Runtime Monitors for Markov Decision Processes

Sebastian Junges, Hazem Torfah, and Sanjit A. Seshia

University of California at Berkeley, USA

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 convex hull algorithms improve the practical runtime, they do not 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 deployment of safety critical (cyber-physical) systems [41,12,29,44]. 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 deployed 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 their 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 Figure 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). The sensor is a stochastic process [50] that translates the environment state into an observation. For example, this could be a perception module on a plane that estimates the movements of an on-ground vehicle, as depicted in Figure 1(a). We are interested in measures of the system state, e.g., whether the plane will crash with the vehicle within a given number of steps, or what the expected number of steps to a dangerous state is. We abstract these measures by a *state risk*. The monitor must infer from a sequence 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 after observing the trace, times the risk imposed by this state. The monitoring problem is to decide whether for every possible scheduler resolving the nondeterminism the trace risk of a given trace exceeds a threshold.

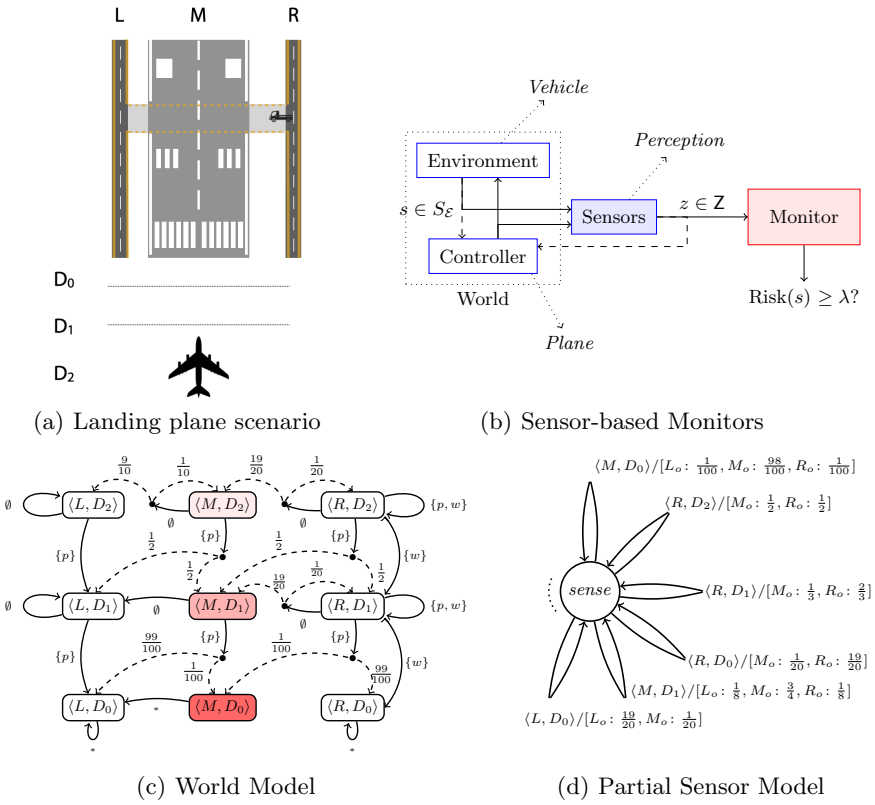
Monitoring of systems that contain either only stochastic or only nondeterministic behavior is typically done based on a filtering approach. Intuitively, the monitor estimates the possible system states based on the modelled dynamics. These approaches are rather efficient. For purely nondeterministic systems, a set of states needs to be tracked, and purely probabilistic systems require tracking a distribution over states. For systems that contain both probabilistic and nondeterministic behavior, the problem more challenging. In particular, we show that filtering on MDPs results in an exponential memory blowup. We show that a reduction taking the convex hull is essential for practical performance, but does not alleviate 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 apply probabilistic model checking. This alternative approach yields a polynomial-time algorithm. Indeed, our empirical evaluation show the overall 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 feasible 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 approaches have to deal with new challenges, and show that a tractable solution exists. In Sect. 3.1, we investigate *forward filtering*, used to estimate the possible system states in partially observable settings. We show that this approach is tractable for systems have probabilistic *or* nondeterministic uncertainty, but not for systems that have both. In Sect. 3.2 we consider model checking as a more tractable alternative. This result utilises results from the analysis of *partially observable MDPs* and model checking MDPs with *conditional properties*. In Sect. 4 we present baseline implementations of these algorithm, 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 the remainder of the paper with a motivating example. We review related work at the end of the paper[1].

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 Figure 1(a). On the ground, close to the runway, maintenance vehicles may cross the runway.

[1] Reviewers: supplementary material is available on GitHub: `https://git.io/JTOZF`



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

(c) World Model (d) Partial Sensor Model

Figure 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.

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 [32], i.e., for every position of V, the perception module reports a noisy variant of the position of V. However, it is important to realize 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 Figure 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 the car. The (simple) model of the perception module is given in Figure 1(d). For example, if the plane is in zone D_2, and V is in R then there is high chance that the perception module will return that V is on the runway. The probability of incorrectly detecting the position of V 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? For example, if with high probability the world is 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 under Imprecise Sensors

In this section, we formalize the problem of monitoring with imprecise sensors for MDPs. In the first part of the section, we show that the dynamics of the system under inspection can be modeled by an MDP by defining a joint MDP derived from two MDPs of the sensors and world model of the system. We then state of our results for MDPs in general.

2.1 Model

We start with a recap of MDPs. For a countable set X, let $\mathsf{Distr}(X) \subset (X \to [0,1])$ define all distributions over X, i.e., for $d \in \mathsf{Distr}(X)$ it holds that $\Sigma_{x \in X} d(x) = 1$. For $\mu \in \mathsf{Distr}(X)$, let $\mathsf{supp}(\mu)$ denote the *support*, $\mathsf{supp}(\mu) := \{x \mid \mu(x) > 0\}$. A distribution μ is *Dirac*, if $|\mathsf{supp}(\mu)| = 1$.

Definition 1 (MDP). *A (partially observable)* Markov decision process *(MDP) is a tuple* $\mathcal{M} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}, \iota_{\mathsf{o}} \rangle$ *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* $\mathsf{obs} \colon S \times \mathsf{Act} \to \mathsf{Distr}(\mathsf{Z})$ *is a* observation function *and* $\iota_{\mathsf{o}} \in \mathsf{Distr}(\mathsf{Z})$ *is the* initial observation.

We denote $\mathsf{AvAct}(s) = \{\alpha \mid P(s,\alpha) \neq \bot\}$. 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 $|\mathsf{AvAct}(s)| = 1$, 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, obs, ι_{o} 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 $\Pi_{\mathcal{M}}$. The *length* of the path is given by the number of actions along the path. The set $\Pi^n_{\mathcal{M}}$ for some $n \in \mathbb{N}$ denotes the set of finite paths of length n. We use $\pi_{\downarrow}$ 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:

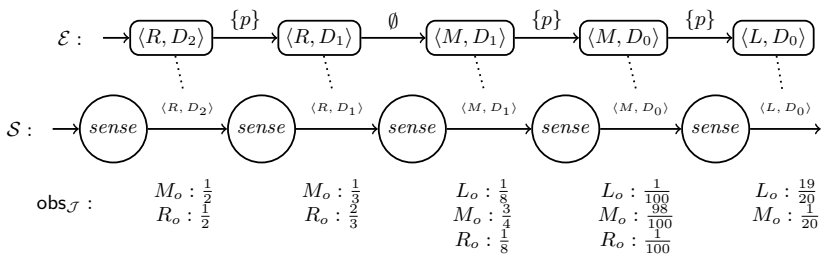


Figure 2. A run with its observations of the inspected system $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!]$ where $\mathcal{E}$ and $\mathcal{S}$ are the models given in Figure 1.

Definition 2 (Trace observation function). *For an observation function* $\mathsf{obs}\colon S \times \mathsf{Act} \to \mathsf{Z}$*, the* trace observation function $\mathsf{obs}_{\mathsf{tr}}\colon \Pi \to \mathsf{Distr}(\mathsf{Z}^+)$ *for* obs *is inductively defined with*

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

and $\mathsf{obs}_{\mathsf{tr}}(\pi)(\tau) = 0$ *otherwise (if the lengths of* π *and* τ *do not match).*

An MDP defining the system dynamics. We define the MDP defining the dynamics of the world and sensors that we use as basis for our monitor as follows. 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 $\mathcal{S} = \langle S_{\mathcal{S}}, \iota_{\mathcal{S}}, S_{\mathcal{E}}, P_{\mathcal{S}}, \mathsf{Z}, \mathsf{obs}, \iota\mathsf{o}\rangle$, the *inspected system* is defined by an MDP $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!] = \langle S_{\mathcal{J}}, \iota_{\mathcal{J}}, \mathsf{Act}_{\mathcal{E}}, P_{\mathcal{J}}, \mathsf{Z}, \mathsf{obs}_{\mathcal{J}}, \iota\mathsf{o}\rangle$ being the synchronous composition of $\mathcal{E}$ and $\mathcal{S}$:

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

In Figure 2 we illustrate a run of $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!]$ for the world and sensor MDPs presented in Figure 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 previously received observations.

In the remainder of the paper, if not stated otherwise, we state our results for general $\mathcal{M} = \langle S, \iota, \mathsf{Act}, P, \mathsf{Z}, \mathsf{obs}, \iota\mathsf{o}\rangle$ rather than the joint MDP $[\![\langle \mathcal{E}, \mathcal{S}\rangle]\!]$.

2.2 Formal Problem Statement

We formalize the monitoring problem. Towards this, we formalize the (conditional) probabilities and risks of paths and traces. As standard, we need resolve any nondeterminism by means of a scheduler.

Definition 3 (Scheduler). *A* scheduler *for MDP $\mathcal{M}$ is a function $\sigma\colon \Pi_{\mathcal{M}} \to \mathsf{Distr}(\mathsf{Act})$ with $\mathsf{supp}\sigma(\pi) \subseteq \mathsf{AvAct}(\pi_\downarrow)$ for every $\pi \in \Pi_{\mathcal{M}}$.*

Let *Sched*($\mathcal{M}$) denote the set of schedulers. We fix a scheduler $\sigma \in \mathit{Sched}(\mathcal{M})$. The probability $\mathsf{Pr}_\sigma(\pi)$ of a path π (under this scheduler) is the product of the transition probabilities in the induced Markov chain, for details see [8]. The probability $\mathsf{Pr}(\tau \mid \pi)$ of a trace τ for a fixed path π is $\mathsf{obs}_{\mathsf{tr}}(\pi)(\tau)$ and the probability $\mathsf{Pr}_\sigma(\tau)$ of a trace τ is $\sum_\pi \mathsf{Pr}_\sigma(\pi) \cdot \mathsf{Pr}_\sigma(\tau \mid \pi)$. Using Bayes' rule, we obtain

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

Let $r\colon S \to \mathbb{R}_{\geq 0}$ map states in $\mathcal{M}$ to some risk in $\mathbb{R}_{\geq 0}$. In our experiments, we flexibly define r using the (expected reward extension of the) temporal logic PCTL [8]. We call r a *state-risk function* for $\mathcal{M}$. The unconditioned risk after n steps, for a fixed scheduler, is then the weighted sum over all paths, multiplied by the state risk from the last state of the path onwards, i.e., $\sum_\pi \mathsf{Pr}_\sigma(\pi) \cdot r(\pi_\downarrow)$. If we take into account a trace τ, then we are interested in the *conditioned risk* $\sum_{\pi \in \Pi_{\mathcal{M}}^{|\tau|}} \mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow)$. Finally, the risk of a trace is conservatively over-approximated by assuming an adversarial scheduler, that is, by taking the supremum risk estimate over all schedulers[2]. We obtain the following problem:

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

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

If $\lambda = 0$, we call the problem the *qualitative monitoring problem*. These problems are (almost) equivalent on Kripke structures:

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

The essential idea in this statement is that a single path to a state with a sufficiently high state-risk suffices.

[2] We later see in Lemma 5 that this is indeed a maximum.

3 Algorithms for Monitoring of MDPs

We present two types of algorithms for the monitoring problem. The first algorithm is based on the widespread (forward) filtering approach, the second is based on model checking conditional probabilities. We show that while the filtering approach is efficacious in a purely nondeterministic or a purely probabilistic setting, it does not scale on models that are both probabilistic and nondeterministic. In those models, model checking provides a tractable alternative.

3.1 Forward Filtering for State Estimation

Below, we show that filtering does not scale well on MDPs. We briefly study how filtering needs limited memory to solve the monitoring problem for either the probabilistic or nondeterministic case. Then, we show that for MDPs, the approach needs to consider a finite but exponential set of distributions.

State estimators for Kripke structures. The monitor has to track the system state. It in general cannot determine this state exactly, but it can maintain a set of possible states that all agree with the observed trace. This set of states is inductively characterized by the function $\mathsf{est}_{\mathsf{KS}} \colon \mathsf{Z}^+ \to 2^S$.

Definition 4 (KS state estimator). *We define* $\mathsf{est}_{\mathsf{KS}} \colon \mathsf{Z}^+ \to 2^S$ *using*

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

For a KS and a given trace τ, the monitoring problem can be solved by computing $\mathsf{est}_{\mathsf{KS}}(\tau)$. For a KS $\mathcal{M}$ and trace τ, it holds that $R_r(\tau) = \max_{\mathsf{est}_{\mathsf{KS}}(\tau)(s)} r(s)$. Computing $R_r(\tau)$ requires time $\mathcal{O}(|\tau| \cdot |P|)$ and space $\mathcal{O}(|S|)$.

State estimators for Markov chains. For MCs, it is not sufficient to keep track of the potential system states. Instead, the transition probabilities need to be taken into account. If the system is (observation-)deterministic, we can adapt the notion of beliefs, similar to RVSE [48], and similar to the construction of belief MDPs for *partially observable MDPs*, cf. [47]:

Definition 5 (Belief). *For a system* $\mathcal{M}$*, 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. We assume further for conciseness that $\frac{0}{0} = 0$. A state estimator based on Bel is then defined as follows [45,51,48][3]:

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

Definition 6 (MC state estimator). *We define* $\mathsf{est}_{\mathsf{MC}} \colon \mathsf{Z}^+ \to \mathsf{Bel}$ *using* $\mathsf{est}_{\mathsf{MC}}(z) := \iota$ *if* $\iota\mathsf{o}(z) > 0$*, and* $\mathsf{est}_{\mathsf{MC}}(z) := \mathbf{0}$ *otherwise. Furthermore,*

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

Example 1. Reconsider Figure 1 and assume our monitor observes the trace $\tau = R_o \cdot M_o \cdot L_o$. Since we are looking into the deterministic case, we assume that the MDP has only actions labeled with $\{p\}$. The belief of τ can be computed as follows:

$$\begin{aligned}
\mathsf{est}_{\mathsf{MC}}(R_o) &= \{\langle R, D_2\rangle \mapsto 1\} \\
\mathsf{est}_{\mathsf{MC}}(R_o \cdot M_o) &= \{\langle R, D_1\rangle \mapsto \tfrac{\frac{1}{2}\cdot\frac{1}{3}}{\frac{1}{2}\cdot\frac{1}{3}+\frac{1}{2}\cdot\frac{3}{4}} = \frac{4}{13}, \langle M, D_1\rangle \mapsto \tfrac{\frac{1}{2}\cdot\frac{3}{4}}{\frac{1}{2}\cdot\frac{1}{3}+\frac{1}{2}\cdot\frac{3}{4}} = \frac{9}{13}\} \\
\mathsf{est}_{\mathsf{MC}}(R_o \cdot M_o \cdot L_o) &= \{\langle M, D_0\rangle \mapsto 0.0001, \langle L, D_0\rangle \mapsto 0.999\}
\end{aligned}$$

For every τ, it holds that $R_r(\tau) = \sum_{s \in S_{\mathcal{J}}} \mathsf{est}_{\mathsf{MC}}(\tau)(s) \cdot r(s)$. Computing $\mathsf{est}_{\mathsf{MC}}(\tau \cdot z)$ from $\mathsf{est}_{\mathsf{MC}}(\tau)$ can be done in $\mathcal{O}(|S| \cdot |P|)$ time, and we can solve the monitoring problem using $|S|$ many rational numbers. The size of the rationals may grow linearly in τ. One may use fixed-precision or floating-point numbers that over-approximates the probability of being in any state—inducing a growing (but conservative) error.

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:

Definition 7 (MDP state estimator). *We define* $\mathsf{est}_{\mathsf{MDP}} \colon \mathsf{Z}^+ \to 2^{\mathsf{Bel}}$ *using*

$$\begin{aligned}
\mathsf{est}_{\mathsf{MDP}}(z) \quad &= \{\iota \mid \iota\mathsf{o}(z) > 0\}, \\
\mathsf{est}_{\mathsf{MDP}}(\tau \cdot z) &= \Big\{ \mathsf{bel}' \in \mathsf{Bel} \;\Big|\; \exists \mathsf{bel} \in \mathsf{est}_{\mathsf{MDP}}(\tau).\ \mathsf{bel}' \in \mathsf{est}^{\mathsf{up}}_{\mathsf{ND}}(\mathsf{bel}, z) \Big\},
\end{aligned}$$

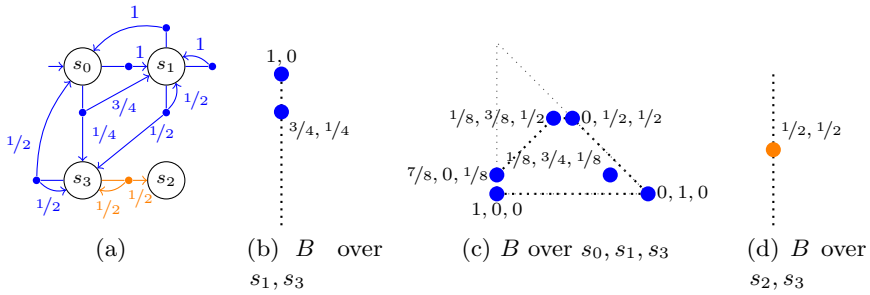
and where $\mathsf{bel}' \in \mathsf{est}^{\mathsf{up}}_{\mathsf{ND}}(\mathsf{bel}, z)$ *if there exists* $\varsigma \colon S \to \mathsf{Distr}(\mathsf{Act})$ *such that:*

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

The definition conservatively extends both Def. 4 and Def. 6. Furthermore, we remark that we do not restrict how the nondeterminism is resolved: any (measurable) distribution over actions can be chosen, and the distributions may be different for different traces.

Theorem 1. *For every* τ*, it holds that* $R_r(\tau) = \sup\limits_{\mathsf{bel} \in \mathsf{est}_{\mathsf{MDP}}(\tau)} \sum_{s \in S} \mathsf{bel}(s) \cdot r(s)$.

In the remainder of 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.



(a) (b) B over s_1, s_3 (c) B over s_0, s_1, s_3 (d) B over s_2, s_3

Figure 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.

Scalability of $\mathsf{est}_{\mathsf{MDP}}$. We can interpret a belief $\mathsf{bel} \in \mathsf{Bel}$ as point in an $\mathbb{R}^{(|S|-1)}$. 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 $B \setminus \{b\}$. All other beliefs are (extremal) vertices. Let the set $\mathcal{V}(B)$ denote the set the vertices of B. Intuitively, $\mathcal{V}(B)$ is the convex hull of B. As a consequence of linear-programming, to optimize the weighted risk, when a trace τ is observed, it suffices to consider $\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))$.

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

Example 2. Consider Fig. 3(a) with initial single observation z_0. All observation are Dirac, and only the action drawn between s_3 and s_2 has observation z_1. The beliefs after observing $z_0 z_0$ 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 after observing $z_0 z_0 z_0$ are distributions over s_0, s_1, s_3, depicted in Fig. 3(c). The convex hull contains five beliefs. The belief after observing $z_0 z_0 z_1$ is in Fig. 3(d).

Remark 1. 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 $\mathsf{est}_{\mathsf{MDP}}(\tau)$.

The convex hull allows us to ignore points that are not a vertex from the computation of risk. In fact, we can discard interior beliefs all together: All operations in computing a new belief are convex-set preserving[4]. This can be determined inductively as follows.

Lemma 2. *Let* $\mathsf{est}^{\mathsf{up}}_{\mathsf{ND}}(B, z)$ *denote* $\bigcup_{\mathsf{bel} \in B} \mathsf{est}^{\mathsf{up}}_{\mathsf{ND}}(\mathsf{bel}, z)$*. Then:*

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

An important consequence of this is that in Def. 7, considering deterministic schedulers suffices. Nevertheless, the estimator may track an exponential number of beliefs.

[4] The scaling is a projection.

Lemma 3. *There exists a family of MDPs $\mathcal{M}_n$ with $2n+3$ states such that $|\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))| = 2^n$ for every τ with $|\tau| > 2$.*

Furthermore, observe that updating bel yields up to $\prod_s |\mathsf{Act}(s)|$ new beliefs, a prohibitively large number. The remaining question is whether we need to track all these beliefs. First, all vertices can induce the maximal weighted trace risk, which is again a simple consequence of linear programming results.

Lemma 4. *For every τ and every $\mathsf{bel} \in \mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))$ there exists an r s.t.*

$$\sum_{s\in S} \mathsf{bel}(s)\cdot r(s) \geq \max_{\mathsf{bel}'\in\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))\setminus\{\mathsf{bel}\}} \sum_{s\in S} \mathsf{bel}'(s)\cdot r(s).$$

Second: even for a fixed state risk, we cannot prune any of these vertices.

Theorem 3. *There exist MDPs $\mathcal{M}_n$ and τ as in Lemma 3 with $B := \mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))$, such that 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 $B' = \mathsf{est}^{\mathsf{up}}_{\mathsf{ND}}(B\setminus\{\mathsf{bel}\},\tau')$.*

It is helpful to think about this theorem as a game between monitor and environment: The statement says that for every belief the monitor decides to drop, the environment may produce an observation trace τ' that will lead the monitor to underestimate the weighted risk at $R_r(\tau\cdot\tau')$.
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.

Example 3. Reconsider observing z_0z_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 $z_0z_0z_1$, 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.

3.2 Unrolling with Model Checking

We present a tractable algorithm for the monitoring problem. Contrary to filtering, this method incorporate 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. In each step, 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 [42], but that algorithm has exponential time complexity.

We now present a tractable algorithm for the quantitative case, where we need to remember all paths. We do this efficiently by storing an MDP with these paths using ideas from [9,18]. 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 [38]. It remains to show a P-time algorithm[5], which is outlined below. Roughly, the algorithm constructs an MDP $\mathcal{M}'''$ from $\mathcal{M}$ in three conceptual steps, such that the 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 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 [18, 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 [40], and can thus be computed in polynomial time.

Example 4. We illustrate the construction in Fig. 4. In Fig. 4(a), we depict an MDP $\mathcal{M}$, with $\iota = \{s_0, s_1 \mapsto 1/2\}$ and $\iota\mathsf{o} = z_0$. Furthermore, let $\tau = z_0 z_0$ and let $r(s_0) = 1$ and $r(s_1) = 2$. Let $\mathsf{obs}(s_0, \alpha) = \{z_0 \mapsto 1\}$, $\mathsf{obs}(s_1, \beta) = \{z_0 \mapsto 1/3, z_1 \mapsto 2/3\}$, $\mathsf{obs}(s_1, \alpha) = \{z_0 \mapsto 1/4, z_1 \mapsto 3/4\}$. State s_1 can be reached with different observations, 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,

[5] 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.

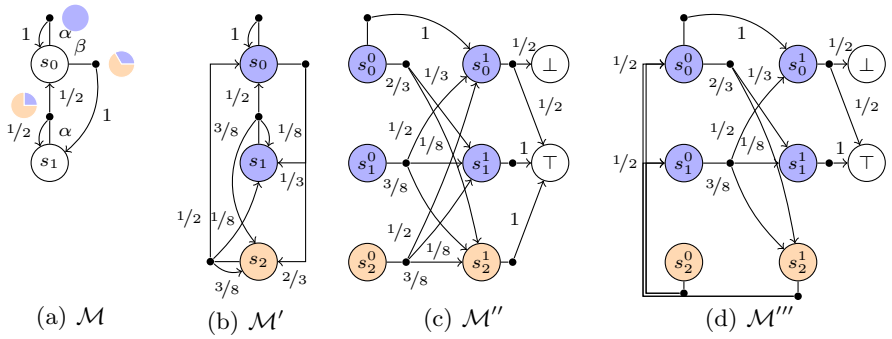


Figure 4. Polynomial-time algorithm for solving Problem 1 illustrated.

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 5. *For every τ, it holds that*

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

By reducing step-bounded reachability: this set of schedulers is necessary [4].

4 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. 5. 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. Actions are resolved uniformly in this simulator. 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 after observing $z_0 \dots z_i$. We flexibly combine these components via a Python API.

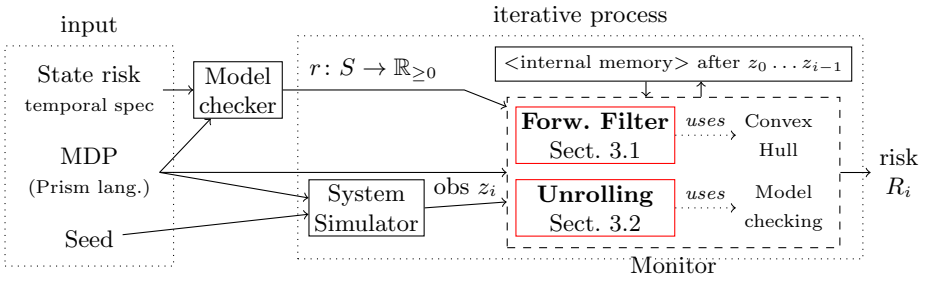


Figure 5. Schematic setup for prototype mapping a stream $z_0 \dots z_k$ to a stream $r_0 \dots r_k$.

For filtering as in Sect. 3.1, we provide a sparse data structure for beliefs that is updated using only deterministic schedulers. This is sufficient, see Lemma 2. To further prune the set of beliefs, we implement an SMT-driven elimination [43] of interior beliefs, inside of the convex hull[6]. We construct the unrolling as described in Sect. 3.2 and apply model checking via any sparse engines in Storm.

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, [40]) 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 [23] 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

[6] 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.

Id	Name	Inst	$\|S\|$	$\|P\|$	$\|\tau\|$	Forward Filtering							Unrolling				
						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

the form of a cardinal direction, and nondeterministically decides to move or 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

Table 2. Summarized performance for online monitoring

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

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 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 approach have its 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 in incentive 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[7]. *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.

[7] 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.

5 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 [45,51,48] and has been extended to hybrid systems [46]. 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 [22] 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 [52] monitors a noisy sensor to reduce energy consumption.

State beliefs are studied when verifying HMMs [53], where the question whether a sequence of observation likely occurs, or which HMM is an adequate representation of a system [35]. State beliefs are prominent in the verification of partially observable MDPs [37,31,16], 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 [19], but automatic techniques for those very general models is lacking. Likewise, the idea of *shielding* (pre)computes all action choices that lead to safe behavior [15,3,24,34,5,33]. For partially observable settings, shielding again requires to compute partial-information schedulers [36,20], 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 [39] 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 [21], 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 [1,49]. 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..

6 Conclusion

We have presented a first framework for monitoring based on a trace on models that combine nondeterminism and probabilities. Future work include heuristics for approximate monitoring and for faster convex hull computations. We also want to extend the work to learned models.

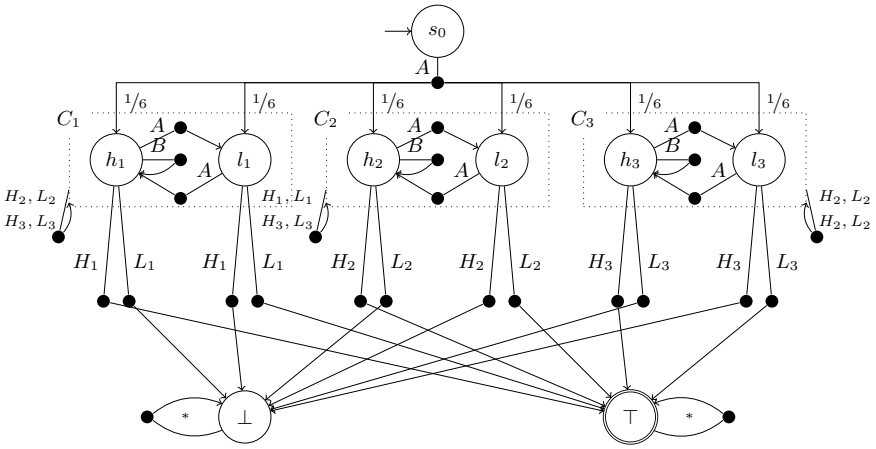
References

1. Bernhard K. Aichernig, Roderick Bloem, Masoud Ebrahimi, Martin Horn, Franz Pernkopf, Wolfgang Roth, Astrid Rupp, Martin Tappler, and Markus Tranninger. Learning a behavior model of hybrid systems through combining model-based testing and machine learning. In *ICTSS*, volume 11812 of *LNCS*, pages 3–21. Springer, 2019.
2. Bernhard K. Aichernig and Martin Tappler. Probabilistic black-box reachability checking (extended version). *Formal Methods Syst. Des.*, 54(3):416–448, 2019.
3. Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI*, pages 2669–2678. AAAI Press, 2018.
4. Suzana Andova, Holger Hermanns, and Joost-Pieter Katoen. Discrete-time rewards model-checked. In *FORMATS*, volume 2791 of *LNCS*, pages 88–104. Springer, 2003.
5. Guy Avni, Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, Bettina Könighofer, and Stefan Pranger. Run-time optimization for learned controllers through quantitative games. In *CAV (1)*, volume 11561 of *LNCS*, pages 630–649. Springer, 2019.
6. Reza Babaee, Arie Gurfinkel, and Sebastian Fischmeister. *P*revent : A predictive run-time verification framework using statistical learning. In *SEFM*, volume 10886 of *LNCS*, pages 205–220. Springer, 2018.
7. Reza Babaee, Arie Gurfinkel, and Sebastian Fischmeister. Predictive run-time verification of discrete-time reachability properties in black-box systems using trace-level abstraction and statistical learning. In *RV*, volume 11237 of *LNCS*, pages 187–204. Springer, 2018.
8. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
9. Christel Baier, Joachim Klein, Sascha Klüppelholz, and Steffen Märcker. Computing conditional probabilities in Markovian models efficiently. In *TACAS*, volume 8413 of *LNCS*, pages 515–530. Springer, 2014.
10. C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.
11. Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009.
12. Ezio Bartocci, Jyotirmoy V. Deshmukh, Alexandre Donzé, Georgios E. Fainekos, Oded Maler, Dejan Nickovic, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS*, pages 135–175. Springer, 2018.
13. Nathalie Bertrand, Eric Fabre, Stefan Haar, Serge Haddad, and Loïc Hélouët. Active diagnosis for probabilistic systems. In *FoSSaCS*, volume 8412 of *LNCS*, pages 29–42. Springer, 2014.
14. Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. A tale of two diagnoses in probabilistic systems. *Inf. Comput.*, 269, 2019.
15. Roderick Bloem, Bettina Könighofer, Robert Könighofer, and Chao Wang. Shield synthesis: - runtime enforcement for reactive systems. In *TACAS*, volume 9035 of *LNCS*, pages 533–548. Springer, 2015.
16. Alexander Bork, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Verification of indefinite-horizon POMDPs. In *ATVA*, volume 12302 of *LNCS*. Springer, 2020. To appear.

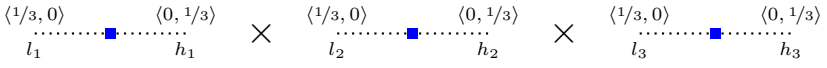
17. Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A. Smolka, and Scott D. Stoller. Neural predictive monitoring. In *RV*, volume 11757 of *LNCS*, pages 129–147. Springer, 2019.
18. Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.*, 234:26–48, 2016.
19. Krishnendu Chatterjee and Laurent Doyen. Partial-observation stochastic games: How to win when belief fails. *ACM Trans. Comput. Log.*, 15(2):16:1–16:44, 2014.
20. Krishnendu Chatterjee, Petr Novotný, Guillermo A. Pérez, Jean-François Raskin, and Dorde Zikelic. Optimizing expectation with guarantees in POMDPs. In *AAAI*, pages 3725–3732. AAAI Press, 2017.
21. Yi Chou, Hansol Yoon, and Sriram Sankaranarayanan. Predictive runtime monitoring of vehicle models using bayesian estimation and reachability analysis. In *IROS*, 2020. to appear.
22. Alessandro Cimatti, Chun Tian, and Stefano Tonetta. Assumption-based runtime verification with partial observability and resets. In *RV*, volume 11757 of *LNCS*, pages 165–184. Springer, 2019.
23. Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
24. Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Permissive controller synthesis for probabilistic systems. *Log. Methods Comput. Sci.*, 11(2), 2015.
25. Marcus Gerhold and Mariëlle Stoelinga. Model-based testing of probabilistic systems. *Formal Asp. Comput.*, 30(1):77–106, 2018.
26. Radu Grigore and Stefan Kiefer. Selective monitoring. In *CONCUR*, volume 118 of *LIPIcs*, pages 20:1–20:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
27. Serge Haddad and Benjamin Monmege. Interval iteration algorithm for MDPs and IMDPs. *Theor. Comput. Sci.*, 735:111–131, 2018.
28. Arnd Hartmanns and Benjamin Lucien Kaminski. Optimistic value iteration. In *CAV (2)*, volume 12225 of *LNCS*, pages 488–511. Springer, 2020.
29. Klaus Havelund and Grigore Rosu. Runtime verification - 17 years later. In Christian Colombo and Martin Leucker, editors, *Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018, Proceedings*, volume 11237 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2018.
30. Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker storm. *CoRR*, abs/2002.07080, 2020.
31. Karel Horák, Branislav Bosanský, and Krishnendu Chatterjee. Goal-HSVI: Heuristic search value iteration for goal POMDPs. In *IJCAI*, pages 4764–4770. ijcai.org, 2018.
32. Nils Jansen, Laura R. Humphrey, Jana Tumova, and Ufuk Topcu. Structured synthesis for probabilistic systems. In *NFM*, volume 11460 of *LNCS*, pages 237–254. Springer, 2019.
33. Nils Jansen, Bettina Könighofer, Sebastian Junges, Alex Serban, and Roderick Bloem. Safe reinforcement learning using probabilistic shields (invited paper). In *CONCUR*, volume 171 of *LIPIcs*, pages 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
34. Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-constrained reinforcement learning for MDPs. In *TACAS*, volume 9636 of *LNCS*, pages 130–146. Springer, 2016.
35. Stefan Kiefer and A. Prasad Sistla. Distinguishing hidden Markov chains. In *LICS*, pages 66–75. ACM, 2016.

36. Wonhong Nam and Rajeev Alur. Active learning of plans for safety and reachability goals with partial observability. *IEEE Trans. Syst. Man Cybern. Part B*, 40(2):412–420, 2010.
37. Gethin Norman, David Parker, and Xueyi Zou. Verification and control of partially observable probabilistic systems. *Real Time Syst.*, 53(3):354–402, 2017.
38. Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.
39. Dung Phan, Nicola Paoletti, Radu Grosu, Nils Jansen, Scott A. Smolka, and Scott D. Stoller. Neural simplex architecture. *CoRR*, abs/1908.00528, 2019. Appears at NFM2020.
40. Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.
41. César Sánchez, Gerardo Schneider, Wolfgang Ahrendt, Ezio Bartocci, Domenico Bianculli, Christian Colombo, Yliès Falcone, Adrian Francalanza, Srdan Krstic, João M. Lourenço, Dejan Nickovic, Gordon J. Pace, José Rufino, Julien Signoles, Dmitriy Traytel, and Alexander Weiss. A survey of challenges for runtime verification from advanced application domains (beyond software). *Formal Methods Syst. Des.*, 54(3):279–335, 2019.
42. Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.
43. Raimund Seidel. Convex hull computations. In *Handbook of Discrete and Computational Geometry, 2nd Ed*, pages 495–512. Chapman and Hall/CRC, 2004.
44. Sanjit A. Seshia. Introspective environment modeling. In *RV*, volume 11757 of *LNCS*, pages 15–26. Springer, 2019.
45. A. Prasad Sistla and Abhigna R. Srinivas. Monitoring temporal properties of stochastic systems. In *VMCAI*, volume 4905 of *LNCS*, pages 294–308. Springer, 2008.
46. A. Prasad Sistla, Miloš Žefran, and Yao Feng. Runtime monitoring of stochastic cyber-physical systems with hybrid state. In *RV*. Springer, 2012.
47. Matthijs T. J. Spaan. Partially observable Markov decision processes. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 387–414. Springer, 2012.
48. Scott D. Stoller, Ezio Bartocci, Justin Seyster, Radu Grosu, Klaus Havelund, Scott A. Smolka, and Erez Zadok. Runtime verification with state estimation. In *RV*, volume 7186 of *LNCS*, pages 193–207. Springer, 2011.
49. Martin Tappler, Bernhard K. Aichernig, Giovanni Bacci, Maria Eichlseder, and Kim G. Larsen. L^*-based learning of Markov decision processes. In *FM*, volume 11800 of *LNCS*, pages 651–669. Springer, 2019.
50. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.
51. Cristina M. Wilcox and Brian C. Williams. Runtime verification of stochastic, faulty systems. In Howard Barringer, Yliès Falcone, Bernd Finkbeiner, Klaus Havelund, Insup Lee, Gordon J. Pace, Grigore Rosu, Oleg Sokolsky, and Nikolai Tillmann, editors, *Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, volume 6418 of *LNCS*, pages 452–459. Springer, 2010.
52. Honguk Woo and Aloysius K. Mok. Real-time monitoring of uncertain data streams using probabilistic similarity. In *RTSS*, pages 288–300. IEEE Computer Society, 2007.

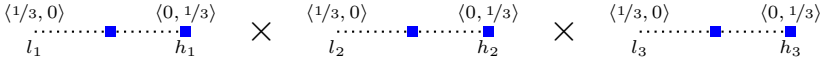
53. Lijun Zhang, Holger Hermanns, and David N. Jansen. Logic and model checking for hidden Markov models. In *FORTE*, volume 3731 of *LNCS*, pages 98–112. Springer, 2005.



(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. Self-loops at the components C_1, C_2, C_3 reflect self-loops on the states in the components.



(b) Beliefs after AA



(c) Beliefs after AAA

(d) Beliefs after $AAAA^+$

Figure 6. Construction for the correctness of Thm. 3.

A Construction for Theorem 3

Below, we outline a construction to prove the correctness of Theorem 3.

We construct $\mathcal{M}_n$ with $n = 3$, that is, $\mathcal{M}_3$ in Fig. 6(a). First, observe that for the subset of actions $\{A, B\}$, the induced sub-MDP (the MDP without actions labelled H_* or L_* induces 2^3 beliefs which are vertices. For this, observe how the belief factorizes into a belief within each component $C_i = \{h_i, l_i\}$, with the extremal probabilities 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. Thus, for any τ with $|\tau|$ we can compactly represent $\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(\tau))$ as bit-strings of length n. Concretely, the belief

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

It is easy to see that these are exponentially many beliefs for bit strings of length n.

Now consider the full MDP, with all actions. By taking an action H_i or L_i, the probability mass leaves both the low and high-state. We can represent states with extended bit-strings, concretely, we can think of

$$\{h_1, l_3, \top \mapsto 1/3, l_1, l_2, h_2, h_3, \bot \mapsto 0\} \text{ as } 1\checkmark 0, \text{ and}$$
$$\{\bot \mapsto 2/3, l_3, \top \mapsto 1/3, l_1, l_2, h_1, h_2, h_3, \bot \mapsto 0\} \text{ as } \times\times 0.$$

We define the risk as $r(\top) = 1$ and $r(s) = 0$ for all $s \neq \top$. Thus, the belief $\checkmark\checkmark\checkmark$ has maximal risk.

Now, we show that if we have $B := \mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(AAA)) = \mathbb{B}^n$. Let us fix $\mathsf{bel} := 100$. Then $B' = \mathsf{est}^{\mathsf{up}}_{\mathsf{ND}}(B \setminus \{\mathsf{bel}\}, H_1L_2L_3)$ does not contain $\mathsf{bel}' := \checkmark\checkmark\checkmark$, but $\mathcal{V}(\mathsf{est}_{\mathsf{MDP}}(AAAH_1L_2L_3))$ does. Thus, the estimated riks after $AAAH_1L_2L_3$ will be below the maximum if we remove bel' after seeing AAA. Symmetrically, we can show that all other beliefs need to remain in B.

B On qualitative variants of the problem

We may reformulate the qualitative problem as follows: First, define the risk as the maximum over reached states, i.e.,

$$R^{\max}_r(\tau) = \max_{\pi \in \Pi \text{ s.t. } \mathsf{obs}_{\mathsf{tr}}(\pi)(\tau) > 0} r(\pi_\downarrow).$$

This *maximum risk estimation* conservatively states the highest risk that can be achieved after an observation. The quantitative (qualitative) *maximal risk estimation problem* analogously is to decide $R^{\max}_r(\tau) > \lambda$ with $\lambda \geq 0$ ($\lambda = 0$).

Lemma 6. *The qualitative risk estimation problem and the quantitative and qualitative maximum risk estimation problem are all logspace interreducible.*

Proof (Lemma 1). The qualitative case is a special case of the quantitative case. Now consider the quantitative case, and assume some fixed $\lambda > 0$. Observe that for any fixed scheduler, $\sum_\pi \mathsf{Pr}_\sigma(\pi \mid \tau) = 1$, and that there is a unique path π such that $\mathsf{Pr}_\sigma(\pi \mid \tau) = 1$. We can reduce the quantitative maximum risk estimation problem to a qualitative monitoring problem as follows. We modify the state risk function by checking state-wise whether the risk at a state is above λ. If it is, then we set the state risk to 1, and to zero otherwise. If $R_r(\tau) > \lambda$ for some $\lambda \in \mathbb{R}_{\geq 0}$, then there is a path $\pi \in \Pi^{|\tau|}_{\mathcal{M}}$ with $r(\pi_\downarrow) > \lambda$. This in turn means that $\mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow) > 0$. The other way around, if the sum over all paths is zero, then there is no such path.

Proof (Lemma 6).

- *The qualitative weighted risk estimation problem and the qualitative maximum risk estimation problem coincide.*

The qualitative weighted risk estimation problem and the qualitative maximum risk estimation problem coincide. In the qualitative case $R_r(\tau) > 0$ holds when at least one path π matches the trace τ with a positive probability and $r(\pi_\downarrow) > 0$. In this case, $R_r^{\max}(\tau) > 0$ also holds, since $r(\pi_\downarrow) > 0$ and $\mathsf{obs}_{\mathsf{tr}}(\pi)(\tau) > 0$.
When $R_r^{\max}(\tau) > 0$ holds then there is a path π such that $r(\pi_\downarrow) > 0$ and $\mathsf{obs}_{\mathsf{tr}}(\pi)(\tau) > 0$. This implies that $\mathsf{Pr}(\sigma)(\pi \mid \tau) > 0$ and in turn that $R_r(\tau) > 0$. □

– *The qualitative weighted risk estimation problem and the quantitative maximum risk estimation problem are logspace interreducible.* From the qualitative weighted risk estimation problem to the qualitative maximum risk estimation problem is trivial and follows from the last lemma since the qualitative maximum risk estimation problem is an instance of the quantitative maximum risk estimation problem.
In the other direction, we can reduce the quantitative maximum risk estimation problem to a qualitative weighted risk estimation problem as follows. We modify the state risk function by checking state-wise whether the risk at a state is above λ. If it is, then we set the state risk to 1, and to zero otherwise. If $R_r^{\max}(\tau) > \lambda$ for some $\lambda \in \mathbb{R}_{\geq 0}$, then there is a path $\pi \in \Pi_{\mathcal{M}}^{|\tau|}$ with $r(\pi_\downarrow) > \lambda$. This in turn means that $\mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow) > 0$. Furthermore, if $R_r^{\max}(\tau) \leq \lambda$, then for all $\pi \in \Pi_{\mathcal{M}}^{|\tau|}$, it holds that $\mathsf{Pr}_\sigma(\pi \mid \tau) \cdot r(\pi_\downarrow) = 0$.
Both reduction can be performed in logspace. In the first direction we only need to set the lambda. In the second direction we just need to iterate over the states and change the risk function accordingly. □