Explainable Representation of Finite-Memory Policies for POMDPs Using Decision Trees

Extended Abstract

Muqsit Azeem
Technical University of Munich
Germany
muqsit.azeem@tum.de

Debraj Chakraborty
Nanyang Technological University
Singapore
debraj.chakraborty@ntu.edu.sg

Sudeep Kanav
Masaryk University
Brno, Czech Republic
kanav@fi.muni.cz

Jan Křetinský
Masaryk University
Brno, Czech Republic
xkretins@fi.muni.cz

ABSTRACT

Partially Observable Markov Decision Processes (POMDPs) are a fundamental framework for decision-making under uncertainty but often require infinite memory, making implementation infeasible and many problems undecidable. While finite-memory policies provide a practical alternative, they remain complex and challenging to interpret. To address this, we propose a novel *representation* of finite-memory policies that is (i) interpretable and (ii) smaller, enhancing explainability without sacrificing optimality. To that end, we combine Mealy machines and decision trees (DTs); the latter describing simple, stationary parts of the policies and the former describing how to switch among them. We design a translation for finite-state-controller (FSC) policies from standard literature into our new representation, enhancing explainability and compactness while preserving optimality. Notably, our method seamlessly generalizes to other variants of finite-memory policies. Finally, through experiments and multiple case studies, we illustrate the improved explainability and practicality of our approach.

KEYWORDS

Explainability; POMDP; Decision Trees; Finite-State Controllers

ACM Reference Format:
Muqsit Azeem, Debraj Chakraborty, Sudeep Kanav, and Jan Křetinský. 2026. Explainable Representation of Finite-Memory Policies for POMDPs Using Decision Trees: Extended Abstract. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/OQGU5189

1 INTRODUCTION

POMDP and FSC. Partially observable Markov decision processes (POMDPs) provide a foundational framework for decision-making under uncertainty, where the agent lacks complete information about the current state. Unfortunately, decision-making in POMDPs is inherently complex, and optimal policies may require infinite memory. Consequently, practitioners often restrict attention to finite-memory policies, e.g. [1, 2]. Finite-memory policies are formalized as *finite-state controllers (FSCs)*, essentially Mealy machines (finite-state automata with both input and output) where each state corresponds to a stationary (a.k.a. memoryless) policy and the automaton transitions describe how to switch among them. However, while FSCs make POMDP policies implementable, certain issues arise concerning their explainability.

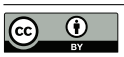
Explainability. The automata representation is often regarded as interpretable and even explainable [5] due to its graphical structure and typically low number of states. However, this issue is more complex as the automaton drawing does not depict the whole policy, notably: (i) Each state corresponds to a stationary policy, which is typically a *table* with an action to be played for each observation; due to its often enormous size it is hard to explain and thus does not expose the goal of that particular policy. (ii) Each transition between states is typically labeled by many observations, effectively hiding the reasons to switch to another policy.

We address this lack of transparency in several steps, exploiting the factored structure of states and observations. First, we translate the tabular representations of the stationary policy in each FSC state into a decision tree (DT). Second, we represent the history-dependent behavior by encoding the memory transitions of the FSC as DTs. Third, we use the resulting DT-FSC representation to provide an *explanation* of the policy.

Our technical contribution can be summarized as follows: In particular, we propose a new data structure to represent finite memory policies, merging the advantages of the Mealy machines and the interpretable *decision trees* (DTs). The latter is used not only for representing the stationary policies similarly to literature, e.g. [3], but importantly also for the automata transitions. Additional structural optimizations and extended experimental results are presented in the full version [4]. Notably, *our transformation procedure alters only the representation obtained from existing tools, preserving both the policies and their optimality.*

2 ILLUSTRATIVE EXAMPLE

We illustrate the representation using the classical cheese-maze POMDP [8] shown in Fig. 1. In this example, the mouse is initially



Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/OQGU5189

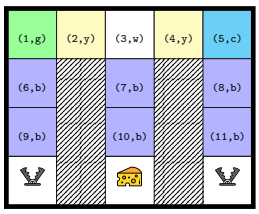
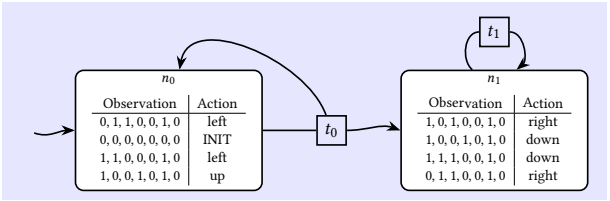


Figure 1: Maze with state-observation labels.



Next observation	Next node
0, 1, 1, 0, 0, 1, 0	n_0
1, 1, 0, 0, 0, 1, 0	n_0
1, 1, 1, 0, 0, 1, 0	n_1
1, 0, 0, 1, 0, 1, 0	n_1
0, 0, 0, 0, 0, 0, 0	n_0
1, 0, 1, 0, 0, 1, 0	n_1

Next observation	Next node
0, 1, 1, 0, 0, 1, 0	n_1
1, 1, 1, 0, 0, 1, 0	n_1
1, 0, 0, 1, 0, 1, 0	n_1
1, 0, 1, 0, 0, 1, 0	n_1

Figure 2: An FSC represented with explicit tables. Each observation is encoded as a vector over the seven observable variables. The transition tables t_0 and t_1 are below nodes n_0 and n_1, respectively.

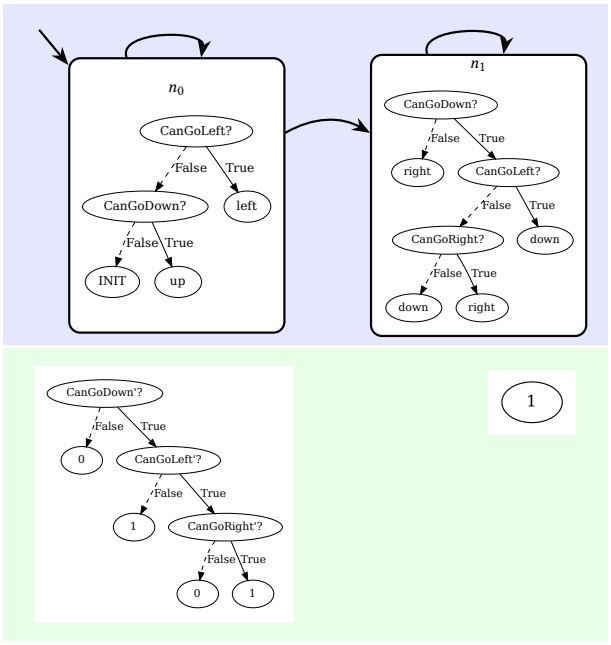


Figure 3: Explaining FSCs using DTs. For an observation variable z, z' would denote the next observation, *i.e.*, the observation the mouse sees after taking the action. A DT in a node describes a stationary policy, a DT below a node describes the next node for the next observation.

placed randomly in the grid in a state with observation blue and it must reach the target while avoiding traps. Observations correspond to local wall configurations (i.e., where the mouse can go to a specific direction), so multiple cells share the same observation.

Finite-State Controller. A winning policy can be represented as the FSC in Fig. 2. Each node defines a stationary policy via an action table, and transitions determine when the controller switches nodes. At each step, the mouse selects an action according to the table of the current node and may transition to another node based on the next observation. *The current node encodes finite memory, implicitly tracking part of the observation history.*

DT-FSC Representation. We translate both the stationary policies and the transition tables into DTs, yielding the DT-FSC in Fig. 3. The FSC structure still captures history, while the decision trees exploit the structure in observations to provide a more compact and interpretable representation. Initially, the mouse follows the policy in n_0: from a blue cell, it repeatedly plays `up` to reach the upper corridor and then `left` to reach the white or green cell. Finally, it switches to the policy in n_1 to reach the target.

Notice, the predicate `CanGoUp` does not appear in any DT, revealing that it is irrelevant for decision-making. The DT-FSC preserves the exact behavior of the original controller while exposing the structure underlying action selection and memory switching.

3 EXPERIMENTAL EVALUATION

We evaluate compactness using representation size as a proxy for explainability. When converting an FSC to a DT-FSC, the number of controller nodes remains unchanged; only the representation of the stationary policies and the memory transitions differs. We therefore compare the number of rows in the tabular FSC with the number of nodes in the corresponding decision trees.

Benchmarks. We consider two benchmark sets: (1) almost-sure reachability benchmarks from [7], and (2) quantitative reachability and expected reward benchmarks from [5]. For the first set, optimal FSCs were obtained using Storm [6]. For the second set, we use the learned FSCs from [5]. DT-FSCs are constructed using dtControl [3].

Results. For the almost-sure reachability benchmarks, DT-FSC reduces the transition function by **13.5**× on average and the action mapping by **1.7**×. For quantitative benchmarks, the transition function is reduced by **5.54**× and the action mapping by **1.66**× on average. The larger reduction for transitions is expected, as transition tables depend on both current and next observations, while action mappings depend only on current observations. This construction requires only seconds per benchmark and serves as a lightweight postprocessing step compared to policy synthesis.

Case Studies. Beyond size, we assess interpretability through detailed case studies on grid navigation and medical decision-making models. These illustrate how DT-FSCs expose structured decision rules that are difficult to extract from tabular FSCs. See the extended version for complete experimental results and case studies [4].

ACKNOWLEDGMENTS

This work has been supported by MUNI Award in Science Humanities grant (MUNI/I/1757/2021), the ERC project InOVationCS (101171844) and the RSS Scheme (NRF-RSS2022-009) by NRF, Singapore.

REFERENCES

[1] Roman Andriushchenko, Alexander Bork, Milan Ceska, Sebastian Junges, Joost-Pieter Katoen, and Filip Macák. 2023. Search and Explore: Symbiotic Policy Synthesis in POMDPs. In *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France (Lncs)*. Springer. https://doi.org/10.1007/978-3-031-37709-9_6

[2] Roman Andriushchenko, Milan Ceska, Sebastian Junges, and Joost-Pieter Katoen. 2022. Inductive synthesis of finite-state controllers for POMDPs. In *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands (Proceedings of Machine Learning Research)*. PMLR. https://proceedings.mlr.press/v180/andriushchenko22a.html

[3] Pranav Ashok, Mathias Jackermeier, Jan Křetínský, Christoph Weinhuber, Maximilian Weininger, and Mayank Yadav. 2021. dtControl 2.0: Explainable Strategy Representation via Decision Tree Learning Steered by Experts. In *TACAS (2) (Lecture Notes in Computer Science, Vol. 12652)*. Springer, 326–345.

[4] Muqsit Azeem, Debraj Chakraborty, Sudeep Kanav, and Jan Kretínský. 2024. Explainable Finite-Memory Policies for Partially Observable Markov Decision Processes. *CoRR* abs/2411.13365 (2024). https://doi.org/10.48550/ARXIV.2411.13365 arXiv:2411.13365

[5] Alexander Bork, Debraj Chakraborty, Kush Grover, Jan Křetínskỳ, and Stefanie Mohr. 2024. Learning Explainable and Better Performing Representations of POMDP Strategies. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer.

[6] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. 2022. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.* 24, 4 (2022), 589–610. https://doi.org/10.1007/s10009-021-00633-z

[7] Sebastian Junges, Nils Jansen, and Sanjit A. Seshia. 2021. Enforcing Almost-Sure Reachability in POMDPs. In *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12760)*. Springer, 602–625. https://doi.org/10.1007/978-3-030-81688-9_28

[8] Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. 1995. Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings 1995*. Elsevier, 362–370.