Learning Robust Markov Models for Safe Runtime Monitoring

Antonina Skurka
Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden
skurka@chalmers.se

Luko van der Maas
Radboud University
Nijmegen, the Netherlands
luko.vandermaas@ru.nl

Sebastian Junges
Radboud University
Nijmegen, the Netherlands
sebastian.junges@ru.nl

Hazem Torfah
Chalmers University of Technology
and University of Gothenburg
Gothenburg, Sweden
hazemto@chalmers.se

ABSTRACT

We present a model-based approach to learning robust runtime monitors for autonomous systems. Runtime monitors play a crucial role in raising the level of assurance by observing system behavior and predicting potential safety violations. In our approach, we propose to capture a system's (stochastic) behavior using interval Hidden Markov Models (iHMMs). The monitor then uses this learned iHMM to derive risk estimates for potential safety violations. The paper makes three key contributions: (1) it provides a formalization of the problem of learning robust runtime monitors, (2) introduces a novel framework that uses conformance-testing-based refinement for learning robust iHMMs with convergence guarantees, and (3) presents an efficient monitoring algorithm for computing risk estimates over iHMMs. Our empirical results demonstrate the efficacy of monitors learned using our approach, particularly when compared to model-free monitoring approaches that rely solely on collected data without access to a system model.

KEYWORDS

Model-based Runtime Monitoring, Decision-Making under Uncertainty, Learning Interval Hidden Markov Models

ACM Reference Format:
Antonina Skurka, Luko van der Maas, Sebastian Junges, and Hazem Torfah. 2026. Learning Robust Markov Models for Safe Runtime Monitoring. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 10 pages. https://doi.org/10.65109/JAKK2294

1 INTRODUCTION

Runtime monitoring is vital for raising the assurance in cyber-physical systems (CPS) [7, 19]. Runtime monitors (RMs) continuously observe the system for possibly unsafe situations. When such situations occur, RMs raise an alarm and may consequently trigger backup fail-safe plans to keep the system safe. In modern autonomous CPS, RMs become even more essential to ensure safety, in particular when integrating machine learning (ML) components [13]. In these settings, RMs must deal with various kinds of uncertainty, both in the decisions made by ML models and in the environments in which these systems operate. Often, system dynamics are probabilistic, and the environment state cannot be fully observed. RMs must take these aspects into account when predicting safety violations, while balancing false positives (too many alarms) and false negatives (dangerous situations go unnoticed). The type of information about the uncertainty in the system available to a monitor has a great impact on achieving this balance.

Consider an example of an autonomous plane approaching a runway with ground traffic. The plane is equipped with camera-based perception sensors. A key safety-critical routine is to observe the behavior of ground traffic and decide that either it is safe to land or that the landing shall be aborted. A safety specification expresses a minimum distance between plane and ground traffic at all times. There will always be uncertainty about the behavior of vehicles, but also in the perceived sensor data, due to induced noise. A RM must (implicitly or explicitly) estimate the risk level: the probability that the specification will be violated during the landing, and if so, by how much. The RM must then raise an alarm if this risk level exceeds some threshold. The accuracy of the monitor will highly depend on what information one can provide about the (stochastic) behavior of the on-ground vehicles and sensors.

In this paper, we study the problem of constructing monitors that robustly estimate the risk of violating a safety specification under uncertainty. We specifically introduce a novel model-based approach for learning *robust* RMs. In our approach, we propose a data-driven method for learning robust uncertainty models in the form of *interval Hidden Markov Models (iHMMs)*, a variant of hidden Markov models where transition probabilities are represented as intervals rather than fixed values. On top of the learned iHMMs, we develop an efficient monitoring algorithm for cautiously (or conservatively) estimating risk levels. Our approach begins with a refinement-based learning procedure that gradually constructs an iHMM from simulation runs collected from the system under observation. The resulting model is then integrated into a monitoring framework capable of estimating risks during system execution. The general flow of learning iHMMs and integrating them into a monitoring framework is depicted in Fig. 1.

Our approach fills a gap in the current state of the art in learning RMs. The state of the art can be broadly categorized into two types

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/JAKK2294

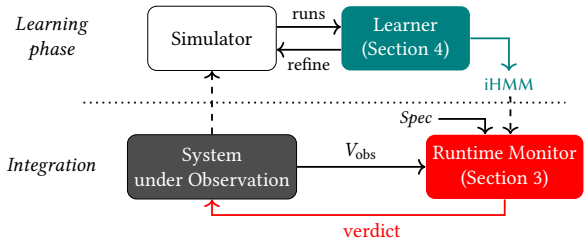


Figure 1: Learning and integrating robust monitors in CPS

of approaches. The first assumes a *known* model that captures the uncertainty of the system and environment using a predefined model. Typically, the models are hidden Markov models (HMM) or variants [3, 10, 18]. A key challenge in this setting is acquiring sufficiently accurate models. The second category of approaches, the *model-free methods*, avoids this challenge. Such methods learn classifiers directly from data, without relying on an explicit system model [9, 30, 36]. While promising, model-free approaches typically suffer from high sample complexity and often tend to have low accuracy, as they learn a monitor purely from observable data and lack deeper structural knowledge of the system.

In our work, we advocate for *learning models* and propose a method to overcome the challenge of acquiring uncertainty models that prevents the application of model-based approaches for RMs. Specifically, we present a data-driven strategy to learn an iHMM that serves as a feasible abstraction of the system. We then construct a RM that reasons (logically) about this learned abstraction. While the monitor still operates on sequences of observations, its decision-making is now grounded in the learned iHMM, effectively combining data-driven learning with model-based reasoning. Our interest in model-based learning approaches is inspired by their success in other areas. One may consider monitoring as a kind of sequential decision making under uncertainty (SDM), where the model-based approaches are dedicated (classical) planners and the model-free approaches are dedicated data-driven variations of reinforcement learning [20]. It has been shown that model-based approaches have great impact in increasing safety [8]. Furthermore, in SDM, model-based learning approaches are known to be more sample efficient. Indeed, in our experiments, we show that given the same amount of data, our models can reach high safety and accuracy, unmatched by the model-free approaches.

The iHMMs allow us to explicitly model uncertainty about the transition dynamics. We present a refinement-based method for learning iHMMs. Here, the model is updated iteratively by linearly adapting the probability intervals [29]. Our refinement approach incorporates a feedback loop driven by conformance testing. This loop assesses the uncertainty in monitoring decisions over the so-far learned models and guides the learning process by refining the model in areas where it underperforms. We particularly prove that the refinement converges to an optimal model with increasing numbers of refinements. Lastly, to enable efficient monitoring, we further introduce a tractable monitoring approach over iHMMs, inspired by [18]. Our monitoring procedure runs in polynomial time with respect to the length of observation sequences. We evaluate our approach on a set of benchmarks from the area of autonomy. They particularly confirm the advantages of the model-based RM setting against model-free approaches.

2 PROBLEM STATEMENT

Our goal is to construct a monitor that raises an alarm when a safety violation is imminent. We operate in a setting where both the system dynamics and the observations made by the monitor are subject to uncertainty.

We first give key notation and definitions. For a set X, we use X^* to define the set of finite sequences over X, and we use $X^{\leq h}$ to denote the set of sequences up to length h. In a state-based system, the set of states Σ_{sys} is defined in terms of valuations over a set of system variables V_{sys}. The monitor does not necessarily observe valuations over V_{sys}, but rather over V_{obs} - the set of observation variables (see Fig. 1 - Integration). In the remainder, we distinguish between system and observed executions, referring to sequences in Σ^*_{sys} as *paths* and sequences in Σ^*_{obs} as *traces*. For a countable set X, let $Dist(X) \subset (X \to [0,1])$ define the set of distributions over X. A distribution $d \in Dist(X)$ particularly satisfies, $\sum_{x \in X} d(x) = 1$.

Definition 1 (System under Observation (SuO)). *A discrete-time system is a tuple $\mathcal{S} = (V_{sys}, V_{obs}, d, obs)$, where V_{sys} and V_{obs} are sets of system and observation variables, the dynamics d is a distribution over Σ^*_{sys}, and $obs\colon \Sigma^*_{sys} \to Dist(\Sigma^*_{obs})$ is a mapping between paths and distributions over observation traces.*

A system under observation implicitly captures the interaction between a system and its environment. In the above definition, d defines the probability of a path occurring, and thus captures the uncertainty in environment behavior. The observation mapping obs captures the (noisy) behavior of the sensors, which for a given path might output different traces, depending on its stochastic behavior.

Definition 2 (Specification). *We define safety constraints as finite linear-time properties [5, 14]. A specification φ over Σ_{sys} is a set $\varphi \subseteq \Sigma^*_{sys}$. If $\pi \in \varphi$, we say π* satisfies *φ, denoted also by $\pi \models \varphi$.*

Based on the latter definition of safety, we can define the probability of violating a safety specification. We call this a risk function [18][1].

Definition 3 (Risk Function). *Given $\mathcal{S} = (V_{sys}, V_{obs}, d, obs)$ and a specification φ over Σ_{sys}, a risk function $r^h_\varphi\colon \Sigma^*_{sys} \to [0,1]$, for a horizon $h \in \mathbb{N}^+$, gives a probability of violating φ within a horizon of h steps for a given initial path $\pi \in \Sigma^*_{sys}$. We set $r^h_\varphi(\pi) = 0$ if $d(\pi) = 0$ and otherwise*

$$r^h_\varphi(\pi) = Pr(\{\pi' \in \Sigma^{\leq h}_{sys} \mid \pi \cdot \pi' \not\models \varphi\}) = \frac{1}{d(\pi)} \sum_{\pi' \in \Sigma^{\leq h}_{sys}} (\pi \cdot \pi' \not\models \varphi) \cdot d(\pi \cdot \pi')$$

The expression on the right is derived by applying Bayes' law on the conditional probability $d(\pi' \mid \pi)$. The specification and risk function are defined on the system level and it is in general not possible to lift them to the observation level without losing preciseness. In terms of the landing scenario from the introduction, we can define the risk of a (near-)collision based on the trajectories of the plane and the ground vehicles, however, a monitor does not know the true locations of the ground vehicles, but only their perceived locations. That is, the challenge in monitoring now is that

[1]Risk is usually defined as the probability of violation times the impact. In fact, the definition can be easily refined to also account for differences in impact.

we cannot observe the system path directly, and thus it is impossible to directly compute the associated risk. Next, we address this challenge by relating the system state space with the observations of the monitor.

A monitor for a SuO is simply a function $M\colon \Sigma^*_{\text{obs}} \to [0,1]$, assigning to each trace in a system a value between 0 and 1, giving a risk of an observation trace. To relate monitors to the system-level specification, we introduce the notion of an ideal monitor.

Definition 4 (Ideal Monitor). *Given $\mathcal{S} = (V_{sys}, V_{obs}, d, obs)$ and a specification φ, for a horizon h, and a loss function $L^{\varphi,h,\mathcal{S}}$, the ideal monitor $M^*\colon \Sigma^*_{obs} \to [0,1]$ is one s.t.*

$$M^* \in \operatorname*{arg\,min}_{M'\colon \Sigma^*_{obs}\to[0,1]} L^{\varphi,h,\mathcal{S}}(M').$$

The notion of the ideal monitor M^*, and consequently also the solution to the problem statement below, are dependent on a specific instantiation of the loss function. In this work, we consider the mean square error loss function. For some monitor $M \in \mathcal{M}$:

$$L^{\varphi,h,\mathcal{S}}_{\text{MSE}}(M) = \sum_{\tau\in\Sigma^*_{\text{obs}}} Pr(\tau) \sum_{\pi\in\Sigma^*_{\text{sys}}} Pr(\pi \mid \tau)\big(M(\tau) - r^h_\varphi(\pi)\big)^2$$

with $Pr(\tau) = \sum_{\pi\in\Sigma^*_{\text{sys}}} d(\pi)\cdot obs(\pi)(\tau)$ and $Pr(\pi\,|\,\tau) = \frac{d(\pi)\cdot obs(\pi)(\tau)}{Pr(\tau)}$.

Using the notion of an ideal monitor, we also define the set of *cautious monitors*, i.e., monitors which overestimate the risk compared to the ideal monitor.

Definition 5 (Cautious Monitors). *Given SuO $\mathcal{S}$ and ideal monitor M^*, the monitor M is cautious, if $M^*(\tau) \le M(\tau)$ for all $\tau \in \Sigma^*_{obs}$. We call the set of cautious monitors $\mathcal{M}^C$.*

Remark. The ideal monitor may still exhibit wrong risk assessments, due to the mismatch between the system and observation world, and we only ask a cautious monitor to have risk assessments that are at least as conservative as the ideal monitor.

Ideal monitors may not be concisely representable or match other constraints that are important for monitors to be deployable to a system [31]. We assume there exists a class of admissible monitors that implement a function $\Sigma^*_{\text{obs}} \to [0,1]$ and a distance function δ comparing two monitors. The goal of this paper is to algorithmically find an admissible cautious monitor that is closest to an ideal monitor:

Problem Statement Let M^* be an ideal monitor, $\mathcal{M}$ a set of admissible monitors, $\mathcal{M}^C$ a set of cautious monitors, and $\delta\colon (\Sigma^*_{\text{obs}} \to [0,1])^2 \to \mathbb{R}$ a distance function. Find a monitor $M \in \mathcal{M} \cap \mathcal{M}^C$, s.t.:

$$M \in \operatorname*{arg\,min}_{M'\in\mathcal{M}\cap\mathcal{M}^C} \delta(M^*, M')$$

The distance function δ can be instantiated in different ways, in this paper, we use the average distance over all traces.

Key to solving the above problem is the access to the ideal monitor M^*. This, in turn, depends heavily on the ability to translate a system-level risk function to the observation level, i.e., having an accurate representation of the function *obs* (see Definition 4).

3 MONITORS FOR iHMMS

In this section, we study SuOs that are described by a (known) hidden Markov model (HMM). We provide the necessary background, define (ideal) monitors on HMMs and then discuss how to account for uncertainty about the transition probabilities in the HMM by defining monitors based on so-called *interval* HMMs.

3.1 (Interval) Hidden Markov Models

Definition 6 (Hidden Markov Model (HMM)). *An HMM is a tuple $H = (S, \iota, P, Z, O)$, where S is a finite set of states, $\iota\colon S \to [0,1]$ is an initial distribution, $P\colon S\times S \to [0,1]$ is a transition probability function, and $O\colon S \to Distr(Z)$ is a observation function, where Z is a set of observations. The transition probability function is such that for every $s \in S$, it holds that $\sum_{s'\in S} P(s,s') = 1$.*

A path in a HMM H is a finite sequence $\pi_{\mathsf{H}} = s_0 s_1 \dots s_n$, where for all $0 \le i < n$, $P(s_i, s_{i+1}) > 0$. Let Π_{H} denote the set of paths of H. Let Π^n_{H} denote the set of paths of H of length $n \in \mathbb{N}$. An observation trace of H associated with some path π_{H} is a sequence $\tau_{\mathsf{H}} = z_0 z_1 \dots z_n$, where for all $0 \le i \le n$, $O(s_i)(z_i) > 0$. Let T_{H} denote the set of observation traces. Let T^n_{H} denote the set of observation traces of length $n \in \mathbb{N}$. Note that there exists a mapping $obs_{\mathsf{H}}\colon \Pi_{\mathsf{H}} \to Distr(T_{\mathsf{H}})$ that assigns to each path a distribution over observation traces, induced by the labeling function O. A system under observation is modeled by an HMM H, if the set of states of H is defined over V_{sys}, the observation function of H maps to the distribution over V_{obs}, and the mapping obs_{H} is equivalent to *obs* for the system. Consequently, the sets of paths and traces between the system and the model coincide. Throughout the remainder of this work, we use Π and T as the sets of paths and traces of both the system and H that models it.

Definition 7 (Interval Hidden Markov Model (iHMM)). *An Interval Hidden Markov Model is a tuple $iH = (S, \iota_l, \iota_u, P_l, P_u, Z, O)$, where S, Z, and O are like in H, $\iota_l, \iota_u : S \to [0,1]$, with $\iota_l(s) \le \iota_u(s)$ for all $s \in S$, are lower and upper bounds on the initial distribution and $P_l, P_u\colon S\times S \to [0,1]$, with $P_l(s,s') \le P_u(s,s')$ for all $s, s' \in S$, are lower and upper bounds on transition probabilities. We lift notions like paths and traces from HMMs to iHMMs.*

An iHMM iH is *refined* by an HMM H if the have the same set of states and set of observations and $\iota_l(s) \le \iota(s) \le \iota_u(s)$ for all and $s \in S$ and $P_l(s,t) \le P(s,t) \le P_u(s,t)$ for all pairs of states $s, t \in S$. Let $\mathcal{H}_{\mathsf{iH}}$ be the set of all HMMs that refine a given iHMM iH.

3.2 Monitors for HMMs

A monitor over an HMM H is a function $M_{\mathsf{H}}\colon T_{\mathsf{H}} \to [0,1]$. Forward filtering [28, 33] is an implementation of such monitor, which computes the expected value for a risk function r^h_φ, conditioned on a trace: $\mathrm{R}_{\mathsf{H}}(\tau) = \mathbb{E}_\pi\left[r^h_\varphi(\pi) \mid \tau\right]$. If we choose our loss function to be the MSE, then the expected value is the optimal estimator for our chosen loss function [24], and thus the optimal monitor for H.

When H models a SuO, R_{H} corresponds to the ideal monitor for $L^{\varphi,h,\mathsf{H}}_{MSE}$. We denote this monitor by M^*_{H}.

Lemma 1. *Given an HMM H, modeling a SuO $\mathcal{S}$, a specification φ, and horizon h, the ideal monitor for $L^{\varphi,h,H}$ is M^*_H.*

3.3 Monitors for iHMMs

Monitoring iHMMs follows the same general principles as monitoring HMMs, but with an additional challenge: an iHMM iH corresponds to a set of possible HMMs, $\mathcal{H}_{\text{iH}}$. When computing the risk of a trace, we must therefore decide which $\text{H} \in \mathcal{H}_{\text{iH}}$ to use. To remain cautious, we overestimate the risk of a trace.

Definition 8 (iHMM Monitor). *Given an iHMM iH, specification φ, horizon h, observation trace τ, the iHMM monitor is defined as $M_{iH}(\tau) = \max_{H \in \mathcal{H}} \text{R}_H(\tau)$.*

Relation to cautious monitors. A safety specification φ, horizon h, and a SuO $\mathcal{S}$, which is modeled by a HMM H, induce a risk function r^h_φ on states, an ideal HMM monitor M^*_{H}, and a set of cautious monitors $\mathcal{M}^C$. Any iHMM monitor M_{iH}, where iH is refined by our system model H, is a cautious monitor.

Lemma 2. *Given a SuO which is modeled by a HMM H, any iHMM iH that is refined by H induces a cautious monitor $M_{iH} \in \mathcal{M}^C$.*

Proof. We know that $\text{H} \in \mathcal{H}_{\text{iH}}$. Thus, for every trace $\tau \in T_{\text{H}}$,

$$M_{\text{iH}}(\tau) = \max_{\text{H}' \in \mathcal{H}} \sum_{\pi \in \Pi^{|\tau|}_{\text{H}'}} Pr_{\text{H}'}(\pi \mid \tau) \cdot r^h_\varphi(\pi) \geq \sum_{\pi \in \Pi^{|\tau|}_{\text{H}}} Pr_{\text{H}}(\pi \mid \tau) \cdot r^h_\varphi(\pi) = M^*_{\text{H}}$$

satisfying the condition for $M_{\text{iH}} \in \mathcal{M}^C$. □

We show that the risk can be computed in polynomial time by model checking a maximum bounded reachability property on the iHMM. Using this, the iHMM monitoring verdicts are computable in polynomial time using the unrolling algorithm from [18].

Theorem 1. *The decision problem $M_{iH}(\tau) > \lambda$ for a threshold $\lambda \geq 0$ is computable in polynomial time.*

The proof of Theorem 1 combines the work from [6, 16, 18, 25]. We illustrate the polynomial time decision procedure in the remainder of this section on the small iHMM, shown in Fig. 2a. The iHMM has three states: the initial state s_0 with exactly observation blue, and states s_1, s_2 with exactly observation orange. The probabilistic observations of the HMM from Definition 6 can be transformed in polynomial time to exact observations as used here [11]. Any point intervals are written as single probabilities. Let the specification be $\varphi = \neg s_2$, and the horizon $h = 1$. The maximizing risk function then becomes $r = [s_0 \mapsto 0.3, s_1 \mapsto 0, s_2 \mapsto 1]$. Suppose we observe the trace $\tau = (blue)(orange)$. To compute $M_{\text{iH}}(\tau)$, we perform the transformation as shown in Fig. 2b. We unroll the model over the length of the trace, $|\tau| = 2$. This gives us the states s^j_i with j being the unrolled depth of the state. At the horizon we add transitions towards two new states $\top$, and $\bot$, with $P(s^h_i, \top) = r(s_i)$, $P(s^h_i, \bot) = 1 - r(s_i)$. In this transformed model, monitoring a trace reduces to a conditional reachability problem,

$$M_{\text{iH}}((blue)(orange)) = Pr^{\max}_{\text{iH}}(Reach(\top) \mid (blue)(orange)).$$

Now, we apply the standard transformation from iHMMs to MDPs [16]. There is a one-to-one correspondence between policies in this MDP and refined HMMs of the iHMM. [25, Proposition 2] showed that the probability measures are equal given that we allow randomized policies in the MDP. Thus, we now have an MDP mdp with randomized policies for which $Pr^{\max}_{\text{iH}}(Reach(\top) \mid (blue)(orange)) = Pr^{\max}_{\text{mdp}}(Reach(\top) \mid (blue)(orange))$. Now [6] has shown that the policy which induces the maximum conditional probability in an MDP is deterministic, and thus we can apply their translation to get a new MDP where $Pr^{\max}_{\text{mdp}}(Reach(\top) \mid (blue)(orange))$ can be computed using non-conditional reachability probability model checking. Thus allowing us to compute $M^{\max}_{\text{iH}}((blue)(orange))$.

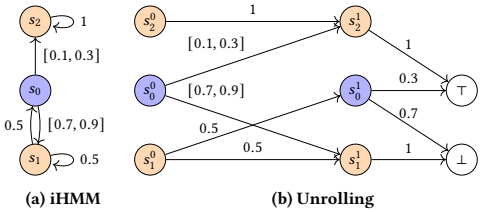


Figure 2: Transformation for monitoring an iHMM

4 LEARNING IDEAL MONITOR MODELS

Above, we derived cautious monitors from iHMMs. In this section, we iteratively learn and refine the iHMMs using paths sampled from the SuO. We learn iHMMs with an adaptation Linearly Updating Intervals [29] and then use conformance checking to decide whether the iHMMs give sufficiently precise monitors.

4.1 Learning interval HMMs

We describe a process of learning iHMMs from a dataset of paths sampled from the SuO. Our process closely follows the learning linearly updating intervals (LUI) method, applied to learning MDPs [29]. To support learning observation distributions, we encode the uncertainty about the distributions into the transitions, motivated by the transformation [11]. As in LUI for MDPs, we assume knowledge of the state and transition space of the model. We define our state space using the relevant variables (V_{sys} and further V_{obs} as explained above). We use domain knowledge (e.g., Newtonian physics) to derive the existing transitions. In contrast to [29], we don't assume a known and unique initial state.

We now outline the initialization. We initialize transition probability intervals by assigning to each transition an initial interval. For example, for a transition between states i and j, we assign $I_{i,j} = [I^{\downarrow}_{i,j}, I^{\uparrow}_{i,j}]$. We additionally introduce a strength interval $[n^{\downarrow}_{i,j}, n^{\uparrow}_{i,j}]$, which captures the minimum and maximum number of samples the current version of the probability interval is based on.

To estimate transition probabilities, we compute transition frequencies. In particular, N_i signifies the total number of times some state i appears in the dataset. $k_{i,j}$ signifies the number of times state j appears after i in the dataset. The empirical transition frequency of a given transition is hence $\frac{k_{i,j}}{N_i}$. The probability intervals are updated as follows:

$$I^{\downarrow}_{i,j} := \begin{cases} \frac{n^{\uparrow}_{i,j} \times I^{\downarrow}_{i,j} + k_{i,j}}{n^{\uparrow}_{i,j} + N_i} & \forall x. \frac{k_{i,x}}{N_i} \geq I^{\downarrow}_{i,j} \\ \frac{n^{\downarrow}_{i,j} \times I^{\downarrow}_{i,j} + k_{i,j}}{n^{\downarrow}_{i,j} + N_i} & \exists x. \frac{k_{i,x}}{N_i} < I^{\downarrow}_{i,j} \end{cases} \quad I^{\uparrow}_{i,j} := \begin{cases} \frac{n^{\uparrow}_{i,j} \times I^{\uparrow}_{i,j} + k_{i,j}}{n^{\uparrow}_{i} + N_i} & \forall x. \frac{k_{i,x}}{N_i} \leq I^{\uparrow}_{i,j} \\ \frac{n^{\downarrow}_{i,j} \times I^{\uparrow}_{i,j} + k_{i,j}}{n^{\downarrow}_{i} + N} & \exists x. \frac{k_{i,x}}{N_i} > I^{\uparrow}_{i,j} \end{cases}$$

We set all initial intervals to $[\epsilon, 1-\epsilon]$ in accordance to the rule $0.5 \geq \epsilon > 0$, and choose strength interval values in accordance to $0 \leq n^{\downarrow}_{i,j} \leq n^{\uparrow}_{i,j}$. We use a small ϵ to make no assumption about prior knowledge and low strength values to allow for the information from the dataset to make a fast impact. If all the empirical frequencies from a given state fall within the prior interval, the posterior intervals are calculated with the upper bound of the strength interval. Else, the lower bound of the strength interval is chosen instead. Using different strength values supports a more conservative reaction when new data does not conform to probability intervals derived from previous data. After updating the probability intervals, the strength intervals are updated as follows: $[n^{\downarrow}_{i,j}, n^{\uparrow}_{i,j}] := [n^{\downarrow}_{i,j} + N_i, n^{\uparrow}_{i,j} + N_i]$.

Learning the initial state probability intervals is done correspondingly, where N_i corresponds to the number of paths in the dataset and $k_{i,j}$ expresses the number of traces starting with state j.

The learning progresses iteratively with access to new data. The LUI method produces valid probability intervals, i.e., they always satisfy: $0 \leq I^{\downarrow}_{i,j} \leq I^{\uparrow}_{i,j} \leq 1$. Additionally, the method guarantees convergence to the true probability. We restate the convergence result from [29], now in the context of our adapted method.

Theorem 2. *For an iHMM with the same state space as some HMM, applying LUI to learning the transition probabilities and initial distribution of the iHMM guarantees that the intervals will converge to the exact transition probabilities when the total number of samples of HMM paths processed tends to infinity, regardless of how many samples are processed per iteration.*

Applying the LUI method in the limit will thus result in a model that induces the ideal monitor.

Corollary 1. *LUI yields a sequence of iHMMs, which induces a sequence of monitors by the pointwise application of Definition 8. That sequence converges to the ideal monitor.*

4.2 Conformance-testing-based refinement

We present a conformance-testing-based refinement framework to learning iHMMs that builds on the LUI procedure from last section. The framework involves an iterative procedure where transition intervals are updated until the iHMM based monitor provides outcomes that meet the stopping condition that we define later. We particularly address some shortcomings of LUI. The LUI method guarantees the convergence to the true HMM, but does not necessarily efficiently converge to this HMM. Empirically, the transitions from states that are frequently visited will converge to narrow intervals very fast, and the transitions from rarely visited states will stay wide for long. This will lead to a situation where the outcomes of the iHMM monitors will likely differ from the outcomes of the ideal monitor. To avoid the situation where we keep sampling and learning the neighborhoods where our transition probability approximations are already accurate, we introduce a method of conformance-testing-based refinement that guides the sampling and hence the learning process to underexplored neighborhoods. Practically, we wish to process as few samples as possible to learn an iHMM that corresponds to an iHMM monitor that is as close as possible to the ideal monitor. This is the central motivation behind the refinement framework.

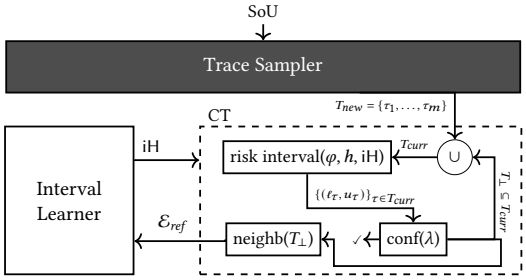


Figure 3: iHMM refinement-based learning.

The framework architecture is depicted in Fig. 3. It is composed of an *Interval Learner*, which updates a current iH using LUI, a *trace sampler*, which samples observation traces from SuO to be used for conformance testing, and a *conformance tester* that given a monitor M_{iH} of the current iHMM iH and a set of traces T, checks if a stopping condition is met. We use a stopping condition defined as a threshold θ on the average width of what we call the 'risk interval'. For each trace, the interval bounds are the minimum and maximum risk $(\ell_\tau, u_\tau) = (M^{\min}_{\text{iH}}(\tau), M^{\max}_{\text{iH}}(\tau))$, where u_τ is the outcome of the iHMM Monitor defined in Section 3.3 and ℓ_τ is the outcome of the corresponding risk minimizing monitor. If the average width of the risk interval for all considered traces is below the stopping condition threshold θ the iHMM learning terminates.

Following Fig. 3, the refinement learning starts with an initial iH and a set of sampled traces T_{new}. The initial iH has transition and initial distribution intervals set to $[\epsilon, 1-\epsilon]$. The sampled traces are forwarded to the risk interval analysis and unless the stopping condition is met, each trace that does not meet the threshold is forwarded to the neighborhood sampler.

The neighborhood sampler samples paths from the extended neighborhood of that trace τ, i.e. for a subsequence τ' of τ we sample a path π with probability $Pr(\pi \mid \tau')$. We discuss details of the sampling method in Appendix A of our Technical Report [27].

All neighborhood samples $\mathcal{E}_{ref}$ are forwarded to the Interval Learner to perform the necessary update. The traces $T_\perp \subseteq T_{new}$ for which the risk interval is above the stopping condition θ are retained for the next conformance testing round along the freshly sampled new traces making up the set $T_{curr} = T_{new} \cup T_\perp$. The learning concludes when the stopping condition is met.

Since the refinement does not interact with the method of learning intervals, but only impacts sampling, the following result holds.

Corollary 2. *Under the assumption that the learned iHMM has the same state as the true HMM and $\theta = 0$, applying the conformance-testing based refinement procedure guarantees that the intervals will converge to the exact transition probabilities when the total number of samples processed tends to infinity, regardless of how many samples are processed per iteration.*

5 EXPERIMENTAL EVALUATION

We address three research questions. First, does our model-based approach improve over model-free approaches? (5.2) Second, does using robust models enhance safety while maintaining accuracy?

(5.3) Third, does conformance-testing-based refinement lead to a better approximation of the Ideal Monitor? (5.4)

5.1 Experimental Setup

Terminology. False Negative Rate (FNR), False Positive Rate (FPR), Area Under the Curve (AUC), Stopping Condition (SC).

Implementation. We provide a prototype implementation for the framework in Fig. 3 and for monitoring over iHMMs as described in Section 3.3. The implementation of the monitoring algorithm builds on Premise [18], a model-checking-based monitoring tool, adapting it to iHMMs. The artifact is available in [1].

Benchmarks. We use benchmarks adapted from [18, 32]. The benchmarks are listed in the table below. The airport benchmarks models a scenario with an autonomous airplane descending for landing with on-ground vehicles moving across the runway. It considers the distance of the airplane to the runway, the position of the on-ground vehicles and the observed position of these vehicles. The models airportB include a larger version of the noise model used in airportA. The evadeV benchmark models the behavior of two robots, with one robot observing the position of the other. The difference between different versions of the same benchmark (ex. airport-A-7-10-10, airport-A-7-40-20) lies in how detailed the state and observation information is. SnL is a toy benchmark modeling the snakes and ladders game and is used mainly for sanity checks.

On each of the benchmarks, we monitor reachability properties, which in the case of airport and evade benchmarks means we are estimating the risk of a future collision and in the case of SnL-10x10 the risk of reaching the final state of the game.

Name	States	Transitions	$\|\tau\|$	h
SnL-10x10$^{(*)}$	101	502	15	5
evadeV-5-3	1001	3878	10	20
evadeV-6-3$^{(*)}$	2 161	8 667	12	20
airport-A-7-10-10$^{(*)}$	1 170	5 557	15	25
airport-A-7-40-20	10 760	56 577	125	35
airport-B-7-40-20$^{(*)}$	21 520	152 170	125	35

The benchmarks marked with an asterisk $^{(*)}$ have been additionally tested in a 'coarse' variant, where part of the system variables are not given to the learner. We use these variants to study the impact of a reduced state space on the accuracy of the learned monitors. Note that the convergence guarantees discussed in Theorem 2 and Corollary 2 do not hold for the coarse models.

Learning setup. We learn iHMM using our adapted LUI method described in Section 4.1 with neighborhood sampling and monitor based on these models, as described in Section 3.3. The monitors are learned from a dataset of paths of length $|\tau| + h$. During testing we use traces of length $|\tau|$ and set the monitoring horizon to h. The values of $|\tau|$ and h are specified in the benchmark table. The learned monitors are tested on 500 traces. For memory reasons, we test airport-A-7-40-20 and airport-B-7-40-20 benchmarks on 200 traces. Benchmark airportB-7-40-20 did not complete within a 12 hour timeout and its results are therefore omitted. Each method for each benchmark is executed 10 times. Unless noted otherwise, we plot averages and (shaded) areas with ±1 standard deviation. All experiments were run on an Intel Xeon Gold 6338 CPU using 16 cores and an NVIDIA A100 GPU, and using a 12h timeout. Only the conformal prediction runs on the GPU, all other computations are CPU based.

5.2 Model-based vs Model-free monitoring

In this experiment, we compare our model-based approach to two model-free monitoring approaches. We consider two model-free approaches: (1) a Stochastic Gradient Descent (SGD) regressor learned via the sklearn library [22], and (2) a conformal-prediction approach [9], that was specifically designed for monitoring under uncertainty; this approach uses a neural approach to estimating risks and conformal prediction to validate the computed risks.

The model-free methods are trained in one shot using the highest number of samples used by the refinement learning. Our comparison is done in terms of FPR, judging conservativeness, and FNR, judging safety, across different thresholds. For all benchmarks, we choose SC on the width of risk intervals based on the size of the benchmark. Choosing a lower SC for smaller benchmarks ensures that learning requires non-trivial amounts of data.

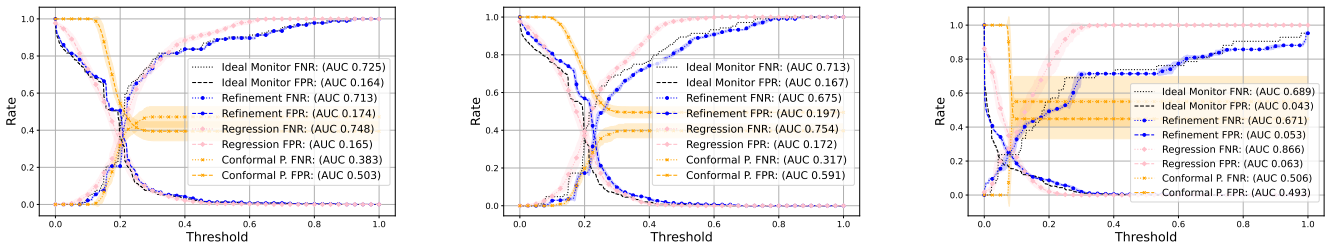
We conducted this experiment over all benchmarks. In Fig. 4, we show a representative set of our results for airportA-7-10-10 and SnL-10x10. The remaining results are found in **??**. Our goal is to create a cautious monitor that is as close to the ideal monitor M^*. In Fig. 4 the FNR and FPR of M^* are depicted by the dotted and dashed black lines, respectively. We plot the rates for increasing thresholds, where a threshold of 1 means no trace is considered unsafe. We aim at monitors with a FNR curve close and consistently below M^*'s FNR and hence with an FNR AUC slightly lower than that of M^*.

We observe the advantages of iHMM-based monitors in terms of supporting safety without being overly conservative. The diagrams clearly show that the iHMM monitors consistently provide a lower FNR than M^* across all thresholds for every benchmark, aside from Coarse SnL-10x10. For some thresholds, the iHMM monitor has a higher FNR than M^* (Fig. 4c). A FNR higher than that of M^* indicates that for some traces the learned monitor underapproximated the risk leading to the trace being misclassified as safe. Although, the iHMM monitor having at times a higher FNR than M^* in the Coarse SnL-10x10 benchmark is concerning, the alternatives perform much poorer, with regression having a far greater FNR. The FPR remains modest and closely follows the FPR of M^* (Fig. 4a). This is also observed in the other benchmarks (Appendix B.1 [27]). In general, the accuracy of the iHMM monitors is improved when considering the true state space, with Fig. 4a showing iHMM monitors that almost perfectly mimic the behavior of M^*.

Model-free monitors lack accuracy and compromise safety. The conformal prediction monitors severely underperform in terms of accuracy. We attribute this to the fact that these monitors have been designed for systems with deterministic dynamics. Regression monitors similarly struggle with accuracy. A shortcoming of regression present in all experiments is that it fails to fit the behavior of M^*. The FPR and FNR curves are overly smooth and don't fit the shape of those of M^*.

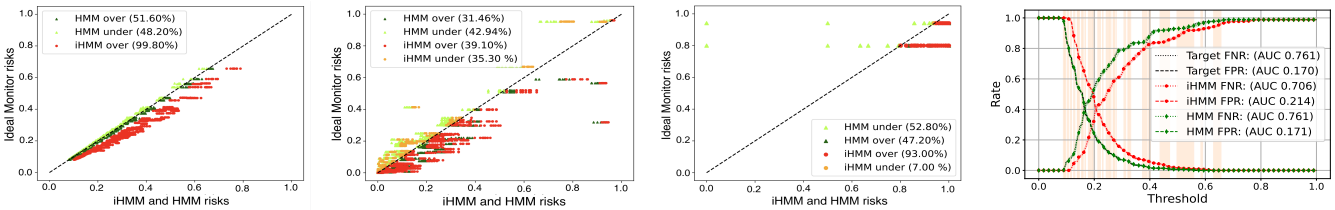
5.3 iHMM vs. HMM-based monitoring

In this section, we explore the impact of using robust models for safe monitoring. We compare the risk values approximated by the iHMM and HMM-based monitors to risk values from M^* and categorize



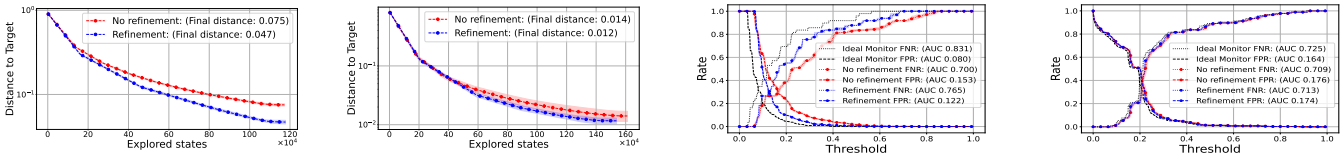
(a) airportA-7-10-10, SC = 0.001 (b) Coarse airportA-7-10-10, SC = 0.01 (c) Coarse SnL-10x10, SC = 0.01

Figure 4: Comparing FNR and FPR between model-based and model-free methods.



(a) evadeV-5-3, SC= 0.01 (b) Coarse SnL-10x10, SC= 0.01 (c) unlikely-15, SC= 0.01 (d) evadeV-5-3 SC = 0.01

Figure 5: Comparison between iHMM and HMM-based monitors, in terms of risk estimation (a-c) and FNR and FPR (d). Scatterplots (a-c) plot the values from all 10 runs for each method.



(a) Coarse evadeV-6-3, SC = 0.01 (b) airportA-7-10-10, SC = 0.001 (c) Coarse evadeV-6-3, SC= 0.01 (d) airportA-7-10-10, SC= 0.001

Figure 6: Comparing learning with and without refinement, in terms of distance to M^* (a-b), FNR and FPR (c-d)

them as under- and overapproximations. Furthermore, we consider the FPR and FNR to study the monitoring outcomes across different thresholds. We learn HMM models using the highest number of samples used by the refinement learning. We split the dataset into 10 equal batches and learn on an increasing number of batches resulting in 10 HMM models per run. The HMMs are learned using a simple frequentist approach.

For all benchmarks (Appendix B.2 [27]), we observe a positive effect of iHMM-based monitors in terms of safety. EvadeV-5-3 (Fig. 5a) is a benchmark that shows the typical performance of iHMM and HMM monitoring. The figure plots risks approximated by HMM and iHMM monitors and compares them to Ideal Monitor risks, where a single point signifies a value of the Ideal Monitor risk and either a value of the iHMM (red) or HMM risk (green). We observe a significant difference between the iHMM and HMM based monitors in terms of under- and overapproximations. As seen Fig. 5a, for almost half of the traces, the HMM-based monitor underapproximates the risk, whereas iHMM-based monitor almost exclusively overapproximates the risk, hence supporting safety. The result is existence of many thresholds for which HMM-based monitors have a higher FNR than that of M^* (orange areas in Fig. 5d). Consequently, for those thresholds, we can expect the HMM-monitor to misclassify some unsafe traces as safe. This is never the case of the iHMM-based monitor. The only benchmark for which we observe significant risk underapproximations produced by iHMM-based monitor is coarse SnL-10x10, Fig. 5b. For this benchmark, however, the iHMM-based monitor underapproximates the risk for fewer traces, than the HMM-based monitor.

The HMM-based monitors can result in stark risk underapproximations. While many traces in HMMs lead to underapproximating risk, the difference tends to be marginal. However, this is not guaranteed to always be the case. We introduce an additional benchmark, unlikely-15, which serves as an example of a HMM monitor failing to recognize high-risk scenarios. The structure of unlikely-15 is such that some of its neighborhoods are extremely unlikely to be encountered, hence the iHMM has in those neighborhoods wide probability intervals, which leads to risk overapproximations. The HMM, however, having access to very few samples, learns probabilities far from the true values, leading at times to sizable risk underapproximations. Fig. 5c plots 100 traces with positive risk, identified through rejection sampling. It includes examples of traces,

where HMM monitors misidentifies high risk traces as low risk, likely leading to violating the safety specification.

5.4 No refinement vs refinement learning

Lastly, we examine the impact of refinement-based iHMM learning on the monitoring outcomes. We consider the change in the distance to M^* and the difference between FPR and FNR. For this experiment, we further learn the iHMM models without refinement using the highest number of samples used the refinement learning. We split the dataset into 10 equal batches and learn by sequentially adding batches, resulting in 10 iHMM models per run.

Learning with refinement consistently leads to monitors that, at the end of the learning process, are closer to M^*. Note that the used distance measure is $\delta = \frac{\sum_{\tau \in T_H} |M^*(\tau) - M_{iH}(\tau)|}{|T_H|}$, where in the experimental evaluation we consider only a sampled subset of T_H. As expected, there is a relationship between the distance to M^* and the FNR and FPR values. For the benchmarks where we observe a significant impact of the refinement method in terms of distance to M^*, we observe that the refinement method results in FNR and FPR curves closer to the ideal curves (Fig. 6a, 6c). The benefit of the refinement procedure is improved sample efficiency. For example, in the case of Fig. 6a, we are able to learn as good a monitor as no refinement learning method while exploring only about two thirds of the states required by the non-refinement approach.

The refinement procedure has a varied impact across different benchmarks. In the case of the coarse evadeV-6-3 benchmark, the monitors corresponding to iHMMs learned with refinement get significantly closer to M^* than with no refinement (Fig. 6a). The effect is that the refinement monitors approximate better the ideal FNR and FPR curves (Fig. 6c). On the other hand, in the case of the airportA-7-10-10 benchmark, there is little difference between refinement and no refinement as in Fig. 6b and 6d.

We additionally considered an alternative sampling method, that showed little difference during experimental evaluation. We explain the method in Appendix A and experimentally compare the two sampling methods in Appendices A.1 and B.4 [27].

5.5 Further Remarks

Results on largest benchmarks. Learning monitors for the two biggest benchmarks, airportA-7-40-20 and airport-B-7-40-20, posed challenges. For this reason, we discuss learning monitors for these benchmarks separately. Some of the previously made conclusions hold, namely the positive effect of refinement on distance to M^*. We also observe that the iHMM, unlike the HMM-based monitors, makes safe risk approximations. Although promising, we refrain from drawing strong conclusions from these experiments, as all monitors from all learning methods produce outputs far removed from M^*. We theorize that the reasons are too high SC, and that the violating paths are significantly rarer in these benchmarks. A detailed discussion is given in Appendix C [27].

Parameter Sensitivity. We note the impact of coarse state space and low stopping condition thresholds. The experimental results highlight the importance of identifying the relevant state variables and choosing a low stopping thresholds for learning monitors that closely approximate M^*.

Computational Cost. All experiments were run with a 12h timeout, but most benchmarks required significantly less time. Taking as an example evadeV-6-3 with SC=0.1, learning an iHMM with refinement took around 110s, learning an iHMM without refinement took around 220s, learning a HMM took around 9s, training a regression model took around 10s and training a conformal prediction took around 345s, tough importantly conformal prediction is the only method trained on a GPU. The evaluated monitors differ as well in terms of execution speed. Taking again as an example evadeV-6-3 tested on traces of length 12 with a horizon set to 20, the HMM, iHMM, regression and conformal prediction monitors produce risk estimations in correspondingly: 1.8, 20, 0.003 and 1.6 milliseconds. It's worth noting that optimizing the speed of learning and executing the iHMM monitors was not the focus of this work.

6 RELATED WORK

Monitoring systems under uncertainty have been studied across many settings, from model-based approaches [2, 3, 10, 17, 18, 35] to model-free approaches [9, 30, 36]. Approaches based on learning Markov models include [3], which introduces a monitoring algorithm for the learned HMMs that relies on selecting a single most likely hidden state during monitoring, which compared to considering a distribution over possible states, lacks the precision for accurate monitoring under uncertainty. This is expanded on in [2] to include importance sampling (IS) to learn rare events more efficiently, similarly to our refinement procedure. However, their IS is based on using a modified simulation in which the rare events occur with a known higher likelihood and is thus not applicable in our setting. An example of model-free approaches can be taken from [9], which uses a neural approach for state estimation and conformal prediction on top for risk estimation. A bottleneck of this approach is that it is designed for monitoring systems with deterministic dynamics, which stands in contrast to ours. The literature also includes several works addressing the problem of learning Markov models from data. These include active learning method for learning Markov Decision Processes [4] or an approach for learning Hidden Markov Models [3]. Both of these are based on Baum-Welch algorithm and hence highly sensitive to the training priors used. Related is also the idea of shielding in reinforcement learning, which has been applied to partially observable systems [12, 21]. In contrast, however, shielding requires to (pre-)compute partial-information schedulers, and foremost the availability of models. Several works have been also dedicated to runtime assurance (enforcement), where the question on how to use the monitors verdict to further change the behavior of a system [15, 23, 26, 34]. Our focus was on monitoring, and our approach can be integrated into these approaches for a complete assurance pipeline.

7 CONCLUSION

We presented a framework for learning robust Markov model-based monitors. Our methods come with guarantees on convergence to an Ideal Monitor. As shown experimentally, the learned monitors outperform the alternative methods in terms of accurate and cautious monitoring, especially compared to model-free approaches.

ACKNOWLEDGMENTS

This work was partly supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation, and by the NWO grant FuRoRe (OCENW.M.22.282). The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

REFERENCES

[1] Skurka Antonina, Luko van der Maas, Sebastian Junges, and Hazem Torfah. 2026. *Artifact for "Learning Robust Markov Models for Safe Runtime Monitoring"*. https://doi.org/10.5281/zenodo.18631204

[2] Reza Babaee, Vijay Ganesh, and Sean Sedwards. 2019. Accelerated Learning of Predictive Runtime Monitors for Rare Failure. In *RV (Lecture Notes in Computer Science, Vol. 11757)*. Springer, 111–128.

[3] Reza Babaee, Arie Gurfinkel, and Sebastian Fischmeister. 2018. Prevent: a Predictive Run-time Verification Framework Using Statistical Learning. *16th International Conference on Software Engineering and Formal Methods*, 205–220.

[4] Giovanni Bacci, Anna Ingolfsdottir, Kim G. Larsen, and Raphaël Reynouard. 2021. Active Learning of Markov Decision Processes using Baum-Welch algorithm. In *Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021 (Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021)*, M. Arif Wani, Ishwar K. Sethi, Weisong Shi, Guangzhi Qu, Daniela Stan Raicu, and Ruoming Jin (Eds.). IEEE (Institute of Electrical and Electronics Engineers), United States, 1203–1208. https://doi.org/10.1109/ICMLA52953.2021.00195

[5] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press.

[6] Christel Baier, Joachim Klein, Sascha Klüppelholz, and Steffen Märcker. 2014. Computing Conditional Probabilities in Markovian Models Efficiently. In *TACAS (Lecture Notes in Computer Science, Vol. 8413)*. Springer, 515–530.

[7] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. 2018. *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. Springer International Publishing, Cham, 135–175. https://doi.org/10.1007/978-3-319-75632-5_5

[8] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. 2017. Safe model-based reinforcement learning with stability guarantees. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 908–919.

[9] Francesca Cairoli, Luca Bortolussi, and Nicola Paoletti. 2021. Neural predictive monitoring under partial observability. In *International Conference on Runtime Verification*. Springer, 121–141.

[10] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2021. Runtime Equilibrium Verification for Resilient Cyber-Physical Systems. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 71–80. https://doi.org/10.1109/ACSOS52086.2021.00025

[11] Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. 2015. Optimal Cost Almost-Sure Reachability in POMDPs. In *AAAI*. AAAI Press, 3496–3502.

[12] Krishnendu Chatterjee, Petr Novotnỳ, Guillermo Pérez, Jean-François Raskin, and Đorđe Žikelić. 2017. Optimizing expectation with guarantees in POMDPs. In *AAAI*. 3725–3732.

[13] David Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, Alessandro Abate, Joe Halpern, Clark W. Barrett, Ding Zhao, Tan Zhi-Xuan, Jeannette M. Wing, and Joshua B. Tenenbaum. 2024. Towards Guaranteed Safe AI: A Framework for Ensuring Robust and Reliable AI Systems. *CoRR* abs/2405.06624 (2024).

[14] Giuseppe De Giacomo and Moshe Y. Vardi. 2013. Linear temporal logic and linear dynamic logic on finite traces *(IJCAI '13)*. AAAI Press, 854–860.

[15] Ankush Desai, Shromona Ghosh, Sanjit A. Seshia, Natarajan Shankar, and Ashish Tiwari. 2019. SOTER: A Runtime Assurance Framework for Programming Safe Robotics Systems. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 138–150. https://doi.org/10.1109/DSN.2019.00027

[16] Garud N. Iyengar. 2005. Robust Dynamic Programming. *Math. Oper. Res.* 30, 2 (2005), 257–280.

[17] Sebastian Junges, Sanjit A. Seshia, and Hazem Torfah. 2024. Active Learning of Runtime Monitors Under Uncertainty. In *Integrated Formal Methods - 19th International Conference, IFM 2024, Manchester, UK, November 13-15, 2024, Proceedings (Lecture Notes in Computer Science, Vol. 15234)*, Nikolai Kosmatov and Laura Kovács (Eds.). Springer, 297–306. https://doi.org/10.1007/978-3-031-76554-4_18

[18] Sebastian Junges, Hazem Torfah, and Sanjit A. Seshia. 2021. Runtime Monitors for Markov Decision Processes. In *CAV (2) (Lecture Notes in Computer Science, Vol. 12760)*. Springer, 553–576.

[19] Stefan Mitsch and André Platzer. 2016. ModelPlex: verified runtime validation of verified cyber-physical system models. *Formal Methods Syst. Des.* 49, 1-2 (2016), 33–74. https://doi.org/10.1007/s10703-016-0241-z

[20] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. 2023. Model-based Reinforcement Learning: A Survey. *Found. Trends Mach. Learn.* 16, 1 (2023), 1–118.

[21] Wonhong Nam and Rajeev Alur. 2010. Active Learning of Plans for Safety and Reachability Goals With Partial Observability. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 40, 2 (2010), 412–420. https://doi.org/10.1109/TSMCB.2009.2025657

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[23] Dung Phan, Junxing Yang, Matthew Clark, Radu Grosu, John Schierman, Scott Smolka, and Scott Stoller. 2017. A Component-Based Simplex Architecture for High-Assurance Cyber-Physical Systems . In *2017 17th International Conference on Application of Concurrency to System Design (ACSD)*. IEEE Computer Society, Los Alamitos, CA, USA, 49–58. https://doi.org/10.1109/ACSD.2017.23

[24] Hossein Pishro-Nik. 2014. *Introduction to probability, statistics, and random processes*. Kappa Research, LLC Blue Bell, PA, USA.

[25] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2006. Model-Checking Markov Chains in the Presence of Uncertainties. In *TACAS (Lecture Notes in Computer Science, Vol. 3920)*. Springer, 394–410.

[26] Lui Sha. 2001. Using simplicity to control complexity. *IEEE Software* 18, 4 (July 2001), 20–28. https://doi.org/10.1109/MS.2001.936213

[27] Antonina Skurka, Luko van der Maas, Sebastian Junges, and Hazem Torfah. 2026. Learning Robust Markov Models for Safe Runtime Monitoring. arXiv:2602.14987 [cs.LO] https://arxiv.org/abs/2602.14987

[28] Scott D. Stoller, Ezio Bartocci, Justin Seyster, Radu Grosu, Klaus Havelund, Scott A. Smolka, and Erez Zadok. 2011. Runtime Verification with State Estimation. In *RV (Lecture Notes in Computer Science, Vol. 7186)*. Springer, 193–207.

[29] Marnix Suilen, Thiago D Simão, David Parker, and Nils Jansen. 2022. Robust anytime learning of Markov decision processes. *Advances in Neural Information Processing Systems* 35 (2022), 28790–28802.

[30] Hazem Torfah, Aniruddha R. Joshi, Shetal Shah, S. Akshay, Supratik Chakraborty, and Sanjit A. Seshia. 2023. Learning Monitor Ensembles for Operational Design Domains. In *23rd International Conference on Runtime Verification (RV) (Lecture Notes in Computer Science, Vol. 14245)*, Panagiotis Katsaros and Laura Nenzi (Eds.). Springer, 271–290.

[31] Hazem Torfah, Carol Xie, Sebastian Junges, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. 2022. Learning Monitorable Operational Design Domains for Assured Autonomy. Springer-Verlag, Berlin, Heidelberg, 3–22. https://doi.org/10.1007/978-3-031-19992-9_1

[32] Luko van der Maas and Sebastian Junges. 2025. Learning Verified Monitors for Hidden Markov Models. *CoRR* abs/2504.05963 (2025).

[33] Cristina M. Wilcox and Brian C. Williams. 2010. Runtime Verification of Stochastic, Faulty Systems. In *RV (Lecture Notes in Computer Science, Vol. 6418)*. Springer, 452–459.

[34] Beyazit Yalcinkaya, Hazem Torfah, Ankush Desai, and Sanjit A. Seshia. 2023. Ulgen: A Runtime Assurance Framework for Programming Safe Cyber-Physical Systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 42, 11 (2023), 3679–3692. https://doi.org/10.1109/TCAD.2023.3246386

[35] Hansol Yoon, Yi Chou, Xin Chen, Eric W. Frew, and Sriram Sankaranarayanan. 2019. Predictive Runtime Monitoring for Linear Stochastic Systems and Applications to Geofence Enforcement for UAVs. In *RV (Lecture Notes in Computer Science, Vol. 11757)*. Springer, 349–367.

[36] Yiqi Zhao, Bardh Hoxha, Georgios Fainekos, Jyotirmoy V. Deshmukh, and Lars Lindemann. 2024. Robust Conformal Prediction for STL Runtime Verification under Distribution Shift. In *15th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2024, Hong Kong, May 13-16, 2024*. IEEE, 169–179. https://doi.org/10.1109/ICCPS61052.2024.00022