Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State

A. Prasad Sistla, Miloš Žefran, and Yao Feng

University of Illinois at Chicago
{sistla,mzefran,yfeng9}@uic.edu

Abstract. Correct functioning of cyber-physical systems is of critical importance. This is more so in the case of safety critical systems such as in medical, automotive and many other applications. Since verification of correctness, in general, is infeasible and testing is not exhaustive, it is of critical importance to monitor such system during their operation and detect erroneous behaviors to be acted on. A distinguishing property of cyber-physical systems is that they are described by a mixture integer-valued and real-valued variables. As a result, approaches that assume countable number of states are not applicable for runtime monitoring of such systems. This paper proposes a formalism, called Extended Hidden Markov systems, for specifying behavior of systems with such hybrid state. Using measure theory, it exactly characterizes when such systems are monitorable with respect to a given property. It also presents monitoring algorithms and experimental results showing their effectiveness.

1 Introduction

As traditional control systems are replaced by ever more powerful networks of microprocessors, cyber-physical systems are becoming an integral part of modern society. Correct functioning of such systems is of paramount importance since a malfunction can lead to a serious injury or a loss of life. Monitoring such systems at run time and shutting them down safely, in case of malfunctioning, can provide a mechanism for safe operation of such systems.

The monitor observes the inputs and outputs of the component and checks whether the behavior of the system is consistent with the expected behavior. The fundamental advantage of monitors is that they are in principle easy to implement, and they are independent of the design procedures used to develop a component. While wrong assumptions might lead to a faulty design, the monitor is independent of design decisions and can therefore easily detect that the component is failing to perform its function.

In our earlier works [10, 29], we addressed the problem of monitoring a system, modeled as a Hidden Markov Chain (HMC) H, when the correctness specification is given by a deterministic Streett automaton $\mathcal{A}$ on the outputs generated by the system. In these works, we defined two measures, called *Acceptance Accuracy* (AA) and *Rejection Accuracy* (RA) that capture the effectiveness of the monitor. Here $(1 - AA)$ and $(1 - RA)$ give measures of false alarms and missed alarms, respectively. Monitoring algorithms for achieving arbitrary high values of accuracies were presented when H and $\mathcal{A}$ are finite state systems.

In a more recent paper [31], we considered the case of internal monitoring, i.e., when the property $\mathcal{A}$ to be monitored is specified on the states of the system and when both the system H and the automaton $\mathcal{A}$ may have countably infinite number of states. In this setting, we defined a notion of monitorability, which states that a system is monitorable with respect to a property if arbitrary high levels of accuracy, close to one, can be achieved. We proved a fundamental theorem, called *monitorability theorem*, that exactly characterizes when a system is monitorable with respect to a property. The paper also gave monitoring algorithms that achieve high accuracies for monitorable cases. We applied the algorithms to systems modeled by probabilistic hybrid automata by approximating its hybrid state space by a discrete state space through quantization.

In this paper, we consider Extended Hidden Markov Systems (EHMS), which are like HMCs, but in which the state space is a hybrid state space. In these systems, the system variables and the output variables are partitioned into discrete and continuous variables. We consider probability functions over such hybrid state spaces. A probability function can be considered as a set of sub-probability density functions on the continuous part of the state space where each density function is indexed by a discrete part of the state. An EHMS is given by a next state function and an output function that give the probability functions on the next state and outputs, given the current state. We extend the monitorability theorem to EHMSes. This extension is non-trivial, and relies heavily on measure theory, as we have to deal with continuous as well as discrete probability distributions. We present monitoring algorithms and experimental results showing the effectiveness of our approach for a more complex version of the example considered in [31].

In summary the main contributions of the paper are as follows: (1) an Extended Hidden Markov model for modeling Hybrid systems, without discretization; (2) exact characterizations of systems that are monitorable with respect to a property; (3) monitoring algorithms when the system and property automata are specified by probabilistic hybrid automata; and (4) experimental results showing the effectiveness of our approach.

The paper is organized as follows. Section 2 contains related work. Section 3 presents the models and the definitions of accuracy measures and monitorability. Section 4 gives exact characterizations of systems that are monitorable. Section 5 presents monitoring algorithms for systems specified using probabilistic Hybrid automata. Section 6 shows how a monitor can designed for a simple autonomous driving system and demonstrates the effectiveness of the approach. Section 7 contains concluding remarks.

2 Related Work

A wealth of literature is available for the modeling and control of hybrid systems and we refer the reader to the overview articles [1,5] and the books [20,33]. In these systems, safety requirements are described by a set of system states which are permissible, or equivalently, by a set of system states that are forbidden. A closely related problem is checking liveness properties, where in general we require that a set of states is visited infinitely often. Formally, safety and liveness properties can be described using temporal logic [16,18]. Safety and liveness verification thus becomes a verification problem for a hybrid automaton modeling the robotic system [12,24]. It was shown that except for the simplest hybrid automata this verification problem is undecidable [12].

A problem that has been extensively studied is monitoring and diagnosis of hybrid automata [2,3,9,13,17,19,23,34], where the aim is to detect when the automaton enters a fail state so that the system can appropriately react. Similar work has been done in the Artificial Intelligence community on failure detection and recovery from failures using Hidden Markov models [27]. In most cases, these works employ techniques that depend on the specific possible modes of failure. Furthermore, even if such methods are employed, one still needs to monitor the correct functioning of the overall system for correct functioning. None of the above works addresses this general problem of monitoring system behaviors against specifications given in an expressive formal system such as the hybrid automata. Furthermore, they do not address the problem of monitoring liveness properties.

An alternative to verification is to directly incorporate the safety requirements into the design process itself so that no verification step is necessary [16,18,21]. These approaches are not yet able to adequately address systems with complex continuous dynamics, nor can they deal with stochastic phenomena such as sensor and actuator failures.

There has been much work done in the literature on monitoring violations of safety properties in distributed systems, for example [4]. This work assumes that it can fully observe the system state and it instruments the program with commands to gather its state information and use it for monitoring. In contrast, we assume that the system is not directly observable. A method for monitoring and checking quantitative and probabilistic properties of real-time systems has been given in [11,28]. These works take specifications in a probabilistic temporal logic (called CSL) and monitor for its satisfaction. The probabilities are deduced from the repeated occurrence of events in a computation. The work presented in [25] considers monitoring interfaces for faults using game-theoretic framework. Run-time monitoring is used to verify that the interface has a winning strategy. *Conservative* run time monitors were proposed in [22, 30]. In this scheme, one identifies a safety property that implies the given property f (in general, f is the intersection/conjunction of a safety and a liveness property). None of these works is intended for monitoring of hybrid systems.

3 Definitions and Notation

Sequences. Let S be a 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 either a finite or an 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 $\{\alpha\beta\ : \beta \in C\}$.

Safety Properties. For any $\sigma \in S^\omega$, let prefixes(σ) denote the set of prefixes of σ and for any $C \subseteq S^\omega$, let prefixes$(C) = \cup_{\sigma\in C}(\text{prefixes}(\sigma))$. We say that $C \subseteq S^\omega$ is a *safety* property if the following condition holds: for any $\sigma \in S^\omega$, if prefixes$(\sigma) \subseteq$ prefixes(C) then $\sigma \in C$. For any $C \subseteq S^\omega$, let $closure(C)$ be the smallest safety property such that $C \subseteq closure(C)$.

Extended Hidden Markov systems. We assume that the reader is familiar with basic probability theory, random variables and Markov chains. We consider stochastic systems over discrete time with both discrete and continuous states. Let $\mathbf{R}, \mathbf{N}$ denote the set of real numbers and non-negative integers, respectively. Throughout the paper, we will be using integrals over measurable functions which are taken to be Lebesgue integrals (see [26] for definitions).

Let $\sigma = (\sigma_0, ..., \sigma_{n-1})$ be a vector in $\{0, 1\}^n$. For each such σ, we define the hybrid domain $S_\sigma = T_0 \times T_1 \times ... \times T_{n-1}$ where $T_i = \mathbf{N}$ or $T_i = \mathbf{R}$ depending on whether $\sigma_i = 0$ or $\sigma_i = 1$, respectively. We define a class $\mathcal{D}_\sigma$ of measurable subsets of S_σ as follows. Let I be the set of all values of i such that $\sigma_i = 0$ and J be the set of all values of j such that $\sigma_j = 1$. Let n_1, n_2 be the cardinalities of I, J respectively. Clearly $n_1 + n_2 = n$. For any $s \in S_\sigma$, let $s|I$ and $s|J$,respectively, be the projections of s on to the coordinates in I and J. That is, $s|I$, $s|J$ give the values of discrete and continuous elements in s. For any $a \in \mathbf{N}^{n_1}$ and any $C \subseteq \mathbf{R}^{n_2}$, let $D_{a,C} = \{s\ :\ s|I = a,\ s|J \in C\}$. Now, we define $\mathcal{D}_\sigma$ to be the σ-algebra generated by the sets in $\{D_{a,C}\ :\ a \in \mathbf{N}^{n_1},\ C \subseteq \mathbf{R}^{n_2}\ \textit{is a Borelset}\}$. A function $\mu\ :\ S_\sigma \to [0, \infty)$ is called a *probability function* if it is a measurable function and satisfies the following property:

$$\sum_{x\in\mathbf{N}^{n_1}} \int_{\mathbf{R}^{n_2}} \mu(x, x')dx' \ =\ 1,$$

where x is a vector of n_1 variables ranging over $\mathbf{N}$, and x' is a vector of n_2 variables ranging over $\mathbf{R}$. Note that, we first integrate over the continuous variables, keeping the discrete variables constant, and then sum over all possible values for the discrete variables. Note that, if σ has no 0s, i.e., $S_\sigma = \mathbf{R}^n$, then μ will be the standard probability density function. We say that μ is a *sub probability* function if it is a measurable function and

$$\sum_{x\in\mathbf{N}^{n_1}} \int_{\mathbf{R}^{n_2}} \mu(x, x')dx' \ \leq\ 1.$$

Let $n_1, n_2, m_1, m_2 \geq 0$ be integers and σ_1, σ_2, respectively be the vectors $0^{n_1}1^{n_2}$ and $0^{m_1}1^{m_2}$. Intuitively, n_1, n_2 give the number of discrete and continuous state variables, while m_1, m_2 give the number discrete and continuous outputs of the system being described.

An Extended Hidden Markov System (EHMS) H of dimensions (n_1, n_2, m_1, m_2), is a triple (f, g, μ) where $f : (S_{\sigma_1} \times S_{\sigma_1}) \rightarrow [0, \infty)$, $g : (S_{\sigma_1} \times S_{\sigma_2}) \rightarrow [0, \infty)$ and μ are functions, satisfying the following properties. Let x, y be sequences of $n_1 + n_2$ variables each and z be a sequence of $m_1 + m_2$ variables. The function $f(x, y)$, called *next state function*, is a probability function in the arguments in y; that is, for any appropriate fixed values for variables in x, f is a probability function on S_{σ_1} in the argument y. Similarly $g(x, z)$, called *output function*, is a probability function in the arguments in z. Finally, μ is a probability function on S_{σ_1}.

Intuitively, the first set of arguments in $f()$ and $g()$ give the discrete and continuous parts of the current state. In $f()$ the last set of arguments give discrete and continuous parts of the next state, while in $g()$ they give the discrete and continuous parts of the output values.

For convenience, from here onwards, we let S denote S_{σ_1} and Σ denote S_{σ_2} which represent the set of possible system states and outputs respectively. Intuitively, H describes a dynamic stochastic system with n_1, n_2 discrete, continuous state variables respectively, and with m_1, m_2 discrete,continuous output values respectively. The state of such a system, at any instance of time, is given by a value in S, denoting the values of the state variables. The outputs generated by the system at any instance is given by an element in Σ denoting the values of the $m_1 + m_2$ outputs. Essentially, given that the current state of the system is given by x, $f(x, y)$ denotes the probability density that the system is in state given by y at the next time instance. Similarly, $g(x, z)$ denotes the probability function in variables in z denoting the probability that the system generates the output values given by z. The function μ gives the probability distribution on the initial state. Many times we write the function $f()$ to be over four vectors of variables, i.e., $f(x, x', y, y')$ where x, x' denote the discrete and continuous part of the current state, and y, y' denote the discrete and continuous parts of the next state. Similar convention is followed for $g()$ and $\mu()$ as well.

We consider an ω-sequence over S as a computation/trajectory of H. The semantics of the EHMS H are given by three probability spaces $(S^\omega, \mathcal{E}_H, \phi_H)$, $(\Sigma^\omega, \mathcal{F}_H, \psi_H)$ and $((S \times \Sigma)^\omega, \mathcal{P}_H, \zeta_H)$ defined as follows.

First, we define the following notation. Let $n', n'' \geq 0$ be integers. For a finite sequence $a = (a_0, ..., a_{l-1})$ of elements from $\mathbf{N}^{n'}$ and a finite sequence $C = (C_0, C_1, ..., C_{l-1})$ of Borel subsets of $\mathbf{R}^{n''}$, let $E_{(a,C)}$ be the set of all ω-sequences $(s_0, ..., s_i, ...)$ of states in $S_{0^{n'}1^{n''}}$ such that $s_i \in \{a_i\} \times C_i$ for $0 \leq i < l$. Note that, if a, C are empty sequences, i.e., sequences of length zero, then $E_{(a,C)} = (S_{0^{n'}1^{n''}})^\omega$. Note that $E_{a,C}$ is well defined, for any $n', n'' \geq 0$, if a, C are of the same length.

Now, $\mathcal{E}_H$ is the smallest σ-algebra which contains the class of sets $\{E_{(a,C)} : \exists\, l \geq 0, a \in (\mathbf{N}^{n_1})^l \textit{ and } C$ is a finite sequences of Borel sets in $\mathbf{R}^{n_2}$ of length $l\}$.

For any μ', which is a probability function or is a sub-probability function, we define a unique probability measure $\Theta(\mu')$ on $\mathcal{E}_H$ such that for any $a \in (\mathbf{N}^{n_1})^l$, and any finite sequence $C = (C_0, ..., C_{l-1})$ of Borel sets in $\mathbf{R}^{n_2}$,

$$\Theta(\mu')(E_{(a,C)}) = \int_{\mathcal{C}} \mu'(a_0, x_0) \Big(\prod_{0 \leq i < l-1} f(a_i, x_i, a_{i+1}, x_{i+1}) dx_{i+1} \Big) dx_0,$$

where the integral is taken over the region $\mathcal{C}$ in which variables in x_i range over the region C_i, for $0 \leq i < l - 1$. The probability measure ϕ_H is defined to be $\Theta(\mu)$.

The class $\mathcal{F}_H$ is the smallest σ-algebra which contains the class of sets $\{E_{(b,C)} : \exists\, l \geq 0,\ b \in (\mathbf{N}^{m_1})^l,\ C = (C_0, ..., C_{l-1})$ is a sequence of Borel sets in $\mathbf{R}^{m_2}\}$. ψ_H is the unique probability measure on $\mathcal{F}_H$ such that

$$\psi_H(E_{(b,C)}) = \sum_{a_i \in \mathbf{N}^{n_1}, i < l} \int_{\mathcal{D}} \int_{\mathcal{C}} \mu(a_0, x_0) g(a_0, x_0, b_0, y_0) \Big(\prod_{0 \leq i < l-1} F_i \, dx_{i+1} dy_{i+1} \Big) dx_0 dy_0,$$

where

$$F_i = f(a_i, x_i, a_{i+1}, x_{i+1}) g(a_{i+1}, x_{i+1}, b_{i+1}, y_{i+1}).$$

and $\mathcal{C}$ is the region in which the variables in $x_0, ..., x_{l-1}$ range over $\mathbf{R}^{n_2}$ and $\mathcal{D}$ is the region where the variables in $y_0, ..., y_{l-1}$ range over $C_0, ..., C_{l-1}$, respectively.

For any $l \geq 0$ and $a \in (\mathbf{N}^{n_1+m_1})^l$ and any finite sequence $C = (C_0, ..., C_{l-1})$ of Borel sets of $\mathbf{R}^{n_2+m_2}$, let $F_{a,C}$ be the set of all ω-sequences in $(S \times \Sigma)^\omega$ of the form $((s_0, t_0), ..., (s_i, t_i), ...)$ such that, for $i \geq 0$, $s_i = (b_i, u_i)$, $t_i = (c_i, v_i)$ where b_i, c_i are, respectively, the vectors consisting of the first n_1 and the last m_1 elements of a_i, and u_i, v_i are, respectively, the vectors consisting of the fist n_2 and the last m_2 elements of a vector in C_i.

The class $\mathcal{P}_H$ is the smallest σ-algebra which contains the class of sets $\{F_{(a,C)} : \exists\, l \geq 0, a \in (\mathbf{N}^{n_1+m_1})^l, C = (C_0, ..., C_{l-1})$ is a finite sequence of open sets in $\mathbf{R}^{n_2+m_2}\}$. ζ_H is the unique probability measure on $\mathcal{F}_H$ such that

$$\zeta_H(F_{(a,C)}) = \int_{\mathcal{C}} \mu(b_0, x_0) g(b_0, x_0, c_0, y_0) dx_0 dy_0 \prod_{0 \leq i < l-1} G_i \, dx_{i+1} dy_{i+1},$$

where

$$G_i = f(b_i, x_i, b_{i+1}, x_{i+1}) g(b_{i+1}, x_{i+1}, c_{i+1}, y_{i+1}),$$

and b_i is the vector consisting of the first n_1 values and c_i is the vector consisting of the last m_1 values of the vector a_i and $\mathcal{C}$ is the region in which the $n_2 + m_2$ variables in x_i and y_i range over C_i, for each $i = 0, ..., l-1$.

For any $s = (s_0, s_1, ... s_{n_1+n_2-1}) \in S$ and $t = (t_0, t_1, ... t_{m_1+m_2-1}) \in \Sigma$, let $s \otimes t$ denote the vector obtained by concatenating elements of s, t in that order. For any $u \in S^\omega$ and $v \in \Sigma^\omega$, given by $u = (u_0, ..., u_i, ...)$ and $v = (v_0, ..., v_i, ...)$, let $u \otimes v$ denote the unique $w = (w_0, ..., w_i, ...)$ where $w_i = u_i \otimes v_i$, for each $i \geq 0$. Now, for any $X \subseteq S^\omega$ and $Y \subseteq \Sigma^\omega$, let $X \otimes Y = \{u \otimes v : u \in X, v \in Y\}$. Observe that $X \otimes Y \subseteq (S \times \Sigma)^\omega$.

The following lemma is fairly straightforward to prove and is left to the reader.

Lemma 1. *If $X \in \mathcal{E}_H$ and $Y \in \mathcal{F}_H$ then $X \otimes Y \in \mathcal{P}_H$.*

Probabilistic Hybrid Automata. Probabilistic Hybrid Automata (PHA) were defined in [13, 14]. They provide a convenient formalism for specifying systems. A probabilistic hybrid automaton $\mathcal{A}$ is a tuple $(Q, V, \Delta t, \mathcal{E}, \mathcal{T}, c_0)$ where Q is a countable set of *discrete* states (modes); V is a disjoint union of three sets V_1, V_2 and V_3 called the continuous state variables, output variables and noise variables, respectively, that all take values in $\mathbf{R}$; Δt is the sampling time; $\mathcal{E}$ is a function that with each $q \in Q$ associates a set $\mathcal{E}(q)$ of discrete-time state equations [8] describing the evolution of the continuous state (value of the variables in V_1) and the output (value of the variables in V_2) at time $t + \Delta t$ as functions of the state at t and the noise variables[1]; $\mathcal{T}$ is a function that assigns to each $q \in Q$ a set of *transitions* (ϕ, p), where the *guard* ϕ is a measurable predicate over the set of continuous (and possibly discrete) state variables and p is a probability distribution over Q; and c_0 is a pair giving the initial discrete state and an initial continuous probability distribution on the variables in V_1. We require that for each $q \in Q$, the state equations in $\mathcal{E}(q)$ have noise variables on the right hand side and that the set of guards on the transitions in $\mathcal{T}(q)$ be mutually exclusive and exhaustive.

We next give the semantics of the PHA. Within each mode q, the evolution of the PHA is given by the difference equations. When the guard ϕ of a transition $(\phi, p) \in \mathcal{T}(q)$ becomes

[1] Additional explicit discrete state variables can be added to the variables in V provided their update equations only involve deterministic functions of such discrete state variables.

satisfied, a transition takes place from q to some target mode q' according to the probability distribution p. Since there are no deterministic resets to variables in the transitions, it is quite easy to see that the semantics of $\mathcal{A}$ can be given in terms of an EHMS system $H_{\mathcal{A}}$.

Monitors. We consider monitors that take outputs Σ, generated by a EHMC H, as inputs and accept or reject them. Formally, a monitor $M : \Sigma^* \rightarrow \{0, 1\}$ is a 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. We require that $L(M)$ is a measurable set, i.e., is in $\mathcal{F}_H$.

It is to be noted that, in practice, a monitor is given as an algorithm that takes a finite sequence of elements from Σ as input, and raises an alarm or not. Here raising an alarm is considered as rejection. Associated with such an algorithm, there is a monitor function as defined above. Note that such a function is computable. Thus, for a monitor M to be feasible it has to be computable.

Accuracy Measures. Let $H = (f, g, \mu)$ be a EHMC over (n_1, n_2, m_1, m_2)-dimensions. Let S, Σ be the states and outputs of H as defined earlier. Let $GOOD$ be a set in $\mathcal{E}_H$ denoting a measurable set. We fix $GOOD$ and call its members as good computations of H. Let $M : \Sigma^* \rightarrow \{0, 1\}$ be a monitor as given above. The *acceptance accuracy* of M for $GOOD$ with respect to H, denoted by $AA(M, H, GOOD)$, is defined to be the conditional probability that M accepts the output generated by H given that the computation of H is in $GOOD$. Formally, it is defined to be the value given by

$$AA(M, H, GOOD) = \frac{\zeta_H(GOOD \otimes L(M))}{\zeta_H(GOOD \otimes \Sigma^\omega)}.$$

Note that $\zeta_H(GOOD \otimes L(M))$ denotes the probability that a computation in $GOOD$ generates an output in $L(M)$, while $\zeta_H(GOOD \otimes \Sigma^\omega)$ denotes the probability that the system generates a computation in $GOOD$. Also note that the later value is same as $\phi_H(GOOD)$.

All computations that are not in $GOOD$ are called bad and we denote this set by BAD, i.e.,$BAD = S^\omega - GOOD$. The *rejection accuracy* of M for $GOOD$ with respect to H, denoted by $RA(M, H, GOOD)$, is defined to be the conditional probability that M rejects the output generated by a bad computation of H. Formally, it is defined to be the value given by

$$RA(M, H, GOOD) = \frac{\zeta_H(BAD \otimes (\Sigma^\omega - L(M)))}{\zeta_H(BAD \otimes \Sigma^\omega)}.$$

Note that the $\zeta_H(BAD \otimes (\Sigma^\omega - L(M)))$ denotes the probability that a bad computation is rejected by M, while $\zeta_H(BAD \otimes \Sigma^\omega)$ denotes the probability that the system generates a bad computation. Clearly, both the above accuracies are defined only when $0 < \phi_H(GOOD) < 1$.

Monitorability. We say that a system H is *monitorable* with respect to $GOOD$ if for every $x \in [0, 1)$ there exists a monitor M such that $AA(M, H, GOOD) \geq x$ and $RA(M, H, GOOD) \geq x$. In the next section, we give necessary and sufficient conditions for these properties to be satisfied.

It is worth noting that monitorability, while related to the classical notion of observability, is fundamentally different from it. It is not difficult to construct hybrid systems that are not observable or even discrete-state observable but are monitorable.

4 Monitorability

In this subsection, we give necessary and sufficient conditions for monitorability. Let $H = (f, g, \mu)$ be the given EHMS over (n_1, n_2, m_1, m_2)-dimensions. Let $GOOD$ be a any member of $\mathcal{E}_H$, i.e., is a measurable subset of S^ω such that $0 < \phi_H(GOOD) < 1$.

Consider an integer $l > 0$ and let $O = (y_0, ..., y_l)$ denote a sequence of variables denoting an output sequence of elements from Σ^m; each y_i consists of m_1+m_2 variables corresponding to discrete as well as continuous outputs. Define a function $I(y_0, ..., y_l)$ as follows.

$$I(y_0, ..., y_l) \; = \; \sum \int \mu(u_0, x'_0) g(u_0, x'_0, y_0) (\prod_{0 \le i < l} F_i \; dx'_{i+1}) dx'_0$$

where $F_i \; = \; f(u_i, x'_i, u_{i+1}, x'_{i+1}) g(u_{i+1}, x'_{i+1}, y_{i+1})$.

In the above equation, u_i, x'_i denote n_1 discrete state variables and n_2 continuous state variables, respectively. The integration is over variables in x'_i, for $i = 0, ...,$, where each such variable ranges over $\mathbf{R}$. The summation is over all discrete variables in u_i, for $i = 0, ..., l$. Each such variable ranges over $\mathbf{N}$.

Essentially, $I(y_0, ..., y_l)$ defines a probability function, over $(l+1)m_1$ discrete variables and $(l+1)m_2$ continuous variables, giving the probability of generation of the output sequence O by all the computations of the system. Using the fact that sums and products of measurable functions are also measurable, and using Fubini's theorem (see [26]) we see that $I(y_0, ..., y_l)$ is a measurable function.

Now, we define sub-probability function $J(y_0, ..., y_l)$ for the probability of generation of the output sequence O by computations of H that are in GOOD. For any sequence $\alpha = (x_0, ..., x_l)$ of states of the system, let $GOOD|\alpha$ denote the set of all sequences $\beta \in S^\omega$ such that $\alpha\beta \in GOOD$. For $i = 0, ..., l$, each x_i has n_1 discrete and n_2 continuous variables. Intuitively, $GOOD|\alpha$ is the set of sequences obtained by dropping the prefix α from those sequences in $GOOD$ that have α as a prefix. It is easy to show that $GOOD|\alpha$ is a measurable set. Now, let $\mu'_{O,\alpha}(z)$ be a probability function on S defined by:

$$\mu'_{O,\alpha}(z) \; = \; \mu(x_0) (\prod_{0 \le i < l} g(x_i, y_i) f(x_i, x_{i+1})) g(x_l, y_l) f(x_l, z)$$

Essentially, it gives the probability that the output sequence O is generated by the sequence of states given by α followed by the state given by z. Note that, in the above equation each x_i has n_1 discrete variables and n_2 continuous variables. Let $U_O(\alpha)$ be the measure of the set $GOOD|\alpha$ defined by the initial probability density function $\mu'_{O,\alpha}$, i.e., $U_O(\alpha) \; = \; \Theta(\mu'_{O,\alpha})(GOOD|\alpha)$. Now, define

$$J(y_0, ..., y_l) \; = \; \sum \int U_O(u_0, x'_0, ..., u_l, x'_l) dx_0 ... dx_l$$

Lemma 2. *$GOOD|\alpha$ is a measurable set and $J(y_0, ..., y_l)$ is a measurable function.*

In the above equation, for each $i = 0, ..., l$, each u_i denotes n_1 discrete variables and each x'_i denotes n_2 continuous variables. Further more the integration is over all variables in x'_i, for $i = 0, ..., l$, while the summation is over all variables in u_i, for $i = 0, ..., l$. Each continuous variable ranges over $\mathbf{R}$, while each discrete variable ranges over $\mathbf{N}$. It should be easy to see that $J(y_0, ..., y_l)$ is the probability function for the probability of generation of outputs given by O by computations of the system that are in $GOOD$. It should be easy to see that $J(y_0, ..., y_l) \leq I(y_0, ..., y_l)$ for all appropriate values for variables in y_i, $i = 0, ..., l$. Let $GoodProb(O)$ be the value defined by

$$GoodProb(O) \; = \; \frac{J(y_0, ..., y_l)}{I(y_0, ..., y_l)}$$

Observe that $GoodProb(O) \leq 1$. Essentially $GoodProb(O)$ gives the conditional probability that the computation of the system is good, i.e. in $GOOD$, given that the output generated in the first $l+1$ states is given by O. It is not difficult to see that $GoodProb(y_0, ..., y_l)$ is a measurable function since $I(y_0, ..., y_l)$, $J(y_0, ..., y_l)$ are measurable functions.

Recall that for any $\beta \in \Sigma^\omega$ and integer $i \geq 0$, $\beta[0, i]$ denotes the prefix of β of length $i+1$. Now, let $OneSeq(H, GOOD)$ be the set of all $\beta \in \Sigma^\omega$ such that $\lim_{i \to \infty} GoodProb(\beta[0, i])$ exists and it's value is 1. Similarly, let $ZeroSeq(H, GOOD)$ be the set of all $\beta \in \Sigma^\omega$ such that the above limit exists and is equal to 0. Let $ZeroOneSeq(H, GOOD) = OneSeq(H, GOOD) \cup ZeroSeq(H, GOOD)$. The following lemma states that the sets $OneSeq(H, GOOD)$ and $ZeroSeq(H, GOOD)$ are measurable. It also states that the measure of the computations of H that generate output sequences in $OneSeq(H, GOOD)$ (resp., sequences in $ZeroSeq(H, GOOD)$) and that are bad (resp. that are good) is zero.

Lemma 3. *The sets $OneSeq(H, GOOD)$ and $ZeroSeq(H, GOOD)$ are measurable (both are members of $\mathcal{F}_H$). Furthermore,*

$$\begin{aligned} \zeta_H(BAD \otimes OneSeq(H, GOOD)) &= 0 \quad \textit{and} \\ \zeta_H(GOOD \otimes ZeroSeq(H, GOOD)) &= 0. \end{aligned}$$

The following *monitorability theorem* gives a necessary and sufficient condition for the monitorability of H with respect to $GOOD$ and is a generalization of the corresponding theorem for HMCs, i.e.,discrete state systems proved in [31]. The theorem states that H is monitorable with respect to $GOOD$ iff with probability 1, any random infinite output sequence generated by H is in $ZeroOneSeq(H, GOOD)$.

Theorem 1. *For any EHMS H and measurable set $GOOD$ in $\mathcal{E}_H$, H is monitorable with respect to $GOOD$ iff $\psi_H(ZeroOneSeq(H, GOOD)) = 1$.*

Proof. Let $H = (f, g, \mu)$ be a EHMS of dimension (n_1, n_2, m_1, m_2) and $GOOD$ be a measurable set in $\mathcal{E}_H$. Assume that H is monitorable with respect to $GOOD$.

Suppose that $\psi_H(ZeroOneSeq(H, GOOD)) < 1$. Let $F = \Sigma^\omega - ZeroOneSeq(H, GOOD)$. Clearly $\psi_H(F) > 0$. Consider any $\beta \in F$. It should be easy to see that for some $u > 0$, the following property (*) holds (otherwise β is in $ZeroOneSeq(H, GOOD)$).
(*) For infinitely many values of i, $GoodProb(\beta[0, i]) < (1 - \frac{1}{2^u})$ and for infinitely many values of j, $GoodProb(\beta[0, j]) > \frac{1}{2^u}$

For each $u > 0$, define F_u to be the set of sequences $\beta \in F$ such that u is the smallest integer that satisfies (*). It is easy to see that the set $\{F_u : u > 0\}$ is a partition of F. It should also be easy to see that $F_u \in \mathcal{F}_H$, i.e., is measurable, for each $u > 0$. Since $\psi_H(F) > 0$, it follows that for some $u > 0$, $\psi_H(F_u) > 0$. Fix such an u and let $x = \psi_H(F_u)$. From property (*), it can be shown that for any $C \in \mathcal{F}_H$, such that $C \subseteq F_u$ and $y = \psi_H(C) > 0$, the following property (**) holds:
(**) $\zeta_H(GOOD \otimes C) \geq \frac{y}{2^u}$ and $\zeta_H(BAD \otimes C) \geq \frac{y}{2^u}$.

Now, consider any monitor M. Recall that $L(M)$ is the set of infinite sequences in Σ^ω that are accepted by M. Further more, $L(M)$ is a safety property and is measurable, i.e., $L(M) \in \mathcal{F}_H$. Now, since $x = \psi_H(F_u)$ and $F_u = (F_u \cap L(M)) \cup (F_u - L(M))$, it is the case that either $\psi_H(F_u \cap L(M)) \geq \frac{x}{2}$ or $\psi_H(F_u - L(M)) \geq \frac{x}{2}$. In the former case, by taking $C = F_u \cap L(M)$ and using property (**), we see that the measure of bad computations of the system (i.e., those in BAD) that are accepted by the monitor is $\geq \frac{x}{2^{u+1}}$ and in the later case, by taking $C = (F_u - L(M))$, the measure of good computations of the system (i.e., those that are in $GOOD$) that are rejected is $\geq \frac{x}{2^{u+1}}$. Now, let $z = \max\{\phi_H(GOOD), \phi_H(BAD))\}$. From the above arguments, we see that , for every monitor M, either $RA(M, H, GOOD) \leq 1 - \frac{x}{z \cdot 2^{u+1}}$ or $AA(M, H, GOOD) \leq 1 - \frac{x}{z \cdot 2^{u+1}}$. This contradicts our assumption that H is monitorable with respect to $\mathcal{A}$.

Now, assume that $\psi_H(ZerOneSeq(H, GOOD)) = 1$. Let $z \in (0, 1)$. Let $M_z : \Sigma^* \to \{0, 1\}$ be a function such that for any $\alpha \in \Sigma^*$, $M_z(\alpha) = 0$ iff there exists a prefix α' of α such that $GoodProb(\alpha') < 1 - z$. Clearly M_z is a monitor. When extended to infinite sequences $M_z(\beta) = 0$ for every $\beta \in ZeroSeq(H, GOOD)$, i.e., it rejects all of them. The second part of Lemma 3 implies that $\zeta_H(GOOD \otimes ZeroSeq(H, GOOD)) = 0$, and it can further be deduced that $\zeta_H(BAD \otimes ZeroSeq(H, GOOD)) = 1$. From these observations, it follows that $RA(M_z, H, GOOD) = 1$. It should be easy to see that the measure of good computations of H that are rejected by M_z is $\leq \min\{y, 1 - z\}$ where $y = \phi_H(GOOD)$. Therefore, $AA(M_z, H, GOOD) \geq 1 - \frac{\min\{y, 1-z\}}{y}$. Now, for any given $x \in (0, 1)$, we can chose a value of z such that $AA(M_z, H, GOOD) \geq x$ and $RA(M_z, H, GOOD) = 1$. This implies that H is monitorable with respect to $GOOD$. □

5 Monitoring Algorithms

In this section we describe how the formal methodology developed above can be used in applications. We assume that the system to be monitored is specified by a probabilistic hybrid automaton (PHA) [13,14] as described in Sec. 3. Let the system under consideration be specified by a PHA $\mathcal{A}$. Recall that $\mathcal{H}_{\mathcal{A}}$ denotes the EHMS capturing the behavior of $\mathcal{A}$.

Property Automaton. We consider traditional deterministic Buchi automata to specify properties over sequences of states of $H_{\mathcal{A}}$. We allow these automata to have possibly infinite, but countable, number of states. Formally, a Buchi automaton $\mathcal{P}$ is given by a tuple $(Q_{\mathcal{P}}, \delta, q_0, F)$ where $Q_{\mathcal{P}}$ is a countable number of states, $q_0 \in Q_{\mathcal{P}}$ is the initial state, $F \subseteq Q_{\mathcal{P}}$ is the set of accepting states and δ is a set of triples of the form (q, ϕ, q') where $q, q' \in Q_{\mathcal{P}}$ and ϕ is a measurable predicate over the states of $H_{\mathcal{A}}$. We require that, for each state q, the set of predicates ϕ in the transitions starting from q form a partition of the state space of $H_{\mathcal{A}}$. $\mathcal{P}$ takes the states of $H_{\mathcal{A}}$ as inputs. When in state q, on input s, $\mathcal{P}$ changes from state q to q' if there is a transition $(q, \phi, q') \in \delta$ such that s satisfies the predicate ϕ. Given an infinite sequence of states of $H_{\mathcal{A}}$, we define a run of $\mathcal{P}$ starting from q_0 in the natural way. Such a run is said to be an accepting run if there are infinite instances on the run where the state of the run is an accepting state. An input is accepted if there is an accepting run of $\mathcal{P}$ on the input. It is well known that the set of inputs accepted by $\mathcal{P}$ is a measurable set, i.e., is a member of $\mathcal{E}_{H_{\mathcal{A}}}$. A safety automaton is a property automaton having a special absorbing state called *error* state with all the other states being the accepting states. The set of sequences accepted by a safety automaton is a safety property.

Product Automaton. In order to monitor whether the system specified by $\mathcal{A}$ satisfies the property specified by $\mathcal{P}$, we construct the product of $\mathcal{A}$ and $\mathcal{P}$ in a natural way. This product is a hybrid automaton and we designate it by $\mathcal{B}$. The formal specification of $\mathcal{B}$ is left out due to space limitations. Each discrete state of $\mathcal{B}$ is a pair (q, q') where q is the discrete state of $\mathcal{A}$ and q' is a state of $\mathcal{P}$. Essentially, $\mathcal{B}$ behaves like $\mathcal{A}$ and at the same time runs $\mathcal{P}$ on the sequence of states of $\mathcal{A}$. Observe that the outputs generated by $\mathcal{A}$ and $\mathcal{B}$ are same.

5.1 Monitoring Safety Properties

Let $\mathcal{A}$ be a system automaton, $\mathcal{P}$ be a safety property automaton specified on $\mathcal{A}$ and $\mathcal{B}$ be the product hybrid automaton. Let $GOOD$ be the set of all input sequences accepted by $\mathcal{P}$.

Consider an infinite sequence α of outputs generated by the EHMS $H_{\mathcal{A}}$. Recall that $\alpha[0,i]$ denotes the prefix of α of length $i+1$. For EHMS $H_{\mathcal{A}}$, let $u_i = 1 - \mathit{GoodProb}(\alpha[0,i])$ and v_i be the probability that the EHMS $H_{\mathcal{B}}$ is in an error state given that $\alpha[0,i]$ is the sequence of outputs generated by it. The following lemma shows that u_i and v_i converge to the same value.

Lemma 4. *For all $i \geq 0$, $v_i \leq u_i$ and $\lim_{i\to\infty}(v_i) = \lim_{i\to\infty}(u_i)$.*

Assume that $\mathcal{A}$ is monitorable with respect to $\mathcal{P}$. For any $z > 0$, let M_z be a monitor that works as follows. On any infinite sequence α of outputs generated by the system given by $\mathcal{A}$ and at instance of time $i \geq 0$, M_z estimates the probability v_i using the product system $\mathcal{B}$, and rejects if $v_i > z$. Using the proof of Theorem 1 and Lemma 4, we see that by increasing z arbitrarily close to 1, we can get monitors whose accuracies are arbitrarily close to 1.

5.2 Liveness Monitoring

Now we give an approach for monitoring properties specified by liveness automata using the methods given in [22, 30]. Let $\mathcal{P}$ be a property automaton which is a liveness automaton. We convert it into a safety automaton $\mathcal{P}'$ by using timeouts as follows. Let T be a large time constant. We modify $\mathcal{P}$ so that it goes to a special error state q_{error} if it does not reach an accepting state with in T steps initially, and also after each occurrence of an accepting state. It is fairly straightforward to obtain such an automaton $\mathcal{P}'$. $\mathcal{P}'$ is a safety automaton. It is fairly easy to show that any input sequence that is rejected by $\mathcal{P}$ is also rejected by $\mathcal{P}'$; however $\mathcal{P}'$ rejects more input sequences. Thus, $\mathcal{P}'$ is an approximation of $\mathcal{P}$. Note that we get better approximations by choosing larger values of T.

We can also construct a safety automaton $\mathcal{P}'$ that is even a better approximation by increasing the time outs after each occurrence of an accepting state. Let h be a monotonically increasing function from $\mathbf{N}^+$ to $\mathbf{N}^+$ where $\mathbf{N}^+$ is the set of natural numbers. We construct $\mathcal{P}'$ which behaves like $\mathcal{P}$ except that it goes to the errors state q_{error} if the next accepting state is not reached with in $h(i)$ steps after the i^{th} occurrence of an accepting state, for each $i \geq 0$. Such an automaton can be defined using countable number of states or concisely using the hybrid automata formalism using discrete state variables. When h is a constant function then we get constant time outs; when it is a linear function we get linearly increasing time outs. By using functions h that start with high initial values and that grow fast, we can get monitors with higher accuracies.

6 Example

Consider the operation of a train with electronically-controlled pneumatic (ECP) brakes [7]. In this case, a braking signal is sent to each of the N cars (($N = 5$ for the simulations) of the train that subsequently engage their own brakes. We consider the case when the braking systems of individual cars can fail. If this happens to more than a given number of cars ($2N/3$ in our example) the train might not be able to stop and it should start an emergency stopping procedure (for example, engaging the brakes using the traditional pneumatic system). Furthermore, when some of the brakes can fail, the links that connect the cars can be subject to excessive levels of stress. To prevent possible damage to the links we also want to trigger the emergency stopping procedure if any of the links are subject to a stress force above a safe threshold. We would thus like to develop a monitor that can correctly trigger the emergency stopping procedure when either the number of failed brakes is excessive, or when the braking pattern might result in unsafe level of stress on links between the cars, allowing the train operators to take the advantage of the superior braking performance of the ECP while not sacrificing the safety of the train.

Figure 1 describes how the train velocity v evolves. The train starts in the discrete state $q_v = 1$ and remains in that state until the velocity exceeds a threshold $V_U = 28.5$, when it

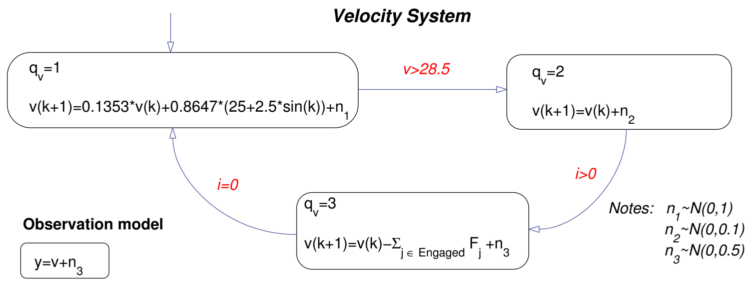

Fig. 1: Velocity subsystem for the train with ECP brakes.

switches to the discrete state $q_v = 2$. The train remains in the state $q_v = 2$ until one of the brakes engages and it switches to state $q_v = 3$. The velocity in states $q_v = 2$ and $q_v = 3$ depends on the number of brakes that have been engaged through the braking force term $-\sum_{j \in Engaged} F_j$, where F_j is the braking force of car j and $Engaged$ is the set of indeces of the cars whose brakes are engaged. When all the brakes disengage, the velocity system switches back to the state $q_v = 1$. When in the state $q_v = 1$, the train accelerates to a constant velocity $V_C = 25$ and oscillates around it with the amplitude 2.5. The measured variable y is assumed to be v, however the measurements are corrupted by a measurement noise n_3. It is worth noting that the dynamics of the system (and thus statistical properties of output sequences) are different for $q_v = 1$ and $q_v = 3$. Furthermore, we will assume that the braking forces have the form $F_j = \kappa_n + \kappa_p b^j$ for some appropriate constants κ_n, κ_p and b so that the value of $\sum_{j \in Engaged} F_j$ uniquely determines which brakes are engaged (for simulations, $\kappa_n = 11.7$, $\kappa_p = 9.6$ and $b = 1.2$). These two properties can be used to show that the system is monitorable. It can be also easily seen that if all the braking forces are equal, the system is not monitorable since there is no way for the monitor to distinguish between different braking patterns.

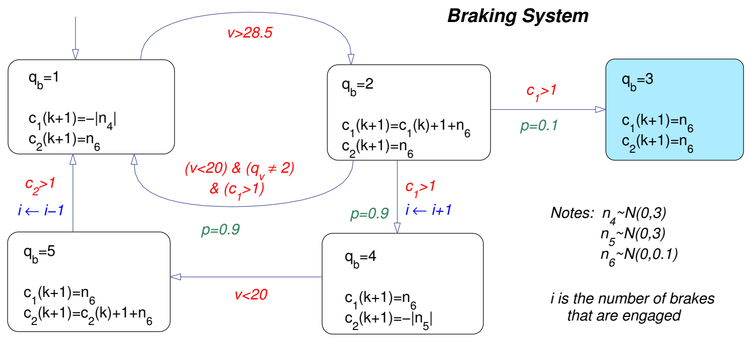

Fig. 2: ECP braking subsystem of each car of the train.

Figure 2 describes the operation of the braking system of each of the cars. The braking system starts in the discrete state $q_b = 1$ and remains in that state until the velocity exceeds a threshold $V_U = 28.5$, when it switches to the discrete state $q_b = 2$. The braking system remains in the state $q_b = 2$ until the timer c_1 reaches $L_1 = 1$ (modeling delays in actuation and computational delays). Note that the initial value of the timer c_1 in the state $q_b = 2$ is not deterministic, so the duration of time the system remains in $q_b = 2$ is a random variable. After the timer reaches L_1, the braking system can fail with a probability $p = 0.1$ and permanently switch to $q_b = 3$. With the probability $p = 0.9$ it either returns to state $q_b = 1$ if the velocity already fell below the threshold $V_L = 20$, or switches to $q_b = 4$ and engages in braking sequence otherwise. When the brake engages the variable i is increased by 1, thereby affecting the velocity of the train as described above. When the velocity falls below $V_L = 20$, the brake disengages after a random amount of time (modeled by the timer c_2 in the state $q_b = 5$), when it switches to the state $q_b = 1$.

Since a braking system is defined for each car, the overall model of the system is roughly a product of N copies of the braking system with the velocity system. For $N = 5$, the number of discrete states of the resulting automaton is more than 9000. Observe that if the system above is allowed to run forever then all the breaks will eventually fail with probability one. To prevent this, we assume that the brakes can only fail in the first τ units of time. To capture this, we add an additional counter in the breaking subsystem that allows the transition from $q_b = 2$ to the $q_b = 3$ only if this counter is less than τ; this is not shown in the figure. For the simulations, $\tau = 500$.

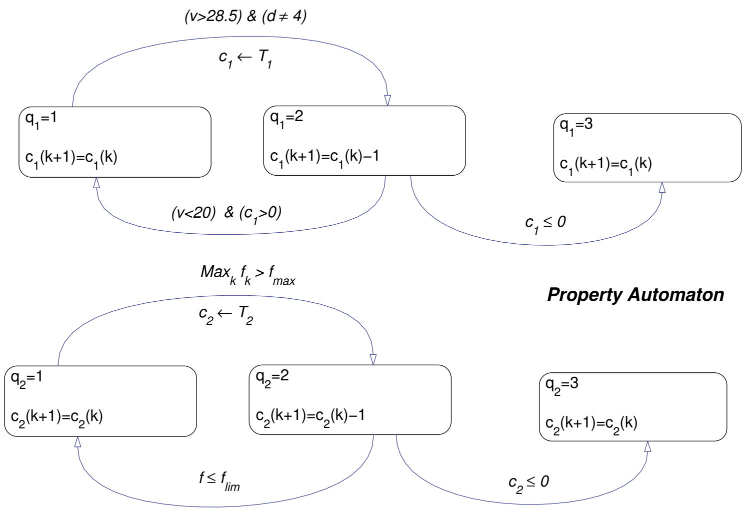

Fig. 3: Property automaton for the train with ECP brakes.

The desired behavior of the train is given by the following specification: every time the train velocity increases beyond V_U, the train should brake so that the velocity decreases below V_L and the link force f_k for any link k should never stay above the safe threshold f_{max} for more than T_2 ($T_2 = 4$ for simulations). The first property can be described by a liveness automaton that has two states $q_1 = 1$ and $q_1 = 2$, and whose initial state as well as the single acceptance state is $q_1 = 1$. This automaton can be converted to a safety automaton using a static time out T_1 according to the approach given in Section 5.2. Figure 3 shows the two automata that describe both safety properties, where $q_1 = 3$ and $q_2 = 3$ are the error states. Note that formally the property automaton is the product of the two automata shown in the figure, with the product automaton entering the error state any time either of the two automata enters an error state. For the simulations, the force threshold was set to $f_{max} = 30$. Out of 32 possible braking patterns, 5 cause the link force to exceed this threshold.

State estimation. Let $\mathcal{S}$ be the system automaton and $\mathcal{P}$ the property automaton given by the product of the two automata in Fig. 3. We construct the product of $\mathcal{S}$ and $\mathcal{P}$ to obtain the product automaton $\mathcal{S} \times \mathcal{P}$. Using this product automaton and using the outputs generated by the actual system, our Monitor M estimates the probability that the state component of the property automaton $\mathcal{P}$ is the bad state. Thus, it becomes necessary to estimate the probability that the property automaton $\mathcal{P}$ enters the bad state. This can be achieved by propagating the belief (probability distribution over the states of the product automaton) from the current state to the next state, given the new observation [27]. A similar approach has been used in [35]. Particle filters were developed as a computationally efficient approximation of the belief propagation [6,15,32]. They have been successfully applied in the hybrid system community for state estimation [3,17,23,34]. These methods become impractical for realistic systems with high number of states and several improvements have been suggested in recent years. It is also worth noting that for particle filters, both estimation accuracy and time complexity increase with the number of particles. The exact relationships depend on the structure of the system and transition probabilities and are difficult to characterize. All these issues are beyond the scope of the present paper.

Experimental results. As described in Section 4, the monitor M computes the probability that the property automaton $\mathcal{P}$ is in a bad state and raises an alarm when this probability surpasses a given threshold P. In order to evaluate the performance of the monitor numerically, the system was run 100 times. Particle filter was used to estimate the probability of each state of the product automaton $\mathcal{S} \times \mathcal{P}$. The number of particles for the particle filter was $\eta = 2400$. This number was chosen so that particles were not depleted during the transients corresponding to discrete state transitions. The simulation was terminated when either an alarm was raised, or the discrete time (number of steps the system has taken) reached $T_d = 700$. As explained above, the brakes can only fail during the first $\tau = 500$ units of time. For each run, the state of the property automaton was recorded, as well as the state of the monitor. The acceptance and rejection accuracies, respectively, denoted by $AA(M, \mathcal{S}, \mathcal{P})$ and $RA(M, \mathcal{S}, \mathcal{P})$ were computed according to:

$$AA(M, \mathcal{S}, \mathcal{P}) = \frac{g_a}{g_a + g_r} \qquad RA(M, \mathcal{S}, \mathcal{P}) = \frac{b_r}{b_a + b_r},$$

where g_a (resp., g_r) is the number of good runs that were accepted (resp., rejected), and b_r (resp., b_a) is the number of bad runs that were rejected (resp., accepted). Note that g_r corresponds to the number of false alarms, and b_a to the number of missed alarms; accuracies approach 1 as these numbers approach 0. A run was considered good if the state of the property automaton at $T_d = 700$ was not an error state, and bad otherwise.

z	$T_1 = 20$					
	g_a	g_r	b_a	b_r	RA	AA
0.050	28	46	0	26	1.000	0.378
0.100	32	37	0	31	1.000	0.464
0.300	44	18	1	37	0.974	0.710
0.500	47	13	2	38	0.950	0.783
0.750	48	9	4	39	0.907	0.842
0.875	50	6	8	36	0.818	0.893
0.950	51	5	15	29	0.659	0.911

(a) Monitor outcomes for 100 runs.

z	$T_1 = 20$		$T_1 = 60$		$T_1 = 80$	
	RA	AA	RA	AA	RA	AA
0.050	1.000	0.378	1.000	0.316	0.958	0.355
0.100	1.000	0.464	1.000	0.423	0.966	0.465
0.300	0.974	0.710	0.968	0.667	0.968	0.725
0.500	0.950	0.783	0.939	0.806	0.939	0.836
0.750	0.907	0.842	0.909	0.851	0.909	0.881
0.875	0.818	0.893	0.848	0.866	0.853	0.894
0.950	0.659	0.911	0.727	0.881	0.647	0.909

(b) Monitor accuracies.

Table 1: Results of experiments for different values of z and T_1.

An example of the monitor performance for $T_1 = 20$ and different values of the threshold probability z is shown in Table 1a. As expected, if z increases the acceptance accuracy AA increases as it becomes more difficult to reject a run. Also, as z increases, the rejection accuracy decreases since it becomes more difficult for the particle filter to estimate that the property automaton entered the fail state with such a high probability.

Table 1b shows accuracy measures of the monitor for different values of the probability threshold z and time out value T_1. The example shows that in general, there is a trade-off between rejection and acceptance accuracies. As z increases the acceptance accuracy in general increases and the rejection accuracy decreases. The reason for this is that as z

increases, the estimate of the probability that the state of the property automaton $\mathcal{P}$ is bad must be higher before the monitor raises an alarm. Clearly for $z_1 < z_2$, if the monitor with the threshold z_2 would raise an alarm so would the monitor with the threshold z_1, while the reverse is not true. So with lower z, the probability that a false alarm is declared is higher. Similar argument applies to rejection accuracy.

The effect of T_1 on accuracies is not as clear. Recall that T_1 is the time out used to (conservatively) approximate a liveness property with a safety property. As T_1 increases, the acceptance accuracy for monitoring the liveness property increases. However, the property automaton for the example is a product between liveness property and another safety property (see Fig. 3). The effect of T_1 on monitor accuracy is thus not obvious.

It is worth noting that while Theorem 1 provides necessary and sufficient conditions for monitorability, in order for the monitor to be implementable, the system also needs to allow robust state estimation. When state estimation can not be performed reliably, a system designer might use additional sensors that provide more information about the system and thus better state estimation.

7 Conclusion

In this paper, we introduced Extended Hidden Markov Systems as models for cyber physical systems and considered the monitoring problem for them. This model does not discretize continuous variables. We exactly characterized when such systems are monitorable. We presented highly accurate monitoring algorithms when the system is specified by probabilistic hybrid automata and when it is monitorable with respect to the given property.

The monitors have been implemented for an automotive example using particle filters as state estimators. Experimental results showing the effectiveness of our approach are presented.

We used certain class of probabilistic hybrid automata for specifying systems. However, we required such automata should not contain resets of continuous variables on transitions. Removing this restriction, will be part of future work.

References

1. R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.
2. A. Balluchi, L. Benvenuti, M. Di Benedetto, and A. Sangiovanni-Vincentelli. Design of observers for hybrid systems. In *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 59–80. Springer, 2002.
3. H. Blom and E. Bloem. Particle filtering for stochastic hybrid systems. In *43rd IEEE Conference on Decision and Control, 2004. CDC*, volume 3, 2004.
4. M. d'Amorim and G. Roşu. Efficient monitoring of ω-languages. In *Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 311–318. Springer, 2005.
5. R. A. DeCarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, 2000.
6. A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, 2000.
7. Federal Railroad Administration. ECP brake system for freight service. `http://www.fra.dot.gov/downloads/safety/ecp_report_20060811.pdf`, 2006.
8. G. Franklin, J. Powell, and M. Workman. *Digital control of dynamic systems*. Addison-Wesley world student series. Addison-Wesley, 1998.
9. S. Funiak and B. Williams. Multi-modal particle filtering for hybrid systems with autonomous mode transitions. In *Workshop on Principles of Diagnosis*, 2003.
10. K. Gondi, Y. Patel, and A. Sistla. Monitoring the full range of ω-regular properties of stochastic systems. In *Verification, Model Checking, and Abstract Interpretation*, volume 5403 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2009.

11. L. Grunske and P. Zhang. Monitoring probabilistic properties. In *Proceedings of 7th Joint meeting of European Software Engineering Conference and ACM Symposium on Foundations of Software Engineering ESEC-FSE'09*, pages 183–192. ACM, 2009.
12. T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of computer and system sciences*, 57(1):94–124, 1998.
13. M. Hofbaur and B. Williams. Mode estimation of probabilistic hybrid systems. In *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 81–91. Springer, 2002.
14. M. W. Hofbaur. *Hybrid Estimation of Complex Systems*, volume 319 of *Lecture Notes in Control and Information Sciences*. Springer, 2005.
15. M. Isard and A. Blake. Condensation–conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.
16. M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from LTL specifications. In *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes on Computer Science*, pages 333–347. Springer, 2006.
17. X. Koutsoukos, J. Kurien, and F. Zhao. Estimation of distributed hybrid systems using particle filtering methods. In *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 298–313. Springer, 2003.
18. H. Kress-Gazit, G. Fainekos, and G. Pappas. Where's Waldo? Sensor-based temporal logic motion planning. In *IEEE Int. Conf. on Robotics and Automation*, pages 3116–3121, 2007.
19. U. Lerner, B. Moses, M. Scott, S. McIlraith, and D. Koller. Monitoring a complex physical system using a hybrid dynamic bayes net. In *Proceedings of the 18th Annual Conference on Uncertainty in AI (UAI)*, pages 301–310, 2002.
20. D. Liberzon. *Switching in Systems and Control*. Birkhäuser, Boston, MA, 2003.
21. J. Lygeros, D. N. Godbole, and S. S. Sastry. A game theoretic approach to hybrid system design. In *LNCS 1066*, pages 1–12. Springer, 1996.
22. T. Margaria, A. Sistla, B. Steffen, and L. Zuck. Taming interface specifications. In *CONCUR 2005 Concurrency Theory*, volume 3653 of *Lecture Notes in Computer Science*, pages 548–561. Springer, 2005.
23. S. McIlraith, G. Biswas, D. Clancy, and V. Gupta. Hybrid systems diagnosis. In *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 282–295. Springer, 2000.
24. A. Pnueli. Verifying liveness properties of reactive systems. In *Hybrid and Real-Time Systems*, volume 1201 of *Lecture Notes on Computer Science*. Springer, New York, NY, 1997.
25. A. Pnueli, A. Zaks, and L. D. Zuck. Monitoring interfaces for faults. In *Proceedings of the 5th Workshop on Runtime Verification (RV'05)*, 2005. To appear in a special issue of ENTCS.
26. W. Rudin. *Real and Complex Analysis*. McGrawHill, NewYork, 1987.
27. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, December 2002.
28. 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.
29. A. Sistla and A. Srinivas. Monitoring temporal properties of stochastic systems. In *Verification, Model Checking, and Abstract Interpretation*, volume 4905 of *Lecture Notes in Computer Science*, pages 294–308. Springer, 2008.
30. A. Sistla, M. Zhou, and L. Zuck. Monitoring off-the-shelf components. In *Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2006.
31. A. P. Sistla, M. Žefran, and Y. Feng. Monitorability in stochastic dynamic systems. In *Computer Aided Verification (CAV 2011)*, 2011.
32. S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.
33. A. J. van der Schaft and J. M. Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences*. Springer, 1999.
34. V. Verma, G. Gordon, R. Simmons, and S. Thrun. Real-time fault diagnosis. *IEEE Robotics & Automation Magazine*, 11(2):56–66, 2004.

35. C. Wilcox and B. Williams. Runtime verification of stochastic, faulty systems. In *Runtime Verification*, volume 6418 of *Lecture Notes in Computer Science*, pages 452–459. Springer Berlin / Heidelberg, 2010.