Correct-by-construction Policies for POMDPs

Nils Jansen
Radboud University, Nijmegen
n.jansen@science.ru.nl

Sebastian Junges
Joost-Pieter Katoen
Tim Quatmann
sebastian.junges@cs.rwth-aachen.de
katoen@cs.rwth-aachen.de
tim.quatmann@cs.rwth-aachen.de
RWTH Aachen University

Bernd Becker
Ralf Wimmer
Leonore Winterer
becker@informatik.uni-freiburg.de
wimmer@informatik.uni-freiburg.de
winterer@informatik.uni-freiburg.de
Albert-Ludwigs-Universität Freiburg

Figure 1: The correspondence between policy synthesis for POMDPs and parameter synthesis for parametric Markov chains.

CCS CONCEPTS
• Computing methodologies → Planning and scheduling; Planning under uncertainty.

KEYWORDS
POMDP, MDP, Parameter Synthesis, Formal Verification, Artificial Intelligence

ACM Reference Format:

Summary. In this extended abstract, we discuss how to compute policies with finite memory—so-called finite-state controllers (FSCs)—for partially observable Markov decision processes (POMDPs) that are provably correct with respect to given specifications. In particular, for a POMDP $M$ and a specification $\varphi$, we want to solve the decision problem whether there is a policy $\sigma$ for $M$ with $k$ memory states, such that $M^{\sigma} \models \varphi$. The underlying method is achieved via a marriage of formal verification and artificial intelligence. The key insight is that computing (randomized) FSCs on POMDPs is equivalent to—and computationally as hard as—synthesis for parametric Markov chains (pMCs). The parameter synthesis problem is to decide whether for a parametric Markov chain (pMC) $P$ and a specification $\varphi$ there is a parameter instantiation $u$ such that in the Markov chain (MC) induced by $u$ the specification is satisfied ($P[u] \models \varphi$). The correspondence—depicted in Figure 1—allows to utilize efficient tools for synthesis in pMCs to compute correct-by-construction FSCs on POMDPs.

The POMDP Synthesis Problem. A popular formal model for planning subject to stochastic behavior are Markov decision processes (MDPs) [19]. 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 [13, 21, 24, 27]. 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, the belief MDP [22].
Example. Every POMDP has an underlying MDP which is extended by an observation function. In Fig. 2(a) we see a fragment of the underlying MDP of a POMDP. For instance, from the initial state \( s_1 \), upon choosing action \( a_1 \), there is a probability of 0.6 to reach state \( s_2 \), and probability 0.4 to reach state \( s_3 \). We have two different observations, indicated by the state coloring. In particular, we have \( O(s_1) = O(s_3) = z_0 \) (white) and \( O(s_2) = O(s_4) = O(s_5) = z_1 \) (red).

For (PO)MDPs, a randomized policy is a function that resolves the nondeterminism by providing a probability distribution over actions at each time step. In general, policies depend on the full history of the current evolution of the (PO)MDP. If a policy depends only on the current state of the system, it is called memoryless. For MDPs, memoryless policies suffice to induce optimal values according to infinite-horizon objectives. Moreover, policies inducing optimal values can be computed by solving the entire belief MDP [2, 16, 18, 23], rendering the problem undecidable [5].

POMDP policies can be represented by infinite-state controllers. For computational tractability, policies may be restricted to finite memory; this amounts to using randomized finite-state controllers (FSCs) [17]. We often refer to policies as FSCs. Already the computation of a memoryless (but randomized) policy adhering to a specification is NP-hard, SQRT-SUM-hard, and in PSPACE [25]. While optimal values cannot be guaranteed, small memory in combination with randomisation may superseed large memory in many cases [1, 6].

Example. Fig. 2(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 over actions. For readability, all distributions are uniform and we omit the action mapping for node \( n_2 \).

Now recall the POMDP \( M \) from Fig. 2(a). Applying the FSC \( \mathcal{A} \) to \( M \) yields an induced Markov chain \( M^{\mathcal{A}} \), which is shown in Fig. 2(c). Assume \( M \) is in state \( s_1 \) and \( \mathcal{A} \) in node \( n_1 \). Based on the observation \( z_0 := O(s_1) \), \( \mathcal{A} \) 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, \( M \) 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 \( (s_1, n_1) \) to \( (s_2, n_1) \) in \( M^{\mathcal{A}} \) has probability \( 0.5 \cdot 0.6 \cdot 0.5 = 0.15 \).

Correct-by-Construction Policy Computation. Our aim is to synthesize FSCs for POMDPs. We require these FSCs to be provably correct for given 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 [26], which are a subclass of indefinite horizon properties [14].

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) [7]. 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 instantiations that satisfy (or refute) a given specification. We show that the POMDP parameter synthesis problem and the POMDP policy synthesis problem are equally hard. This correspondence not only yields complexity results [11], but particularly enables using a plethora of methods for parameter synthesis implemented in sophisticated and optimized parameter synthesis tools like PARAM [10], PRISM [15], and PROPhESY [8]. They turn out to be competitive alternatives to dedicated POMDP solvers. Moreover, as we are solving slightly different problems, our methods are orthogonal to, e.g., PRISM-POMDP [18] and solve-POMDP [26].

Example. Consider the POMDP in Fig. 3(a) and let \( k = 1 \). The induced pMC is given in Fig. 3(b). The three actions from \( s_4 \) 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.

**Related Work.** This work has been presented to the AI community in [12]. In addition to the cited works, [17] uses a branch-&-bound method to find optimal FSCs for POMDPs. A SAT-based approach computes FSCs for qualitative properties [3]. For a survey of decidability results and algorithms for broader classes of properties refer to [4, 5]. Work on parameter synthesis [9, 11] might contain additions to the methods considered here.

**Conclusion.** This paper connects two active research areas, namely synthesis for POMDPs and parameter synthesis for Markov models. We see benefits for both areas. On the one hand, the rich application cidability results and algorithms for broader classes of properties refer to [4, 5]. Work on parameter synthesis [9, 11] might contain additions to the methods considered here.

**REFERENCES**


