcal{A}_q)) = 1$ and **bad** if $\mathcal{F}_s(L(\mathcal{A}_q))$ is 0. \mathcal{A}_q same as \mathcal{A} with starting state q .
 - Compute good and bad states of G' .

Monitoring Algorithm

- Preprocessing
 - Compute Markov chain G' — the product of G and \mathcal{A} .
 - A state (s, q) in G' is **good** if $\mathcal{F}_s(L(\mathcal{A}_q)) = 1$ and **bad** if $\mathcal{F}_s(L(\mathcal{A}_q))$ is 0. \mathcal{A}_q same as \mathcal{A} with starting state q .
 - Compute good and bad states of G' .
- Simulates \mathcal{A} on the sequence of system outputs .

Monitoring Algorithm Contd

- Maintains the following variables
 - X : possible system states, initialized to $\{s_0\}$.
 - q : the automaton state, initialized to q_0 .
 - i : denotes the number of times an accepting automaton state is reached. Initialized to 0.
 - *counter* : denotes the number of expected outputs before an accepting automaton state.

Monitoring Algorithm Contd

- Maintains the following variables
 - X : possible system states, initialized to $\{s_0\}$.
 - q : the automaton state, initialized to q_0 .
 - i : denotes the number of times an accepting automaton state is reached. Initialized to 0.
 - *counter* : denotes the number of expected outputs before an accepting automaton state.

Monitoring Algorithm Contd

- Maintains the following variables
 - X : possible system states, initialized to $\{s_0\}$.
 - q : the automaton state, initialized to q_0 .
 - i : denotes the number of times an accepting automaton state is reached. Initialized to 0.
 - *counter* : denotes the number of expected outputs before an accepting automaton state.

Monitoring Algorithm Contd

- Maintains the following variables
 - X : possible system states, initialized to $\{s_0\}$.
 - q : the automaton state, initialized to q_0 .
 - i : denotes the number of times an accepting automaton state is reached. Initialized to 0.
 - *counter* : denotes the number of expected outputs before an accepting automaton state.

Monitoring Algorithm Contd

- Maintains the following variables
 - X : possible system states, initialized to $\{s_0\}$.
 - q : the automaton state, initialized to q_0 .
 - i : denotes the number of times an accepting automaton state is reached. Initialized to 0.
 - *counter* : denotes the number of expected outputs before an accepting automaton state.

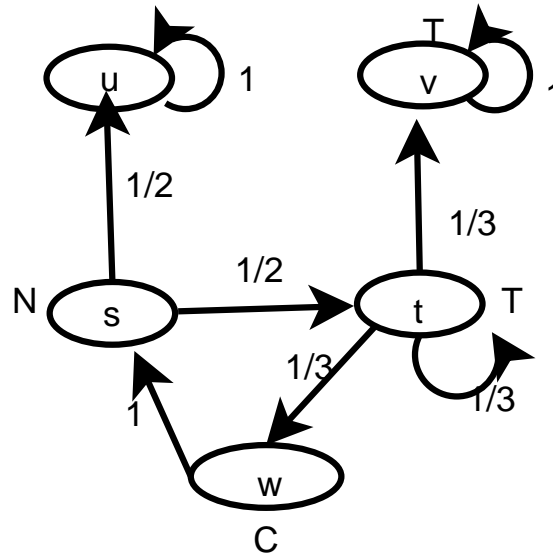
Monitoring Algorithm Contd

Loop forever

- Get next output from the system;
- Update q and X ;
- $counter := counter - 1$;
- If all states in $X \times \{q\}$ are good then accept;
- If all states in $X \times \{q\}$ are bad then reject;
- If $counter = 0$ and q is not an accepting state of \mathcal{A} then reject;
- If q is an accepting state of \mathcal{A} then
 $\{i := i + 1; counter := f(q, X, i)\}$

- **Theorem:** For any y , $0 \leq y \leq 1$, there exists a constant c such that if $f(q, X, i) = c \cdot i$ then the accuracy of the monitor is at least y .
- **Theorem:** If the HMC is **fully visible**, then the monitor has accuracy 1.

Example: Resource Acquisition



- s is the initial state.
- v denotes state when the server crashed.
- Property to be monitored: $\square(T \rightarrow \diamond C)$.
- Accuracy of 0.9 can be achieved by choosing $K = 3$.

Probabilistic Monitors

- A *probabilistic monitor* M :
For $i = 1, \dots, \infty$
 - Get the i^{th} input from the system;
 - With some computed probability p_i , reject and stop

Probabilistic Monitors

- A *probabilistic monitor* M :
For $i = 1, \dots, \infty$
 - Get the i^{th} input from the system;
 - With some computed probability p_i , reject and stop
- For a finite sequence α of length k , $pr(\alpha)$ — probability that α is accepted— is $(1 - p_1) \cdot \dots \cdot (1 - p_k)$.

Probabilistic Monitors

- A *probabilistic monitor* M :
For $i = 1, \dots, \infty$
 - Get the i^{th} input from the system;
 - With some computed probability p_i , reject and stop
- For a finite sequence α of length k , $pr(\alpha)$ — probability that α is accepted— is $(1 - p_1) \cdot \dots \cdot (1 - p_k)$.
- For $\sigma \in \Sigma^\omega$, $\text{Prob} \{ \sigma \text{ is accepted} \} = \lim_{k \rightarrow \infty} pr(\sigma_k)$,
 σ_k — prefix of σ of length k .

Probabilistic Monitors

- A *probabilistic monitor* M :
For $i = 1, \dots, \infty$
 - Get the i^{th} input from the system;
 - With some computed probability p_i , reject and stop
- For a finite sequence α of length k , $pr(\alpha)$ — probability that α is accepted— is $(1 - p_1) \cdot \dots \cdot (1 - p_k)$.
- For $\sigma \in \Sigma^\omega$, $\text{Prob} \{ \sigma \text{ is accepted} \} = \lim_{k \rightarrow \infty} pr(\sigma_k)$,
 σ_k — prefix of σ of length k .
- M is a *probabilistic monitor* for $L \subseteq \Sigma^\omega$, if for every $\sigma \notin L$, M rejects σ with probability 1.

Probabilistic Monitors

- A *probabilistic monitor* M :
For $i = 1, \dots, \infty$
 - Get the i^{th} input from the system;
 - With some computed probability p_i , reject and stop
- For a finite sequence α of length k , $pr(\alpha)$ — probability that α is accepted— is $(1 - p_1) \cdot \dots \cdot (1 - p_k)$.
- For $\sigma \in \Sigma^\omega$, $\text{Prob} \{ \sigma \text{ is accepted} \} = \lim_{k \rightarrow \infty} pr(\sigma_k)$,
 σ_k — prefix of σ of length k .
- M is a *probabilistic monitor* for $L \subseteq \Sigma^\omega$, if for every $\sigma \notin L$, M rejects σ with probability 1.

Existence of Monitors

- A **strong monitor** for L , is a probabilistic monitor that accepts every $\sigma \in L$ with non-zero probability.

Existence of Monitors

- A **strong monitor** for L , is a probabilistic monitor that accepts every $\sigma \in L$ with non-zero probability.
- **Strong Monitor for $\diamond P$** : Reject with a fixed probability p until the first P . Has **graceful degradation** property.

Existence of Monitors

- A **strong monitor** for L , is a probabilistic monitor that accepts every $\sigma \in L$ with non-zero probability.
- **Strong Monitor for $\diamond P$** : Reject with a fixed probability p until the first P . Has **graceful degradation** property.
- **Theorem**: There is a strong monitor for L iff \bar{L} is accepted by finite/infinite state deterministic Buchi automaton.

Existence of Monitors

- A **strong monitor** for L , is a probabilistic monitor that accepts every $\sigma \in L$ with non-zero probability.
- **Strong Monitor for $\diamond P$** : Reject with a fixed probability p until the first P . Has **graceful degradation** property.
- **Theorem**: There is a strong monitor for L iff \bar{L} is accepted by finite/infinite state deterministic Buchi automaton.
- There exist strong monitors for $\diamond P, \diamond \square P$.

Existence of Monitors

- A **strong monitor** for L , is a probabilistic monitor that accepts every $\sigma \in L$ with non-zero probability.
- **Strong Monitor for $\diamond P$** : Reject with a fixed probability p until the first P . Has **graceful degradation** property.
- **Theorem**: There is a strong monitor for L iff \bar{L} is accepted by finite/infinite state deterministic Buchi automaton.
- There exist strong monitors for $\diamond P, \diamond \square P$.
- There are no strong monitors for $\square \diamond P$ and $\square \diamond P \rightarrow \square \diamond Q$.

Monitors for $\square\diamond P$

- A probabilistic monitor M for $\square\diamond P$:
 - Until the first P , reject with probability $\frac{1}{2}$ after each input.
 - For each $i > 0$, from the i^{th} P until the next P , reject with probability $(\frac{1}{2})^i$ after each input.

Monitors for $\square\diamond P$

- A **probabilistic monitor M** for $\square\diamond P$:
 - Until the first P , reject with probability $\frac{1}{2}$ after each input.
 - For each $i > 0$, from the i^{th} P until the next P , reject with probability $(\frac{1}{2})^i$ after each input.
- **Theorem:** Any sequence, in which the distance between consecutive P s is bounded, is accepted with non-zero probability.

Monitors for $\square\diamond P$

- A **probabilistic monitor M** for $\square\diamond P$:
 - Until the first P , reject with probability $\frac{1}{2}$ after each input.
 - For each $i > 0$, from the i^{th} P until the next P , reject with probability $(\frac{1}{2})^i$ after each input.
- **Theorem**: Any sequence, in which the distance between consecutive P s is bounded, is accepted with non-zero probability.
- **Monitor for $\square\diamond P \rightarrow \square\diamond Q$** : After each P , reject with probability $\frac{1}{2^{i+1}}$; i is the number of Q s before this symbol.

Monitors for $\square\diamond P$

- A **probabilistic monitor M** for $\square\diamond P$:
 - Until the first P , reject with probability $\frac{1}{2}$ after each input.
 - For each $i > 0$, from the i^{th} P until the next P , reject with probability $(\frac{1}{2})^i$ after each input.
- **Theorem**: Any sequence, in which the distance between consecutive P s is bounded, is accepted with non-zero probability.
- **Monitor for $\square\diamond P \rightarrow \square\diamond Q$** : After each P , reject with probability $\frac{1}{2^{i+1}}$; i is the number of Q s before this symbol.
- can give prob. monitors for det. Streett/Buchi automata.

Hybrid Algorithms

- Combine counter based methods with probabilities.

Hybrid Algorithms

- Combine counter based methods with probabilities.
- Hybrid Algorithm for $\diamond P$:
 - After each input toss a fair coin.
 - After every k^{th} input: If no P in the last k inputs and all the last k coin tosses were “heads” then reject.

Hybrid Algorithms

- Combine counter based methods with probabilities.
- Hybrid Algorithm for $\diamond P$:
 - After each input toss a fair coin.
 - After every k^{th} input: If no P in the last k inputs and all the last k coin tosses were “heads” then reject.
- The k -counter serves dual purpose: as reusable timeout and also for generating low probability, i.e. $\frac{1}{2^k}$.

Hybrid Algorithms

- Combine counter based methods with probabilities.
- Hybrid Algorithm for $\diamond P$:
 - After each input toss a fair coin.
 - After every k^{th} input: If no P in the last k inputs and all the last k coin tosses were “heads” then reject.
- The k -counter serves dual purpose: as reusable timeout and also for generating low probability, i.e. $\frac{1}{2^k}$.
- Highly accurate. It can be more accurate by a factor of 2^k compared to deterministic counter based methods

Experimental Results

- Conducted simple experiments monitoring for $\diamond P$ and $\square\diamond P$.

Experimental Results

- Conducted simple experiments monitoring for $\diamond P$ and $\square\diamond P$.
- Considered strings that are very long (length $> 10^6$)

Experimental Results

- Conducted simple experiments monitoring for $\diamond P$ and $\square\diamond P$.
- Considered strings that are very long (length $> 10^6$)
- Generated strings in which the distance between consecutive P s is normally distributed and uniformly distributed.

Experimental Results

- Conducted simple experiments monitoring for $\diamond P$ and $\square\diamond P$.
- Considered strings that are very long (length $> 10^6$)
- Generated strings in which the distance between consecutive P s is normally distributed and uniformly distributed.
- Used k -counters with $k = 50, 100, \dots$

Experimental Results

- Conducted simple experiments monitoring for $\diamond P$ and $\square\diamond P$.
- Considered strings that are very long (length $> 10^6$)
- Generated strings in which the distance between consecutive P s is normally distributed and uniformly distributed.
- Used k -counters with $k = 50, 100, \dots$
- The Hybrid algorithms never rejected any corrected sequence.

Related Work

- Monitoring for safety properties done by many people [Si87], [KV99], etc.

Related Work

- Monitoring for safety properties done by many people [Si87], [KV99], etc.
- Recent work— Amorium and Rosu (CAV2005)— handle some liveness. Concentrate on evaluating efficiently atomic propositions in system states.

Related Work

- Monitoring for safety properties done by many people [Si87], [KV99], etc.
- Recent work— Amorium and Rosu (CAV2005)— handle some liveness. Concentrate on evaluating efficiently atomic propositions in system states.
- The paper [PZZ 200] uses game theoretic approach.

Conclusions

- Other cost measures for tuning deterministic algs for HMCs.

Conclusions

- Other cost measures for tuning deterministic algs for HMCs.
- How to monitor for complex systems? Use Assume/guarantee paradigms.

Conclusions

- Other cost measures for tuning deterministic algs for HMCs.
- How to monitor for complex systems? Use Assume/guarantee paradigms.
- Monitoring in a distributed environment!