Permissive Finite-state Controllers of POMDPs using Parameter Synthesis*

Sebastian Junges[1], Nils Jansen[2], Ralf Wimmer[3], Tim Quatmann[1], Leonore Winterer[3], Joost-Pieter Katoen[1], and Bernd Becker[3]

[1] RWTH Aachen University, Aachen, Germany
[2] Radboud University, Nijmegen, The Netherlands
[3] Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany

Abstract. We study finite-state controllers (FSCs) for partially observable Markov decision processes (POMDPs). The key insight is that computing (randomized) FSCs on POMDPs is equivalent to synthesis for parametric Markov chains (pMCs). This correspondence enables using parameter synthesis techniques to compute FSCs for POMDPs in a black-box fashion. We investigate how typical restrictions on parameter values affect the quality of the obtained FSCs. Permissive strategies for POMDPs are obtained as regions of parameter values, a natural output of parameter synthesis techniques. Experimental evaluation on several POMDP benchmarks shows promising results.

1 Introduction

We aim at providing guarantees for planning scenarios given by dynamical systems with uncertainties. In particular, we want to compute a *strategy* for an agent that ensures certain desired behavior [17]. A popular formal model for planning subject to stochastic behavior is a Markov decision process (MDPs). An MDP is a nondeterministic model in which the agent chooses to perform an action under full knowledge of the environment it is operating in. The outcome of the action is a probability distribution over the system states. Many applications, however, allow only *partial observability* of the current system state [21,35,38,31]. For such applications, MDPs are extended to *partially observable Markov decision processes* (POMDPs). While the agent acts within the environment, it encounters certain *observations*, according to which it can infer the likelihood of the system being in a certain state. This likelihood is called the *belief state*. Executing an action leads to an update of the belief state according to new observations. The belief state together with an update function form a (possibly uncountably infinite) MDP, commonly referred to as the *belief MDP* [33].

Strategy Computation. For (PO)MDPs, a strategy is a function that resolves the nondeterminism by either deterministically choosing an action at each time step, or providing a probability distribution over actions. The former is a *deterministic*

* This work is supported by the DFG RTG 2236 "UnRAVeL".

strategy, the latter a *randomized strategy*. In general, strategies depend on the full history of the current evolution of the (PO)MDP. For MDPs, the history is a sequence of states, and for POMDPs a sequence of observations. If a strategy depends only on the current state of the system, it is called *memoryless*. For MDPs, deterministic memoryless strategies suffice to induce optimal values according to our measures of interest [27]. Contrarily, POMDPs require randomized strategies and the full history [29]. These strategies can be represented by *infinite-state controllers*, for instance in case of infinite-horizon objectives. On top of that, strategies inducing *optimal* values are computed by assessing the entire belief MDP [23,3,34,25], rendering the problem undecidable [6].

For computational tractability, strategies are often restricted to finite memory; this amounts to using *randomized finite-state controllers* (FSCs) [24]. In what follows, we refer to such strategies as FSCs. The product of a POMDP and the memory of an FSC yields a POMDP with a larger state space. In this product, it suffices to compute a memoryless randomized strategy. Computing such a strategy is NP-hard, SQRT-SUM-hard, and in PSPACE [36]. While optimal values cannot be guaranteed, finite memory in combination with *randomization* may supersede infinite memory without randomization in many cases [7,1].

Problem and Approach. Computing FSCs for POMDPs is the problem we tackle in this paper. The key observation is that the *set of all FSCs* for a POMDP and a fixed memory bound can be succinctly represented by a *parametric Markov chain* (pMC). Such pMCs have transitions whose probabilities are not constants but symbolically given by *functions* over a finite set of parameters. The *parameter synthesis* problem for pMCs is to determine parameter instantiations that satisfy (or refute) a given specification. We show the direct correspondence between pMC parameter synthesis and the computation of a randomized strategy adhering to a specification. The advantage is that this correspondence enables to use sophisticated and optimized parameter synthesis tools like PARAM [16], PRISM [22], or PROPhESY [12]. However, several intricacies regarding the definition of FSCs and the parameter synthesis method need to be discussed. We detail our contributions and the structure of the paper below.

Table 1. Correspondence

POMDP under FSC	pMC
states $\times$ memory	states
same observation	same parameter
strategy	parameter instantiation
permissive strategy	region of instantiations

- In Sect. 3, we establish the correspondence between the set of all induced Markov chains for FSCs for a concrete POMDP on the one hand, and a pMC on the other hand. Consider Tab. 1. The product of a POMDP and an FSC yields a larger POMDP with state-memory pairs, these are mapped to states in the pMC. If POMDP states share an *observation*, the corresponding pMC states will share *parameters* at their emanating transitions. A *strategy*, potentially satisfying a quantitative reachability specification of a POMDP, corresponds to a *parameter instantiation* in the pMC. We discuss specific types of FSCs from the literature and their effect in our setting.

- Section 4 identifies the class of *simple pMCs* where parameters only occur in linear polynomials of a specific form. This class is practically relevant as most available pMC benchmarks are of this form. We discuss how this class coincides with POMDPs under memoryless strategies. We furthermore relate restrictions on parameter instantiations to restrictions on strategies.
- In Sect. 5, we show how to use parameter synthesis to obtain optimal FSCs.
- We introduce *permissive strategies* [14,20] as permissive FSCs for POMDPs in Sect. 6. Intuitively, permissiveness refers to *sets of FSCs* and corresponds to regions of parameter instantiations, see again Tab. 1.
- We evaluate the approach on a range of typical POMDP benchmarks. The approach seems orthogonal to, e. g., PRISM-POMDP [25], and is able to synthesize high-quality FSCs.

Related Work. In addition to the cited works, in [24] a branch-&-bound method finds optimal FSCs for POMDPs. A SAT-based approach computes FSCs for qualitative properties [4]. For a survey of decidability results and algorithms for broader classes of properties refer to [6,5]. Work on parameter synthesis [18,15,10] might contain valuable additions to the methods considered here.

2 Preliminaries

A *probability distribution* over a finite or countably infinite set X is a function $\mu\colon X \to [0,\,1] \subseteq \mathbb{R}$ with $\sum_{x\in X} \mu(x) = \mu(X) = 1$. The set of all distributions on X is $\mathit{Distr}(X)$. The support of a distribution μ is $\mathit{supp}(\mu) = \{x \in X \mid \mu(x) > 0\}$.

Let $V = \{p_1, \ldots, p_n\}$ be a set of *parameters* over the domain $\mathbb{R}$; $\mathbb{Q}[V]$ is the set of multivariate polynomials over V. An *instantiation* for V is a function $u\colon V \to \mathbb{R}$.

2.1 Parametric Probabilistic Models

Definition 1 (pMDP). *A* parametric Markov decision process *(pMDP) M is a tuple $M = (S,\, s_{\mathrm{I}}, \mathit{Act}, V, \mathcal{P})$ with a finite (or countably infinite) set S of* states*,* initial state *$s_{\mathrm{I}} \in S$, a finite set Act of* actions*, a finite set V of parameters, and a* transition function *$\mathcal{P}\colon S \times \mathit{Act} \times S \to \mathbb{Q}[V]$.*

The *available actions* in $s \in S$ are $\mathit{Act}(s) = \{a \in \mathit{Act} \mid \exists s' \in S : \mathcal{P}(s, a, s') \neq 0\}$. We assume the pMDP M contains no deadlock states, i. e., $\mathit{Act}(s) \neq \emptyset$ for all $s \in S$. A *parametric discrete-time Markov chain* (pMC) is a pMDP with $|\mathit{Act}(s)| = 1$ for all $s \in S$. For pMCs, we may omit the actions and use the notation $D = (S, s_{\mathrm{I}}, V, P)$ with a transition function P of the form $P\colon S \times S \to \mathbb{Q}[V]$.

A *path* of a pMDP M is an (in)finite sequence $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$, where $s_0 = s_{\mathrm{I}}$, $s_i \in S$, $a_i \in \mathit{Act}(s_i)$, and $\mathcal{P}(s_i, a_i, s_{i+1}) \neq 0$ for all $i \in \mathbb{N}$. For finite π, $\mathit{last}(\pi)$ denotes the last state of π. The set of (in)finite paths is $\mathsf{Paths}^{M}_{\mathit{fin}}$ (Paths^{M}).

Definition 2 (MDP). *A pMDP M is a* Markov decision process (MDP) *if* $\mathcal{P}\colon S \times Act \times S \to [0,1]$ *and* $\sum_{s' \in S} \mathcal{P}(s,a,s') = 1$ *for all* $s \in S$ *and* $a \in Act(s)$*.*

This is analogous for Markov chains, where the actions are omitted.

Applying an *instantiation* $u\colon V \to \mathbb{R}$ to a pMDP or pMC M, denoted $M[u]$, replaces each polynomial f in M by $f[u]$. We also call $M[u]$ the *instantiation* of M at u. An instantiation u is *well-defined* for M if the replacement yields probability distributions, i. e., if $M[u]$ is an MDP or an MC, respectively.

Strategies. To resolve the nondeterministic action choices in MDPs, so-called *strategies* determine at each state a distribution over actions to take. This decision may be based on the *history* of the current path.

Definition 3 (Strategy). *A* strategy σ *for M is a function* $\sigma\colon \mathsf{Paths}^M_{\mathit{fin}} \to \mathit{Distr}(Act)$ *s. t.* $\mathit{supp}\big(\sigma(\pi)\big) \subseteq Act\big(\mathit{last}(\pi)\big)$ *for all* $\pi \in \mathsf{Paths}^M_{\mathit{fin}}$*. Let* Σ^M *denote the set of all strategies of M.*

A strategy σ is *memoryless* if $\mathit{last}(\pi) = \mathit{last}(\pi')$ implies $\sigma(\pi) = \sigma(\pi')$ for all $\pi, \pi' \in \mathrm{dom}(\sigma)$. It is *deterministic* if $\sigma(\pi)$ is a Dirac distribution for all $\pi \in \mathrm{dom}(\sigma)$. A strategy that is not deterministic is *randomized*.

A strategy σ for an MDP M resolves all nondeterministic choices, yielding an *induced Markov chain* (MC) M^σ, for which a *probability measure* over the set of infinite paths is defined by the standard cylinder set construction [2].

Definition 4 (Induced Markov Chain). *Let* $M = (S, s_{\mathrm{I}}, Act, \mathcal{P})$ *be an MDP and* $\sigma \in \Sigma^M$ *a strategy for M. The MC induced by M and σ is given by* $M^\sigma = (\mathsf{Paths}^M_{\mathit{fin}}, s_{\mathrm{I}}, P^\sigma)$ *where:*

$$P^\sigma(\pi, \pi') = \begin{cases} \mathcal{P}(\mathit{last}(\pi), a, s') \cdot \sigma(\pi)(a) & \text{if } \pi' = \pi a s', \\ 0 & \text{otherwise.} \end{cases}$$

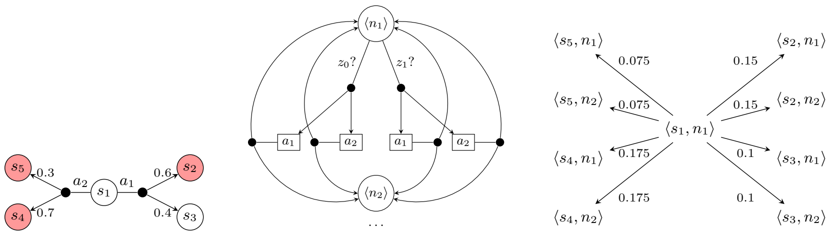
2.2 Partial Observability

We introduce MDPs in which states are only partially observable.

Definition 5 (POMDP). *A* partially observable MDP (POMDP) *is a tuple* $\mathcal{M} = (M, Z, O)$ *such that* $M = (S, s_{\mathrm{I}}, Act, \mathcal{P})$ *is the* underlying MDP *of* $\mathcal{M}$*,* Z *is a finite set of observations and* $O\colon S \to Z$ *is the observation function.*

We require that states with the same observations have the same set of enabled actions, i. e., $O(s) = O(s')$ implies $Act(s) = Act(s')$ for all $s, s' \in S$. We define $Act(z) = Act(s)$ if $O(s) = z$. More general observation functions [30,33] take the last action into account and provide a distribution over Z. There is a transformation of the general case to the POMDP definition used here that blows up the state space polynomially [5]. In Fig. 1(a), a fragment of the underlying MDP of a POMDP has two different observations, indicated by the state coloring.

We lift the observation function to paths: For a path $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots s_n \in \mathsf{Paths}^M_{\mathit{fin}}$, the associated *observation sequence* is $O(\pi) = O(s_0) \xrightarrow{a_0} O(s_1) \xrightarrow{a_1} \cdots O(s_n)$. Several paths in the underlying MDP may yield the same observation sequence. Strategies have to take this restricted observability into account.



(a) POMDP $\mathcal{M}$ (b) FSC $\mathcal{A}$ (c) Induced MC $\mathcal{M}^{\sigma_{\mathcal{A}}}$

Fig. 1. The (a) POMDP $\mathcal{M}$ has two observations $O(s_1) = O(s_3) = z_0$ (white) and $O(s_2) = O(s_4) = O(s_5) = z_1$ (red). The (b) associated (partial) FSC $\mathcal{A}$ has two memory nodes. The (c) MC $\mathcal{M}^{\sigma_{\mathcal{A}}}$ induced by $\mathcal{M}$ and $\mathcal{A}$ has then 9 states.

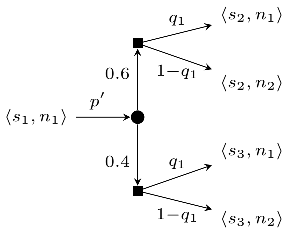
Definition 6. *An* observation-based strategy *for a POMDP $\mathcal{M}$ is a strategy σ for the underlying MDP M such that for all paths $\pi, \pi' \in \mathsf{Paths}_{fin}^{M}$ with $O(\pi) = O(\pi')$ we have $\sigma(\pi) = \sigma(\pi')$. $\Sigma^{\mathcal{M}}$ denotes the set of observation-based strategies for $\mathcal{M}$.*

An observation-based strategy selects actions based on the observations encountered along a path and the past actions. Applying the strategy to a POMDP yields an induced MC as in Def. 4, resolving all nondeterminism and partial observability. To represent observation-based strategies with finite memory, we introduce *finite-state controllers* (FSCs): We fix the syntax and semantics here, and discuss other definitions from the literature in Sect. 3. A randomized observation-based strategy for a POMDP $\mathcal{M}$ with (finite) k memory is represented by an FSC $\mathcal{A}$ with k memory nodes. If $k = 1$, the FSC describes a *memoryless strategy.* In the following, we directly refer to observation-based strategies as FSCs.

Definition 7 (FSC). *A* finite-state controller (FSC) *for a POMDP $\mathcal{M}$ is a tuple $\mathcal{A} = (N, n_0, \gamma, \delta)$ where N is a finite set of* memory nodes*, $n_0 \in N$ is the* initial memory node*, γ is the* action mapping *$\gamma \colon N \times Z \to Distr(Act)$, and δ is the* memory update *$\delta \colon N \times Z \times Act \to Distr(N)$. The set $FSC_k^{\mathcal{M}}$ denotes the set of FSCs with k memory nodes, called k-*FSC*s. Let $\sigma_{\mathcal{A}} \in \Sigma^{\mathcal{M}}$ denote the observation-based strategy represented by $\mathcal{A}$.*

From a node n and the observation z in the current state of the POMDP, the action a to be executed next is chosen from $Act(z)$ randomly as given by $\gamma(n, z)$. Then, the successor node of the FSC is determined randomly as described by $\delta(n, z, a)$. The following example illustrates an FSC and the induced MC for an FSC.

Example 1. Figure 1(b) represents an excerpt of an FSC $\mathcal{A}$ with two memory nodes. From node n_1, the action mapping distinguishes observations z_0 and z_1. The solid dots indicate a probability distribution from $Distr(Act)$. For readability, all distributions are uniform and we omit the action mapping for node n_2.



(a) Induced pMC

Act	$\mathcal{P}$	Node	Result	Substituted
$a_1 : p'$	0.6	$n_1 : q_1$	$0.6 \cdot p' \cdot q_1$	$0.6 \cdot p_1$
		$n_2 : 1 - q_1$	$0.6 \cdot p' \cdot (1{-}q_1)$	$0.6 \cdot p_2$
	0.4	$n_1 : q_1$	$0.4 \cdot p' \cdot q_1$	$0.4 \cdot p_1$
		$n_2 : 1 - q_1$	$0.4 \cdot p' \cdot (1{-}q_1)$	$0.4 \cdot p_2$
$a_2 : 1 - p'$	0.7	$n_1 : q_2$	$0.7 \cdot (1{-}p') \cdot q_2$	$0.7 \cdot p_3$
		$n_2 : 1 - q_2$	$0.7 \cdot (1{-}p') \cdot (1{-}q_2)$	$0.7 \cdot (1{-} \sum_{i=1}^{3} p_i)$
	0.3	$n_1 : q_2$	$0.3 \cdot (1{-}p') \cdot q_2$	$0.3 \cdot p_3$
		$n_2 : 1 - q_2$	$0.3 \cdot (1{-}p') \cdot (1{-}q_2)$	$0.3 \cdot (1{-} \sum_{i=1}^{3} p_i)$

(b) Symbolic transition probabilities

Fig. 2. Induced parametric Markov chain for symbolic FSCs.

Recall the POMDP $\mathcal{M}$ from Fig. 1(a). We illustrate the induced MC $\mathcal{M}^{\sigma_\mathcal{A}}$, depicted in Fig. 1(c). Assume we are in state s_1 and memory node n_1. Based on the observation $z_0 := O(s_1)$ for s_1, with probability $\delta(n_1, z_0)(a_1) = 0.5$, we choose action a_1 leading to the probabilistic branching in the POMDP. With probability 0.6, we reach state s_2. Next, we update the memory node; with probability $\delta(n_1, z_0, a_1)(n_1) = 0.5$ we stay in memory node n_1. The corresponding transition from $\langle s_1, n_1\rangle$ to $\langle s_2, n_1\rangle$ in $\mathcal{M}^{\sigma_\mathcal{A}}$ has probability $0.5 \cdot 0.6 \cdot 0.5 = 0.15$.

2.3 Specifications

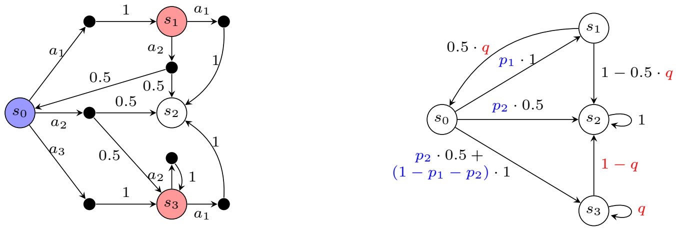
For a POMDP $\mathcal{M}$, a set $G \subseteq S$ of *goal states* and a set $B \subseteq S$ of *bad states*, we consider *quantitative reach-avoid specifications* $\varphi = \mathbb{P}_{>\lambda}(\neg B \mathbin{\mathsf{U}} G)$. The specification φ is satisfied for a strategy $\sigma \in \Sigma^\mathcal{M}$ ($\sigma \models \varphi$) if the probability $\Pr^{\mathcal{M}^\sigma}(\neg B \mathbin{\mathsf{U}} G)$ of reaching a goal state in $\mathcal{M}^\sigma$ without entering a bad state in between exceeds λ, denoted by $\mathcal{M}^\sigma \models \varphi$. The task is to compute such a strategy if one exists. For an MDP M, there is a memoryless deterministic strategy inducing the maximal probability $\Pr^M_{\max}(\neg B \mathbin{\mathsf{U}} G)$ [9]. For a POMDP $\mathcal{M}$, however, observation-based strategies with infinite memory as in Def. 6 are necessary [29] to attain $\Pr^\mathcal{M}_{\max}(\neg B \mathbin{\mathsf{U}} G)$. The problem of proving the satisfaction of φ is therefore undecidable [6]. In our experiments, we also use *expected reward properties*.

3 Induced MCs under k-FSCs for POMDPs

Formal Problem Statement. Given a POMDP $\mathcal{M}$, a specification φ, and a memory bound k, we want to synthesize a k-FSC $\mathcal{A}$ such that $\mathcal{M}^{\sigma_\mathcal{A}} \models \varphi$.

The degrees of freedom to select a k-FSC are given by the choice of γ and δ. For each choice of γ and δ, we get a different induced MC, but these MCs are *structurally similar* and can be represented by a single pMC.

Example 2. Recall Fig. 1 and Ex. 1. The action mapping γ and the memory update δ have arbitrary but fixed probability distributions. For a_1, we represent the probability $\gamma(n_1, z_0)(a_1) =: p'$ by $p' \in [0, 1]$. The memory update yields $\delta(n_1, z_0, a_1)(n_1) =: q_1 \in [0, 1]$ and $\delta(n_1, z_0, a_1)(n_2) =: 1{-}q_1$, respectively. Fig. 2(a)



(a) POMDP $\mathcal{M}$ (b) Induced pMC $D_{\mathcal{M},1}$

Fig. 3. From POMDPs to pMCs

shows the induced MC for action choice a_1. For instance, the transition from $\langle s_1, n_1 \rangle$ to $\langle s_2, n_1 \rangle$ has now probability $p' \cdot 0.6 \cdot q$, which is a polynomial over p', q.

We collect all these polynomials for memory node n_1 and observation z_0 in Tab. 2(b). Note that the *result* column basically describes a *parameterized distribution* over tuples of states and memory nodes, thus, instantiations for these polynomials need to sum up to one. The *substitute* column is discussed later.

In the following definition we assign parameters as arbitrary probabilities to action choices. We assume at each state one action to be the *remaining action* to which, after choosing probabilities for all other actions, we have to assign the remaining probability. A similar principle holds for the remaining memory node.

Definition 8 (Induced pMC for a k-FSC on POMDPs). *Let $\mathcal{M} = (M, Z, O)$ be a POMDP with $M = (S, s_{\mathrm{I}}, Act, \mathcal{P})$ and k a memory bound. Let the* remaining action *be fixed by a mapping $Remain \colon Z \to Act$, $Remain(z) \in Act(z)$. We construct the* induced pMC *$D_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{\mathrm{I},\mathcal{M},k}, V_{\mathcal{M},k}, P_{\mathcal{M},k})$:*

– $S_{\mathcal{M},k} = S \times \{0, \ldots, k-1\}$
– $s_{\mathrm{I},\mathcal{M},k} = \langle s_{\mathrm{I}}, 0 \rangle$
– $V_{\mathcal{M},k} = \big\{ p_a^{z,n} \bigm| z \in Z, n \in \{0, \ldots, k-1\}, a \in Act(z), a \neq Remain(z) \big\}$
$\qquad \cup \big\{ q_{a,n'}^{z,n} \bigm| z \in Z, n, n' \in \{0, \ldots, k-1\}, n' \neq k-1, a \in Act(z) \big\}$
– $P_{\mathcal{M},k}(s, s') = \sum_{a \in Act(s)} H(s, s', a)$ *for all* $s, s' \in S'$,

where $H \colon S_{\mathcal{M},k} \times S_{\mathcal{M},k} \times Act \to \mathbb{R}$ is for $z = O(s)$ defined by $H\big(\langle s, n \rangle, \langle s', n' \rangle, a\big) =$

$$\mathcal{P}(s, a, s') \cdot \left\{ \begin{matrix} p_a^{z,n}, & \text{if } a \neq Remain\big(z\big) \\ 1 - \sum\limits_{a' \neq a} p_{a'}^{z,n}, & \text{if } a = Remain\big(z\big) \end{matrix} \right\} \cdot \left\{ \begin{matrix} q_{a,n'}^{z,n}, & \text{if } n' \neq k-1 \\ 1 - \sum\limits_{\bar{n} \neq n'} q_{a,\bar{n}}^{z,n}, & \text{if } n' = k-1 \end{matrix} \right\}$$

Intuitively, $H(s, s', a)$ describes the probability mass from s to s' in the induced pMC that is contributed by a particular action $a \in Act$. The three terms correspond to the three terms as seen in Tab. 2(b).

Example 3. For the POMDP in Fig. 3(a) and memory bound $k = 1$ there is only one memory node. We get the induced pMC (Fig. 3(b)). The three actions from s_0 have probability p_1, p_2, or $1 - p_1 - p_2$ for the *remaining action* a_3. From the indistinguishable states s_1, s_3, actions have probability q or $1 - q$.

By construction, the induced pMC describes the set of all induced MCs:

Theorem 1 (Correspondence Theorem). *Let $\mathcal{M}$ be a POMDP, k a memory bound, and $D_{\mathcal{M},k}$ the induced pMC. Then:*

$$\{D_{\mathcal{M},k}[u] \mid u \textit{ well-defined}\} = \{\mathcal{M}^{\sigma_\mathcal{A}} \mid \mathcal{A} \in \mathit{FSC}_k^\mathcal{M}\}.$$

In particular, every well-defined instantiation u describes an FSC $\mathcal{A}_u \in \mathit{FSC}_k^\mathcal{M}$.

By the correspondence, we can thus evaluate an instantiation of the induced pMC to assess whether the corresponding k-FSC satisfies a given specification.

Corollary 1. *Let $D_{\mathcal{M},k}$ be an induced pMC with variables $V_{\mathcal{M},k}$. For every instantiation u of variables $V_{\mathcal{M},k}$ and the corresponding k-FSC $\mathcal{A}_u$, we have:*

$$\mathcal{M}^{\sigma_{\mathcal{A}_u}} \models \varphi \iff D_{\mathcal{M},k}[u] \models \varphi.$$

Lemma 1 (Number of Parameters). *The number of parameters in the induced pMC is given by $\mathcal{O}\big(|Z| \cdot k^2 \cdot \max_{z\in Z} |Act(z)|\big)$.*

Towards a simplified treatment of FSCs with $k > 1$, we substitute the polynomials in the transition probability function. In Tab. 2(b), we observe that the polynomials of the form $p' \cdot q_i$ and $p' \cdot (1 - q_i)$ for $i \in \{0, 1\}$ are independent from each other. We substitute them with single variables in the *substituted* column.

Definition 9 (Substituted Induced pMC). *Reconsider Def. 8. We define $D^{\text{subs}}_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{\mathrm{I},\mathcal{M},k}, V^{\text{subs}}_{\mathcal{M},k}, P^{\text{subs}}_{\mathcal{M},k})$ by modifying $V_{\mathcal{M},k}$ and H^{subs} as follows:*

- $V^{\text{subs}}_{\mathcal{M},k} = \big\{z \in Z, r^{z,n}_{a,n'} \;\big|\; n, n' \in \{0, \dots, k-1\}, a \in Act(z) \textit{ with } n' \neq k-1 \vee a \neq Remain(z)\big\}$,
- $H^{\text{subs}}\big(\langle s, n\rangle, \langle s', n'\rangle, a\big) =$

$$\mathcal{P}(s, a, s') \cdot \begin{Bmatrix} r^{z,n}_{a,n'}, & \textit{if } a \neq Remain(z) \vee n' \neq k-1 \\ 1 - \sum_{a'\neq a \vee \bar{n}\neq k-1} r^{z,n}_{a',\bar{n}}, & \textit{if } a = Remain(z) \wedge n' = k-1 \end{Bmatrix}$$

  with $z = O(s)$, and
- $P^{\text{subs}}_{\mathcal{M},k}(s, s') = \sum_{a\in Act(s)} H^{\text{subs}}(s, s', a)$ *for all $s, s' \in S_{\mathcal{M},k}$.*

The obtained pMC is called the *substituted induced pMC*. It follows:

$$\big\{D_{\mathcal{M},k}[u] \;\big|\; u \text{ well-defined}\big\} = \big\{D^{\text{subs}}_{\mathcal{M},k}[u'] \;\big|\; u' \text{ well-defined}\big\}.$$

Remark 1 (k-Unfolding). We observe that we can remove the multiplications which originate from the two different distributions in FSCs. The substitution not only yields structurally simpler pMCs, but also allows us to restrict the further treatment to 1-FSCs. The main idea is the following: The substituted induced pMC for a k-FSC is isomorphic to the induced pMC on a 1-FSC on so-called k-unfolded POMDPs. The k-unfolding constructs a POMDP as the Cartesian product of the memory and the original POMDP as state space. See Appendix A for the formal construction. The substitution is thus elementary for some of the obtained results.

In the literature, several formalisms for FSCs occur, in particular [5,24,2] do not agree upon a common model. Some of the choices made in these sources prevent the application of the substitution. For details, we refer to Appendix B.

Ignoring the Taken Action for Updates. In [24,2], the memory update is of the form $\delta' \colon N \times Z \to \mathit{Distr}(N)$. This is a restriction of the FSCs considered here, represented by the constraint $\delta(n, z, a_1) = \delta(n, z, a_2)$. This yields dependencies between different actions, preventing substitution.

Taking the Next Observation into Account. The memory node update depends on the observation at the state *before* executing the action. Instead, the update may also be based on the observation *after* the update [24]. This notion introduces dependencies between actions from states with different observations that reach the same observation. This can be alleviated by transforming the POMDP.

Ignoring the Current Observation when Selecting the Action. In [5], the action mapping is modeled as $\gamma' \colon N \to \mathit{Distr}(\mathit{Act})$, which restricts our FSC to $\gamma(n, z) = \gamma(n, z')$. This type of FSC is more general in the sense that it can assign memory usage more freely than the rather uniform assignment used here. In particular, a model with one memory node is now not memoryless anymore, but weaker (it has to select the same action distribution regardless of the observation). It also contains some restrictions: In particular, every POMDP state requires the same action set. Therefore, we do not consider this model any further.

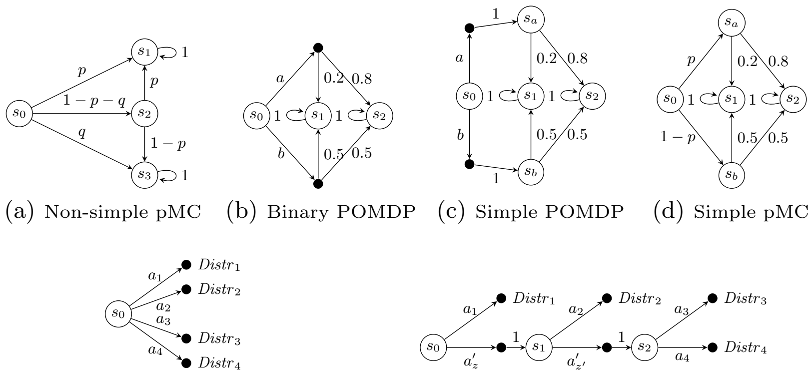
4 Relationships between POMDPs and pMCs

In the previous section we showed the *direct correspondence between computing reachability probabilities for pMCs and synthesizing finite-memory strategies for POMDPs.* Due to the unfolding, we restrict ourselves to memoryless strategies (see Remark 1). We now explore the relationship between typical classes of pMCs (called *simple pMCs*) and POMDPs. Furthermore, the state-of-the art methods for pMC verification pose certain restrictions on parameter values. Accordingly, we discuss the impact on strategies for POMDPs.

4.1 Transforming General POMDPs to Simple pMCs and Back

First, we examine how general pMCs relate to POMDPs. In general, there is no transformation from pMCs to POMDPs that preserves the topology of the pMC, i. e., that induces the same graph in terms of states and transitions given by positive transition probabilities.

Example 4. Consider the pMC in Fig. 4(a), parameter p occurs in two distinct distributions (at s_0 and s_2). To obtain a topology-preserving POMDP, there are two options: Either $O(s_0) = O(s_2)$, or $O(s_0) \neq O(s_2)$. The intuition is that every (parametric) transition in the pMC corresponds to an action in the POMDP.



(a) Non-simple pMC (b) Binary POMDP (c) Simple POMDP (d) Simple pMC

(e) Four actions in a POMDP (f) Four actions in a binary POMDP

Fig. 4. POMDP $\leftrightarrow$ simple pMC

Then $O(s_0) = O(s_2)$ is impossible due to the different outdegrees. $O(s_0) \neq O(s_2)$ is impossible, as a strategy can distinguish s_0 and s_2, and thereby assign different probabilities to p and q. Adding a self-loop to s_2 does not alleviate the problem.

We will now show that for an important subclass of pMCs called *simple pMCs*, there is a one-to-one correspondence with POMDPs under finite-memory schedulers. A pMC is simple, if for all states s, s', $P(s, s') \in \mathbb{Q} \cup \{p, 1-p \mid p \in V\}$. An elementary benefit of considering simple pMCs is that the set of well-defined parameter instantiations has a hyper-rectangular shape. Note that the typical pMC benchmarks as from the PARAM webpage [26] are simple pMCs.

Every POMDP can be translated into a simple pMC such that parameter instantiations and FSCs coincide. We first translate arbitrary POMDPs into *binary POMDPs* and subsequently into *simple POMDPs*, which yields simple pMCs. Examples are given in Fig. 4(b–f).

Definition 10 (Binary/Simple POMDP). *A POMDP is* binary*, if* $|Act(s)| \leq 2$ *for all* $s \in S$. *A binary POMDP is* simple*, if for all* $s \in S$

$$|Act(s)| = 2 \implies \forall a \in Act(s)\ \exists s' \in S\ s.\,t.\ P(s, a, s') = 1.$$

Proposition 1. *If $\mathcal{M}$ is a simple POMDP, then $D_{\mathcal{M},1}$ is a simple pMC.*

Theorem 2. *Let D be a simple pMC with n states and parameters V_D, $|V_D| = m$. D is isomorphic to $D_{\mathcal{M},1}$ for some simple POMDP $\mathcal{M}$ with n states and m observations.*

We refrain from a formal proof: The construction is the reverse of Definition 9, with observations $\{z_p \mid p \in V_D\}$. The important observation is that in a simple pMC, the outgoing transitions are either all parameter free, or of the form $p, 1-p$. The parameter-free case is transformed into a POMDP state with a single action (and any observation). The parametric case is transformed into a state with two actions with Dirac-distributions attached. As observation, we use z_p.

The transformation from binary POMDP to simple POMDP is illustrated by Fig. 4(b) to Fig. 4(c). We introduce a fresh observation z'. After each state with a choice of two actions, auxiliary states are introduced, such that the outcome of the action becomes deterministic and the probabilistic choice is delayed to the auxiliary state. This construction is similar to the conversion of Segala's probabilistic automata into Hansson's alternating model [32].

There are several ways to transform a POMDP $\mathcal{M}$ into a binary POMDP. We illustrate one in Fig. 4(e) to Fig. 4(f). The idea is to split actions at a state with more than two actions into two sets, which are then handled by fresh states: This iteratively reduces the number of actions until every state has at most two outgoing actions. All states with the same observation should be handled the same way. After the transformation to a simple POMDP, the number of parameters in the induced pMC is equal to the number of parameters in the induced pMC of the original POMDP. Utilizing state elimination [11], the auxiliary states that have been introduced in this phase can be eliminated.

4.2 Strategy/Parameter Restrictions

We observe two typical restrictions on the parameters made in pMC research:

– Each transition gets assigned a strictly positive probability (graph-preserving).
– Each transition gets assigned at least probability $\varepsilon > 0$ (ε-preserving).

For simple pMCs, this corresponds to selecting parameters instantiations from $(0, 1)$ or $[\varepsilon, 1 - \varepsilon]$, respectively.

As parameter instantiations describe strategies for POMDPs, the search space for strategies is restricted. We define the following restricted strategies.

Definition 11 (Non-zero Strategies). *A strategy σ is* non-zero *if $\sigma(\pi)(a) > 0$ for all $\pi \in \mathsf{Paths}^{M}_{fin}, a \in Act(last(\pi))$. If additionally $\sigma(\pi)(a) \geq \varepsilon > 0$, then σ is* min-ε.

Non-zero strategies ensure that $supp(\sigma(s)) = Act(s)$.

Proposition 2. *Let $\mathcal{M}$ be a POMDP. An instantiation u on $D_{\mathcal{M},1}$ is graph-preserving (ε-preserving), iff $\sigma_{\mathcal{A}_u}$ is non-zero (min-ε).*

For the considered specifications, we can restrict the FSCs which we consider.

Theorem 3. *Let $\mathcal{M}$ be POMDP, k a memory bound and $\varphi = \mathbb{P}_{>\lambda}(\neg B \cup G)$. Either $\forall \mathcal{A} \in k\text{-}FSC : \mathcal{M}^{\sigma_\mathcal{A}} \not\models \varphi$ or $\exists \mathcal{A}' \in k\text{-}FSC : \mathcal{M}^{\sigma_{\mathcal{A}'}} \models \varphi$ with $\sigma_{\mathcal{A}'}$ non-zero.*

The fundamental insight here is that the supremum of the reachability probabilities induced by any instantiation of a simple pMC coincides with the supremum over the graph-preserving instantiations. The essential idea is that the only reason for a discontinuity of a function mapping instantiations to reachability is a change in the set $S_{=0}$ – states in the pMC from which the probability to reach the target is zero. $S_{=0}$ is the smallest under a graph preserving assignment, and a discontinuity thus implies a reduced reachability probability. As a consequence, if we have to construct a strategy which reaches a goal with probability $> \kappa$, we can look for such a strategy among the non-zero strategies. In particular, this also means that the set of states $S_{=0}$ can be precomputed.

5 Synthesis of FSCs for POMDPs

Above, we established the correspondence between FSCs and parameter instantiations of pMCs. Now, we discuss the available methods for parameter synthesis, and how they may be exploited or adapted to search for optimal FSCs. The tools PARAM [16], PRISM [22], and PROPhESY [12] employ a technique called *state elimination* [11] to compute a rational function f over the parameters, where f maps well-defined parameter valuations to the reachability probability of the instantiated MC. f can serve as a starting point for further evaluation.

For a POMDP $\mathcal{M}$ and a memory bound k, we want to synthesize a k-FSC inducing the maximal reach-avoid probability $\mathrm{Pr}^{\mathcal{M}}_{\max}(\neg B \cup G)$, for instance to show satisfaction of a lower probability bound λ, as before. We first construct the substituted induced pMC $D^{\mathrm{subs}}_{\mathcal{M},k}$ as in Def. 9. Then, we apply the following steps.

- *Find a good solution.* By searching the parameter space, we heuristically determine parameter instantiations, i. e., valuations u inducing probability $\kappa_u = \mathrm{Pr}^{D^{\mathrm{subs}}_{\mathcal{M},k}[u]}(\neg B \cup G)$. Surely, $\kappa_u \leq \mathrm{Pr}^{\mathcal{M}}_{\max}(\neg B \cup G)$, but we strive to find an u which is close to optimal, i. e., $\kappa_u \geq \mathrm{Pr}^{\mathcal{M}}_{\max}(\neg B \cup G) - \epsilon$ holds, where ϵ is an acceptable deviation from the optimum.
- *Prove that a solution is (nearly) optimal.* For a presumably optimal candidate valuation u, we aim to show that there is no $u' \neq u$ such that $\kappa_{u'} > \kappa_u + \epsilon$.

Sampling. To find a suitable parameter valuation, we search the parameter space by iterative sampling. Heuristics like *particle swarm optimization* (PSO) [8] provide an efficient way to find near-optimal solutions. There are two options for *evaluating* instantiations: On the one hand, model checking is employed for the resulting (instantiated) MC. On the other hand, a previously computed rational function may directly be instantiated and evaluated. For simple pMCs, it is significantly simpler to find *well-defined instantiations*. For general pMCs, we resort to SAT-modulo-theories (SMT)-based approaches (over non-linear real arithmetic) to ensure well-defined sample points, but sampling becomes hard to guide and generally slow.

Proving Bounds on Optimality. Given a candidate probability κ_u, we want to compute an upper bound $b \geq \mathrm{Pr}^{\mathcal{M}}_{\max}(\neg B \cup G)$ with $b \leq \mathrm{Pr}^{\mathcal{M}}_{\max}(\neg B \cup G) + \epsilon$ for some preferably small ϵ. The distance between b and κ_u then indicates whether there is justified hope to improve the candidate. *Parameter lifting* (PL) [28] is a technique that computes such an upper bound. Basically, relaxing parameter dependencies enables to efficiently compute bounds on optimal probabilities.

These bounds are computed for *regions* of the parameter space, containing only well-defined instantiations. The bounds can be refined by *splitting* the regions into smaller parts. Iterating this to infinitesimally small regions, the optimum can be approximated arbitrarily closely. In practice, the algorithm terminates once a computed bound is ϵ-close to the given candidate κ_u. For our case, we can only consider min-ε strategies. Note that in case no candidate solution is necessary, we can use PL to iteratively tighten the bound on the optimum.

SMT-based Binary Search. Instead of finding instantiations and proving bounds independently, we can resort to a binary search driven by SMT solving. Starting with a lower bound κ_u as before, we guess an upper bound b. Trivial bounds may be $\kappa_u = 0$ and $b = 1$. SMT solving determines whether there is a valuation u'' such that $\kappa_{u''} > {}^1\!/_2(\kappa_u + b)$. If not, ${}^1\!/_2(\kappa_u + b)$ is the new upper bound. If yes, we use $\kappa_{u''}$ as the new lower bound and continue the search until $b - \kappa_u < \epsilon$.

6 Permissive Strategies via Parameter Synthesis

Permissive strategies are sets of strategies satisfying a specification for MDPs [14,20]. One interesting application of permissive strategies is *robustness* in the sense that we can assess if a slight change to a strategy will preserve the satisfaction of the specification. For observation-based strategies, a permissive scheduler may be defined as a function $\mathsf{Paths}^{M}_{\mathit{fin}} \to 2^{Distr(Act)}$ mapping paths to (infinite) sets of distributions over actions. Here, we can restrict ourselves to sets of k-FSCs.

Definition 12. *A* permissive k-FSC *for a POMDP $\mathcal{M}$ is a subset $FSC_k^{\mathcal{M}}$.*

Using the correspondence from Theorem 1, we transfer the notion of permissive schedulers to *sets of parameter instantiations* that satisfy a given specification on the induced pMCs. We aim at finding (preferably large) regions of well-defined instantiations that all satisfy the given specification.

As in the previous section, several methods may be employed. Sampling to identify a contiguous set of instantiations that satisfy the specification delivers a *region candidate.* As in [12], SMT solving is used to verify whether all points within this region are satisfying. If they are, the region forms a permissive FSC. If not, a counterexample provides a non-satisfying point along which the region can be split into smaller region candidates.

Alternatively, we use *parameter lifting (PL)* [28]; contrary to its original purpose, we do not aim to partition the full parameter space into satisfying and rejecting strategies. The established correspondence allows to rephrase PL for POMDPs. First, PL renders a POMDP fully observable, i. e., it works on the underlying MDP. On the induced pMC this means to relax parameter dependencies. Memoryless strategies suffice to induce minimal and maximal probabilities for the MDP, forming upper and lower bounds on the optimal solution induced by observation-based strategies. These bounds may be very coarse; however, in an iterative region refinement approach the influence of nondeterministic choices is reduced. This tightening of the bounds corresponds to reducing the influence of neglecting different observations.

7 Experiments

Implementation and Setup. We extended the model checker Storm [13] to parse and store POMDPs, and implemented several transformation options to pMCs. Most notably, Storm supports k-unfolding and several types of transformation to simple pMCs: To this end, we integrated the steps towards a simple POMDP and a possible state elimination into a single step. We extended PROPhESY to handle parameter optimization and use z3 [19] as backend.

We took *all* POMDPs from PRISM-POMDP [25], additional maze, load/unload examples from [24], and a slippery gridworld with traps inspired by [31]. From each benchmark set, we showcase a challenging instance with memory bounds $k \in \{1, 2\}$. The experiments contain minimal expected rewards, which are analyzed by a straightforward extension of maximal reachability probabilities.

Experiments were conducted on a single core of an Intel Xeon E5-1620v4 CPU running at 3.5 GHz with memory limit 16 GB and a 1800 s time out. We executed state elimination on both the pMC and simple pMC. Moreover, we applied PSO and either parameter lifting (PL) or SMT for the simple pMCs. To judge the performance of our approach, we compare with PRISM-POMDP, which computes optimal memory-unbounded strategies and is still in prototype stage. We take the settings used in [25] and the default settings otherwise.

Results. Table 2(a) gives information about the POMDP instances (id, name), their size (states, choices, transitions, observations), the number of states, transitions and parameters of the simple pMCs (with $k = 1$ and $k = 2$, resp.) for which any auxiliary states were eliminated and bisimulation minimization was applied. Translation times from the POMDP to pMCs were within seconds. The last column gives the optimal value for the fully observable case. Table 2(b) gives run times in seconds for *elimination* on the pMC and the equivalent simple pMC. The next columns give run times for *sampling* by PSO and *proving* the bounds by PLA or SMT, and the obtained *results* for k=1 and k=2. The last columns refer to the run time and result as given by PRISM-POMDP.

Evaluation. We see that for some benchmarks (1,4,7) memory does not seem to improve the quality of the strategies. For others (2,6), moving to $k = 2$ already drastically improves quality. Run times with PRISM-POMDP are incomparable, there is no trend, both approaches favor different instances.

We stress that typical pMC benchmarks in the literature contain up to 4 parameters. The considered pMCs here are significantly different. Based on the preliminary results presented here, we already obtained several insights. Searching in the high dimensional search space for memory bound $k = 2$ is difficult: Although the strategy may now use memory, in (3,5) we find a worse result, (4) time-outs during preprocessing. We sampled directly on the model. Benchmarks (3,4) show that state elimination can be successfully applied even though there are hundreds of parameters in the bisimulation quotient. For (4), the result is a constant function which is trivial to analyze, while for (3), we obtained rational functions with roughly one million terms, rendering further

Table 2. Benchmarks

(a) Instances

		POMDP $\mathcal{M}$				$D_{\mathcal{M},1}$			$D_{\mathcal{M},2}$			MDP
ID	Name	States	Choices	Trans	Obs	States	Trans	Pars	States	Trans	Pars	Res
1	NRP (8)	125	161	168	21	75	118	8	148	378	50	1.0
2	Grid (4)	17	62	76	3	17	66	3	33	257	16	3.2
3	Netw (3,4,8)	3349	5941	17736	429	2324	11573	276	5190	53862	1726	0.83
4	Crypt (6)	22406	64966	65285	2458	2248	2887	256	40974	248342	14116	1.0
5	Maze-2	16	58	70	8	16	57	9	31	223	50	4.0
6	Load (8)	16	28	33	5	16	33	1	31	127	12	10.5
7	Slippery (4)	17	59	114	4	17	72	3	32	270	16	1.0

(b) Results

	Elimination		$D_{\mathcal{M},1}$			$D_{\mathcal{M},2}$			PRISM-POMDP	
ID	std.	simple	sampl.	proof	result	sampl.	proof	result	time	result
1	<0.1	<0.1	3	105	[0.125,0.225]	7	TO	[0.125,1]	94	[0.125, 0.178]
2	1.0	1.5	13	2	[5.1,5.1]	42	613	[3.1,4.3]	53	[3.97, 4.13]
3	192	TO	125	TO	[0,13.3]	170	TO	[0, 17.2]	290	[1.30, 1.45]
4	<0.1	<1	2	<0.1	[0.2, 0.2]	TO	TO	[0, 1.0]	963	[0.2, 1.0]
5	221	37	16	101	[10.8, 14.8]	85	8	[4.2, 16.2]	1	[5.11,5.23]
6	<0.1	<0.1	6	<1	[82.5, 82.5]	14	12	[11.3, 11.8]	101	[10.5,10.5]
7	13	20	1	2	[0.927, 0.927]	35	1603	[0.929, 0.996]	1	[0.24, 0.99]

evaluation using SMT unsuitable. PL cannot cope with that many parameters as it often requires exponentially many model checking calls in the number of parameters. For lower dimensional benchmarks, though, PL quickly finds tight bounds. While SMT-based binary search is applicable in smaller cases, *PL was superior in all cases*. Generally, without further preprocessing or restrictions to the memory, tackling higher memory bounds requires further improvements in parameter synthesis.

For obtaining rational functions, a direct POMDP to pMC translation is faster while simple pMCs have significant advantages: If omitting the transformation to simple, from 10K samples of (3) none is well-defined, rendering PSO application hard and ruling out PL. How to obtain simple pMCs influences the performance, just swapping p and $1{-}p$ for (7) increased the runtime by a factor 20.

Overall, even out-of-the-box, the pMC approaches yield non-trivial results for POMDPs. A full-scale evaluation on more POMDPs and comparison with more tools, e.g. [37], is left for future work.

8 Conclusion

This paper connects two active research areas, namely verification and synthesis for POMDPs and parameter synthesis for Markov models. We see benefit for both areas. On the one hand, the rich application area for POMDPs in, e. g., robotics, yields new challenging benchmarks for parameter synthesis and can drive the development of more efficient methods. On the other hand, the parameter synthesis tools made available for POMDP verification help the state-of-the-art there. Besides application-driven improvement of parameter synthesis methods, future work will concern a thorough investigation of permissive schedulers in concrete motion planning scenarios.

References

1. Amato, C., Bernstein, D.S., Zilberstein, S.: Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. Autonomous Agents and Multi-Agent Systems 21(3), 293–320 (2010)
2. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
3. Braziunas, D.: POMDP solution methods. University of Toronto (2003)
4. Chatterjee, K., Chmelik, M., Davies, J.: A symbolic SAT-based algorithm for almost-sure reachability with small strategies in POMDPs. In: AAAI. pp. 3225–3232. AAAI Press (2016)
5. Chatterjee, K., Chmelík, M., Gupta, R., Kanodia, A.: Optimal cost almost-sure reachability in POMDPs. Artif. Intell. 234, 26–48 (2016)
6. Chatterjee, K., Chmelík, M., Tracol, M.: What is decidable about partially observable Markov decision processes with ω-regular objectives. J. Comput. Syst. Sci. 82(5), 878–911 (2016)
7. Chatterjee, K., De Alfaro, L., Henzinger, T.A.: Trading memory for randomness. In: QEST. IEEE (2004)
8. Chen, T., Hahn, E.M., Han, T., Kwiatkowska, M.Z., Qu, H., Zhang, L.: Model repair for Markov decision processes. In: TASE. pp. 85–92. IEEE CS (2013)
9. Condon, A.: The complexity of stochastic games. Inf. Comput. 96(2), 203–224 (1992)
10. Cubuktepe, M., Jansen, N., Junges, S., Katoen, J., Papusha, I., Poonawala, H.A., Topcu, U.: Sequential convex programming for the efficient verification of parametric MDPs. In: TACAS (2). LNCS, vol. 10206, pp. 133–150 (2017)
11. Daws, C.: Symbolic and parametric model checking of discrete-time Markov chains. In: ICTAC. LNCS, vol. 3407, pp. 280–294. Springer (2004)
12. Dehnert, C., Junges, S., Jansen, N., Corzilius, F., Volk, M., Bruintjes, H., Katoen, J.P., Ábrahám, E.: PROPhESY: A probabilistic parameter synthesis tool. In: CAV. LNCS, vol. 9206, pp. 214–231. Springer (2015)
13. Dehnert, C., Junges, S., Katoen, J., Volk, M.: A storm is coming: A modern probabilistic model checker. In: CAV (2). LNCS, vol. 10427, pp. 592–600. Springer (2017)
14. Dräger, K., Forejt, V., Kwiatkowska, M.Z., Parker, D., Ujma, M.: Permissive controller synthesis for probabilistic systems. Logical Methods in Computer Science 11(2) (2015)
15. Filieri, A., Ghezzi, C., Tamburrelli, G.: Run-time efficient probabilistic model checking. In: ICSE. pp. 341–350. ACM (2011)
16. Hahn, E.M., Hermanns, H., Zhang, L.: Probabilistic reachability for parametric Markov models. Software Tools for Technology Transfer 13(1), 3–19 (2010)
17. Howard, R.A.: Dynamic Programming and Markov Processes. The MIT Press (1960)
18. Hutschenreiter, L., Baier, C., Klein, J.: Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination. In: GandALF. EPTCS, vol. 256, pp. 16–30 (2017)
19. Jovanovic, D., de Moura, L.M.: Solving non-linear arithmetic. In: IJCAR. LNCS, vol. 7364, pp. 339–354. Springer (2012)
20. Junges, S., Jansen, N., Dehnert, C., Topcu, U., Katoen, J.: Safety-constrained reinforcement learning for MDPs. In: TACAS. LNCS, vol. 9636, pp. 130–146. Springer (2016)

21. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artif. Intell. 101(1), 99–134 (1998)
22. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: CAV. LNCS, vol. 6806, pp. 585–591. Springer (2011)
23. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In: AAAI. pp. 541–548. AAAI Press (1999)
24. Meuleau, N., Kim, K.E., Kaelbling, L.P., Cassandra, A.R.: Solving POMDPs by searching the space of finite policies. In: UAI. pp. 417–426. Morgan Kaufmann Publishers Inc. (1999)
25. Norman, G., Parker, D., Zou, X.: Verification and control of partially observable probabilistic systems. Real-Time Systems 53(3), 354–402 (2017)
26. PARAM Website: (2015), `http://depend.cs.uni-sb.de/tools/param/`
27. Puterman, M.L.: Markov Decision Processes. John Wiley and Sons (1994)
28. Quatmann, T., Dehnert, C., Jansen, N., Junges, S., Katoen, J.: Parameter synthesis for Markov models: Faster than ever. In: ATVA. LNCS, vol. 9938, pp. 50–67. Springer (2016)
29. Ross, S.M.: Introduction to Stochastic Dynamic Programming. Academic Press, Inc. (1983)
30. Roy, N., Gordon, G.J., Thrun, S.: Finding approximate POMDP solutions through belief compression. J. Artif. Intell. Res. 23, 1–40 (2005)
31. Russell, S.J., Norvig, P.: Artificial Intelligence – A Modern Approach (3. internat. ed.). Pearson Education (2010)
32. Segala, R., Turrini, A.: Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In: QEST. pp. 44–53. IEEE CS (2005)
33. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based POMDP solvers. Autonomous Agents and Multi-Agent Systems 27(1), 1–51 (2013)
34. Szer, D., Charpillet, F.: An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In: ECML. pp. 389–399. Springer (2005)
35. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. The MIT Press (2005)
36. Vlassis, N., Littman, M.L., Barber, D.: On the computational complexity of stochastic controller optimization in POMDPs. ACM Trans. on Computation Theory 4(4), 12:1–12:8 (2012)
37. Walraven, E., Spaan, M.T.J.: Accelerated vector pruning for optimal POMDP solvers. In: AAAI. pp. 3672–3678. AAAI Press (2017)
38. Wongpiromsarn, T., Frazzoli, E.: Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications. In: CDC. pp. 7644–7651. IEEE (2012)

A Memory Aspects of the Induced MC

We now consider $k > 1$ in more detail. Instead of constructing the FSC for the k-memory bound, we show that we can describe the induced pMC in two steps: First, we construct a larger POMDP by a product construction, making the memory nodes part of the POMDP state space. We then consider the induced pMC for memoryless strategies on the larger POMDP. The advantage of this is that for later results, we can restrict ourselves to memoryless strategies, which significantly simplifies some constructions.

Definition 13 (Unfolding). *Let $\mathcal{M} = (M, Z, O)$ be a POMDP with $M = (S, s_{\mathrm{I}}, Act, \mathcal{P})$, and $k > 1$. The POMDP $\mathcal{M}' = (M', Z', O')$ with $M' = (S', s_{\mathrm{I}}, Act, \mathcal{P}')$ is the k-*unfolding *of $\mathcal{M}$ if:*

- $S' = S \times \{0, \ldots k{-}1\}$
- $s'_{\mathrm{I}} = \langle s_{\mathrm{I}}, 0\rangle$
- $Act' = Act \times \{0, \ldots, k{-}1\}$
- $P\big(\langle s, n\rangle, \langle a, \bar{n}\rangle, \langle s', n'\rangle\big) = \begin{cases} P(s, a, s') & \textit{if } n' = \bar{n} \\ 0 & \textit{otherwise.} \end{cases}$
- $Z' = Z \times \{0, \ldots, k{-}1\}$
- $O\big(\langle s, n\rangle\big) = \big\langle O(s), n\big\rangle$

Fig. 5 shows this process for $k = 2$. All states of the POMDP are copied once. By observation, we can still distinguish in which copy of a state – and therefore, which memory cell – we currently are. But when choosing an action, a strategy now has the option to switch into the other memory cell in order to store additional information on the history of the current execution.

Proposition 3. *Let $\mathcal{M}$ be a POMDP. Then $D^{\mathrm{subs}}_{\mathcal{M},k}$ is isomorphic to $D_{U_k(\mathcal{M}),1}$, where $U_k(\mathcal{M})$ denotes the k-unfolding of $\mathcal{M}$.*

Intuitively, this is true since the unfolded POMDP can make use of its state copies to store the same information a FSC could store in its memory cells.

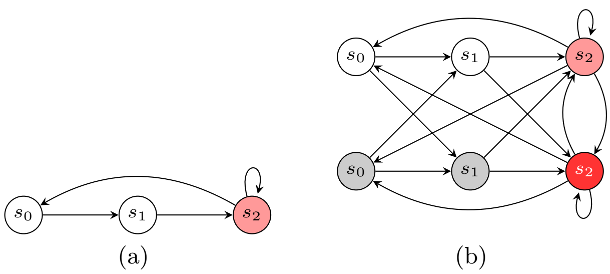


Fig. 5. Unfolding a POMDP for two memory states

Table 3. Alternative induced pMCs

(a) Action Restricted

Obs	Act	P	Node	Result
z_1	$a_1 : p'$	0.6	$n_1 : q_1$	$0.6 \cdot p' \cdot q_1$
			$n_2 : 1-q_1$	$0.6 \cdot p' \cdot (1{-}q_1)$
		0.4	$n_1 : q_1$	$0.4 \cdot p' \cdot q_1$
			$n_2 : 1{-}q_1$	$0.4 \cdot p' \cdot (1{-}q_1)$
	$a_2 : 1{-}p'$	0.7	$n_1 : \mathbf{q_1}$	$0.7 \cdot (1{-}p') \cdot q_1$
			$n_2 : \mathbf{1{-}q_1}$	$0.7 \cdot (1{-}p') \cdot (1{-}q_1)$
		0.3	$n_1 : \mathbf{q_1}$	$0.3 \cdot (1{-}p') \cdot q_1$
			$n_2 : \mathbf{1{-}q_1}$	$0.3 \cdot (1{-}p') \cdot (1{-}q_1)$

(b) Next observation dependent

Obs	Act	P	Node	Result
z_1	$a_1 : p'$	0.6	$n_1 : \mathbf{q_1}$	$0.6 \cdot p' \cdot q_1$
			$n_2 : \mathbf{1{-}q_1}$	$0.6 \cdot p' \cdot (1{-}q_1)$
		0.4	$n_1 : q_2$	$0.4 \cdot p' \cdot q_2$
			$n_2 : 1-q_2$	$0.4 \cdot p' \cdot (1{-}q_2)$
	$a_2 : 1{-}p'$	0.7	$n_1 : \mathbf{q_1}$	$0.7 \cdot (1{-}p') \cdot q_1$
			$n_2 : \mathbf{1{-}q_1}$	$0.7 \cdot (1{-}p') \cdot (1{-}q_1)$
		0.3	$n_1 : \mathbf{q_1}$	$0.3 \cdot (1{-}p') \cdot q_1$
			$n_2 : \mathbf{1{-}q_1}$	$0.3 \cdot (1{-}p') \cdot (1{-}q_1)$

B Alternative FSCs

Ignoring the Taken Action for Updates. In [24,2], the memory update is of the form $\delta' \colon N \times Z \to \mathit{Distr}(N)$. This is a restriction of the FSCs considered here, represented by the constraint $\delta(n, z, a_1) = \delta(n, z, a_2)$.

Example 5. Recall Ex. 2, with the induced pMC for the POMDP fragment, as also given in Tab. 2(b). Tab. 3(a) presents the induced pMC with the restriction in place. Notice that we have no parameter q_2 anymore. Based on Tab. 2(b) we could set $p' = 0.5, q_1 = 0$, and the following transition probabilities for each target: $\{\langle s_2, n_1\rangle \mapsto 0, \langle s_2, n_2\rangle \mapsto 0.3, \langle s_4, n_1\rangle \mapsto 0.35\}$. Based on Tab. 3(a), this assignment is not possible.

We conclude from the example above that we get an additional parameter dependency.

Definition 14 (Action-Restricted Induced pMC for FSCs on POMDPs). *Reconsider Def. 8. We define $D^{\text{restr}}_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{\text{I},\mathcal{M},k}, V^{\text{restr}}_{\mathcal{M},k}, P^{\text{restr}}_{\mathcal{M},k})$ by modifying $V_{\mathcal{M},k}$ and H^{restr} as follows:*

– $V^{\text{restr}}_{\mathcal{M},k} = \{p^{z,n}_a \mid z \in Z, n \in \{0, \ldots, k-1\}, a \in \mathit{Act}(z), a \neq \mathit{Remain}(z)\}$
$\cup \{q^{z,n}_{n'} \mid n, n' \in \{0, \ldots, k-1\}, n' \neq k-1, z \in Z\}$
– $H^{\text{restr}}\big(\langle s, n\rangle, \langle s', n'\rangle, a\big) =$

$$\mathcal{P}(s, a, s') \cdot \begin{Bmatrix} p^{z,n}_a, & \text{if } a{\neq}\mathit{Remain}\big(z\big) \\ 1 - \sum\limits_{a' \neq a} p^{z,n}_{a'}, & \text{if } a{=}\mathit{Remain}\big(z\big) \end{Bmatrix} \cdot \begin{Bmatrix} q^{z,n}_{n'}, & \text{if } n'{\neq}k{-}1 \\ 1 - \sum\limits_{\bar{n} \neq n'} q^{z,n}_{\bar{n}}, & \text{if } n'{=}k{-}1 \end{Bmatrix}$$

with $z = O(s)$
– $P^{\text{restr}}_{\mathcal{M},k}(s, s') = \sum_{a \in \mathit{Act}(s)} H^{\text{next}}(s, s', a)$ *for all $s, s' \in S'$.*

The obtained pMC is then called the action-restricted induced pMC.

For these pMCs, we can no longer perform the substitution as proposed in Def. 9. As a consequence this restriction breaks the proposed unfolding.

Taking the Next Observation into Account. Instead of basing the memory node update on the observation from the state before executing the action, the memory node may also be updated based on the observation after the update [24].

Example 6. Recall Ex. 2, with the induced pMC for the POMDP fragment, as also given in Tab. 2(b). Tab. 3(a) presents the induced pMC with the restriction in place. Notice that the memory update probabilities now depend on the observation of the resulting state. In particular, the action probability depends on the current observation, and features dependencies between source states, while the memory update features dependencies between target states.

Definition 15 (Next-observation induced pMC). *Reconsider Def. 8. We define* $D^{\text{next}}_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{\text{I},\mathcal{M},k}, V^{\text{next}}_{\mathcal{M},k}, P^{\text{next}}_{\mathcal{M},k})$ *by modifying* $V_{\mathcal{M},k}$ *and* H^{next} *as follows:*

– $V^{\text{next}}_{\mathcal{M},k} = \big\{p^{z,n}_a \;\big|\; z \in Z, n \in \{0,\dots,k-1\}, a \in \mathit{Act}(z), a \neq \mathit{Remain}(z)\big\}$
$\quad\cup \big\{q^{z,n}_{a,n'} \;\big|\; z \in Z, n, n' \in \{0,\dots,k-1\}, n' \neq k-1, a \in \mathit{Act}\big\}$,

– $H^{\text{next}}\big(\langle s,n\rangle, \langle s',n'\rangle, a\big) =$

$$\mathcal{P}(s,a,s') \cdot \left\{\begin{matrix} p^{z,n}_a, & \text{if } a{\neq}\mathit{Remain}\big(z\big) \\ 1 - \sum\limits_{a'\neq a} p^{z,n}_{a'}, & \text{if } a{=}\mathit{Remain}\big(z\big) \end{matrix}\right\} \cdot \left\{\begin{matrix} q^{z',n}_{a,n'}, & \text{if } n'{\neq}k{-}1 \\ 1 - \sum\limits_{\bar{n}\neq n'} q^{z',n}_{a,\bar{n}}, & \text{if } n'{=}k{-}1 \end{matrix}\right\}$$

with $z = O(s), z' = O(s')$*, and*

– $P^{\text{next}}_{\mathcal{M},k}(s,s') = \sum_{a\in \mathit{Act}(s)} H^{\text{next}}(s,s',a)$ *for all* $s,s' \in S'$.

The obtained pMC is then called the next-induced pMC.

Notice that due to the dependencies, we cannot substitute monomials, and we cannot simply unfold the memory into the POMDP.

We observe that compared to taking the next observation into account, the defined FSC lags behind, and needs an additional step. We can modify the POMDP to give the memory structure time to update.