Runtime Monitors for Markov Decision Processes

Sebastian Junges(✉), Hazem Torfah, and Sanjit A. Seshia

University of California at Berkeley, Berkeley, USA
sjunges@berkeley.edu

Abstract. We investigate the problem of monitoring partially observable systems with nondeterministic and probabilistic dynamics. In such systems, every state may be associated with a risk, e.g., the probability of an imminent crash. During runtime, we obtain partial information about the system state in form of observations. The monitor uses this information to estimate the risk of the (unobservable) current system state. Our results are threefold. First, we show that extensions of state estimation approaches do not scale due the combination of nondeterminism and probabilities. While exploiting a geometric interpretation of the state estimates improves the practical runtime, this cannot prevent an exponential memory blowup. Second, we present a tractable algorithm based on model checking conditional reachability probabilities. Third, we provide prototypical implementations and manifest the applicability of our algorithms to a range of benchmarks. The results highlight the possibilities and boundaries of our novel algorithms.

1 Introduction

Runtime assurance is essential in the deployment of safety-critical (cyber-physical) systems [12,29,45,49,50]. Monitors observe system behavior and indicate when the system is at risk to violate system specifications. A critical aspect in developing reliable monitors is their ability to handle noisy or missing data. In cyber-physical systems, monitors observe the system state via sensors, i.e., sensors are an interface between the system and the monitor. A monitor has to base its decision solely on the obtained sensor output. These sensors are not perfect, and not every aspect of a system state can be measured.

This paper considers a model-based approach to the construction of monitors for systems with imprecise sensors. Consider Fig. 1(b). We assume a model for the environment together with the controller. Typically, such a model contains both nondeterministic and probabilistic behavior, and thus describes a Markov decision process (MDP): In particular, the sensor is a stochastic process [56] that

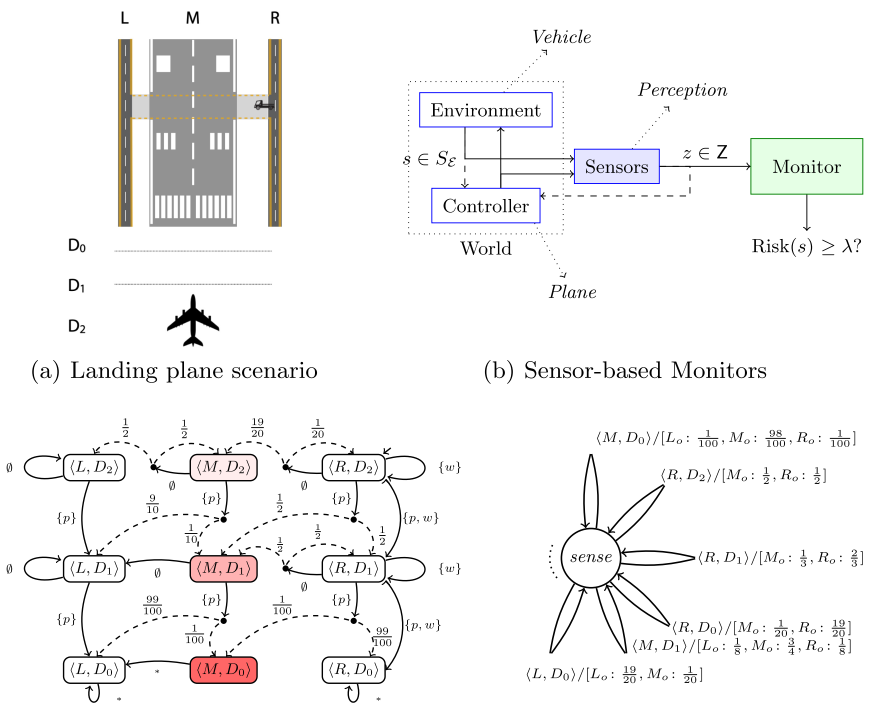
This work is partially supported by NSF grants 1545126 (VeHICaL), 1646208 and 1837132, by the DARPA contracts FA8750-18-C-0101 (AA) and FA8750-20-C-0156 (SDCPS), by Berkeley Deep Drive, and by Toyota under the iCyPhy center.

translates the environment state into an observation. For example, this could be a perception module on a plane that during landing estimates the movements of an on-ground vehicle, as depicted in Fig. 1(a). Due to lack of precise data, the vehicle movements itself may be most accurately described using nondeterminism.

We are interested in the associated *state risk* of the current system state. The state risk may encode, e.g., the probability that the plane will crash with the vehicle within a given number of steps, or the expected time until reaching the other side of the runway. The challenge is that the monitor cannot directly observe the current system state. Instead, the monitor must infer from a trace of observations the current state risk. This cannot be done perfectly as the system state cannot be inferred precisely. Rather, we want a sound, conservative estimate of the system state. More concretely, for a fixed resolution of the nondeterminism, the *trace risk* is the weighted sum over the probability of being in a state having observed the trace, times the risk imposed by this state. The monitoring problem is to decide whether for any possible scheduler resolving the nondeterminism the trace risk of a given trace exceeds a threshold.

Monitoring of systems that contain either only probabilistic or only nondeterministic behavior is typically based on *filtering*. Intuitively, the monitor then estimates the current system states based on the model. For purely nondeterministic systems (without probabilities) a set of states needs to be tracked, and purely probabilistic systems (without nondeterminism) require tracking a distribution over states. This tracking is rather efficient. For systems that contain both probabilistic and nondeterministic behavior, filtering is more challenging. In particular, we show that filtering on MDPs results in an exponential memory blowup as the monitor must track sets of distributions. We show that a reduction based on the geometric interpretation of these distributions is essential for practical performance, but cannot avoid the worst-case exponential blowup. As a tractable alternative to filtering, we rephrase the monitoring problem as the computation of conditional reachability probabilities [9]. More precisely, we unroll and transform the given MDP, and then model check this MDP. This alternative approach yields a polynomial-time algorithm. Indeed, our experiments show the feasibility of computing the risk by computing conditional probabilities. We also show benchmarks on which filtering is a competitive option.

Contribution and Outline. This paper presents the first runtime monitoring for systems that can be adequately abstracted by a combination of *probabilities and nondeterminism* and where the system state is *partially observable*. We describe the use case, show that typical filtering approaches in general fail to deal with this setting, and show that a tractable alternative solution exists. In Sect. 3, we investigate *forward filtering*, used to estimate the possible system states in partially observable settings. We show that this approach is tractable for systems that have probabilistic *or* nondeterministic uncertainty, but not for systems that have both. To alleviate the blowup, Sect. 4 discusses an (often) efficacious pruning strategy and its limitations. In Sect. 5 we consider model checking as a more tractable alternative. This result utilizes constructions from the analysis


L
M
R
D0
D1
D2
Vehicle
Environment
Perception
$s \in S_{\mathcal{E}}$
Sensors
$z \in Z$
Monitor
Controller
World
Plane
Risk(s) $\geq \lambda$?
$\langle L, D_2\rangle$
$\langle M, D_2\rangle$
$\langle R, D_2\rangle$
$\langle L, D_1\rangle$
$\langle M, D_1\rangle$
$\langle R, D_1\rangle$
$\langle L, D_0\rangle$
$\langle M, D_0\rangle$
$\langle R, D_0\rangle$
$\{p\}$
$\{w\}$
$\{p, w\}$
$\emptyset$
sense
$\langle M, D_0\rangle/[L_o\colon \frac{1}{100}, M_o\colon \frac{98}{100}, R_o\colon \frac{1}{100}]$
$\langle R, D_2\rangle/[M_o\colon \frac{1}{2}, R_o\colon \frac{1}{2}]$
$\langle R, D_1\rangle/[M_o\colon \frac{1}{3}, R_o\colon \frac{2}{3}]$
$\langle R, D_0\rangle/[M_o\colon \frac{1}{20}, R_o\colon \frac{19}{20}]$
$\langle M, D_1\rangle/[L_o\colon \frac{1}{8}, M_o\colon \frac{3}{4}, R_o\colon \frac{1}{8}]$
$\langle L, D_0\rangle/[L_o\colon \frac{19}{20}, M_o\colon \frac{1}{20}]$

(a) Landing plane scenario (b) Sensor-based Monitors

(c) World Model (d) Partial Sensor Model

Fig. 1. A probabilistic world and sensor model represented by two MDPs for the scenario of an airplane in landing approach with on-ground vehicle movements.

of *partially observable MDPs* and model checking MDPs with *conditional properties*. In Sect. 6 we present baseline implementations of these algorithms, on top of the open-source model checker STORM, and evaluate their performance. The results show that the implementation allows for monitoring of a variety of MDPs, and reveals both strengths and weaknesses of both algorithms. We start with a motivating example and review related work at the end of the paper.

Motivating Example. Consider a scenario where an autonomous airplane is in its final approach, i.e., lined up with a designated runway and descending for landing, see Fig. 1(a). On the ground, close to the runway, maintenance vehicles may cross the runway. The airplane tracks the movements of these vehicles and has to decide, depending on the movements of the vehicles, whether to abort the landing. To simplify matters, assume that the airplane (P) is tracking the movement of one vehicle (V) that is about to cross the runway. Let us further assume that P tracks V using a perception module that can only determine the position of the vehicle with a certain accuracy [33], i.e., for every position of V, the perception module reports a noisy variant of the position of V. However, it is important to know that the plane obtains a sequence of these measurements.

Figure 1 illustrates the dynamics of the scenario. The world model describing the movements of V and P is given in Fig. 1(c), where D_2, D_1, and D_0 define how close P is to the runway, and R, M, and L define the position of V. Depending on what information V perceives about P, given by the atomic proposition $\{(p)rogress\}$, and what commands it receives $\{(w)ait\}$, it may or may not cross the runway. The perception module receives the information about the state of the world and reports with a certain accuracy (given as a probability) the position of V. The (simple) model of the perception module is given in Fig. 1(d). For example, if P is in zone D_2 and V is in R then there is high chance that the perception module returns that V is on the runway. The probability of incorrectly detecting V's position reduces significantly when P is in D_0.

A monitor responsible for making the decision to land or to perform a go-around based on the information computed by the perception module, must take into consideration the accuracy of this returned information. For example, if the sequence of sensor readings passed to the monitor is the sequence $\tau = R_o{\cdot}R_o{\cdot}M_o$, and each state is mapped to a certain risk, then how risky is it to land after seeing τ? If, for instance, the world is with high probability in state $\langle M, D_0\rangle$, a very risky state, then the plane should go around. In the paper, we address the question of computing the risk based on this observation sequence. We will use this example as our running example.

2 Monitoring with Imprecise Sensors

In this section, we formalize the problem of monitoring with imprecise sensors when both the world and sensor models are given by MDPs. We start with a recap of MDPs, define the monitoring problem for MDPs, and finally show how the dynamics of the system under inspection can be modeled by an MDP defined by the composition of two MDPs of the sensors and world model of the system.

2.1 Markov Decision Processes

For a countable set X, let $\mathsf{Distr}(X) \subset (X \to [0,1])$ define the set of all distributions over X, i.e., for $d \in \mathsf{Distr}(X)$ it holds that $\Sigma_{x\in X} d(x) = 1$. For $d \in \mathsf{Distr}(X)$, let the *support* of d be defined by $\mathsf{supp}(d) := \{x \mid d(x) > 0\}$. We call a distribution d *Dirac*, if $|\mathsf{supp}(d)| = 1$.

Definition 1 (Markov decision process). *A* Markov decision process *is a tuple* $\mathcal{M} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}\rangle$*, where* S *is a finite set of* states*,* $\iota \in \mathsf{Distr}(S)$ *is an* initial distribution*,* Act *is a finite set of* actions*,* $P\colon S \times \mathsf{Act} \to \mathsf{Distr}(S)$ *is a* partial transition function*,* Z *is a finite set of* observations*, and* $\mathsf{obs}\colon S \to \mathsf{Distr}(\mathsf{Z})$ *is a* observation function*.*

Remark 1. The observation function can also be defined as a state-action observation function $\mathsf{obs}\colon S \times \mathsf{Act} \to \mathsf{Distr}(\mathsf{Z})$. MDPs with state-action observation function can be easily transformed into equivalent MDPs with a state observation function using auxiliary states [19]. Throughout the paper we use state-action observations to keep (sensor) models concise.

For a state $s \in S$, we define $\mathsf{AvAct}(s) = \{\alpha \mid P(s,\alpha) \neq \bot\}$. W.l.o.g., $|\mathsf{AvAct}(s)| \geq 1$. If all distributions in $\mathcal{M}$ are Dirac, we refer to $\mathcal{M}$ as a *Kripke structure* (KS). If $|\mathsf{AvAct}(s)| = 1$ for all $s \in S$, we refer to $\mathcal{M}$ as a *Markov chain* (MC). When $\mathsf{Z} = S$, we refer to $\mathcal{M}$ as *fully observable* and omit Z and obs from its definition. A *finite path* in an MDP $\mathcal{M}$ is a sequence $\pi = s_0 a_0 s_1 \dots s_n \in S \times \big(\mathsf{Act} \times S\big)^*$ such that for every $0 \leq i < n$ it holds that $P(s_i, a_i)(s_{i+1}) > 0$ and $\iota(s_0) > 0$. We denote the set of finite paths of $\mathcal{M}$ by $\varPi_{\mathcal{M}}$. The *length* of the path is given by the number of actions along the path. The set $\varPi^n_{\mathcal{M}}$ for some $n \in \mathbb{N}$ denotes the set of finite paths of length n. We use $\pi_\downarrow$ to denote the last state in π. We omit $\mathcal{M}$ whenever it is clear from the context. A *trace* is a sequence of observations $\tau = z_0 \dots z_n \in \mathsf{Z}^+$. Every path induces a distribution over traces.

As standard, any nondeterminism is resolved by means of a scheduler.

Definition 2 (Scheduler). *A* scheduler *for an MDP $\mathcal{M}$ is a function $\sigma\colon \varPi_{\mathcal{M}} \to \mathsf{Distr}(\mathsf{Act})$ with $\mathsf{supp}(\sigma(\pi)) \subseteq \mathsf{AvAct}(\pi_\downarrow)$ for every $\pi \in \varPi_{\mathcal{M}}$.*

We use $\mathit{Sched}(\mathcal{M})$ to denote the set of schedulers. For a fixed scheduler $\sigma \in \mathit{Sched}(\mathcal{M})$, the probability $\mathsf{Pr}_\sigma(\pi)$ of a path π (under the scheduler σ) is the product of the transition probabilities in the induced Markov chain. For more details we refer the reader to [8].

2.2 Formal Problem Statement

Our goal is to determine the risk that a system is exposed to having observed a trace $\tau \in \mathsf{Z}^+$. Let $r\colon S \to \mathbb{R}_{\geq 0}$ map states in $\mathcal{M}$ to some risk in $\mathbb{R}_{\geq 0}$. We call r a *state-risk function* for $\mathcal{M}$. This function maps to the risk that is associated with being in every state. For example, in our experiments, we flexibly define the state risk using the (expected reward extension of the) temporal logic PCTL [8], to define the probability of reaching a fail state. For example, we can define risk as the probability to crash within H steps. The use of expected rewards allows for even more flexible definitions.

Intuitively, to compute this risk of the system we need to determine the current system state having observed τ, considering both the probabilistic and nondeterministic context. To this end, we formalize the (conditional) probabilities and risks of paths and traces. Let $\mathsf{Pr}_\sigma(\pi \mid \tau)$ define the probability of a path π, under a scheduler σ, having observed τ. Since a scheduler may define many paths that induce the observation trace τ, we are interested in the weighted risk over all paths, i.e., $\sum_{\pi \in \varPi^{|\tau|}_{\mathcal{M}}} \mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow)$. The monitoring problem for MDPs then conservatively over-approximates the risk of a trace by assuming an adversarial scheduler, that is, by taking the supremum risk estimate over all schedulers[1].

[1] We later see in Lemma 8 that this is indeed a maximum.

The Monitoring Problem. Given an MDP $\mathcal{M}$, a state-risk $r\colon S \to \mathbb{R}_{\geq 0}$, an observation trace $\tau \in \mathsf{Z}^+$, and a threshold $\lambda \in [0, \infty)$, decide $R_r(\tau) > \lambda$, where the *weighted risk function* $R_r\colon \mathsf{Z}^+ \to \mathbb{R}_{\geq 0}$ is defined as

$$R_r(\tau) \quad := \quad \sup_{\sigma \in Sched(\mathcal{M})} \sum_{\pi \in \Pi_{\mathcal{M}}^{|\tau|}} \mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow).$$

The conditional probability $\mathsf{Pr}_\sigma(\pi \mid \tau)$ can be characterized using Bayes' rule[2]:

$$\mathsf{Pr}_\sigma(\pi \mid \tau) = \frac{\mathsf{Pr}(\tau \mid \pi) \cdot \mathsf{Pr}_\sigma(\pi)}{\mathsf{Pr}_\sigma(\tau)}.$$

The probability $\mathsf{Pr}(\tau \mid \pi)$ of a trace τ for a fixed path π is $\mathsf{obs}_{\mathsf{tr}}(\pi)(\tau)$, where

$$\mathsf{obs}_{\mathsf{tr}}(s) := \mathsf{obs}(s), \quad \mathsf{obs}_{\mathsf{tr}}(\pi \alpha s') := \{\tau \cdot z \mapsto \mathsf{obs}_{\mathsf{tr}}(\pi)(\tau) \cdot \mathsf{obs}(s')(z)\},$$

when $|\pi| = |\tau|$, and $\mathsf{obs}_{\mathsf{tr}}(\pi)(\tau) = 0$ otherwise. The probability $\mathsf{Pr}_\sigma(\tau)$ of a trace τ is $\sum_\pi \mathsf{Pr}_\sigma(\pi) \cdot \mathsf{Pr}(\tau \mid \pi)$.

We call the special variant with $\lambda = 0$ the *qualitative monitoring problem.* The problems are (almost) equivalent on Kripke structures, where considering a single path to an adequate state suffices. Details are given in [36, Appendix].

Lemma 1. *For Kripke structures the monitoring and qualitative monitoring problems are logspace interreducible.*

In the next sections we present two types of algorithms for the monitoring problem. The first algorithm is based on the widespread (forward) filtering approach [44]. The second is new algorithm based on model checking conditional probabilities. While filtering approaches are efficacious in a purely nondeterministic or a purely probabilistic setting, it does not scale on models such as MDPs that are both probabilistic and nondeterministic. In those models, model checking provides a tractable alternative. Before going into details, we first connect the problem statement more formally to our motivating example.

2.3 An MDP Defining the System Dynamics

We show how the weighted risk for a system given by a world and sensor model can be formalized as a monitoring problem for MDPs. To this end, we define the dynamics of the world and sensors that we use as basis for our monitor as the following joint MDP.

For a fully observable world MDP $\mathcal{E} = \langle S_\mathcal{E}, \iota_\mathcal{E}, \mathsf{Act}_\mathcal{E}, P_\mathcal{E} \rangle$ and a sensor MDP $\mathcal{S} = \langle S_\mathcal{S}, \iota_\mathcal{S}, S_\mathcal{E}, P_\mathcal{S}, \mathsf{Z}, \mathsf{obs} \rangle$, where obs is state-action based, the *inspected system* is defined by an MDP $[\![\langle \mathcal{E}, \mathcal{S} \rangle]\!] = \langle S_\mathcal{J}, \iota_\mathcal{J}, \mathsf{Act}_\mathcal{E}, P_\mathcal{J}, \mathsf{Z}, \mathsf{obs}_\mathcal{J} \rangle$ being the synchronous composition of $\mathcal{E}$ and $\mathcal{S}$:

[2] For conciseness we assume throughout the paper that $\frac{0}{0} = 0$.

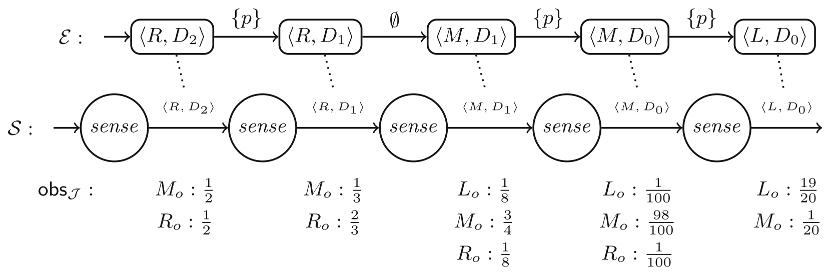


Fig. 2. A run with its observations of the inspected system $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!]$ where $\mathcal{E}$ and $\mathcal{S}$ are the models given in Fig. 1.

- $S_{\mathcal{J}} := S_{\mathcal{E}} \times S_{\mathcal{S}}$,
- $\iota_{\mathcal{J}}$ is defined as $\iota_{\mathcal{J}}(\langle u, s\rangle) := \iota_{\mathcal{E}}(u) \cdot \iota_{\mathcal{S}}(s)$ for each $u \in S_{\mathcal{E}}$ and $s \in S_{\mathcal{S}}$,
- $P_{\mathcal{J}} \colon S_{\mathcal{J}} \times \mathsf{Act}_{\mathcal{E}} \to \mathsf{Distr}(S_{\mathcal{J}})$ such that for all $\langle u, s\rangle \in S_{\mathcal{J}}$ and $\alpha \in \mathsf{Act}_{\mathcal{E}}$;

$$P_{\mathcal{J}}(\langle u, s\rangle, \alpha) = d_{u,s} \in \mathsf{Distr}(S_{\mathcal{J}}),$$

  where for all $u' \in S_{\mathcal{E}}$ and $s' \in S_{\mathcal{S}}$: $d_{u,s}(\langle u', s'\rangle) = P_{\mathcal{E}}(u, \alpha)(u') \cdot P_{\mathcal{S}}(s, u)(s')$,
- $\mathsf{obs}_{\mathcal{J}} \colon S_{\mathcal{J}} \to \mathsf{Distr}(\mathsf{Z})$ with $\mathsf{obs}_{\mathcal{J}} \colon \langle u, s\rangle \mapsto \mathsf{obs}(s, u)$.

In Fig. 2 we illustrate a run of $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!]$ for the world and sensor MDPs presented in Fig. 1. We particularly show the observations of the joint MDP given by the distributions over the observations for each transition in the run (we omitted the probabilistic transitions for simplicity). The observations of the MDP $\mathcal{M}$ present the output of the sensor upon a path through $\mathcal{M}$. These observations in turn are the inputs to a monitor on top of the system. The role of the monitor is then to compute the risk of being in a critical state based on the received observations.

3 Forward Filtering for State Estimation

We start by showing why standard forward filtering does not scale well on MDPs. We briefly show how filtering can be used to solve the monitoring problem for purely nondeterministic systems (Kripke structures) or purely probabilistic systems (Markov Chains). Then, we show why for MDPs, the forward filtering needs to manage, although finite but an exponential set of distributions. In Sect. 4 we present a new improved variant of forward filtering for MDPs based on filtering with vertices of the convex hull. In Sect. 5 we present a new polynomial-time model checking-based algorithm for solving the problem.

3.1 State Estimators for Kripke Structures.

For Kripke structures, we maintain a set of possible states that agree with the observed trace. This set of states is inductively characterized by the function

$\mathsf{est}_{\mathsf{KS}} \colon \mathsf{Z}^+ \to 2^S$ which we define formally below. For an observation trace τ, $\mathsf{est}_{\mathsf{KS}}(\tau)$ defines the set of states that can be reached with positive probability. This set can be computed by a forward state traversal [31]. To illustrate how $\mathsf{est}_{\mathsf{KS}}(\tau)$ is computed for τ, consider the underlying Kripke structure of the inspected system $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!]$ for our running example in Fig. 1 (to make this a Kripke structure, we remove the probabilities). Consider further the observation trace $\tau = R_o \cdot M_o \cdot L_o$. Since $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!]$ has only one initial state $\langle\langle R, D_2\rangle, \mathit{sense}\rangle$ and R_o is observable with a positive probability in this state, $\mathsf{est}_{\mathsf{KS}}(R_o) = \{\langle\langle R, D_2\rangle, \mathit{sense}\rangle\}$. As M_o is observed next, $\mathsf{est}_{\mathsf{KS}}(R_o \cdot M_o)$ computes the states reached from $\langle\langle R, D_2\rangle, \mathit{sense}\rangle$ and where M_o can be observed with a positive probability, i.e., $\mathsf{est}_{\mathsf{KS}}(R_o \cdot M_o) = \{\langle\langle R, D_1\rangle, \mathit{sense}\rangle, \langle\langle R, M_1\rangle, \mathit{sense}\rangle\}$. Finally, the current state having observed $R_o \cdot M_o \cdot L_o$ may be one of the states $\mathsf{est}_{\mathsf{KS}}(\tau) = \{\langle\langle M, D_1\rangle, \mathit{sense}\rangle,\ \langle\langle L, D_1\rangle, \mathit{sense}\rangle,\ \langle\langle L, D_0\rangle, \mathit{sense}\rangle,\ \langle\langle M, D_0\rangle, \mathit{sense}\rangle\}$, which especially shows that we might be in the high-risk world state $\langle M, D_0\rangle$.

Definition 3 (KS state estimator). *For $\mathsf{KS} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}\rangle$, the state estimation function $\mathsf{est}_{\mathsf{KS}} \colon \mathsf{Z}^+ \to 2^S$ is defined as*

$$\mathsf{est}_{\mathsf{KS}}(z) := \{s \in S \mid \iota(s) > 0 \wedge \mathsf{obs}(s)(z) > 0\}$$

$$\mathsf{est}_{\mathsf{KS}}(\tau \cdot z) := \Big\{s' \in S \mid \exists s \in \mathsf{est}_{\mathsf{KS}}(\tau), \exists \alpha \in \mathsf{Act}, P(s,\alpha)(s') > 0 \wedge \mathsf{obs}(s')(z) > 0\Big\}.$$

For a Kripke structure KS and a given trace τ, the monitoring problem can be solved by computing $\mathsf{est}_{\mathsf{KS}}(\tau)$, using [31] and Lemma 1.

Lemma 2. *For a Kripke stucture $\mathsf{KS} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}\rangle$, a trace $\tau \in \mathsf{Z}^+$, and a state-risk function $r \colon S \to \mathbb{R}_{\geq 0}$, it holds that $R_r(\tau) = \max_{s \in \mathsf{est}_{\mathsf{KS}}(\tau)} r(s)$. Computing $R_r(\tau)$ requires time $\mathcal{O}(|\tau| \cdot |P|)$ and space $\mathcal{O}(|S|)$.*

A proof can be found in [36, Appendix]. The time and space requirements follow directly from the inductive definition of $\mathsf{est}_{\mathsf{KS}}$ which resembles solving a forward state traversal problem in automata [31]. In particular, the algorithm allows updating the result after extending τ in $\mathcal{O}(|P|)$.

3.2 State Estimators for Markov Chains

For Markov chains, in addition to tracking the potential reachable system states, we also need to take the transition probabilities into account. When a system is (observation-)deterministic, we can adapt the notion of beliefs, similar to RVSE [54], and similar to the construction of belief MDPs for *partially observable MDPs*, cf. [53]:

Definition 4 (Belief). *For an MDP $\mathcal{M}$ with a set of states S, a belief bel is a distribution in $\mathsf{Distr}(S)$.*

In the remainder of the paper, we will denote the function $S \to \{0\}$ by $\mathbf{0}$ and the set $\mathsf{Distr}(S) \cup \{\mathbf{0}\}$ by Bel. A state estimator based on Bel is then defined as follows [51,54,57][3]:

[3] For the deterministic case, we omit the unique action for brevity.

Definition 5 (MC state estimator). *For* $\mathsf{MC} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}\rangle$*, a trace* $\tau \in \mathsf{Z}^+$ *the state estimation function* $\mathsf{est}_{\mathsf{MC}} \colon \mathsf{Z}^+ \to \mathsf{Bel}$ *is defined as*

$$\mathsf{est}_{\mathsf{MC}}(z) := \begin{cases} \left\{s \mapsto \frac{\iota(s)\cdot \mathsf{obs}(s)(z)}{\sum\limits_{\hat{s}\in S} \iota(\hat{s})\cdot\mathsf{obs}(\hat{s})(z)}\right\} & \exists s \in S.\ \iota(s)\cdot \mathsf{obs}(z) > 0, \\ \mathbf{0} & \textit{otherwise.} \end{cases}$$

$$\mathsf{est}_{\mathsf{MC}}(\tau\cdot z) := \left\{ s' \mapsto \frac{\sum\limits_{s\in S} \mathsf{est}_{\mathsf{MC}}(\tau)(s)\cdot P(s,s')\cdot \mathsf{obs}(s')(z)}{\sum\limits_{s\in S} \mathsf{est}_{\mathsf{MC}}(\tau)(s)\cdot \big(\sum\limits_{\hat{s}\in S} P(s,\hat{s})\cdot \mathsf{obs}(\hat{s})(z)\big)} \right\}$$

To illustrate how $\mathsf{est}_{\mathsf{MC}}$ is computed, consider again our system in Fig. 1 and assume that the MDP has only the actions labeled with $\{p\}$ (reducing it to the Markov chain induced by the a scheduler that only performs the $\{p\}$ actions). Again we consider the observation trace $\tau = R_o \cdot M_o \cdot L_o$ and compute $\mathsf{est}_{\mathsf{MC}}(\tau)$. For the first observation R_o, and since there is only one initial state, it follows that $\mathsf{est}_{\mathsf{MC}}(R_o) = \{\langle R, D_2\rangle \mapsto 1\}$[4]. From $\langle R, D_2\rangle$ and having observed M_o we can reach the states $\langle R, D_1\rangle$ and $\langle M, D_1\rangle$ with probabilities $\mathsf{est}_{\mathsf{MC}}(R_o \cdot M_o) = \{\langle R, D_1\rangle \mapsto \frac{\frac{1}{2}\cdot\frac{1}{3}}{\frac{1}{2}\cdot\frac{1}{3}+\frac{1}{2}\cdot\frac{3}{4}} = \frac{4}{13}, \langle M, D_1\rangle \mapsto \frac{\frac{1}{2}\cdot\frac{3}{4}}{\frac{1}{2}\cdot\frac{1}{3}+\frac{1}{2}\cdot\frac{3}{4}} = \frac{9}{13}\}$. Finally, from the later two states, when observing L_o, the states $\langle M, D_0\rangle$ and $\langle L, D_0\rangle$ can be reached with probabilities $\mathsf{est}_{\mathsf{MC}}(R_o \cdot M_o \cdot L_o) = \{\langle M, D_0\rangle \mapsto 0.0001, \langle L, D_0\rangle \mapsto 0.999\}$. Notice that although the state $\langle R, D_0\rangle$ can be reached from $\langle R, D_1\rangle$, the probability of being in this state is 0 since the probability of observing L_o in this state is $\mathsf{obs}(\langle R, D_0\rangle)(L_o) = 0$.

Lemma 3. *For a Markov chain* $\mathsf{MC} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}\rangle$*, a trace* $\tau \in \mathsf{Z}^+$*, and a state-risk function* $r\colon S \to \mathbb{R}_{\geq 0}$*, it holds that* $R_r(\tau) = \sum_{s\in S} \mathsf{est}_{\mathsf{MC}}(\tau)(s)\cdot r(s)$*. Computing* $R_r(\tau)$ *can be done in time* $\mathcal{O}(|\tau|\cdot|S|\cdot|P|)$ *, and using* $|S|$ *many rational numbers. The size of the rationals*[5] *may grow linearly in* τ*.*

Proof Sketch. Since the system is deterministic, there is a unique scheduler σ, thus $R_r(\tau) = \sum_{\pi\in\Pi_{\mathsf{MC}}^{|\tau|}} \mathsf{Pr}_\sigma(\pi \mid \tau)\cdot r(\pi_\downarrow)$ by definition. We can show by induction over the length of τ that $\mathsf{Pr}_\sigma(\pi \mid \tau) = \mathsf{est}_{\mathsf{MC}}(\tau)(\pi_\downarrow)$ and conclude that $R_r(\tau) = \sum_{\pi\in\Pi_{\mathcal{M}}^{|\tau|}} \mathsf{est}_{\mathsf{MC}}(\tau)(\pi_\downarrow)\cdot r(\pi_\downarrow) = \sum_{s\in S} \mathsf{est}_{\mathsf{MC}}(\tau)(s)\cdot r(s)$ because $\mathsf{est}_{\mathsf{MC}}(\tau)(s) = 0$ for all $s \in S$ for which there is no path $\pi \in \Pi_{\mathcal{M}}^{|\tau|}$ with $\pi_\downarrow = s$. The complexity follows from the inductive definition of $\mathsf{est}_{\mathsf{MC}}$ that requires in each inductive step to iterate over all transitions of the system and maintain a belief over the states of the system. □

3.3 State Estimators for Markov Decision Processes

In an MDP, we have to account for every possible resolution of nondeterminism, which means that a belief can evolve into a set of beliefs:

[4] We omit the (single) sensor state for conciseness.

[5] To avoid growth, one may use fixed-precision numbers that over-approximate the probability of being in any state—inducing a growing (but conservative) error.

Definition 6 (MDP state estimator). *For an MDP* $\mathcal{M} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}\rangle$*, a trace* $\tau \in \mathsf{Z}^+$*, and a state-risk function* $r\colon S \to \mathbb{R}_{\geq 0}$*, the state estimation function* $\mathsf{est}_{\mathsf{MDP}}\colon \mathsf{Z}^+ \to 2^{\mathsf{Bel}}$ *is defined as*

$$\begin{aligned}\mathsf{est}_{\mathsf{MDP}}(z) &= \{\mathsf{est}_{\mathsf{MC}}(z)\},\\ \mathsf{est}_{\mathsf{MDP}}(\tau \cdot z) &= \left\{\mathsf{bel}' \in \mathsf{Bel} \;\middle|\; \exists \mathsf{bel} \in \mathsf{est}_{\mathsf{MDP}}(\tau).\ \mathsf{bel}' \in \mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(\mathsf{bel}, z)\right\},\end{aligned}$$

and where $\mathsf{bel}' \in \mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(\mathsf{bel}, z)$ *if there exists* $\varsigma_{\mathsf{bel}}\colon S \to \mathsf{Distr}(\mathsf{Act})$ *such that:*

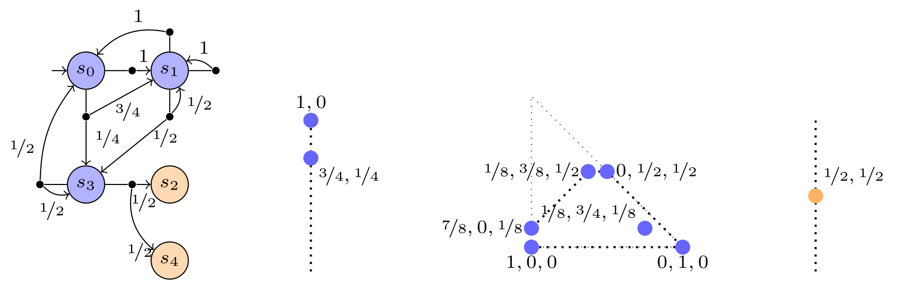
$$\forall s'.\mathsf{bel}'(s') = \frac{\sum\limits_{s \in S} \mathsf{bel}(s) \cdot \sum\limits_{\alpha \in \mathsf{Act}} \varsigma_{\mathsf{bel}}(s)(\alpha) \cdot P(s, \alpha, s') \cdot \mathsf{obs}(s')(z)}{\sum\limits_{s \in S} \mathsf{bel}(s) \cdot \sum\limits_{\alpha \in \mathsf{Act}} \varsigma_{\mathsf{bel}}(s)(\alpha) \cdot \sum\limits_{\hat{s} \in S} P(s, \alpha, \hat{s}) \cdot \mathsf{obs}(\hat{s})(z)}.$$

The definition conservatively extends both Definition 3 and Definition 5. Furthermore, we remark that we do not restrict how the nondeterminism is resolved: any distribution over actions can be chosen, and the distributions may be different for different traces.

Consider our system in Fig. 1. For the trace $\tau = R_o \cdot M_o \cdot L_o$, $\mathsf{est}_{\mathsf{MDP}}(\tau)$ is computed as follows. First, when observing R_o, the state estimator computes the initial belief set $\mathsf{est}_{\mathsf{MDP}}(R_o) = \{\{\langle R, D_2\rangle \mapsto 1\}\}$. From this set of beliefs, when observing M_o, a set $\mathsf{est}_{\mathsf{MDP}}(R_o \cdot M_o)$ can be computed since all transitions $\emptyset, \{p\}, \{w\}, \{p, w\}$ (as well as their convex combinations) are possible from $\langle R, D_2\rangle$. One of these beliefs is for example $\{\langle R, D_1\rangle \mapsto \frac{4}{13}, \langle M, D_1\rangle \mapsto \frac{9}{13}\}$ when a scheduler takes the transition $\{p\}$ (as was computed in our example for the Markov chain case). Having additionally observed L_o a new set $\mathsf{est}_{\mathsf{MDP}}(R_o M_o L_o)$ of beliefs can be computed based on the beliefs in $\mathsf{est}_{\mathsf{MDP}}(R_o M_o)$. For example from the belief $\{\langle R, D_1\rangle \mapsto \frac{4}{13}, \langle M, D_1\rangle \mapsto \frac{9}{13}\}$, two of the new beliefs are $\{\langle L, D_0\rangle \mapsto 0.999, \langle M, D_0\rangle \mapsto 0.0001\}$ and $\{\langle M, D_1\rangle \mapsto 0.0287, \langle M, D_0\rangle \mapsto 0.0001, \langle L, D_0\rangle \mapsto 0.9712\}$. The first belief is reached by a scheduler that takes a transition $\{p\}$ at both $\langle R, D_1\rangle$ and $\langle M, D_1\rangle$. Notice that the belief does not give a positive probability to the state $\langle R, D_0\rangle$ because L_o cannot be observed in this state. The second belief is reached by considering a scheduler that takes transition $\{p\}$ at $\langle M, D_1\rangle$ and transition $\emptyset$ at $\langle R, D_1\rangle$.

Theorem 1. *For an MDP* $\mathcal{M} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}\rangle$*, a trace* $\tau \in \mathsf{Z}^+$*, and a state-risk function* $r\colon S \to \mathbb{R}_{\geq 0}$*, it holds that* $R_r(\tau) = \sup_{\mathsf{bel} \in \mathsf{est}_{\mathsf{MDP}}(\tau)} \sum_{s \in S} \mathsf{bel}(s) \cdot r(s)$*.*

Proof Sketch. For a given trace τ, each (history-dependent, randomizing) scheduler induces a belief over the states of the Markov chain induced by the scheduler. Also, each belief in $\mathsf{est}_{\mathsf{MDP}}(\tau)$ corresponds to a fixed scheduler, namely that one used to compute the belief recursively (i.e., an arbitrary randomizing memoryless scheduler for every time step). Once a scheduler σ and its corresponding belief bel is fixed, or vice versa, we can show using induction over the length of τ that $\sum_{\pi \in \Pi^{|\tau|}_{\mathcal{M}}} \mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow) = \sum_{s \in S} \mathsf{bel}(s) \cdot r(s)$. □



(a) MDP $\mathcal{M}$ (b) Over s_1, s_3 (c) Over s_0, s_1, s_3 (d) Over s_2, s_3

Fig. 3. Beliefs in $\mathbb{R}^n$ on $\mathcal{M}$ for $\tau = z_0 z_0$, $z_0 z_0 z_0$ and $z_0 z_0 z_1$, respectively.

4 Convex Hull-Based Forward Filtering

In this section, we show that we can use a finite representation for $\mathsf{est}_{\mathsf{MDP}}(\tau)$, but that this representation is exponentially large for some MDPs.

4.1 Properties of $\mathsf{est}_{\mathsf{MDP}}(\tau)$.

First, observe that $\mathbf{0}$ never maximizes the risk. Furthermore, $\mathbf{0}$ is closed under updates, i.e., $\mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(\mathbf{0}, z) = \{\mathbf{0}\}$. We can thus w.l.o.g. assume that $\mathbf{0} \notin \mathsf{est}_{\mathsf{MDP}}(\tau)$. Second, observe that $\mathsf{est}_{\mathsf{MDP}}(\tau) \neq \emptyset$ if $\mathsf{Pr}_\sigma(\tau) > 0$.

We can interpret a belief $\mathsf{bel} \in \mathsf{Bel}$ as point in (a bounded subset of) $\mathbb{R}^{(|S|-1)}$. We are in particular interested in convex sets of beliefs. A set $B \subseteq \mathsf{Bel}$ is convex if the convex hull $\mathsf{CH}(B)$ of B, i.e. all convex combination of beliefs in B[6], coincides with B, i.e., $\mathsf{CH}(B) = B$. For a set $B \subseteq \mathsf{Bel}$, a belief $\mathsf{bel} \in B$ is an interior belief if it can be expressed as convex combination of the beliefs in $B \setminus \{\mathsf{bel}\}$. All other beliefs are (extremal) points or *vertices*. Let the set $\mathcal{V}(B) \subseteq B$ denote the set of *vertices of the convex hull* of B.

Example 1. Consider Fig. 3(a). All observation are Dirac, and only states s_2 and s_4 have observation z_1. The beliefs having observed $z_0 z_0$ are distributions over s_1, s_3, and can thus be depicted in a one-dimensional simplex. In particular, we have $\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(z_0 z_0)) = \{\{s_1 \mapsto 1\}, \{s_1 \mapsto 3/4, s_3 \mapsto 1/4\}\}$, as depicted in Fig. 3(b). The six beliefs having observed $z_0 z_0 z_0$ are distributions over s_0, s_1, s_3, depicted in Fig. 3(c). Five out of six beliefs are vertices. The belief having observed $z_0 z_0 z_1$ is in Fig. 3(d).

Remark 2. Observe that we illustrate the beliefs over only the states $\mathsf{est}_{\mathsf{KS}}(\tau)$. We therefore call $|\mathsf{est}_{\mathsf{KS}}(\tau)|$ the dimension of $\mathsf{est}_{\mathsf{MDP}}(\tau)$.

From the fundamental theorem of linear programming [47, Ch. 7] it immediately follows that the trace risk R_τ is obtained at a vertex of the beliefs of $\mathsf{est}_{\mathsf{MDP}}\tau$. We obtain the following refinement over Theorem 1:

[6] That is, $\mathsf{CH}(B) = \{\sum_{\mathsf{bel} \in B} w(\mathsf{bel}) \cdot \mathsf{bel} \mid \text{for all } w \in \mathbb{R}^B_{\geq 0} \text{ with } \sum w(\mathsf{bel}) = 1\}$.

Theorem 2. *For every τ and r: $R_r(\tau) = \max_{\mathsf{bel} \in \mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))} \sum_{s \in S} \mathsf{bel}(s) \cdot r(s)$.*

Lemma 5 below clarifies that this maximum indeed exists.

We make some observations that allow us to compute the vertices more efficiently: Let $\mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(B, z)$ denote $\bigcup_{\mathsf{bel} \in B} \mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(\mathsf{bel}, z)$. From the properties of convex sets [18, Ch. 2], we make the following observations: If B is convex, $\mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(B, z)$ is convex, as all operations in computing a new belief are convex-set preserving[7]. Furthermore, if B has a finite set of vertices, then $\mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(B, z)$ has a finite set of vertices. The following lemma which is based on the observations above clarifies how to compute the vertices:

Lemma 4. *For a convex set of beliefs B with a finite set of vertices and an observation z:*

$$\mathcal{V}(\mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(B, z)) = \mathcal{V}(\mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(\mathcal{V}(B), z)).$$

By induction and using the facts above we obtain:

Lemma 5. *Any $\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))$ is finite.*

A monitor thus only needs to track the vertices. Furthermore, $\mathsf{est}^{\mathsf{up}}_{\mathsf{MDP}}(B, z)$ can be adapted to compute only vertices by limiting ς_{bel} to $S \to \mathsf{Act}$.

4.2 Exponential Lower Bounds on the Relevant Vertices

We show that a monitor in general cannot avoid an exponential blow-up in the beliefs it tracks. First observe that updating bel yields up to $\prod_s |\mathsf{Act}(s)|$ new beliefs (vertex or not), a prohibitively large number. The number of vertices is also exponential:

Lemma 6. *There exists a family of MDPs $\mathcal{M}_n$ with $2n + 1$ states such that $|\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))| = 2^n$ for every τ with $|\tau| > 2$.*

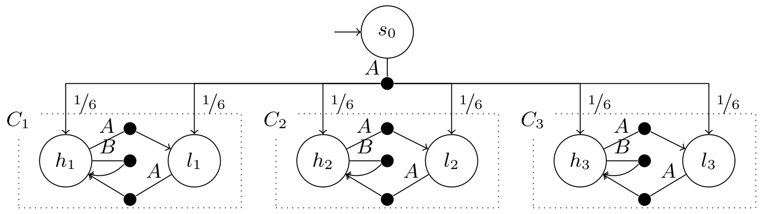
Proof Sketch. We construct $\mathcal{M}_n$ with $n = 3$, that is, $\mathcal{M}_3$ in Fig. 4(a). For this MDP and $\tau = AAA$, $|\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))| = 2^3$. In particular, observe how the belief factorizes into a belief within each component $C_i = \{h_i, l_i\}$ and notice that $\mathcal{M}_n$ has components C_1 to C_n. In particular, for each component, the belief being that we are with probability mass $1/n$ (for $n = 3$, $1/3$) in the 'low' state l_i or the 'high' state h_i. We depict the beliefs in Fig. 4(b,c,d). Thus, for any τ with $|\tau| > 2$ we can compactly represent $\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))$ as bit-strings of length n. Concretely, the belief

$$\{h_1, l_2, l_3 \mapsto 1/3, l_1, h_2, h_3 \mapsto 0\} \text{ maps to } 100, \text{ and}$$
$$\{h_1, l_2, h_3 \mapsto 1/3, l_1, h_2, l_3 \mapsto 0\} \text{ maps to } 101.$$

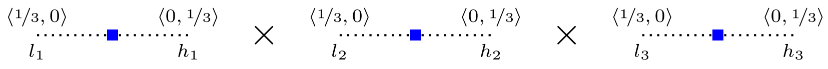
These are exponentially many beliefs for bit strings of length n. □

One might ask whether a symbolic encoding of an exponentially large set may result in a more tractable approach to filtering. While Theorem 2 allows

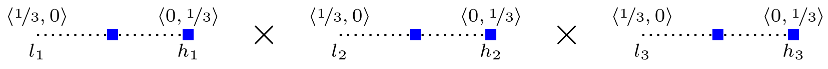
[7] The scaling is called a *projection*.



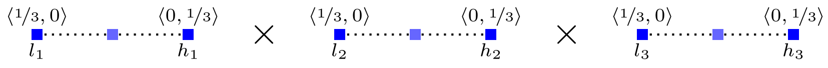
(a) $\mathcal{M}_3$. Observations $\mathsf{Z} = \mathsf{Act}$ with $\mathsf{obs}(s, \alpha) = \alpha$ if $\alpha \neq B$ and $\mathsf{obs}(s, B) = A$ for every s. Initial belief is A, all probabilities are 1, unless stated otherwise.



(b) Beliefs after AA



(c) Beliefs after AAA



(d) Beliefs after $AAAA^+$

Fig. 4. Construction for the correctness of Lemma 6.

to compute the associated risk from a set of linear constraints with standard techniques, it is not clear whether the concise set of constraints can be efficiently constructed and updated in every step. We leave this concern for future work.

In the remainder we investigate whether we need to track all these beliefs. First, when the monitor is unaware of the state-risk, this is trivially unavoidable. More precisely, all vertices may induce the maximal weighted trace risk by choosing an appropriate state-risk:

Lemma 7. *For every τ and every $\mathsf{bel} \in \mathcal{V}(\mathsf{est_{MDP}}(\tau))$ there exists an r s.t.*

$$\sum_{s \in S} \mathsf{bel}(s) \cdot r(s) \geq \max_{\mathsf{bel}' \in \mathcal{V}(\mathsf{est_{MDP}}(\tau)) \setminus \{\mathsf{bel}\}} \sum_{s \in S} \mathsf{bel}'(s) \cdot r(s) \text{ with } \max_{\mathsf{bel} \in \emptyset} = -\infty.$$

Proof Sketch. We construct r such that $r(s) > r(s')$ if $\mathsf{bel}(s) > \mathsf{bel}(s')$. □

Second, even if the monitor is aware of the state risk r, it may not be able to prune enough vertices to avoid exponential growth. The crux here is that while some of the current beliefs may induce a smaller risk, an extension of the trace may cause the belief to evolve into a belief that induces the maximal risk.

Theorem 3. *There exist MDPs $\mathcal{M}_n$ a τ with $B := \mathcal{V}(\mathsf{est_{MDP}}(\tau))$ and a state-risk r such that $|B| = 2^n$ and for all $\mathsf{bel} \in B$ exists $\tau' \in \mathsf{Z}^+$ with $R_r(\tau \cdot \tau') > \sup_{\mathsf{bel} \in B'} \sum_s \mathsf{bel}(s) \cdot r(s)$, where $B' = \mathsf{est^{up}_{MDP}}(B \setminus \{\mathsf{bel}\}, \tau')$.*

It is helpful to understand this theorem as describing the outcome of a game between monitor and environment: The statement says if the monitor decides to drop some vertices from $\mathsf{est}_{\mathsf{MDP}}\tau$, the environment may produce an observation trace τ' that will lead the monitor to underestimate the weighted risk at $R_r(\tau{\cdot}\tau')$.

Proof Sketch. We extend the construction of Fig. 4(a) with choices to go to a final state. The full proof sketch can be found in [36, Appendix].

4.3 Approximation by Pruning

Finally, we illustrate that we cannot simply prune small probabilities from beliefs. This indicates that an approximative version of filtering for the monitoring problem is nontrivial. Reconsider observing $z_0 z_0$ in the MDP of Fig. 3, and, for the sake of argument, let us prune the (small) entry $s_3 \mapsto 1/4$ to 0. Now, continuing with the trace $z_0 z_0 z_1$, we would update the beliefs from before and then conclude that this trace cannot be observed with positive probability. With pruning, there is no upper bound on the difference between the *computed* R_τ and the *actual* R_τ. Thus, forward filtering is, in general, not tractable on MDPs.

5 Unrolling with Model Checking

We present a tractable algorithm for the monitoring problem. Contrary to filtering, this method incorporates the state risk. We briefly consider the qualitative case. An algorithm that solves that problem iteratively guesses a successor such that the given trace has positive probability, and reaches a state with sufficient risk. The algorithm only stores the current and next state and a counter.

Theorem 4. *The Monitoring Problem with $\lambda = 0$ is in NLOGSPACE.*

This result implies the existence of a polynomial time algorithm, e.g., using a graph-search on a graph growing in $|\tau|$. There also is a deterministic algorithm with space complexity $\mathcal{O}(log^2(|\mathcal{M}|+|\tau|))$, which follows from applying Savitch's Theorem [46] , but that algorithm has exponential time complexity.

We now present a tractable algorithm for the quantitative case, where we need to store all paths. We do this efficiently by storing an unrolled MDP with these paths using ideas from [9,19]. In particular, on this MDP, we can efficiently obtain the scheduler that optimizes the risk by model checking rather than enumerating over all schedulers explicitly. We give the result before going into details.

Theorem 5. *The Monitoring Problem (with $\lambda > 0$) is P-complete.*

The problem is P-hard, as unary-encoded step-bounded reachability is P-hard [41]. It remains to show a P-time algorithm[8], which is outlined below. Roughly, the algorithm constructs an MDP $\mathcal{M}'''$ from $\mathcal{M}$ in three conceptual steps, such that the

[8] On first sight, this might be surprising as step-bounded reachability in MDPs is PSPACE-hard and only quasi-polynomial. However, our problem gets a trace and therefore (assuming that the trace is not compressed) can be handled in time polynomial in the length of the trace.

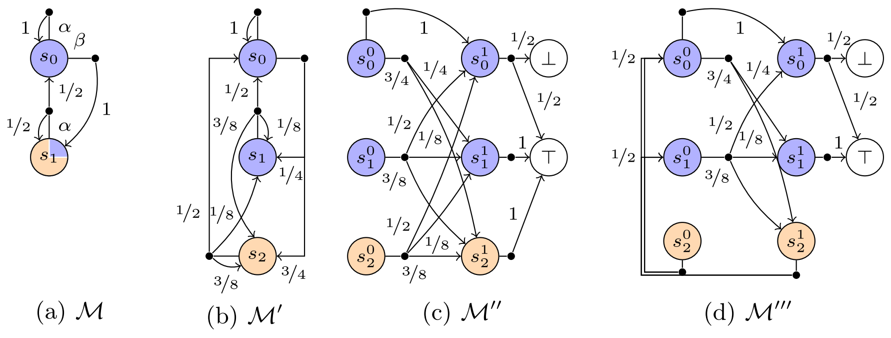


Fig. 5. Polynomial-time algorithm for solving Problem 1 illustrated.

maximal probability of reaching a state in $\mathcal{M}'''$ coincides with the $R_r(\tau)$. The former can be solved by linear programming in polynomial time. The downside is that even in the best case, the memory consumption grows linearly in $|\tau|$.

We outline the main steps of the algorithm and exemplify them below: First, we transform $\mathcal{M}$ into an MDP $\mathcal{M}'$ with *deterministic state* observations, i.e., with $\mathsf{obs}' \colon S \to \mathsf{Z}$. This construction is detailed in [19, Remark 1], and runs in polynomial time. The new initial distribution takes into account the initial observation and the initial distribution. Importantly, for each path π and each trace τ, $\mathsf{obs}_{\mathsf{tr}}(\pi)(\tau)$ is preserved. From here, the idea for the algorithm is a tailored adaption of the construction for conditional reachability probabilities in [9]. We ensure that $r(s) \in [0,1]$ by scaling r and λ accordingly. Now, we construct a new MDP $\mathcal{M}'' = \langle S'', \iota'', \mathsf{Act}'', P'' \rangle$ with state space $S'' := (S' \times \{0, \dots, |\tau|-1\}) \cup \{\bot, \top\}$ and an n-times unrolled transition relation. Furthermore, from the states $\langle s, |\tau|-1 \rangle$, there is a single outgoing action that with probability $r(s)$ leads to $\top$ and with probability $1 - r(s)$ leads to $\bot$. Observe that the risk is now the supremum of conditioned reachability probabilities over paths that reach $\top$, conditioned by the trace τ. The MDP $\mathcal{M}''$ is only polynomially larger. Then, we construct MDP $\mathcal{M}'''$ by copying $\mathcal{M}''$ and replacing (part of) the transition relation P'' by P''' such that paths π with $\tau \notin \mathsf{obs}_{\mathsf{tr}}(\pi)$ are looped back to the initial state (resembling rejection sampling). Formally,

$$P'''(\langle s, i \rangle, \alpha) = \begin{cases} P''(\langle s, i \rangle, \alpha) & \text{if } \mathsf{obs}'(s) = \tau_i, \\ \iota & \text{otherwise.} \end{cases}$$

The maximal conditional reachability probability in $\mathcal{M}''$ is the maximal reachability probability in M''' [9]. Maximal reachability probabilities can be computed by solving a linear program [43], and can thus be computed in polynomial time.

Example 2. We illustrate the construction in Fig. 5. In Fig. 5(a), we depict an MDP $\mathcal{M}$, with $\iota = \{s_0, s_1 \mapsto 1/2\}$. Furthermore, let $\tau = z_0 z_0$ and let $r(s_0) = 1$ and $r(s_1) = 2$. Let $\mathsf{obs}(s_0) = \{z_0 \mapsto 1\}$ and $\mathsf{obs}(s_1) = \{z_0 \mapsto 1/4, z_1 \mapsto 3/4\}$. State s_1 has two possible observations, so we split s_1 into s_1 and s_2 in MDP

$\mathcal{M}'$, each with their own observations. Any transition into s_1 is now split. As $|\tau| = 2$, we unroll the MDP $\mathcal{M}'$ into MDP $\mathcal{M}''$ to represent two steps, and add goal and sink states. After rescaling, we obtain that $r(s_0) = 1/2$, whereas $r(s_1) = r(s_2) = 2/2 = 1$, and we add the appropriate outgoing transitions to the states s_*^1. In a final step, we create MDP $\mathcal{M}'''$ from $\mathcal{M}''$: we reroute all probability mass that does not agree with the observations to the initial states. Now, $R_r(z_0 z_0)$ is given by the probability to reach, in $\mathcal{M}'''$, in an unbounded number of steps, $\top$.

The construction also implies that maximizing over a finite set of schedulers, namely the deterministic schedulers with a counter from 0 to $|\tau|$, suffices. We denote this class $\Sigma_{\mathrm{DC}}(|\tau|)$. Formally, a scheduler is in $\Sigma_{\mathrm{DC}}(k)$ if for all π, π':

$$\Big(\pi_\downarrow = \pi'_\downarrow \wedge \big(|\pi| = |\pi'| \vee (|\pi| > k \wedge |\pi'| > k)\big)\Big) \text{ implies } \sigma(\pi) = \sigma(\pi').$$

Lemma 8. *For every τ, it holds that*

$$R_r(\tau) \quad = \quad \max_{\sigma \in \Sigma_{DC}(|\tau|)} \sum_{\pi \in \Pi_M^{|\tau|}} \mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow).$$

The crucial idea underpinning this lemma is that memoryless schedulers suffice for the unrolling, and that the states of the unrolling can be uniquely mapped to a state and the length of the history for every π through $\mathcal{M}$. By reducing step-bounded reachability we can also show that this set of schedulers is necessary [4].

6 Empirical Evaluation

Implementation. We provide prototype implementations for both filtering- and model-checking-based approaches from Sect. 3, built on top of the probabilistic model checker Storm [30]. We provide a schematic setup of our implementation in Fig. 6. As input, we consider a symbolic description of MDPs with state-based observation labels, based on an extended dialect of the Prism language. We define the state risk in this MDP via a temporal property (given as a PCTL formula), and obtain the concrete state-risk by model checking. We take a seed that yields a trace using the simulator. For the experiments, actions are resolved uniformly in this simulator[9]. The simulator iteratively feeds observations into the monitor, running either of our two algorithms (implemented in C++). After each observation z_i, the monitor computes the risk R_i having observed $z_0 \dots z_i$. We flexibly combine these components via a Python API[10].

For filtering as in Sect. 4, we provide a sparse data structure for beliefs that is updated using only deterministic schedulers. This is sufficient, see Lemma 4. To further prune the set of beliefs, we implement an SMT-driven elimination [48]

[9] This is not an assumption but rather our evaluation strategy.
[10] Available at https://github.com/monitoring-MDPs/premise.

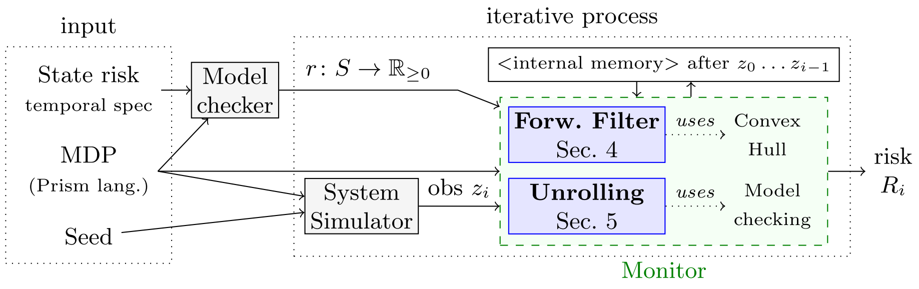


Fig. 6. Schematic setup for prototype mapping stream $z_0 \dots z_k$ to stream $R_0 \dots R_k$.

of interior beliefs, inside of the convex hull[11]. We construct the unrolling as described in Sect. 5 and apply model checking via any sparse engines in STORM.

Reproducibility. We archived a container with sources, benchmarks, and scripts to reproduce our experiments: https://doi.org/10.5281/zenodo.4724622.

Set-Up. For each benchmark described below, we sampled 50 random traces using seeds 0–49 of lengths up to $|\tau| = 500$. We are interested in the *promptness*, that is, the delay of time between getting an observation z_i and returning corresponding risk r_i, as well as the *cumulative performance* obtained by summing over the promptness along the trace. We use a timeout of 1 second for this query. We compare the forward filtering (FF) approach with and without convex hull (CH) reduction, and the model unrolling approach (UNR) with two model checking engines of STORM: exact policy iteration (EPI, [43]) and optimistic value iteration (OVI, [28]). All experiments are run on a MacBook Pro MV962LL/A, using a single core. The memory limit of 6GB was not violated. We use Z3 [38] as SMT-solver [11] for the convex hull reduction.

Benchmarks. We present three benchmark families, all MDPs with a combination of probabilities, nondeterminism and partial observability.

AIRPORT-A is as in Sect. 1, but with a higher resolution for both ground vehicle in the middle lane and the plane. AIRPORT-B has a two-state sensor model with stochastic transitions between them.

REFUEL-A models robots with a depleting battery and recharging stations. The world model consists of a robot moving around in a $D \times D$ grid with some dedicated charging cells, where each action costs energy. The risk is to deplete the battery within a fixed horizon. REFUEL-B is a two-state sensor variant.

EVADE-I is inspired by a navigation task in a multi-agent setting in a $D \times D$ grid. The monitored robot moves randomly, and the risk is defined as the probability of crashing with the other robot. The other robot has an internal incentive in the form of a cardinal direction, and nondeterministically decides to move or

[11] Advanced algorithms like Quickhull [10] are not without significant adaptions applicable as the set of beliefs can be degenerate (roughly, a set without full rank).

Table 1. Performance for promptness of online monitoring on various benchmarks.

						CH Forward Filtering							Unrolling				
Id	Name	Inst	$\|S\|$	$\|P\|$	$\|\tau\|$	N	T_{avg}	T_{max}	B_{avg}	B_{max}	D_{avg}	D_{max}	N	T_{avg}	T_{max}	$\|S_u\|_{\text{avg}}$	$\|S_u\|_{\text{max}}$
1	AIRPORT-A	7,50,30	20910	114143	100	**50**	0.01	0.01	4.5	7	4.6	7	**50**	0.04	0.11	524	599
					500	**50**	0.01	0.01	1.0	1	1.0	1	**50**	0.01	0.01	1075	1258
2	AIRPORT-B	3,50,30	20232	106012	100	**0**							**50**	0.09	0.16	556	629
					500	**0**							**50**	0.01	0.01	1460	1647
3	AIRPORT-B	7,50,30	41820	308474	100	**0**							**50**	0.14	0.33	1000	1183
					500	**0**							**11**	0.02	0.02	2097	2297
4	REFUEL-A	12,50	45073	2431691	100	**50**	0.01	0.01	2.2	4	2.8	5	**50**	0.01	0.05	325	409
					500	**50**	0.01	0.01	1.5	4	1.7	5	**50**	0.01	0.19	1071	2409
5	REFUEL-B	12,50	90145	9725277	100	**50**	0.06	0.23	4.2	8	5.6	10	**50**	0.04	0.17	608	732
					500	**50**	0.01	0.01	2.9	8	3.3	10	**46**	0.04	0.09	2171	4688
6	EVADE-I	15	377101	2022295	100	**50**	0.01	0.02	2.6	10	3.3	4	**49**	0.01	0.06	332	363
					500	**50**	0.01	0.01	2.4	5	3.4	4	**45**	0.08	0.90	1655	1891
7	EVADE-V	5,3	1001	5318	100	**26**	0.01	0.01	1.0	1	1.0	1	**50**	0.00	0.02	134	241
					500	**25**	0.01	0.01	1.0	1	1.0	1	**50**	0.00	0.01	538	671
8	EVADE-V	6,3	2161	11817	100	**1**	0.01	0.01	1.0	1	1.0	1	**50**	0.02	0.32	319	861
					500	**1**	0.01	0.01	1.0	1	1.0	1	**49**	0.01	0.02	777	1484

to uniformly randomly change its incentive. The monitor observes everything except the incentive of the other robot. EVADE-V is an alternative navigation task: Contrary to above, the other robot does not have an internal state and indeed navigates nondeterministically in one of the cardinal directions. We only observe the other robot location is within the view range.

Results. We split our results in two tables. In Table 1, we give an ID for every benchmark name and instance, along with the size of the MDP (nr. of states $|S|$ and transitions $|P|$) our algorithms operate on. We consider the promptness after prefixes of length $|\tau|$. In particular, for forward filtering with the convex hull optimization, we give the number N of traces that did not time out before, and consider the average T_{avg} and maximal time T_{max} needed (over all sampled traces that did not time-out before). Furthermore, we give the average, B_{avg}, and maximal, B_{max}, number of beliefs stored (after reduction), and the average, D_{avg}, and maximal, D_{max}, dimension of the belief support. Likewise, for unrolling with exact model checking, we give the number N of traces that did not time out before, and we consider average T_{avg} and maximal time T_{max}, as well as the average size and maximal number of states of the unfolded MDP.

In Table 2, we consider for the benchmarks above the cumulative performance. In particular, this table also considers an alternative implementation for both FF and UNR. We use the IDs to identify the instance, and sum for each prefix of length $|\tau|$ the time. For filtering, we recall the number of traces N that did not time out, the average and maximal cumulative time along the trace, the average cumulative number of beliefs that were considered, and the average cumulative number of beliefs eliminated. For the case without convex hull, we do not eliminate any vertices. For unrolling, we report average T_{avg} and maximal cumulative time using EPI, as well as the time required for model building, *Bld*$^{\%}$ (relative to the total time, per trace). We compare this to the average

Table 2. Summarized performance for online monitoring

		FF w/o CH				FF w/ CH					UNR (EPI)					UNR (OVI)		
Id	$\lvert\tau\rvert$	N	T_{avg}	T_{max}	B_{avg}	N	T_{avg}	T_{max}	B_{avg}	E_{avg}	N	T_{avg}	T_{max}	$Bld^{\%}_{avg}$	$Bld^{\%}_{max}$	N	T_{avg}	T_{max}
1	100	**0**				**50**	0.9	1.1	493	241	**50**	2.9	3.6	6	56	**50**	0.0	0.1
	500	**0**				**50**	3.7	4.3	1040	316	**50**	7.5	10.7	21	24	**50**	0.4	0.8
2	100	**0**				**0**					**50**	3.7	4.7	6	54	**50**	0.1	0.1
	500	**0**				**0**					**50**	11.9	17.1	18	23	**50**	0.6	0.8
3	100	**0**				**0**					**50**	7.6	10.6	5	55	**50**	0.1	0.2
	500	**0**				**0**					**11**	21.3	28.7	19	23	**50**	0.9	1.7
4	100	**1**	0.9	0.9	1473	**50**	0.7	0.8	241	138	**50**	0.7	1.0	35	69	**50**	0.0	0.1
	500	**1**	0.9	0.9	1873	**50**	3.4	3.7	868	226	**50**	5.6	21.2	57	67	**50**	0.5	0.9
5	100	**0**				**50**	7.4	10.7	442	2267	**50**	2.5	4.4	32	57	**50**	0.1	0.2
	500	**0**				**50**	16.5	42.2	1781	4249	**46**	19.5	64.2	55	70	**50**	1.3	2.3
6	100	**13**	0.7	2.9	2055	**50**	1.1	4.8	273	160	**49**	0.5	2.0	34	65	**47**	0.0	0.1
	500	**2**	4.4	6.8	20524	**50**	5.1	11.5	1237	632	**45**	22.4	53.6	13	29	**43**	0.5	0.7
7	100	**13**	0.1	0.5	274	**26**	0.8	1.2	106	11	**50**	0.4	1.0	19	45	**48**	0.0	0.1
	500	**13**	0.1	0.5	674	**25**	3.7	4.2	505	7	**50**	1.3	4.4	46	58	**47**	0.2	0.3
8	100	**0**				**1**	1.3	1.3	124	109	**50**	1.5	7.0	15	39	**36**	0.4	5.6
	500	**0**				**1**	4.3	4.3	524	109	**49**	4.9	28.1	37	56	**35**	0.7	6.4

and maximal cumulative time for using OVI (notice that building times remain approximately the same).

Discussion. The results from our prototype show that conservative (sound) predictive modeling of systems that combine probabilities, nondeterminism and partial observability is within reach with the methods we proposed and state-of-the-art algorithms. Both forward filtering and an unrolling-based approaches have their merits. The practical results thus slightly diverge from the complexity results in Sect. 3.1, due to structural properties of some benchmarks. In particular, for AIRPORT-A and REFUEL-A, the nondeterminism barely influences the belief, and so there is no explosion, and consequentially the dimension of the belief is sufficiently small that the convex hull can be efficiently computed. Rather than the number of states, this belief dimension makes EVADE-V a difficult benchmark[12]. *If many states can be reached with a particular trace, and if along these paths there are some probabilistic states, forward filtering suffers significantly.* We see that if the benchmark allows for efficacious forward filtering, it is not slowed down in the way that unrolling is slower on longer traces. For UNR, we observe that OVI is typically the fastest, but EPI does not suffer from the numerical worst-cases as OVI does. *If an observation trace is unlikely, the unrolled MDP constitutes a numerically challenging problem, in particular for value-iteration based model checkers*, see [27]. For FF, the convex hull computation is essential for any dimension, and eliminating some vertices in every step keeps the number of belief states manageable.

[12] The max dimension =1 in EVADE-V is only over the traces that did not time-out. The dimension when running in time-outs is above 5.

7 Related Work

We are not the first to consider model-based runtime verification in the presence of partial observability and probabilities. Runtime verification with state estimation on hidden Markov models (HMM)—without nondeterminism has been studied for various types of properties [51,54,57] and has been extended to hybrid systems [52]. The tool Prevent focusses on black-box systems by learning an HMM from a set of traces. The HMM approximates (with only convergence-in-the-limit guarantees) the actual system [6], and then estimates during runtime the most likely trace rather than estimating a distribution over current states. Extensions consider symmetry reductions on the models [7]. These techniques do not make a conservative (sound) risk estimation. The recent framework for runtime verification in the presence of partial observability [23] takes a more strict black-box view and cannot provide state estimates. Finally, [26] chooses to have partial observability to make monitoring of software systems more efficient, and [58] monitors a noisy sensor to reduce energy consumption.

State beliefs are studied when verifying HMMs [59], where the question whether a sequence of observations likely occurs, or which HMM is an adequate representation of a system [37]. State beliefs are prominent in the verification of partially observable MDPs [16,32,40], where one can observe the actions taken (but the problem itself is to find the right scheduler). Our monitoring problem can be phrased as a special case of verification of partially observable stochastic games [20], but automatic techniques for those very general models are lacking. Likewise, the idea of *shielding* (pre)computes all action choices that lead to safe behavior [3,5,15,24,34,35]. For partially observable settings, shielding again requires to compute partial-information schedulers [21,39], contrary to our approach. Partial observability has also been studied in the context of diagnosability, studying if a fault has occurred (in the past) [14], or what actions uncover faults [13]. We, instead assume partial observability in which we do detect faults, but want to estimate the risk that these faults occur in the future.

The assurance framework for reinforcement learning [42] implicitly allows for stochastic behavior, but cannot cope with partial observability or nondeterminism. Predictive monitoring has been combined with deep learning [17] and Bayesian inference [22], where the key problem is that the computation of an imminent failure is too expensive to be done exactly. More generally, learning automata models has been motivated with runtime assurance [1,55]. Testing approaches statistically evaluate whether traces are likely to be produced by a given model [25]. The approach in [2] studies stochastic black-box systems with controllable nondeterminism and iteratively learns a model for the system.

8 Conclusion

We have presented the first framework for monitoring based on a trace of observations on models that combine nondeterminism and probabilities. Future work includes heuristics for approximate monitoring and for faster convex hull computations, and to apply this work to gray-box (learned) models.

Acknowledgments. The example in Fig. 1 is inspired by a challenge problem provided by Boeing in the DARPA Assured Autonomy program.

References

1. Aichernig, B.K., et al.: Learning a behavior model of hybrid systems through combining model-based testing and machine learning. In: Gaston, C., Kosmatov, N., Le Gall, P. (eds.) ICTSS 2019. LNCS, vol. 11812, pp. 3–21. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31280-0_1
2. Aichernig, B.K., Tappler, M.: Probabilistic black-box reachability checking (extended version). Formal Methods Syst. Des. **54**(3), 416–448 (2019)
3. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: AAAI, pp. 2669–2678. AAAI Press (2018)
4. Andova, S., Hermanns, H., Katoen, J.-P.: Discrete-time rewards model-checked. In: Larsen, K.G., Niebert, P. (eds.) FORMATS 2003. LNCS, vol. 2791, pp. 88–104. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-40903-8_8
5. Avni, G., Bloem, R., Chatterjee, K., Henzinger, T.A., Könighofer, B., Pranger, S.: Run-time optimization for learned controllers through quantitative games. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 630–649. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_36
6. Babaee, R., Gurfinkel, A., Fischmeister, S.: *Prevent*: a predictive run-time verification framework using statistical learning. In: Johnsen, E.B., Schaefer, I. (eds.) SEFM 2018. LNCS, vol. 10886, pp. 205–220. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92970-5_13
7. Babaee, R., Gurfinkel, A., Fischmeister, S.: Predictive run-time verification of discrete-time reachability properties in black-box systems using trace-level abstraction and statistical learning. In: Colombo, C., Leucker, M. (eds.) RV 2018. LNCS, vol. 11237, pp. 187–204. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03769-7_11
8. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press, Cambridge (2008)
9. Baier, C., Klein, J., Klüppelholz, S., Märcker, S.: Computing conditional probabilities in Markovian models efficiently. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 515–530. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_43
10. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The Quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**(4), 469–483 (1996)
11. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 825–885. IOS Press (2009)
12. Bartocci, E., et al.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification. LNCS, vol. 10457, pp. 135–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5_5
13. Bertrand, N., Fabre, É., Haar, S., Haddad, S., Hélouët, L.: Active diagnosis for probabilistic systems. In: Muscholl, A. (ed.) FoSSaCS 2014. LNCS, vol. 8412, pp. 29–42. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54830-7_2
14. Bertrand, N., Haddad, S., Lefaucheux, E.: A tale of two diagnoses in probabilistic systems. Inf. Comput. **269** (2019)

15. Bloem, R., Könighofer, B., Könighofer, R., Wang, C.: Shield synthesis: runtime enforcement for reactive systems. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 533–548. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_51
16. Bork, A., Junges, S., Katoen, J.-P., Quatmann, T.: Verification of indefinite-horizon POMDPs. In: Hung, D.V., Sokolsky, O. (eds.) ATVA 2020. LNCS, vol. 12302, pp. 288–304. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59152-6_16
17. Bortolussi, L., Cairoli, F., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural predictive monitoring. In: Finkbeiner, B., Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 129–147. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32079-9_8
18. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2014)
19. Chatterjee, K., Chmelik, M., Gupta, R., Kanodia, A.: Optimal cost almost-sure reachability in POMDPs. Artif. Intell. **234**, 26–48 (2016)
20. Chatterjee, K., Doyen, L.: Partial-observation stochastic games: how to win when belief fails. ACM Trans. Comput. Log. **15**(2), 16:1–16:44 (2014)
21. Chatterjee, K., Novotný, P., Pérez, G.A., Raskin, J., Zikelic, D.: Optimizing expectation with guarantees in POMDPs. In: AAAI, pp. 3725–3732. AAAI Press (2017)
22. Chou, Y., Yoon, H., Sankaranarayanan, S.: Predictive runtime monitoring of vehicle models using Bayesian estimation and reachability analysis. In: IROS (2020, to appear)
23. Cimatti, A., Tian, C., Tonetta, S.: Assumption-based runtime verification with partial observability and resets. In: Finkbeiner, B., Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 165–184. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32079-9_10
24. Dräger, K., Forejt, V., Kwiatkowska, M.Z., Parker, D., Ujma, M.: Permissive controller synthesis for probabilistic systems. Log. Methods Comput. Sci. **11**(2) (2015)
25. Gerhold, M., Stoelinga, M.: Model-based testing of probabilistic systems. Formal Asp. Comput. **30**(1), 77–106 (2018)
26. Grigore, R., Kiefer, S.: Selective monitoring. In: CONCUR. LIPIcs, vol. 118, pp. 20:1–20:16. Dagstuhl - LZI (2018)
27. Haddad, S., Monmege, B.: Interval iteration algorithm for MDPs and IMDPs. Theor. Comput. Sci. **735**, 111–131 (2018)
28. Hartmanns, A., Kaminski, B.L.: Optimistic value iteration. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12225, pp. 488–511. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53291-8_26
29. Havelund, K., Roşu, G.: Runtime verification - 17 years later. In: Colombo, C., Leucker, M. (eds.) RV 2018. LNCS, vol. 11237, pp. 3–17. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03769-7_1
30. Hensel, C., Junges, S., Katoen, J., Quatmann, T., Volk, M.: The probabilistic model checker storm. CoRR abs/2002.07080 (2020)
31. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: Algorithmic analysis of nonlinear hybrid systems. IEEE Trans. Autom. Control **43**(4), 540–554 (1998)
32. Horák, K., Bosanský, B., Chatterjee, K.: Goal-HSVI: heuristic search value iteration for goal POMDPs. In: IJCAI, pp. 4764–4770. ijcai.org (2018)
33. Jansen, N., Humphrey, L., Tumova, J., Topcu, U.: Structured synthesis for probabilistic systems. In: Badger, J.M., Rozier, K.Y. (eds.) NFM 2019. LNCS, vol. 11460, pp. 237–254. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20652-9_16

34. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe reinforcement learning using probabilistic shields (invited paper). In: CONCUR. LIPIcs, vol. 171, pp. 3:1–3:16. Dagstuhl - LZI (2020)
35. Junges, S., Jansen, N., Dehnert, C., Topcu, U., Katoen, J.-P.: Safety-constrained reinforcement learning for MDPs. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 130–146. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_8
36. Junges, S., Torfah, H., Seshia, S.A.: Runtime monitoring for Markov decision processes. CoRR abs/2105.12322 (2021)
37. Kiefer, S., Sistla, A.P.: Distinguishing hidden Markov chains. In: LICS, pp. 66–75. ACM (2016)
38. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
39. Nam, W., Alur, R.: Active learning of plans for safety and reachability goals with partial observability. IEEE Trans. Syst. Man Cybern. Part B **40**(2), 412–420 (2010)
40. Norman, G., Parker, D., Zou, X.: Verification and control of partially observable probabilistic systems. Real Time Syst. **53**(3), 354–402 (2017)
41. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Math. Oper. Res. **12**(3), 441–450 (1987)
42. Phan, D.T., Grosu, R., Jansen, N., Paoletti, N., Smolka, S.A., Stoller, S.D.: Neural simplex architecture. In: Lee, R., Jha, S., Mavridou, A., Giannakopoulou, D. (eds.) NFM 2020. LNCS, vol. 12229, pp. 97–114. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-55754-6_6
43. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics. Wiley (1994)
44. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
45. Sánchez, C., et al.: A survey of challenges for runtime verification from advanced application domains (beyond software). Formal Methods Syst. Des. **54**(3), 279–335 (2019)
46. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. J. Comput. Syst. Sci. **4**(2), 177–192 (1970)
47. Schrijver, A.: Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley (1999)
48. Seidel, R.: Convex hull computations. In: Handbook of Discrete and Computational Geometry, 2nd edn, pp. 495–512. Chapman and Hall/CRC (2004)
49. Seshia, S.A.: Introspective environment modeling. In: Finkbeiner, B., Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 15–26. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32079-9_2
50. Seshia, S.A., Sadigh, D., Sastry, S.S.: Towards verified artificial intelligence. arXiv e-prints, July 2016
51. Sistla, A.P., Srinivas, A.R.: Monitoring temporal properties of stochastic systems. In: Logozzo, F., Peled, D.A., Zuck, L.D. (eds.) VMCAI 2008. LNCS, vol. 4905, pp. 294–308. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78163-9_25
52. Sistla, A.P., Žefran, M., Feng, Y.: Runtime monitoring of stochastic cyber-physical systems with hybrid state. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 276–293. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29860-8_21

53. Spaan, M.T.J.: Partially observable Markov decision processes. In: Wiering, M., van Otterlo, M. (eds.) Reinforcement Learning, Adaptation, Learning, and Optimization, vol. 12, pp. 387–414. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27645-3_12
54. Stoller, S.D., et al.: Runtime verification with state estimation. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 193–207. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29860-8_15
55. Tappler, M., Aichernig, B.K., Bacci, G., Eichlseder, M., Larsen, K.G.: L^*-based learning of Markov decision processes. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) FM 2019. LNCS, vol. 11800, pp. 651–669. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30942-8_38
56. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. Intelligent Robotics and Autonomous Agents. MIT Press (2005)
57. Wilcox, C.M., Williams, B.C.: Runtime verification of stochastic, faulty systems. In: Barringer, H., et al. (eds.) RV 2010. LNCS, vol. 6418, pp. 452–459. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16612-9_34
58. Woo, H., Mok, A.K.: Real-time monitoring of uncertain data streams using probabilistic similarity. In: RTSS, pp. 288–300. IEEE CS (2007)
59. Zhang, L., Hermanns, H., Jansen, D.N.: Logic and model checking for hidden Markov models. In: Wang, F. (ed.) FORTE 2005. LNCS, vol. 3731, pp. 98–112. Springer, Heidelberg (2005). https://doi.org/10.1007/11562436_9