Finite-state Controllers of POMDPs via 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) that are provably correct with respect to given specifications. The key insight is that computing (randomised) FSCs on POMDPs is equivalent to (and computationally as hard as) synthesis for parametric Markov chains (pMCs). This correspondence enables using black-box techniques to compute correct-by-construction FSCs for POMDPs for a wide range of properties. Our experimental evaluation on typical POMDP problems shows that we are competitive to state-of-the-art POMDP solvers.

1 Introduction

Partially Observable MDPs. We intend to provide guarantees for planning scenarios given by dynamical systems with uncertainties. In particular, we want to synthesise a *strategy* for an agent that ensures certain desired behaviour [18]. A popular formal model for planning subject to stochastic behaviour are Markov decision processes (MDPs) [27]. 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 [20, 31, 36, 39]. 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 (typically uncountably infinite) MDP, referred to as the *belief MDP* [34].

The POMDP Synthesis Problem. For (PO)MDPs, a *randomised strategy* is a function that resolves the nondeterminism by providing a probability distribution over actions at each time step. In general, strategies depend on the full history of the current evolution of the (PO)MDP. If a strategy depends only on the current state of the system, it is called *memoryless*. For MDPs, memoryless strategies suffice to induce optimal values according to our measures of interest [27]. Contrarily, POMDPs require strategies taking the full observation history into account [29], e. g. in case of infinite-horizon objectives. Moreover, strategies inducing *optimal* values are computed by assessing the entire belief MDP [3, 22, 24, 35], rendering the problem undecidable [6].

POMDP strategies can be represented by *infinite-state controllers*. For computational tractability, strategies are often restricted to finite memory; this amounts to using *randomised finite-state controllers* (FSCs) [23]. We often refer to strategies as FSCs. The product of a POMDP and an FSC yields a POMDP with a larger state space. In this product, it suffices to compute a memoryless randomised strategy. Computing such a strategy is NP-hard, SQRT-SUM-hard, and in PSPACE [37]. While optimal values cannot be guaranteed, finite memory in combination with *randomisation* may supersede infinite memory in many cases [1, 7].

Correct-by-Construction Strategy Computation. In this paper, we synthesise FSCs for POMDPs. We require these FSCs to be provably correct for specifications such as indefinite horizon properties like expected reward or reach-avoid probabilities. State-of-the-art POMDP solvers mainly consider expected discounted reward measures [38]. Note that reach-avoid probabilities cannot sufficiently be simulated by *reward engineering*, as complex requirements may trigger negative side-effects or hide potential bugs [32].

Our key observation is that for a POMDP the *set of all FSCs* with a fixed memory bound can be succinctly represented by a *parametric Markov chain* (pMC) [12]. Transitions of pMCs are given by functions over a finite set of parameters rather than constant probabilities. The *parameter synthesis* problem for pMCs is to determine parameter

Table 1: Correspondence

POMDP under FSC	pMC
states × memory	states
same observation	same parameter
strategy	parameter instantiation

instantiations that satisfy (or refute) a given specification. We show that the pMC parameter synthesis problem and the POMDP strategy synthesis problem are equally hard. This correspondence not only yields complexity results [19], but particularly enables using a plethora of methods for parameter synthesis implemented in sophisticated and optimised parameter synthesis tools like PARAM [17], PRISM [21], and PROPhESY [14]. They turn out to be competitive alternatives to dedicated POMDP solvers. Moreover, as we are solving partially different problems, our methods are orthogonal to, e. g., PRISM-POMDP [24] and solve-POMDP [38].

We detail our contributions and the structure of the paper, which starts with necessary formalisms in Sect. 2.

Section 3: We establish the correspondence between POMDPs and pMCs. Consider Table 1. The product of a POMDP and an FSC yields a POMDP with state-memory pairs. These are mapped to states in the pMC. If POMDP states share an *observation*, the corresponding pMC states share *parameters* at their emanating transitions. A *strategy* of the POMDP corresponds to a *parameter instantiation* in the pMC.

Section 4: We show the opposite direction, namely a transformation from pMCs to POMDPs. This result establishes that the synthesis problems for POMDPs and pMCs are equally hard. Technically, we identify the practically relevant class of *simple pMCs*, which coincides with POMDPs under memoryless strategies.

Section 5: Specific types of FSCs differ in the information they take into account, e. g. the last action that has been taken by an agent. We compare existing definitions from the literature and discuss their effect in our setting.

Section 6: Typical restrictions on parameter instantiations concern whether parameters may be assigned the probability zero. We discuss effects of such restrictions to the resulting POMDP strategies.

Section 7: We show how to compute correct-by-construction FSCs using our techniques. To that end, we explain how particular parameter synthesis approaches deliver optimal or near-optimal FSCs. Then, we evaluate the approach on a range of typical POMDP benchmarks. We observe that often, only little memory suffices. Our approach is competitive to state-of-the-art POMDP solvers and is able to synthesise small, almost-optimal FSCs.

Related Work. In addition to the cited works, [23] uses a branch-&-bound method to find 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 [5, 6]. Work on parameter synthesis [10, 16, 19] 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 $Distr(X)$. The support of a distribution μ is $\mathrm{supp}(\mu) = \{x \in X \mid \mu(x) > 0\}$. A distribution is *Dirac* if $|\mathrm{supp}(\mu)| = 1$.

Let $V = \{p_1, \dots, p_n\}$ be a finite set of *parameters* over the domain $\mathbb{R}$ and let $\mathbb{Q}[V]$ be the set of multivariate polynomials over V. An *instantiation* for V is a function $u\colon V \to \mathbb{R}$. Replacing each parameter p in a polynomial $f \in \mathbb{Q}[V]$ by $u(p)$ yields $f[u] \in \mathbb{R}$; $f \neq 0$ holds if $f[u] \neq 0$ for some instantiation u.

Decision problems can be considered as languages describing all positive instances. A language $L_1 \subseteq \{0,1\}^*$ is *polynomial (many-one or Karp) reducible* to $L_2 \subseteq \{0,1\}^*$, written $L_1 \leqslant_P L_2$, if there exist a polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that for all $w \in \{0,1\}^*$, $w \in L_1 \iff f(w) \in L_2$. Polynomial reductions are essential to define complexity classes, cf. [25].

2.1 Parametric Probabilistic Models

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

The *available actions* in $s \in S$ are $Act(s) = \{a \in Act \mid \exists s' \in S : \mathcal{P}(s, a, s') \neq 0\}$. We assume that pMDP M contains no deadlock states, i. e. $Act(s) \neq \emptyset$ for all $s \in S$. A *path* of a pMDP M is an (in)finite sequence $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots$, where $s_0 = s_I$, $s_i \in S$, $a_i \in Act(s_i)$, and $\mathcal{P}(s_i, a_i, s_{i+1}) \neq 0$ for all $i \in \mathbb{N}$. For finite π, $\mathrm{last}(\pi)$ denotes the last state of π. The set of (in)finite paths of M is Paths^M_{fin} (Paths^M).

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

A *(parametric) discrete-time Markov chain* ((p)MC) is a (p)MDP with $|Act(s)| = 1$ for all $s \in S$. For a pMC D, we may omit the actions and use the notation $D = (S, s_I, V, P)$ with a transition function P of the form $P\colon S \times S \to \mathbb{Q}[V]$. This is analogous for (non-parametric) MCs.

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 $f[u]$. $M[u]$ is also called the *instantiation* of M at u. 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* $\sigma\colon \mathsf{Paths}^M_{fin} \to Distr(Act)$ *s. t.* $\mathrm{supp}(\sigma(\pi)) \subseteq Act(\mathrm{last}(\pi))$ *for all* $\pi \in \mathsf{Paths}^M_{fin}$*. The set of all strategies of* M *is* Σ^M.

A strategy σ is *memoryless* if $\mathrm{last}(\pi) = \mathrm{last}(\pi')$ implies $\sigma(\pi) = \sigma(\pi')$ for all $\pi, \pi' \in \mathsf{Paths}^M_{fin}$. It is *deterministic* if $\sigma(\pi)$ is a Dirac distribution for all $\pi \in \mathsf{Paths}^M_{fin}$. A strategy that is not deterministic is *randomised*.

A strategy σ for an MDP M resolves all nondeterministic choices, yielding an *induced Markov chain* 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) *For an MDP* $M = (S, s_\mathrm{I}, Act, \mathcal{P})$ *and a strategy* $\sigma \in \Sigma^M$*, the MC induced by* M *and* σ *is given by* $M^\sigma = (\mathsf{Paths}^M_{fin}, s_\mathrm{I}, P^\sigma)$ *where:*

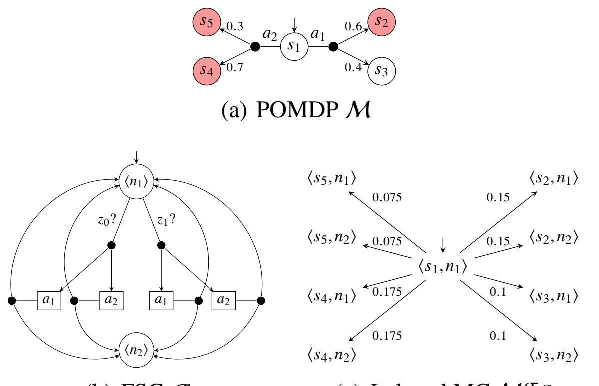
$$P^\sigma(\pi, \pi') = \begin{cases} \mathcal{P}(\mathrm{last}(\pi), a, s') \cdot \sigma(\pi)(a) & \text{if } \pi' = \pi a s', \\ 0 & \text{otherwise.} \end{cases}$$

2.2 Partial Observability

Definition 5 (POMDP) *A* partially observable MDP (POMDP) *is a tuple* $\mathcal{M} = (M, Z, O)$*, with* $M = (S, s_\mathrm{I}, Act, \mathcal{P})$ *the* underlying MDP of $\mathcal{M}$*,* Z *a finite set of observations and* $O\colon S \to Z$ *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, 34] 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 colouring.

We lift the observation function to paths: For $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots s_n \in \mathsf{Paths}^M_{fin}$, 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.


s_5
0.3
a_2
s_1
a_1
0.6
s_2
s_4
0.7
0.4
s_3
$\langle n_1\rangle$
z_0?
z_1?
a_1
a_2
a_1
a_2
$\langle n_2\rangle$
$\langle s_5, n_1\rangle$
0.075
0.15
$\langle s_2, n_1\rangle$
$\langle s_5, n_2\rangle$
0.075
0.15
$\langle s_2, n_2\rangle$
$\langle s_1, n_1\rangle$
$\langle s_4, n_1\rangle$
0.175
0.1
$\langle s_3, n_1\rangle$
0.175
0.1
$\langle s_4, n_2\rangle$
$\langle s_3, n_2\rangle$

(a) POMDP $\mathcal{M}$

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

Figure 1: (a) The 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). (b) The associated (partial) FSC $\mathcal{A}$ has two memory nodes. (c) The 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* $\sigma(\pi) = \sigma(\pi')$ *for all* $\pi, \pi' \in \mathsf{Paths}^M_{fin}$ *with* $O(\pi) = O(\pi')$. $\Sigma^\mathcal{M}$ *is the set of observation-based strategies for* $\mathcal{M}$.

An observation-based strategy selects actions based on observations 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 define *finite-state controllers* (FSCs). We discuss alternative definitions from the literature in Sect. 5. A randomised 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 refer to observation-based strategies as FSCs.

Definition 7 (FSC) *A* finite-state controller (FSC) *for a POMDP* $\mathcal{M}$ *is a tuple* $\mathcal{A} = (N, n_\mathrm{I}, \gamma, \delta)$*, where* N *is a finite set of* memory nodes, $n_\mathrm{I} \in N$ *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* $FSC^\mathcal{M}_k$ *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 next action a 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 $\delta(n, z, a)$.

Example 1 *Fig. 1(b) shows 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.*

Now recall the POMDP $\mathcal{M}$ from Fig. 1(a). The induced MC $\mathcal{M}^{\sigma_\mathcal{A}}$ is shown in Fig. 1(c). Assume $\mathcal{M}$ is in state s_1 and $\mathcal{A}$ in node n_1. Based on the observation $z_0 := O(s_1)$, $\sigma_\mathcal{A}$ chooses action a_1 with probability $\delta(n_1, z_0)(a_1) = 0.5$ leading to the probabilistic branching in the POMDP. With probability 0.6, $\mathcal{M}$ evolves to state s_2. Next, the FSC $\mathcal{A}$ updates its memory node; with probability $\delta(n_1, z_0, a_1)(n_1) = 0.5$, $\mathcal{A}$ stays in 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*, a set $B \subseteq S$ of *bad states*, and a *threshold* $\lambda \in [0,1]$, we consider *quantitative reach-avoid specifications* $\varphi = \mathbb{P}_{>\lambda}(\neg B \cup G)$. The specification φ is satisfied for a strategy $\sigma \in \Sigma^\mathcal{M}$ if the probability $\mathrm{Pr}^{\mathcal{M}^\sigma}(\neg B \cup G)$ 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 provided that one exists. For an MDP M, there is a memoryless deterministic strategy inducing the maximal probability $\mathrm{Pr}^M_{\max}(\neg B \cup G)$ [9]. For a POMDP $\mathcal{M}$, however, observation-based strategies with infinite memory as in Def. 6 are necessary [29] to attain $\mathrm{Pr}^\mathcal{M}_{\max}(\neg B \cup G)$. The problem of proving the satisfaction of φ is therefore undecidable [6]. In our experiments, we also use *undiscounted expected reachability reward properties* [2].

3 From POMDPs to pMCs

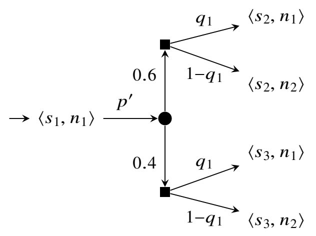
Our goal is to make pMC synthesis methods available for POMDPs. In this section we provide a transformation from a POMDP $\mathcal{M}$ to a pMC D. We consider the following decision problems.

Problem 1 ($\exists k$-FSC) *Given a POMDP $\mathcal{M}$, a specification φ, and a (unary encoded) memory bound $k > 0$, is there a k-FSC $\mathcal{A}$ with $\mathcal{M}^{\sigma_\mathcal{A}} \models \varphi$?*

Problem 2 ($\exists$INST) *Given a pMC D and a specification φ, is there a well-defined instantiation u such that $D[u] \models \varphi$?*

Theorem 1 *$\exists k$-FSC $\leqslant_P$ $\exists$INST.*

The converse direction is addressed in Sect. 4. Consider a POMDP $\mathcal{M}$, a specification φ, and a memory bound $k > 0$ for which $\exists k$-FSC is to be solved. The degrees of freedom to select a k-FSC are given by the possible choices for γ and δ. For each γ and δ, we get a different induced MC, but these MCs are *structurally similar* and can be represented by a single pMC.



q_1
$\langle s_2, n_1\rangle$
0.6
$1{-}q_1$
$\langle s_2, n_2\rangle$
p'
$\to \langle s_1, n_1\rangle$
0.4
q_1
$\langle s_3, n_1\rangle$
$1{-}q_1$
$\langle s_3, n_2\rangle$

(a) Induced pMC

Act	$\mathcal{P}$	Node	Result
$a_1 : p$	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	$n_1 : q_2$	$0.7 \cdot (1 - p) \cdot q_2$
		$n_2 : 1 - q_2$	$0.7 \cdot (1 - p) \cdot (1 - q_2)$
	0.3	$n_1 : q_2$	$0.3 \cdot (1 - p) \cdot q_2$
		$n_2 : 1 - q_2$	$0.3 \cdot (1 - p) \cdot (1 - q_2)$

(b) Parameterised transition probabilities

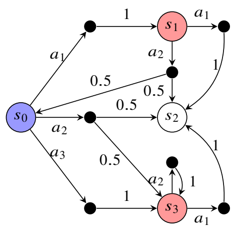
Figure 2: Induced parametric Markov chain for FSCs.

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) shows the induced pMC for action choice a_1. For instance, the transition from $\langle s_1, n_1\rangle$ to $\langle s_2, n_1\rangle$ is labelled with polynomial $p \cdot 0.6 \cdot q_1$.*
We collect all polynomials for observation z_0 in Fig. 2(b). The result *column describes a* parameterised distribution *over tuples of states and memory nodes. Thus, instantiations for these polynomials need to sum up to one.*

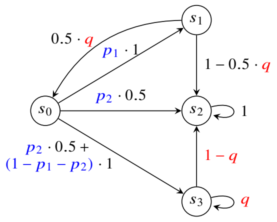
As the next step, we define the pMC that results from combining a k-FSC with a POMDP. The idea is to assign parameters as arbitrary probabilities to action choices. Each observation has one *remaining action* given by a mapping *Remain*: $Z \to Act$. $Remain(z) \in Act(z)$ is the action to which, after choosing probabilities for all other actions in $Act(z)$, the remaining probability is assigned. 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 $M = (S, s_I, Act, \mathcal{P})$ and let $k > 0$ be a memory bound. The* induced pMC *$D_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{I,\mathcal{M},k}, V_{\mathcal{M},k}, P_{\mathcal{M},k})$ is defined by:*

- $S_{\mathcal{M},k} = S \times \{0, \ldots, k-1\}$
- $s_{I,\mathcal{M},k} = \langle s_I, 0\rangle$
- $V_{\mathcal{M},k} = \{p_a^{z,n} \mid z \in Z, n \in \{0, \ldots, k-1\},$
  $a \in Act(z), a \neq Remain(z)\}$



(a) POMDP $\mathcal{M}$



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

Figure 3: From POMDPs to pMCs ($k = 1$)

$$\cup \{q_{a,n'}^{z,n} \mid z \in Z, n, n' \in \{0, \ldots, k-1\}, \\ n' \neq k-1, a \in Act(z)\}$$

- $P_{\mathcal{M},k}(s, s') = \sum_{a \in Act(s)} H(s, s', a)$ *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* $H(\langle s, n\rangle, \langle s', n'\rangle, a) =$

$$\mathcal{P}(s, a, s') \cdot \begin{Bmatrix} p_a^{z,n}, & \text{if } a \neq Remain(z) \\ 1 - \sum\limits_{b \neq a} p_b^{z,n}, & \text{if } a = Remain(z) \end{Bmatrix}$$
$$\cdot \begin{Bmatrix} q_{a,n'}^{z,n}, & \text{if } n' \neq k-1 \\ 1 - \sum\limits_{\tilde{n} \neq n'} q_{a,\tilde{n}}^{z,n}, & \text{if } n' = k-1 \end{Bmatrix}.$$

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

Example 3 *Consider the POMDP in Fig. 3(a) and let $k = 1$. The induced pMC is given in Fig. 3(b). The three actions from s_0 have probability p_1, p_2, and $1 - p_1 - p_2$ for the* remaining action *a_3. From the indistinguishable states s_1, s_3, actions have probability q and $1 - q$, respectively.*

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

Theorem 2 (Correspondence Theorem) *Let $\mathcal{M}$ be a POMDP, k a memory bound, and $D_{\mathcal{M},k}$ the induced pMC:*

$$\{D_{\mathcal{M},k}[u] \mid u \text{ well-defined}\} = \{\mathcal{M}^{\sigma_{\mathcal{A}}} \mid \mathcal{A} \in FSC_k^{\mathcal{M}}\}.$$

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

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 and let φ be a specification. For every well-defined instantiation u of $D_{\mathcal{M},k}$ and the corresponding k-FSC $\mathcal{A}_u$ we have:*
$$\mathcal{M}^{\sigma_{\mathcal{A}_u}} \models \varphi \iff D_{\mathcal{M},k}[u] \models \varphi.$$

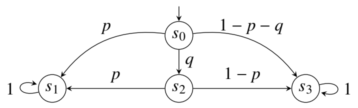


Figure 4: Non-simple pMC

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

4 From pMCs to POMDPs (and back again)

In the previous section we have shown that ∃k-FSC is at least as hard as ∃INST. We now discuss whether both problems are equally hard: The open question is whether we can reduce ∃INST to ∃k-FSC.

A straightforward reduction would maintain the states of the pMC in the POMDP, or even yield a POMDP with the same graph structure (the topology) as the pMC. The following example shows that such a naive reduction is not possible.

Example 4 *In the pMC in Fig. 4 the parameter p occurs in two different distributions (at s_0 and s_2). For defining a reduction where the resulting POMDP has the same set of states, there are two options for the observation function at the states s_0 and s_2: 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 choice in a POMDP. Then $O(s_0) = O(s_2)$ is impossible as s_0 and s_3 have a different number of outgoing transitions (outdegree). Adding a self-loop to s_2 does not alleviate the problem.*
Moreover, $O(s_0) \neq O(s_2)$ is impossible, as a strategy could distinguish s_0 and s_2 and assign different probabilities to p.

The pMC in the example is problematic as the parameters occur at the outgoing transitions of states in different combinations. We restrict ourselves to an important subclass[1] of pMCs which we call *simple pMCs*. A pMC is simple if for all states s, s', $P(s, s') \in \mathbb{Q} \cup \{p, 1-p \mid p \in V\}$. Consequently, we can map states to parameters, and use this map to define the observations. The transformation from a POMDP to a pMC then is the reverse of the transformation from Def. 8. In the remainder, we detail this correspondence. The correspondence also establishes a construction to compute k-FSCs via parameter synthesis on *simple* pMCs. Current tool-support (cf. Sect. 7) for simple pMCs is more mature than for the more general pMCs obtained via Def. 8.

Let simple-∃INST be the restriction of ∃INST to simple pMCs. Similarly, let simple-∃1-FSC be a variant of ∃1-FSC that only considers *simple POMDPs*.

[1]All pMC benchmarks from the PARAM webpage [26] are simple pMCs.

Definition 9 (Binary/Simple POMDP) *A POMDP is* binary, *if* $|Act(s)| \leqslant 2$ *for all* $s \in S$. *A binary POMDP is* simple, *if for all* $s \in S$

$$|Act(s)| = 2 \implies \forall a \in Act(s)\ \exists s' \in S : P(s,a,s') = 1.$$

We establish the following relation between the POMDP and pMC synthesis problems, which asserts that the problems are equivalently hard.

Theorem 3 *For any* $L_1, L_2 \in \{\exists k$-*FSC*, $\exists 1$-*FSC*, *simple*-$\exists 1$-*FSC*, *simple*-$\exists$*INST* $\}$, $L_1 \leqslant_P L_2$.

The proof is a direct consequence of the Lemmas 2-5 below, as well as the facts that every 1-FSC is a k-FSC, and every simple POMDP is a POMDP.

The induced pMC $D_{\mathcal{M},1}$ of a simple POMDP $\mathcal{M}$ is also simple. Consequently, Sect. 3 yields:

Lemma 2 *simple*-$\exists 1$-*FSC* $\leqslant_P$ *simple*-$\exists$*INST.*

4.1 From Simple pMCs to Simple POMDPs

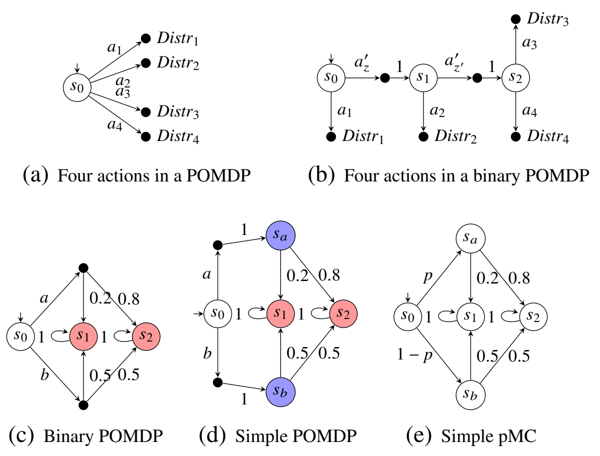
Theorem 4 *Every simple pMC D with n states and m parameters is isomorphic to* $D_{\mathcal{M},1}$ *for some simple POMDP* $\mathcal{M}$ *with n states and m observations.*

We refrain from a formal proof: The construction is the reverse of Def. 8, with observations $\{z_p \mid p \in V_D\}$. 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.

Lemma 3 *simple*-$\exists$*INST* $\leqslant_P$ *simple*-$\exists 1$-*FSC.*

4.2 From General POMDPs to Simple POMDPs

We present a reduction from $\exists 1$-FSC to simple $\exists 1$-FSC by translating a (possibly not simple) POMDP into a *binary* POMDP and subsequently into a *simple* POMDP. Examples are given in Fig. 5(a–e). We emphasise that our construction only preserves the expressiveness of 1-FSCs. There are several ways to transform a POMDP into a binary POMDP. We illustrate one in Fig. 5(a–b). The idea is to split actions at a state with more than two actions into two sets, which are then handled by fresh states with fresh observations. The transformation iteratively reduces the number of actions until every state has at most two outgoing actions. To ensure a one-to-one correspondence between 1-FSCs of the original POMDP and the transformed POMDP, all states with the same observation should be handled the same way.



(a) Four actions in a POMDP (b) Four actions in a binary POMDP

(c) Binary POMDP (d) Simple POMDP (e) Simple pMC

Figure 5: POMDP $\leftrightarrow$ simple pMC

In particular, the same observations should be used for the introduced auxiliary states.

The transformation from binary POMDP to simple POMDP is illustrated by Fig. 5(c–d). 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 [33]. Fig. 5(e) shows the induced simple pMC.

Lemma 4 $\exists 1$-*FSC* $\leqslant_P$ *simple*-$\exists 1$-*FSC.*

4.3 From k-FSCs to 1-FSCs

Given a POMDP $\mathcal{M}$ and a memory bound $k > 1$ we construct a POMDP $\mathcal{M}_k$ such that $\mathcal{M}$ satisfies a specification φ under some k-FSC iff $\mathcal{M}_k$ satisfies φ under some 1-FSC.

Definition 10 (k-Unfolding) *Let* $\mathcal{M} = (M, Z, O)$ *be a POMDP with* $M = (S, s_{\mathrm{I}}, Act, \mathcal{P})$, *and* $k > 1$. *The* k-unfolding *of* $\mathcal{M}$ *is the POMDP* $\mathcal{M}_k = (M_k, Z_k, O_k)$ *with* $M_k = (S_k, s_{\mathrm{I},k}, Act_k, \mathcal{P}_k)$ *defined by:*

- $S_k = S \times \{0, \dots k{-}1\}$
- $s_{\mathrm{I},k} = \langle s_{\mathrm{I}}, 0\rangle$
- $Act_k = Act \times \{0, \dots, k{-}1\}$
- $\mathcal{P}_k(\langle s,n\rangle, \langle a,\bar{n}\rangle, \langle s',n'\rangle) = \begin{cases} \mathcal{P}(s,a,s') & \textit{if } n' = \bar{n} \\ 0 & \textit{otherwise.} \end{cases}$
- $Z_k = Z \times \{0, \dots, k{-}1\}$
- $O_k(\langle s,n\rangle) = \langle O(s), n\rangle.$

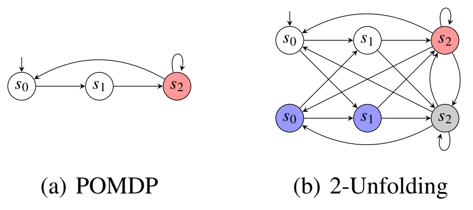


Figure 6: Unfolding a POMDP for two memory states

Intuitively, $\mathcal{M}_k$ stores the current memory node into its state space. At state $\langle s, n \rangle$ of $\mathcal{M}_k$, a 1-FSC can not only choose between the available actions $Act(s)$ in $\mathcal{M}$ but also between different successor memory nodes.

Fig. 6 shows this process for $k = 2$. All states of the POMDP are copied once. Different observations allow to determine in which copy of a state – and therefore, which memory cell – we currently are. Additionally, all actions are duplicated to model the option for a strategy to switch the memory cell.

The induced pMC $D_{\mathcal{M}_k,1}$ of the k-unfolding of $\mathcal{M}$ has the same topology as the induced pMC $D_{\mathcal{M},k}$ of $\mathcal{M}$ with memory bound k. In fact, both pMCs have the same instantiations.

Proposition 1 *For POMDP $\mathcal{M}$ and memory bound k:*

$$\{D_{\mathcal{M}_k,1}[u] \mid u \textit{ well-defined}\} = \{D_{\mathcal{M},k}[u] \mid u \textit{ well-defined}\}.$$

The intuition is that in both pMCs the parameter instantiations reflect arbitrary probability distributions over the same set of successor states. In the transition probability function of the induced pMC $D_{\mathcal{M},k}$ of $\mathcal{M}$ we can also substitute the multiplications of parameters $p_a^{z,n}$ and $q_{a,n'}^{z,n}$ by single parameters. This yields a *substituted induced pMC* which is then isomorphic to the induced pMC $D_{\mathcal{M}_k,1}$ of the k-unfolding of $\mathcal{M}$. More details are given in Appendix A

From Prop. 1 and Thm. 2 we get that induced MCs of $\mathcal{M}$ under k-FSCs coincide with induced MCs of $\mathcal{M}_k$ under 1-FSCs, i.e., $\{\mathcal{M}^{\sigma_{\mathcal{A}}} \mid \mathcal{A} \in FSC_k^{\mathcal{M}}\} = \{\mathcal{M}_k^{\sigma_{\mathcal{A}}} \mid \mathcal{A} \in FSC_1^{\mathcal{M}}\}$

Lemma 5 $\exists k$-*FSC* $\leqslant_P \exists 1$-*FSC.*

5 Alternative FSCs

In the literature, several formalisms for FSCs occur. In particular, [2, 5, 23] do not agree upon a common model. We discuss the applicability of our results with respect to the different variants of FSCs.

Ignoring the Taken Action for Updates. In [2, 23], the memory update is of the form $\delta' \colon N \times Z \to Distr(N)$. The update is a restriction of the FSCs in this paper, represented by the constraint $\delta(n, z, a_1) = \delta(n, z, a_2)$. The constraint yields dependencies between different actions, preventing the k-unfolding as in Def. 10. We present an alternative to the induced pMC as in Def. 8, respecting this variant of k-FSCs in Appendix B.

Taking the Next Observation into Account. In this paper, the memory node update in FSCs depends on the observation at the state *before* executing the action. Instead, the update may also be based on the observation *after* the update [23]. This notion introduces dependencies between actions from states with different observations that reach the same observation. The dependencies can be eliminated by transforming the POMDP. Appendix B details how a transformation to a pMC is possible with this memory model.

Ignoring the Current Observation when Selecting the Action. In [5], the action mapping is modeled as $\gamma' \colon N \to Distr(Act)$, 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, this model is not compatible with our framework.

6 Strategy/Parameter Restrictions

Two typical restrictions on the parameters are usually made in parameter synthesis for pMCs:

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

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

Accordingly, we define restrictions to POMDP strategies that correspond to such restricted parameter instantiations.

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

Non-zero strategies ensure that $\text{supp}(\sigma(s)) = Act(s)$. The example below shows the potential impact on reachability probabilities.

Example 5 *The MDP M in Fig. 6 has a choice between actions a_1 and a_2 at state s_0. If action a_1 is chosen with probability zero, the probability to reach s_1 from becomes zero, and the corresponding parameter instantiation is not*

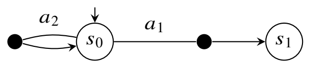


Figure 7: MDP $\mathcal{M}$

graph-preserving. Contrarily, if a_1 is chosen with any positive probability, as would be enforced by a non-zero strategy, the probability to reach s_1 is one.

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).*

Still, for the considered specifications, we can, w.l.o.g., restrict ourselves to FSCs that induce non-zero strategies.

Theorem 5 *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 theorem is a consequence of the corresponding statement for pMCs, which we show in Appendix C.

Lemma 6 *For pMC D and $\varphi = \mathbb{P}_{>\lambda}(\neg B \cup G)$, either $D[u] \not\models \varphi$ for all well-defined instantiations u, or $D[u] \models \varphi$ for some graph-preserving instantiation.*

7 Empirical Evaluation

Above, we established the correspondence between the synthesis problems for POMDPs and pMCs. Now, we discuss the available methods for pMC parameter synthesis, and how they may be exploited or adapted to synthesise FSCs. All techniques mentioned below are supported by the tool PROPhESY [14]. We distinguish three key problems:

(1) *Find a correct-by-construction strategy for a POMDP and a specification.* To construct such a strategy, one needs to find a parameter valuation for the pMC that provably satisfies the specification. Most solution techniques focused on pMCs with a few parameters, rendering the problem at hand infeasible. Recently, efficient approaches emerged that are either based on particle swarm optimisation (PSO) [8] or on convex optimisation [1,10], in particular using quadratically-constrained quadratic programming (QCQP) [11]. We employ PSO and QCQP for our evaluation.

(2) *Prove that no FSC exists for a POMDP and a specification.* Proving the absence of an FSC also allows us to show ε-optimality of a previously synthesised strategy. Two approaches exist: An approximative technique called *parameter lifting* [28] and a method based on SAT-modulo-theories (SMT) solving [13].

Table 2: Benchmarks

		POMDP $\mathcal{M}$			PRISM-POMDP		SolvePOMDP		MDP
Id	Name	States	Bran.	Obs.	Result	Time	Result	Time	Res
1	NRP (8)	125	161	41	[0.125, 0.24]	20		TO	1.0
2	Grid (4)	17	62	3	[3.97, 4.13]	1038	4.13	0.4	3.2
3	Netw (3,4,8)	2729	4937	361		TO		TO	0.83
4	Crypt (5)	4885	11733	890		MO		TO	1.0
5	Maze (2)	16	58	8	[5.11, 5.23]	3.9	5.23	16	4.0
6	Load (8)	16	28	5	[10.5, 10.5]	1356	10.5	7.6	10.5
7	Slippery (4)	17	59	4		TO	0.93	95	1.0

(3) *Provide a closed-form solution* for the underlying measure of a specification in form of a function over the induced parameters of an FSC. The function may be used for further analysis, e.g., of the sensitivity of decisions or parameter values, respectively. To compute this function, all of the parameter synthesis tools PARAM [17], PRISM [21], Storm [15], and PROPhESY [14] employ a technique called *state elimination* [12].

Implementation and Setup. We extended the tool Storm [15] to parse and store POMDPs, and implemented several transformation options to pMCs. Most notably, Storm supports k-unfolding, the product with several restricted FSCs such as counters that can be incremented at will, and several types of transformation to (simple) pMCs.

We evaluate on a HP BL685C G7 with 48 2 GHz cores, a 16 GB memory limit, and 1800 seconds time limit. The methods compared here are single-threaded. We took *all* POMDPs from PRISM-POMDP [24], additional maze, load/unload examples from [23], and a slippery gridworld with traps inspired by [31]. Table 2 gives further details. We list the number of states, branches, and observations in each POMDP. As a baseline, we provide the results and run time of the model-checking tool PRISM-POMDP, and the point-based solver SolvePOMDP [38], obtained with default settings. Both tools compute optimal memory-unbounded strategies and are prototypes. The last column contains the result on the underlying, fully observable MDP. The experiments contain minimal expected rewards, which are analysed by a straightforward extension of maximal reachability probabilities. All pMCs computed are simple pMCs, as PROPhESY typically benefits from the simpler structure. PROPhESY has been invoked with the default set-up.

7.1 Finding strategies

We evaluate how quickly a strategy that satisfies the specification can be synthesised. We vary the threshold used in the specification, as well as the structure of the FSC.

Results. We summarise the obtained results in Table 3. For each instance (Id), we define three thresholds (Ts), ordered from challenging (i.e., close to the optimum) to less challenging. For different types of FSCs (FSC, F=full, C=counter) and memory bounds (k), we obtain pMCs with the given number of states, transitions and parameters. For

Table 3: Synthesing strategies

Id	Ts	FSC/k	States	Trans	Pars	T1		T2		T3	
						pso	qcqp	pso	qcqp	pso	qcqp
1	.124/.11/.09	F/1	75	118	8	<1	<1	<1	<1	<1	<1
		F/2	205	420	47	2	<1	2	<1	2	<1
		F/4	921	1864	215	9	2	9	2	10	2
		F/8	3889	7824	911	43	15	42	14	42	14
2	4.15/4.5/5.5	F/1	47	106	3	-	-	-	-	Err	<1
		F/2	183	390	15	7.4	11	4	9	2	<1
		F/4	719	1486	63	TO	64	39	91	14	8
		F/8	2845	5788	255	TO	700	TO	946	254	69
3	8/10/15	F/1	3268	13094	276	TO	TO	TO	43	22	4
		F/2	16004	46153	1783	TO	TO	TO	877	152	28
		C/2	11270	36171	1168	TO	TO	TO	358	100	62
		C/4	27183	82145	2940	TO	MO	TO	MO	476	MO
4	.25/.2/.15	F/1	3366	6534	364	18	25	18	15	18	12
		F/2	25713	51608	3907	330	MO	350	MO	326	MO
5	5.2/15/25	F/1	30	64	8	-	-	TO	TO	<1	TO
		F/2	137	294	49	TO	TO	14	TO	2	TO
		F/4	587	1214	219	93	TO	TO	TO	26	TO
		F/8	2421	4924	919	TO	TO	1034	TO	115	TO
		C/2	99	212	33	TO	TO	3.7	TO	<1	TO
		C/4	231	476	81	7	TO	6	TO	3	TO
6	10.6/10.9/82.5	F/1	16	33	1	-	-	-	-	<1	TO
		F/2	77	160	11	9	TO	6	TO	<1	TO
		F/4	354	721	63	20	TO	21	63	3	TO
7	.929/.928/.927	F/1	87	184	3	TO	TO	<1	1	<1	<1
		F/2	285	592	15	4	TO	4	20	3	22
		F/4	1017	2080	63	76	767	71	205	67	187
		F/8	3825	7744	255	TO	TO	TO	TO	TO	TO

each threshold (T1,T2,T3), we then report the run time of the two methods PSO and QCQP, respectively. T1 is chosen to be nearly optimal for all benchmarks except Netw (where we do not know the optimum). A dash indicates a combination of memory and threshold that is not realisable according to the results in Sect. 7.2. TO/MO denote violations of the time/memory limit, respectively.

Evaluation. Strategies for thresholds which are suboptimal (T3) are synthesised faster. If the memory bound is increased, the number of parameters quickly grows and the performance of the methods quickly degrades. Additional experiments showed that the number of states has only a minor effect on the performance. Simpler FSC topology for a counter alleviates the blow-up of the pMC and is successfully utilised to find good strategies for larger instances.

Trivially, a k-FSC is also a valid $(k+i)$-FSC for some $i \in \mathbb{N}$. Yet, the larger number of parameters make searching for $(k+i)$-FSCs significantly more difficult. We furthermore observe that the performance of PSO and QCQP is incomparable, and both methods have their merits.

Summarising, many of the POMDPs in the benchmarks only require FSCs with little memory. **We find nearly-optimal, and small, FSCs for POMDP benchmarks with thousands of states within seconds.**

7.2 Proving ε-Optimality

We now focus on evaluating how quickly pMC techniques can prove the absence of a strategy satisfying the specification. Such a proof allows us to draw conclusions about the (ε-)optimality of a strategy synthesised in Sect. 7.1.

Table 4: Proving strategy absence and getting closed-forms

(a) Proving absence

Id	FSC/k	T	time
2	F/1	5	<1
3	F/1	5	8
3	F/4	5	183
4	F/1	0.25	2*
5	F/1	10	3
5	F/2	5	TO
6	F/1	82	<1
6	F/8	10.5	1
7	F/1	0.94	5

(b) Closed-form sol.

Id	FSC/k	time
1	F/1	<1
1	F/2	97
2	F/1	155
3	F/1	464
4	F/1	<1
5	F/1	116
6	F/1	<1
7	F/1	TO

Results. Table 4(a) shows the run times to prove that for the POMDP in column *Id*, there exists no strategy of type *FSC* with k memory that performs better than threshold T. The row indicated by * was obtained with SMT. All other results were obtained with parameter lifting.

Evaluation. We generally can prove tight bounds for k=1. For $k > 1$, the high number of parameters yields a mixed impression, the performance depends on the particular benchmark. Notably, **we can prove non-trivial bounds even for k=8, even if the pMC has hundreds of parameters.**

7.3 Closed-form solutions.

Results. Table 4(b) indicates running times to compute a closed-form solution, i.e., a rational function that maps k-FSCs to the induced probability.

Evaluation. Closed form computation is limited to small memory bounds. The rational functions obtained vary wildly in their structure. For (4), the result is a constant function which is trivial to analyse, while for (3), we obtained rational functions with roughly one million terms, rendering further evaluation expensive.

8 Conclusion

This paper connects two active research areas, namely verification and synthesis for POMDPs and parameter synthesis for Markov models. We see benefits 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, parameter synthesis tools and techniques extend the state-of-the-art approaches for POMDP analysis. Future work will also concern a thorough investigation of *permissive schedulers*, that correspond to regions of parameter instantiations, in concrete motion planning scenarios.

References

[1] Christopher Amato, Daniel S Bernstein, and Shlomo Zilberstein. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, 21(3):293–320, 2010.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[3] Darius Braziunas. POMDP solution methods. *University of Toronto*, 2003.

[4] Krishnendu Chatterjee, Martin Chmelik, and Jessica Davies. A symbolic SAT-based algorithm for almost-sure reachability with small strategies in POMDPs. In *AAAI*, pages 3225–3232. AAAI Press, 2016.

[5] Krishnendu Chatterjee, Martin Chmelík, Raghav Gupta, and Ayush Kanodia. Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.*, 234:26–48, 2016.

[6] Krishnendu Chatterjee, Martin Chmelík, and Mathieu Tracol. What is decidable about partially observable Markov decision processes with ω-regular objectives. *Journal of Computer and System Sciences*, 82(5):878–911, 2016.

[7] Krishnendu Chatterjee, Luca De Alfaro, and Thomas A Henzinger. Trading memory for randomness. In *QEST*. IEEE, 2004.

[8] Taolue Chen, Ernst Moritz Hahn, Tingting Han, Marta Z. Kwiatkowska, Hongyang Qu, and Lijun Zhang. Model repair for Markov decision processes. In *TASE*, pages 85–92. IEEE CS, 2013.

[9] Anne Condon. The complexity of stochastic games. *Inf. Comput.*, 96(2):203–224, 1992.

[10] Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, Ivan Papusha, Hasan A. Poonawala, and Ufuk Topcu. Sequential convex programming for the efficient verification of parametric MDPs. In *TACAS (2)*, volume 10206 of *LNCS*, pages 133–150, 2017.

[11] Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, and Ufuk Topcu. Synthesis in pMDPs: A tale of 1001 parameters. *CoRR*, abs/1803.02884, 2018.

[12] Conrado Daws. Symbolic and parametric model checking of discrete-time Markov chains. In *ICTAC*, volume 3407 of *LNCS*, pages 280–294. Springer, 2004.

[13] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[14] Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Ábrahám. PROPhESY: A probabilistic parameter synthesis tool. In *CAV*, volume 9206 of *LNCS*, pages 214–231. Springer, 2015.

[15] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *CAV (2)*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017.

[16] Antonio Filieri, Carlo Ghezzi, and Giordano Tamburrelli. Run-time efficient probabilistic model checking. In *ICSE*, pages 341–350. ACM, 2011.

[17] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric Markov models. *Software Tools for Technology Transfer*, 13(1):3–19, 2010.

[18] Ronald A. Howard. *Dynamic Programming and Markov Processes*. The MIT Press, 1960.

[19] Lisa Hutschenreiter, Christel Baier, and Joachim Klein. Parametric Markov chains: PCTL complexity and fraction-free Gaussian elimination. In *GandALF*, volume 256 of *EPTCS*, pages 16–30, 2017.

[20] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1):99–134, 1998.

[21] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[22] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *AAAI*, pages 541–548. AAAI Press, 1999.

[23] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R Cassandra. Solving POMDPs by searching the space of finite policies. In *UAI*, pages 417–426. Morgan Kaufmann Publishers Inc., 1999.

[24] Gethin Norman, David Parker, and Xueyi Zou. Verification and control of partially observable probabilistic systems. *Real-Time Systems*, 53(3):354–402, 2017.

[25] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[26] PARAM Website, 2015. `http://depend.cs.uni-sb.de/tools/param/`.

[27] Martin L. Puterman. *Markov Decision Processes*. John Wiley and Sons, 1994.

[28] Tim Quatmann, Christian Dehnert, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Parameter synthesis for Markov models: Faster than ever. In *ATVA*, volume 9938 of *LNCS*, pages 50–67. Springer, 2016.

[29] Sheldon M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, Inc., 1983.

[30] Nicholas Roy, Geoffrey J. Gordon, and Sebastian Thrun. Finding approximate POMDP solutions through belief compression. *J. Artif. Intell. Res.*, 23:1–40, 2005.

[31] Stuart J. Russell and Peter Norvig. *Artificial Intelligence – A Modern Approach (3. ed.)*. Pearson Education, 2010.

[32] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young. Machine learning: The high interest credit card of technical debt. In *SE4ML (NIPS 2014 Workshop)*, 2014.

[33] Roberto Segala and Andrea Turrini. Comparative analysis of bisimulation relations on alternating and non-alternating probabilistic models. In *QEST*, pages 44–53. IEEE CS, 2005.

[34] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[35] Daniel Szer and François Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *ECML*, pages 389–399. Springer, 2005.

[36] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[37] Nikos Vlassis, Michael L. Littman, and David Barber. On the computational complexity of stochastic controller optimization in POMDPs. *ACM Trans. on Computation Theory*, 4(4):12:1–12:8, 2012.

[38] Erwin Walraven and Matthijs T. J. Spaan. Accelerated vector pruning for optimal POMDP solvers. In *AAAI*, pages 3672–3678. AAAI Press, 2017.

[39] Tichakorn Wongpiromsarn and Emilio Frazzoli. Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications. In *CDC*, pages 7644–7651. IEEE, 2012.

A Substituted Induced pMC

We simplify the induced pMC $D_{\mathcal{M},k}$ of a POMDP $\mathcal{M}$ and a memory bound k by substituting the polynomials in the transition probability function as follows.

Consider the POMDP from Fig. 1(a) and its induced pMC for memory bound $k = 2$. Tab. 5 enumerates the outgoing transitions of the pMC state $\langle s_1, n_1\rangle$ (cf. Fig. 3). 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. The obtained pMC is called the *substituted induced pMC*.

Definition 12 (Substituted Induced pMC) *Reconsider Def. 8. We define $D^{\text{subs}}_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{I,\mathcal{M},k}, V^{\text{subs}}_{\mathcal{M},k}, P^{\text{subs}}_{\mathcal{M},k})$ by modifying $V_{\mathcal{M},k}$ and H^{subs} as follows:*

- $V^{\text{subs}}_{\mathcal{M},k} = \{r^{z,n}_{a,n'} \mid z \in Z, n, n' \in \{0, \dots, k-1\}, a \in Act(z) \text{ with } n' \neq k-1 \vee a \neq Remain(z)\}$,
- $H^{\text{subs}}(\langle s,n\rangle, \langle s',n'\rangle, a) =$
$$\mathcal{P}(s,a,s') \cdot \left\{ \begin{array}{l} r^{z,n}_{a,n'}, \quad \text{if } a \neq Remain(z) \vee n' \neq k-1 \\ 1 - \sum\limits_{a' \neq a \vee \bar{n} \neq k-1} r^{z,n}_{a',\bar{n}}, \\ \qquad \text{if } a = Remain(z) \wedge n' = k-1 \end{array} \right\}$$
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* $s,s' \in S_{\mathcal{M},k}$.

It follows:

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

Table 5: Substitution of polynomials in induced pMC

Act	$\mathcal{P}$	Node	induced pMC	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 Alternative FSCs

Ignoring the Taken Action for Updates. In [2, 23], the memory update is of the form $\delta' \colon N \times Z \to Distr(N)$. This is a restriction of the FSCs considered here, represented by the constraint $\delta(n, z, a_1) = \delta(n, z, a_2)$.

Example 6 *Recall Ex. 2, with the induced pMC for the POMDP fragment, as also given in Tab. 2(b). Tab. 6(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. 6(a), this assignment is not possible.*

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

Definition 13 (Action-Restricted Induced pMC) *Reconsider Def. 8. We define $D^{\text{restr}}_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{I,\mathcal{M},k}, V^{\text{restr}}_{\mathcal{M},k}, P^{\text{restr}}_{\mathcal{M},k})$ by modifying $V_{\mathcal{M},k}$ and H^{restr} as follows:*

- $V^{\text{restr}}_{\mathcal{M},k} = \{p^{z,n}_{a} \mid z \in Z, n \in \{0, \dots, k-1\}, a \in Act(z), a \neq Remain(z)\}$
$\cup \{q^{z,n}_{n'} \mid n, n' \in \{0, \dots, k-1\}, n' \neq k-1, z \in Z\}$
- $H^{\text{restr}}(\langle s,n\rangle, \langle s',n'\rangle, a) =$
$$\mathcal{P}(s,a,s') \cdot \left\{ \begin{array}{ll} p^{z,n}_{a}, & \text{if } a{\neq}Remain(z) \\ 1 - \sum\limits_{a' \neq a} p^{z,n}_{a'}, & \text{if } a{=}Remain(z) \end{array} \right\} \cdot \left\{ \begin{array}{ll} q^{z,n}_{n'}, & \text{if } n'{\neq}k{-}1 \\ 1 - \sum\limits_{\bar{n} \neq n'} q^{z,n}_{\bar{n}}, & \text{if } n'{=}k{-}1 \end{array} \right\}$$
with $z = O(s)$
- $P^{\text{restr}}_{\mathcal{M},k}(s,s') = \sum_{a \in Act(s)} H^{\text{next}}(s,s',a)$ *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. 12. 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 [23].

Example 7 *Recall Ex. 2, with the induced pMC for the POMDP fragment, as also given in Tab. 2(b). Tab. 6(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 14 (Next-observation induced pMC) *Reconsider Def. 8. We define $D^{\text{next}}_{\mathcal{M},k} = (S_{\mathcal{M},k}, s_{I,\mathcal{M},k}, V^{\text{next}}_{\mathcal{M},k}, P^{\text{next}}_{\mathcal{M},k})$ by modifying $V_{\mathcal{M},k}$ and H^{next} as follows:*

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

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

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

C Proof of Lemma 6

We show Lemma 6. Assume pMC $\mathcal{P}$ and $\varphi = \mathbb{P}_{>\lambda}(\neg B \cup G)$. We have to show that either $\mathcal{P}[u] \not\models \varphi$ for all well-defined instantiations or $\mathcal{P}[u] \models \varphi$ for some graph-preserving instantiation.

Let f be a function that maps a well-defined instantiation u to the probability $\Pr^{D[u]}(\neg B \cup G)$. The essential idea is that the only reason for a discontinuity of f is a change in the set $S_{=0}$ – states in the pMC from which the probability to reach the target is zero. The number of states in $S_{=0}$ is the smallest under a graph preserving assignment. A discontinuity of f thus implies a reduced reachability probability (there are more states in $S_{=0}$).

As a consequence, if we have to construct an instantiation which reaches a goal with probability $> \kappa$, we can look for such an instantiation among the graph-preserving ones. In particular, this also means that the set of states $S_{=0}$ can be precomputed.

Table 6: 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	$n_1 : \mathbf{q_1}$	$0.3 \cdot (1{-}p') \cdot q_1$
			$n_2 : \mathbf{1{-}q_1}$	$0.3 \cdot (1{-}p') \cdot (1{-}q_1)$