Post-Hoc Interpretation of POMDP Policies

Geoffrey Laforest[1], Olivier Buffet[2], Alexandre Niveau[1], Bruno Zanuttini[1]

[1] Université Caen Normandie, ENSICAEN, CNRS, Normandie Univ
GREYC UMR6072, F-14000 Caen, France
[2] Université de Lorraine, CNRS, Inria
LORIA, F-54000 Nancy, France
{prenom.nom}@unicaen.fr, {prenom.nom}@loria.fr

May 16, 2025

Résumé

Les politiques pour des processus de décision markoviens partiellement observables sont des objets riches, prescrivant des actions en fonction de l'historique entier des observations et actions. Avec l'interpretabilité comme objectif, nous proposons de redécrire des représentations peu interprétables comme des fonctions définies sur les valeurs de descripteurs de l'état de croyance courant, construits de manière systématique à partir de descripteurs de l'état. Ces fonctions peuvent à leur tour être représentées par des objets intelligibles, comme des matrices de poids ou des arbres de décision. Nous étudions les problèmes de calcul afférents, et comparons empiriquement notre approche avec une redescription des politiques par des contrôleurs à états finis.

Mots-clés

POMDP, politique, interprétabilité

Abstract

Policies for partially observable Markov decision processes are rich objects, prescribing actions to take depending on the whole history of observations and actions. Towards interpretability, we propose to redescribe poorly interpretable representations as mappings defined on the value of features of the current belief state, built in a systematic manner from state features. Those mappings can in turn be represented by intelligible objects, like matrices of weights or decision trees. We investigate related computational problems and experimentally compare our approach with redescription of policies as finite-state controllers.

Keywords

POMDP, policy, interpretability

1 Introduction

We are interested in the representation of policies of actions for partially observable Markov decision processes (POMDPs), which are stochastic decision processes with partially observable states. At execution time, a policy prescribes actions to take depending on the whole history of actions and observations. Standard representations of such policies are by finite-state controllers (with states/nodes associated to actions and transitions via observations), and by sets of α-*vectors*, which are hyperplanes in the belief space, that is, the space of probability distributions over the state space (which represent possible beliefs of the agent over the current state).

Arguably, such representations lack interpretability: states in finite-state controllers are simply atoms, and α-vectors are high-dimensional vectors of real values. Hence we propose a new representation of policies for POMDPs, in which a set of *epistemic features* is built in a systematic manner from a given set of state features, and policies are represented by interpretable objects (e.g., matrices of weights, decision trees) defined on the embedding of the belief space onto these features.

In a nutshell, an epistemic feature is a propositional formula over the state features (a small disjunction or conjunction), and its value in a belief state is the probability that it is satisfied by a state drawn from it. This allows us to propose representations of policies over features like "the probability that the goal is at $(0, 0)$" which, we think, makes them more interpretable.

We define these representations and algorithms for computing them. Our approach is *post-hoc*, that is, starts from a (non-interpretable) policy and redescribes it into a more interpretable one. We finally compare our representations to those based on finite-state controllers in terms of size and computational time.

2 Related Work

Solving POMDPs exactly is intractable in general except for the smallest problems, due to the rapid growth in complexity. Point-based algorithms have been a breakthrough for scaling and solving POMDPs, by avoiding the curse of dimensionality and the curse of history. Point-Based Value Iteration (PBVI) [31] samples a small set of representative belief points and locally updates a lower bound of the optimal value function represented by α-vectors, which ensures that it is piece-wise linear and

convex. Heuristic Search Value Iteration (HSVI) [35, 36] draws on the same idea but, contrary to PBVI, it also maintains an upper bound on the optimal value function, and samples beliefs by generating trajectories while acting optimistically. One of the most efficient point-based algorithms is SARSOP [21]. It builds on HSVI, but tries to approximate the reachable belief states under an optimal policy. It also benefits from a pruning strategy based on the α-vector representation of the value function.

Another line of work leverages recurrent neural networks for their ability to encode the history [2]. They are often integrated in another architecture to make predictions, such as Deep Q-Networks [29]. Carefully designed and tuned, they can be a strong baseline for many POMDPs [30]. Recently, transformers have also been considered instead of recurrent networks, with mixed results [24]. A somewhat compact representation of a policy can be obtained by embedding it into an RKHS and truncating the embedding basis up to a certain number [25]. This results in a more compact but approximated policy.

2.1 Interpretability

There is a growing interest and a surge of work in explainable and interpretable machine learning [11]. This is motivated by ethical and legal concerns [7], and the need to understand algorithmic predictions to ensure decisions based on machine learning systems are safe and reliable [23]. In this work, we focus on interpretability, which refers to the degree to which a human can understand the model's results and the causes of its decisions. It often implies that a user can grasp how inputs are processed and mapped to the outputs in a human-comprehensible format [10]. Closely related notions often involve deriving agent behaviors that are more easily interpretable by external observers. For instance, behaviors may be considered *legible* if they facilitate guessing the agent's true objective, *explainable* if the agent's actions clearly align with some objective, or *predictable* if future actions can be easily anticipated [5]. Observer-aware MDPs (OAMDPs) provide a formal framework to address these interpretability challenges within stochastic environments [28].

Interpretability usually takes two roads [8]. Interpretability *by design* means that the models learnt or computed are taken from a language deemed interpretable *per se*. In planning, examples are representations by easy-to-understand trees [19, 37] or expressed with symbolic features [17, 12], sometimes both [6, 20]. In [17], a continuous state-action space is discretized, and the policy is represented with if-else fuzzy rules.

Now *post-hoc* interpretability, which is the setting of our work, means a model is redescribed, after learning or computation, into an interpretable language. In the setting of MDPs and RL, examples build on measuring the importance of reachable states [18, 9] or actions taken under the given policy. For instance, in [34], the importance of actions is assessed regarding their role in satisfying a predicate of interest at each time-step until the end of the episode, starting at a given horizon k. Another approach consists in targeting a simpler language [3].

In the POMDP setting, one can use symbolic features based on a logical language. For example, Meli et al. [26] use logical features to describe interesting traces of a given policy, then use these descriptions as heuristics for other solvers. Hence, though they are many similarities with our work, the focus there is on computing interpretable heuristics. In addition, in their experiments, they use features which require more than the transition model to be known.

2.2 Compression of Policies

Policies computed by search, as with SARSOP, are in practice represented by high-dimensional vectors with many nonzero coefficients. For this reason, symbolic solvers, which use a factored representation of α-vectors, like Symbolic Perseus [32], have been developed. Another representation of policies is by a tree branching on observations and prescribing an action at each node, and the natural generalization of this to an automaton, namely, a finite-state controller (FSC). In [13], the authors propose algorithms for computing such compact representations by FSCs, of policies given by oracles. Their focus is on (post-hoc) compression of policies, which can be seen as a manner of enhancing interpretability. We experimentally compare our approach with theirs in Section 6.

3 Background

Given a finite set S, we write $\mathcal{P}(S)$ for the set of all subsets of S, and $\Delta(S)$ for the probability simplex over S. Given a distribution δ, we write $\mathbf{x} \sim \delta$ to denote the fact that random variable $\mathbf{x}$ is sampled from δ, and $x \in \delta$ to mean that value x is in the support of δ.

A *Partially Observable Markov Decision Process* (POMDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, b_0, \gamma \rangle$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{O}$ finite spaces of states, actions, and observations, resp.; for all $s \in \mathcal{S}, a \in \mathcal{A}$, $T_{s,a} \in \Delta(\mathcal{S})$ is the distribution of next states when a is taken in s; for all $s \in \mathcal{S}, a \in \mathcal{A}, s' \in T_{s,a}$, $Z_{s,a,s'} \in \Delta(\mathcal{O})$ is the distribution of observations upon transition $\langle s, a, s' \rangle$ and $R_{s,a,s'} \in \mathbb{R}$ the reward received for transition $\langle s, a, s' \rangle$; $b_0 \in \Delta(\mathcal{S})$ is an initial belief state; and $\gamma \in [0, 1)$ a discount factor.

Given a POMDP, a *history* is a finite sequence of actions and observations of the form $h := a_0 o_0 \dots a_k o_k$; we write $\langle\rangle$ for the empty history, and $\mathcal{H}$ for the set of all histories. A *belief state* is a probability distribution over $\mathcal{S}$, and we write $\mathcal{B}$ for the set of all belief states ($\mathcal{B} := \Delta(\mathcal{S})$). A history h induces a belief state $\mathrm{bs}(h)$ through Bayesian update:

$$\mathrm{bs}(\langle\rangle) := b_0,$$
$$\mathrm{bs}(h \cdot ao) := s' \mapsto \eta \sum_{s \in \mathcal{S}} Z_{s,a,s'}(o) T_{s,a}(s') \,\mathrm{bs}(h)(s),$$

where η is a normalizing constant. A *non-deterministic policy* $pol : \mathcal{H} \mapsto \mathcal{A}$ maps histories to actions. A *non-deterministic belief-based policy* $p \colon B \to \mathcal{P}(\mathcal{A})$ maps a set $B \subseteq \mathcal{B}$ to sets of actions, and induces a policy

$\mathrm{pol}(p)\colon \mathcal{H} \to \mathcal{P}(\mathcal{A})$ via $\mathrm{pol}(p)(h) := p(\mathrm{bs}(h))$, with $\mathrm{bs}(h) := b$ being the belief-state induced by h.
The *optimal value* at history h and infinite horizon is denoted by $\mathrm{V}^*(h)$ and defined by $\mathrm{V}^*(h) := \max_{a\in\mathcal{A}} \mathrm{Q}^*(h,a)$, with

$$\mathrm{Q}^*(h,a) := \mathop{\mathbb{E}}_{\substack{\mathbf{s}\sim\mathrm{bs}(h)\\ \mathbf{s}'\sim T_{\mathbf{s},a}}} (R_{\mathbf{s},a,\mathbf{s}'} + \gamma \mathop{\mathbb{E}}_{\mathbf{o}\sim Z_{\mathbf{s},a,\mathbf{s}'}} \mathrm{V}^*(h\cdot a\mathbf{o})).$$

We call *optimal nondeterministic policy* of the POMDP, the function $pol^* : \mathcal{H} \to \mathcal{P}(\mathcal{A})$ defined for all histories h by $pol^*(h) := \mathrm{argmax}_{a\in\mathcal{A}} \mathrm{Q}^*(h,a)$. This function maps any history to the set of all optimal actions to take at this history and the initial belief state, given an infinite horizon.
It is well-known that a POMDP can be cast into a fully observed MDP, called the *belief MDP*, whose abstract states correspond to the above belief states. The transition function between belief states is derived using Bayesian update as above. The belief state is a sufficient statistic for optimal decision-making, so that solving the belief MDP at infinite horizon provides a policy $p^* : \Delta(\mathcal{S}) \to \mathcal{P}(\mathcal{A})$ that is optimal for the POMDP at an infinite horizon.

Example 1. As a running example, we use a POMDP containing 4 states, written $\mathsf{s}_{00}, \mathsf{s}_{01}, \mathsf{s}_{10}, \mathsf{s}_{11}$ (read intuitively as s_{xy}). There are three actions: $\mathsf{check}_=$ does not change the state but (deterministically) produces observation true (resp. false) when taken in state s_{xy} with $x = y$ (resp. $x \neq y$); $\mathsf{switch}_\mathsf{x}$ deterministically maps each s_{xy} to $\mathsf{s}_{(1-x)y}$, yielding observation void; noop does not change the state and yields observation void. Finally, b_0 is the uniform distribution $\mathsf{b}^{\mathsf{unif}}$ over $\mathcal{S}$, and $R_{\cdot,\cdot,s'}$ is 1 for $s' \in \{\mathsf{s}_{00}, \mathsf{s}_{11}\}$, 0 otherwise.
Then $\mathsf{h}^= := \langle \mathsf{check}_=, \mathsf{false}, \mathsf{switch}_\mathsf{x}, \mathsf{void}\rangle$ is a history, inducing the belief state $\mathsf{b}^= := \mathrm{bs}(\mathsf{h}^=)$, which assigns probability $\frac{1}{2}$ to $\mathsf{s}_{00}, \mathsf{s}_{11}$. Let moreover $\mathsf{b}^{\neq}$ be the belief state assigning $\frac{1}{2}$ to $\mathsf{s}_{01}, \mathsf{s}_{10}$, and p be defined by $\mathsf{p}(\mathsf{b}^{\mathsf{unif}}) := \{\mathsf{check}_=, \mathsf{noop}\}$, $\mathsf{p}(\mathsf{b}^{\neq}) := \{\mathsf{switch}_\mathsf{x}\}$, and $\mathsf{p}(\mathsf{b}^=) = \{\mathsf{check}_=, \mathsf{noop}\}$.[1] Then p is a (non optimal) belief-based policy, inducing, for example, $\mathrm{pol}(\mathsf{p})(\mathsf{h}^=) := \mathsf{p}(\mathsf{b}^=) = \{\mathsf{check}_=, \mathsf{noop}\}$.
It is easy to see that the optimal policy is given by $\mathsf{p}^*(\mathsf{b}^{\mathsf{unif}}) := \{\mathsf{check}_=\}$, $\mathsf{p}^*(\mathsf{b}^{\neq}) := \{\mathsf{switch}_\mathsf{x}\}$, and $\mathsf{p}^*(\mathsf{b}^=) = \{\mathsf{check}_=, \mathsf{noop}\}$ □

Policies can be naturally represented as trees, a node being a history or a belief state, branching at each time-step on actions that can be taken at this node and subsequent possible observations. The problem of this representation is that it grows exponentially with the horizon.
Manipulating graphs rather than trees yields *finite state controllers* (FSCs) [27, 14, 13]. A non-deterministic FSC consists of a set of nodes N among which a set N_0 of distinguished initial nodes, an action mapping $act\colon N \to \mathcal{A}$ assigning an action to each node, and a partial transition function $\delta\colon N \times \mathcal{O} \to \mathcal{P}(N)$.[2]

[1] The definition over other belief states is irrelevant to our examples.
[2] The transition function is partial because not all observations may be received at each node during execution.

Intuitively, an FSC defines a nondeterministic policy pol as follows. The set of nodes *reachable* by the FSC through a history is defined by $\mathrm{N}(\langle\rangle) := N_0$ and $\mathrm{N}(h\cdot ao) := \bigcup_{n\in\mathrm{N}(h), act(n)=a} \delta(a,o)$. Now for any history h, $pol(h)$ is defined to be $\bigcup_{n\in\mathrm{N}(h)} act(n)$. In words, executing the FSC means taking any action associated with one of the current nodes, and progressing via a and o means restricting to those current nodes where a could indeed be taken, then transiting to their successors via o.
An ϵ-optimal (infinite-horizon) belief-based policy can also be obtained by approximating the optimal value function for a long horizon t. Such approximation is typically expressed as a set of sets Γ_a of α-*vectors*, one set per action a, with $V_{\{\Gamma_a\}}(b) := \max_{a\in\mathcal{A},\alpha\in\Gamma_a} b\cdot\alpha$ for all belief states b; the function $V_{\{\Gamma_a\}}$ has the nice property of being piecewise linear and convex. The induced nondeterministic policy is defined on belief states, by $p(b) := \mathrm{argmax}_{a\in\mathcal{A}}(\max_{\alpha\in\Gamma_a} b\cdot\alpha)$. However, as the dimension of the vectors grows linearly with the number of states and as the sets of alpha vectors Γ_a can become very large to represent the policy up to the desired precision, in practice this representation is neither more compact nor more readable than tree policies.
Optimal value functions (and policies) can also be obtained by relying on factored POMDPs [15], where the state and observation spaces are modeled using multiple random variables, and the transition and observation functions are modeled as 2-layer dynamic Bayesian networks. These functions, as well as the reward function, are often expressed as *algebraic decision diagrams* (ADDs) [1, 16, 32]. Dedicated operators allow computing the optimal value function also as an ADD, as in Symbolic Perseus [32], and typically lead to compact representations. We however leave this aspect for future work.

4 Epistemic Representations

Our proposal is based on the idea of describing policies over what we call *epistemic features*, which are features defined on the belief space. Formally, given a POMDP, we define an *epistemic feature* to be a function $\varphi : \mathcal{B} \to \mathbb{R}$. Epistemic features may take many different forms, for instance, entropy, but we choose to focus on epistemic features built in a systematic manner from a set of *state features*, that is, Boolean features of the form $f : \mathcal{S} \to \mathbb{B}$.[3] The reason is that such state features are typically readily available (from a factored representation of the POMDP, for instance) or easily defined by an expert.

Example 2 (continued)**.** The state space of our running example can be described with two state features, x and y, with $\mathsf{x}(\mathsf{s}_{xy}) := x$ and $\mathsf{y}(\mathsf{s}_{xy}) := y$; for instance, $\mathsf{x}(\mathsf{s}_{00}) = \mathsf{x}(\mathsf{s}_{01}) = 0$. □

To define epistemic features from state features, we use simple logical combinations. A (propositional) *clause* over variables F is a disjunction c of the form $(f_1 \vee \cdots \vee f_p \vee \neg f_{p+1} \vee \cdots \vee \neg f_w)$, where all f_i's are elements of F, all

[3] By $\mathbb{B}$ we mean $\{0,1\}$, with 0 standing for "false" and 1 for "true".

different from each other; dually, a *term* is a conjunction t of the form $(f_1 \wedge \cdots \wedge f_p \wedge \neg f_{p+1} \wedge \cdots \wedge \neg f_w)$. We call w the *width* of the clause or term. A clause or term of the above form is said to be *positive* if $p = w$ holds, that is, no negated literal occurs in it. Given that $\vee$ and $\wedge$ are commutative and idempotent, we consider $(f \vee f')$ and $(f' \vee f)$, for instance, to be the same clause (and similarly for terms).

Definition 1 (epistemic features of width w)**.** *Let $F \subseteq \mathbb{B}^{\mathcal{S}}$ be a set of Boolean state features, and let w be a positive integer. An* epistemic clause of width w *over F is an atom κ of the form $\mathrm{B}\, c$, where c is a clause of width w over F, and an* epistemic term of width w *is an atom τ of the form $\mathrm{B}\, t$, with t a term of width w over F. An* epistemic feature (of width w) *is an epistemic clause or term (of width w).*[4]

If c (resp. t) is positive, then $\mathrm{B}\, c$ (resp. $\mathrm{B}\, t$) is said to be positive as well. The set of all epistemic clauses (resp. terms, positive clauses, positive terms) of width w over F is written $\Phi_F^{\vee,w}$ (resp. $\Phi_F^{\wedge,w}$, $\Phi_F^{\vee,+,w}$, $\Phi_F^{\wedge,+,w}$).
Given a set F of state features over a state space $\mathcal{S}$, a state $s \in \mathcal{S}$ induces an assignment s_F to the features in F: $\forall f \in F : s_F(f) = f(s)$. Given a propositional formula g over F, we define $\mathrm{sat}(s, g)$ to be 1 (resp. 0) if s_F satisfies g under the standard semantics of propostional logic.

Definition 2 (semantics of epistemic features)**.** *The function induced by an epistemic clause $\kappa := \mathrm{B}\, c$ (resp. term $\tau := \mathrm{B}\, t$) is defined for all belief states b by $\kappa(b) := \mathbb{E}_{s\sim b}\, \mathrm{sat}(s, c)$ (resp. $\tau(b) := \mathbb{E}_{s\sim b}\, \mathrm{sat}(s, t)$).*

$\mathrm{B}\, c$ can literally be read as "the probability that c is true".

Example 3 (continued)**.** The epistemic clauses of width $w = 1$ are $\mathrm{B}\,\mathsf{x}$, $\mathrm{B}\,\mathsf{y}$, $\mathrm{B}\,\neg\mathsf{x}$, $\mathrm{B}\,\neg\mathsf{y}$; the first two are positive, the last two are not. For $w = 2$, the epistemic clauses are $\mathrm{B}(\mathsf{x} \vee \mathsf{y})$, $\mathrm{B}(\neg\mathsf{x} \vee \neg\mathsf{y})$, $\mathrm{B}(\mathsf{x} \vee \neg\mathsf{y})$, and $\mathrm{B}(\mathsf{y} \vee \neg\mathsf{x})$; only the first one is positive.
Consider b defined by $b(\mathsf{s}_{00}) := \frac{1}{2}$, $b(\mathsf{s}_{01}) := \frac{1}{4}$, and $b(\mathsf{s}_{10}) := \frac{1}{4}$. For $\kappa := \mathrm{B}(\mathsf{x} \vee \mathsf{y})$, we have $\kappa(b) = \frac{1}{2} \times 0 + \frac{1}{4} \times 1 + \frac{1}{4} \times 1 = \frac{1}{2}$, and for $\kappa := \mathrm{B}\,\neg\mathsf{y}$, we have $\kappa(b) = \frac{1}{2} \times 1 + \frac{1}{4} \times 0 + \frac{1}{4} \times 1 = \frac{3}{4}$. □

4.1 Projections

Given a set of epistemic features, our aim is to represent policies as mappings defined over *projections* of belief states onto these features. Importantly, we only consider policies defined over (or restricted to) a finite set of belief states (e.g., all those reachable by the policy up to a certain horizon), and leave the more general case to future work.

Definition 3 (projection)**.** *Let Φ be an ordered tuple of N epistemic features. The* projection *of belief state b (onto Φ) is defined to be the vector $\Phi(b) := (\varphi(b) \mid \varphi \in \Phi) \in \mathbb{R}^N$.*

In general, the projection of a belief state onto a set of epistemic features is lossy. However, as our experiments show (Section 6), features of small width will in general be enough for unambiguously representing optimal policies over a finite subset of the belief space, for natural sets of state features on natural problems.

Definition 4 (projectable)**.** *Let B be a finite set of belief states, $p\colon B \to \mathcal{P}(\mathcal{A})$ be a nondeterministic policy defined on B, and Φ be a set of epistemic features. Then p is said to be* projectable *onto Φ if there is a function $\pi\colon \{\Phi(b) \mid b \in B\} \to \mathcal{P}(\mathcal{A})$ satisfying $\forall b \in B : \emptyset \neq \pi(\Phi(b)) \subseteq p(b)$.*

In words, a policy is projectable if it can be faithfully represented by a function of the epistemic projection of the belief states, possibly at the price of breaking ties between actions arbitrarily. Note that if we are representing the optimal policy p^*, breaking ties between actions does not hinder optimality, since all actions are optimal at all belief states/histories.
Projectability can be straightforwardly decided in time polynomial in the numbers of belief states, states, epistemic features, and actions.

Example 4 (continued)**.** For $w = 1$, we have $(\mathrm{B}\,\mathsf{x})(b) = (\mathrm{B}\,\mathsf{y})(b) = (\mathrm{B}\,\neg\mathsf{x})(b) = (\mathrm{B}\,\neg\mathsf{y})(b) = \frac{1}{2}$ for $b \in \{\mathsf{b}^{\mathsf{unif}}, \mathsf{b}^{=}, \mathsf{b}^{\neq}\}$, hence

$$\Phi_F^{\vee,1}(\mathsf{b}^{\mathsf{unif}}) = \Phi_F^{\vee,1}(\mathsf{b}^{=}) = \Phi_F^{\vee,1}(\mathsf{b}^{\neq}) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}).$$

Hence p cannot be projected onto $\Phi_F^{\vee,1}$, since it prescribes different sets of actions for $\mathsf{b}^{\mathsf{unif}}$ and $\mathsf{b}^{\neq}$, which have the same projection onto $\Phi_F^{\vee,1}$.
On the other hand, p is projectable onto $\Phi_F^{\wedge,+,1} \cup \Phi_F^{\wedge,+,2} = \{\mathrm{B}\,\mathsf{x}, \mathrm{B}\,\mathsf{y}, \mathrm{B}(\mathsf{x} \wedge \mathsf{y})\}$, since the value v_3 of the third feature is already different on all three belief states (it is $\frac{1}{4}$ on $\mathsf{b}^{\mathsf{unif}}$, $\frac{1}{2}$ on $\mathsf{b}^{=}$, and 0 on $\mathsf{b}^{\neq}$). Hence we can define (for instance) $\pi((v_1, v_2, v_3)) = \{\mathsf{switch}_{\mathsf{x}}\}$ for $v_3 < \frac{1}{8}$, and $\pi((v_1, v_2, v_3)) = \{\mathsf{check}_{=}, \mathsf{noop}\}$ for $\frac{1}{8} \leq v_3$. Another projection (which breaks ties) is given by $\pi'((v_1, v_2, v_3)) = \{\mathsf{switch}_{\mathsf{x}}\}$ for $v_3 < \frac{1}{8}$, $\pi'((v_1, v_2, v_3)) = \{\mathsf{check}_{=}\}$ for $\frac{1}{8} \leq v_3 < \frac{3}{8}$, and $\pi'((v_1, v_2, v_3)) = \{\mathsf{noop}\}$ for $\frac{3}{8} < v_3$.
Observe that these definitions use only the value of the third feature, and hence p is also projectable onto $\Phi_F^{\wedge,+,2}$ alone. □

Obviously, projectability depends on the set of epistemic features. The following results explore this dependency.

Proposition 1. *Let B be a finite set of belief states, $p\colon B \to \mathcal{P}(\mathcal{A})$ be a nondeterministic policy defined on B, and F be a set of state features. Let $w < |F|$ be a nonnegative integer. If p is projectable onto $\Phi_F^{\vee,w}$ (resp. $\Phi_F^{\wedge,w}$), then it is projectable onto $\Phi_F^{\vee,w+1}$ (resp. $\Phi_F^{\wedge,w+1}$).*

Proof. This comes from the fact that the value of an epistemic clause (resp. term) of width w can be retrieved from the values of epistemic clauses (resp. terms) of width w. Indeed, for a propositional term t and a belief state b, we have $(\mathrm{B}\, t)(b) = (\mathrm{B}(t \wedge f))(b) + (\mathrm{B}(t \wedge \neg f))(b)$, for an arbitrary state feature $f \in F$. For a propositional clause c, writing t for a term equivalent to the negation of c, we have $(\mathrm{B}\, c)(b) = 1 - (\mathrm{B}\, t)(b) = 1 - \big((\mathrm{B}(t \wedge f))(b) + (\mathrm{B}(t \wedge$

[4] We use Latin letters for state features and formulas, and Greek letters for epistemic features and formulas.

$\neg f))(b)\big)$ for an arbitrary state feature $f \in F$, and hence $(\mathrm{B}\, c)(b) = -1 + (\mathrm{B}(c \vee \neg f))(b) + (\mathrm{B}(c \vee f))(b)$. □

Proposition 2. *Let F be an ordered set of n state features, and assume that the function $F : \mathcal{S} \to \mathbb{R}^n$ defined by $F(s) := (f(s) \mid f \in F)$ is injective. Then any policy over a finite set B of belief states is projectable onto $\Phi_F^{\vee,n}$ and onto $\Phi_F^{\wedge,n}$.*

Proof. For epistemic terms, this follows directly from the fact that the function $b \mapsto \Phi_F^{\wedge,n}(b)$ is also injective. Indeed, the probability of each state s in b can be retrieved from the projection as the value of the unique epistemic term of width n satisfied by s. The reasoning is similar for clauses: the probability of s can be retrieved as $1 - (\mathrm{B}\, c)(b)$, with c the unique epistemic clause of width n *not* satisfied by s. □

4.2 Representations

Given a finite set of belief states B, once a set of epistemic features Φ is fixed, over which a policy p^* is projectable, our aim is to compute an interpretable representation of p^*, using the epistemic features in Φ.

We view this problem as the problem of computing a classifier for a set of labelled examples, where for each belief state $b \in B$, the vector $\Phi(b)$ is an example, and the set of actions $p^*(b)$ is a set of possible labels for this vector. To keep the approach simple, we use only single-label classifiers, with the semantics that predicting any single action in p^* is correct.

Example 5 (continued)**.** Consider again $B = \{\mathsf{b}^{\mathsf{unif}}, \mathsf{b}^{=}, \mathsf{b}^{\neq}\}$, $\Phi = \Phi_F^{\wedge,+,1} \cup \Phi_F^{\wedge,+,2}$ ordered as $(\mathrm{B}\,\mathsf{x}, \mathrm{B}\,\mathsf{y}, \mathrm{B}\,\mathsf{x} \wedge \mathsf{y})$, and p. Then the examples are $(\frac{1}{2}, \frac{1}{2}, \frac{1}{4})$, $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, and $(\frac{1}{2}, \frac{1}{2}, 0)$, with respective sets of labels $\{\mathsf{check}_{=}, \mathsf{noop}\}$, $\{\mathsf{check}_{=}, \mathsf{noop}\}$, and $\{\mathsf{switch}_{\mathsf{x}}\}$. □

Observe that the set of examples does not cover the whole example space, so that the learned classifier has freedom to generalize them. We however require it to be correct on the given set of examples, that is, to provide an exact representation (modulo ties between optimal actions) of the given policy over the given set of belief states.

In principle, any classifier can be used. However, with our goal of building interpretable representations in mind, we will consider two classes of classifiers.

First, we call *linear representation* of p^* (over B and Φ) a matrix of real numbers $\{\theta_\varphi^a \mid \varphi \in \Phi, a \in \mathcal{A}\}$; the number θ_φ^a is called the *weight* of feature φ for action a. The policy π induced by such a representation is defined for all belief states $b \in \mathcal{B}$ by $\pi(b) := \mathrm{argmax}_{a \in \mathcal{A}} \sum_{\varphi \in \Phi} \theta_\varphi^a \cdot \varphi(b)$. Hence such representations can be seen as akin to α-vectors, but with only one vector per action, and compactly represented belief states (α-vectors can be seen as operating on projections with one feature per state).

The reason why we are interested in this representation is that it provides easy-to-interpret weights: the higher θ_φ^a, the more feature φ "promotes" action a, in the sense that the higher the value of φ on b, the more likely a is prescribed by the policy (all other things, namely, the value of other epistemic features on b, being equal).

Second, we call *decision tree representation* of p^* a decision tree branching on the values of epistemic features, and with actions at leaves. The policy π induced by such a representation is defined to map each belief state $b \in \mathcal{B}$, to the action at the leaf of the (unique) branch activated by b in the tree.

It is important to note that depending on the family of classifiers, an exact representation of a policy needs not always exist. For instance, a linear representation of a policy does not exist in general (though our experiments show that it often does), even if the policy is projectable onto the set of epistemic features. Contrastingly, if we do not bound the depth of trees nor the number of occurrences of an epistemic feature along a branch, a decision tree representation always exists, provided the policy is projectable.

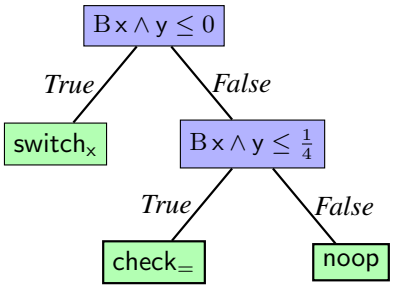

Figure 1: Decision tree representing an optimal policy in our POMDP example, branching on epistemic features.

Example 6 (continued)**.** The matrix of weights defined by $\theta^{\mathsf{check}_=}_{\mathrm{B}\,\mathsf{x}\wedge\mathsf{y}} = 1$ and $\theta^{\mathsf{switch}_\mathsf{x}}_{\mathrm{B}\,\mathsf{x}} = 0.1$ (all other weights being 0) is a linear representation of p over B and $\Phi_F^{\wedge,+,1} \cup \Phi_F^{\wedge,+,2}$. The policy π induced by this matrix indeed prescribes $\{\mathsf{switch}_\mathsf{x}\}$ at $\mathsf{b}^{\neq}$ (with value 0.05 against 0 for noop and $\mathsf{check}_=$) and $\{\mathsf{check}_=\}$ at $\mathsf{b}^{\mathsf{unif}}$ and $\mathsf{b}^{=}$ (with value 0.25 and 0.5, respectively, against 0.05 for $\mathsf{switch}_\mathsf{x}$ and 0 for noop). This can be interpreted as follows: the more the agent thinks it possible that the current state satisfies x and y, the more it should take action $\mathsf{check}_=$. On the other hand, the more it thinks it possible that the current state satisfies x, the more it should take action $\mathsf{switch}_\mathsf{x}$; however, the weight associated to the latter rule is much smaller than the one associated to the former, so that knowing $\mathsf{x} \wedge \mathsf{y}$ promotes more $\mathsf{check}_=$ than knowing x promotes $\mathsf{switch}_\mathsf{x}$.

On the other hand, there is no linear representation of p over B and $\Phi_F^{\wedge,+,2} = \mathrm{B}\,(\mathsf{x} \wedge \mathsf{y})$; indeed, such representation would necessarily give a 0 weight to all actions at $\mathsf{b}^{\neq}$ and, hence, could only represent a policy π with $\pi(\mathsf{b}^{\neq}) = \{\mathsf{check}_=, \mathsf{switch}_\mathsf{x}, \mathsf{noop}\}$.

Finally, Figure 1 depicts a decision tree representation of p (breaking the ties in favour of $\mathsf{check}_=$ at $\mathsf{b}^{\mathsf{unif}}$ and of noop at $\mathsf{b}^{=}$). □

Finally, executing a representation of a policy over Φ proceeds as follows. Given a POMDP, the agent first sets

$t := 0$ and computes $v_0 := \Phi(b_0)$. Then at each timestep t, it executes $a_t := \pi(v_t)$ and, upon receiving observation o_t, computes $v_{t+1} := \Phi(b_t \cdot a_t o_t)$.

Hence, like with α-vector representations, execution requires the POMDP model (transition and observation function) to be known. However, observe that the full belief state needs not be computed in general. Computing $\Phi(b_t \cdot a_t o_t)$ is a *belief tracking* task, which can be performed implicitly, for instance using operations over dynamic Bayesian networks representing the POMDP action and observation models [4]. Still, this is of course more costly than, for instance, following an arc in a finite-state controller. Hence there is a tradeoff between interpretability and efficiency of execution. Note the link with the work on knowledge-based programs [22, 38], which are representations of policies as programs branching on epistemic features (similar to ours) can yield very compact representations. Our work can indeed be seen as one using specific forms of KBPs as a target language for redescribing policies.

5 Computing Representations

We now present algorithms for computing epistemic representations starting from (non epistemic) policies. We start from a given policy p defined on the belief space, and restrict it to a finite set B of belief states (e.g., those reachable in at most t steps from b_0). In our experiments, we use the (near-optimal) policy computed from the POMDP by the off-the-shelf solver SARSOP, which comes with a representation by α-vectors, and restrict it to a large horizon.

We also fix a set of epistemic features. For this, we start from a given set F of state features (it is straightforward to define a meaningful one for practical problems). We then fix a width w, and choose whether to use clauses and/or terms, and whether to include negative features. These choices induce a set of epistemic features (for instance, $\Phi = \Phi_F^{\wedge,+,2}$).

We then proceed in two steps. First, we check whether p is projectable onto Φ (for B). As already mentioned, this can be done by a simple algorithm which goes through each belief state $b \in B$ in turn, and computes its projection $v := \Phi(b)$. If v is not mapped to a set of actions yet, then it gets mapped to $p(b)$; otherwise, if v is already mapped to a set of actions A, then it gets mapped to $A \cap p(b)$. Finally, if at any point a projection gets mapped to the empty set of actions, then the instance is not projectable onto Φ. In that case, we may, for instance, increase the width w, or include negative features if they were excluded.

Second, if the instance is projectable, then as a by-product of the first phase, we get a set of ordered pairs $\langle v_j, A_j \rangle_{j=1,\ldots,m}$ that we are going to use as examples for a classifier.

5.1 Decision Tree Representations

For computing a representation as a decision tree, we view the set $\langle v_j, A_j \rangle_{j=1,\ldots,m}$ as a training set for a learning algorithm. For this, for each $j \in \{1, \ldots, m\}$, we first sample an action a uniformly at random from A, so as to build a set of ordered pairs $\langle v_j, a_j \rangle_{j=1,\ldots,m}$, mapping a unique action to each vector. Such sampling is imposed by the fact that we focus on single-label algorithms, and of course, which action is sampled for each example impacts the final results (in terms of size, for instance).[5]

It is then a standard problem to learn a decision tree from such training set [33]. So as to get an exact representation of the policy, we impose no maximal depth on the learnt decision tree.

Linear constraints: (j ranges over $1, \ldots, m$)

1. $\boldsymbol{m}_j \geq \sum_{\phi \in \Phi} \phi(v_j) \cdot \boldsymbol{\theta}_\phi^a + \boldsymbol{\eta}$ $\quad (\forall j, \forall a \in \mathcal{A} \setminus A_j)$
2. $\boldsymbol{m}_j \geq \sum_{\phi \in \Phi} \phi(v_j) \cdot \boldsymbol{\theta}_\phi^{a^*}$ $\quad (\forall j, \forall a^* \in A_j)$
3. $M \cdot \boldsymbol{k}_{j,a^*} + \sum_{\phi \in \Phi} \phi(v_j) \cdot \boldsymbol{\theta}_\phi^{a^*} \geq \boldsymbol{m}_j$ $\quad (\forall j, \forall a^* \in A_j)$
4. $\sum_{a^* \in A_j} \boldsymbol{k}_{j,a^*} = |A_j| - 1$ $\quad (\forall j)$
5. $M \geq \boldsymbol{\eta} \geq 0$

Integrality constraints: $\boldsymbol{k}_{j,a^*} \in \{0, 1\}$ $\quad (\forall j, \forall a^* \in A_j)$

Objective: maximize $\boldsymbol{\eta}$

Figure 2: MILP for computing a linear representation.

5.2 Linear Representations

Now for computing a linear representation, we propose a formulation as a mixed-integer linear program (MILP; indeed a 0/1 linear program). This program takes advantage of the nondeterminism of the given policy, by choosing one or several actions in each set A_j so that the resulting policy is linear, if possible.

The MILP is given on Figure 2. Variables $\boldsymbol{\theta}_\varphi^a$ represent the weight θ_φ^a of feature φ for action a; variables $\boldsymbol{m}_j$ represent the highest value of an action of A_j in v_j (i.e., $\max_{a^*}(\sum \varphi(v_j) \cdot \theta_\varphi^{a^*})$); and $\boldsymbol{\eta}$ is a nonnegative margin variable (to be maximized).

Constraint 1 ensures that, for all j, the highest value $\boldsymbol{m}_j$ is greater than the value of all actions which are *not* prescribed by the policy. Constraints 2–4 together ensure that $\boldsymbol{m}_j$ receives the greatest value of all actions which *are* prescribed; for this, we use the standard trick with a "big M" constant M and 0/1 selector variables $\boldsymbol{k}_{j,a^*}$; since, as it turns out, the whole program is insensitive to multiplying by a constant, M can be chosen to be any positive value. Finally, Constraint 5 ensures that $\boldsymbol{\eta}$ is upper-bounded.

It is easy to see that this program is correct, in the following sense.

Proposition 3. *Let $\langle v_j, A_j \rangle_j$ be a set of instances, and M be any positive value. Then the optimal value $\boldsymbol{\eta}$ of the MILP of Figure 2 is strictly positive if and only if $\langle v_j, A_j \rangle_j$ has a linear representation; moreover, in this case, the values of variables $\boldsymbol{\theta}_\varphi^a$ define such a linear representation.*

[5] We plan to investigate multi-label algorithms in the future.

Solving a MILP is not known to be feasible in polynomial time. As a matter of fact, we conjecture that deciding whether a policy (given by an oracle) has a linear representation over given sets of belief states and epistemic features is an NP-complete problem.

6 Experiments

We now present experiments run on several standard problems from the POMDP literature. Our goal is to investigate to what extent our approach yields succinct and "interpretable" representations in reasonable time; to compare them with representations by FSCs; and to investigate the impact of the epistemic features chosen. In addition, we aim to investigate what epistemic width is enough for natural benchmarks from the POMDP literature.

6.1 Experimental Protocol

Each experiment consists first in choosing a POMDP and computing a near-optimal policy for it, using SARSOP.[6] Then several methods for redescribing this policy (output by SARSOP in an α-vector representation) are investigated: linear, decision tree, and FSC representations. For each method, we record the time to compute the redescription (without including the time for solving the POMDP) and its size: we define the size of a linear representation to be its number of nonzero weights, and that of a decision tree or an FSC to be its number of nodes.

Apart from SARSOP, all methods are run with our own implementation in Python, which itself uses `scipy` for solving MILPs and `sklearn` for computing decision trees. The experiments were run on servers with Intel Xeon processors, between 2.0 and 2.8 GHz and between 128 and 512 GB of RAM. For a given benchmark, they were all run on the same server, with one core dedicated to each computation. All computations were given a timeout of 1 h CPU time; SARSOP was run with target precision 10^{-3}, with $\gamma = 0.9$. For all methods, reachable belief states were gathered up to an horizon of 100.[7] For epistemic methods, we consider the following sets of epistemic features: $\Phi_i := \bigcup_{j=1}^{i}(\Phi_F^{\vee,j} \cup \Phi_F^{\wedge,j})$ and $\Phi_i^+ := \bigcup_{j=1}^{i}(\Phi_F^{\vee,+,j} \cup \Phi_F^{\wedge,+,j})$, for $i = 1, 2, 3$. So, the richest set of epistemic features includes all epistemic clauses and terms of width 1, 2, or 3, and the poorest includes only features of the form $\mathrm{B}\, f$ (for all $f \in F$).

For computing compact FSC representations, we generalized the algorithms proposed in [13]. Indeed, their approach, like ours, starts from a policy given by an oracle and from a finite set of belief states, but it requires the policy to be deterministic, and all observations to occur with a nonzero probability after any action is taken in any belief state. For lack of space, we do not give the details here, but simply note that it is a rather direct generalization of algorithm "policy2fsc" in [13]. All in all, it comes in two variants: starting from a policy pol, it computes an FSC representing a policy pol' such that, for all histories h, $\emptyset \neq pol'(h) \subseteq pol(h)$ (non-strict variant), or $pol'(h) = pol(h)$ (strict variant) holds.

[6] https://github.com/AdaCompNUS/sarsop

[7] Here it was enough for always fully representing the policy

6.2 Benchmarks

We tested our approach on four different families of POMDPs, with different features (in particular, reliable or noisy observations, fixed or combinatorial set of actions).

Wumpus. An agent must find a treasure as fast as possible on a 2D grid, while avoiding the monsters (Wumpuses). The agent always knows its own location and has deterministic actions for moving by one square in the four cardinal directions. No location can be occupied by a treasure and a Wumpus. The treasures and Wumpuses do not move and are at locations initially unknown, but the agent can "smell", which reveals whether there is a Wumpus on one of the 4 neighboring cells. Therefore, the agent may infer the locations of the Wumpuses by moving around and smelling, and eventually reach a treasure.

A Wumpus POMDP is further specified by the size of the grid, the number of Wumpuses and treasures (both known to the agent), and whether there might be a Wumpus around the agent in the initial state ("unsafe" instances; the initial belief state is otherwise uniform).

Rock-Sample. A rover navigates a 2D grid containing rocks (at known locations), each either good or bad, with equal probability. The rover can sample a rock at its current position, with a positive (resp. negative) reward if it was good (resp. bad); then the rock becomes bad. The rover can also check the status of a rock, with sensor accuracy decreasing exponentially with its distance to the rock. Reaching an exit (at one of known locations) yields a positive reward. Finally, the robot can move deterministically by one square in each cardinal direction, and always knows its position.

A Rock-Sample POMDP is further specified by the size of the grid and the positions of the rocks and exits. An interesting feature of this problem is noisy observations, which imply a large number of belief states can be reached even on small grids.

Mastermind. Our third benchmark is Mastermind, the famous code-breaking game. A Mastermind POMDP is specified by the number of positions, colors, possible repetitions in the secret code, and the number of attempts allowed. An interesting feature of this benchmark is the combinatorial set of actions, which grows with the size of the instance.

Minesweeper. Finally, we used the famous Minesweeper game. A Minesweeper POMDP is specified by the size of the grid and the number of mines.

6.3 Results

We start with the size of the representations. Figure 3 presents the results for each family of benchmarks and four representative approaches: a linear representation over Φ_1 ("epistemic-1-1-linear-withneg"), a decision tree representation over Φ_3^+ ("epistemic-1-2-3-1-2-3-tree-noneg"), and an FSC computed with the strict ("fsc-

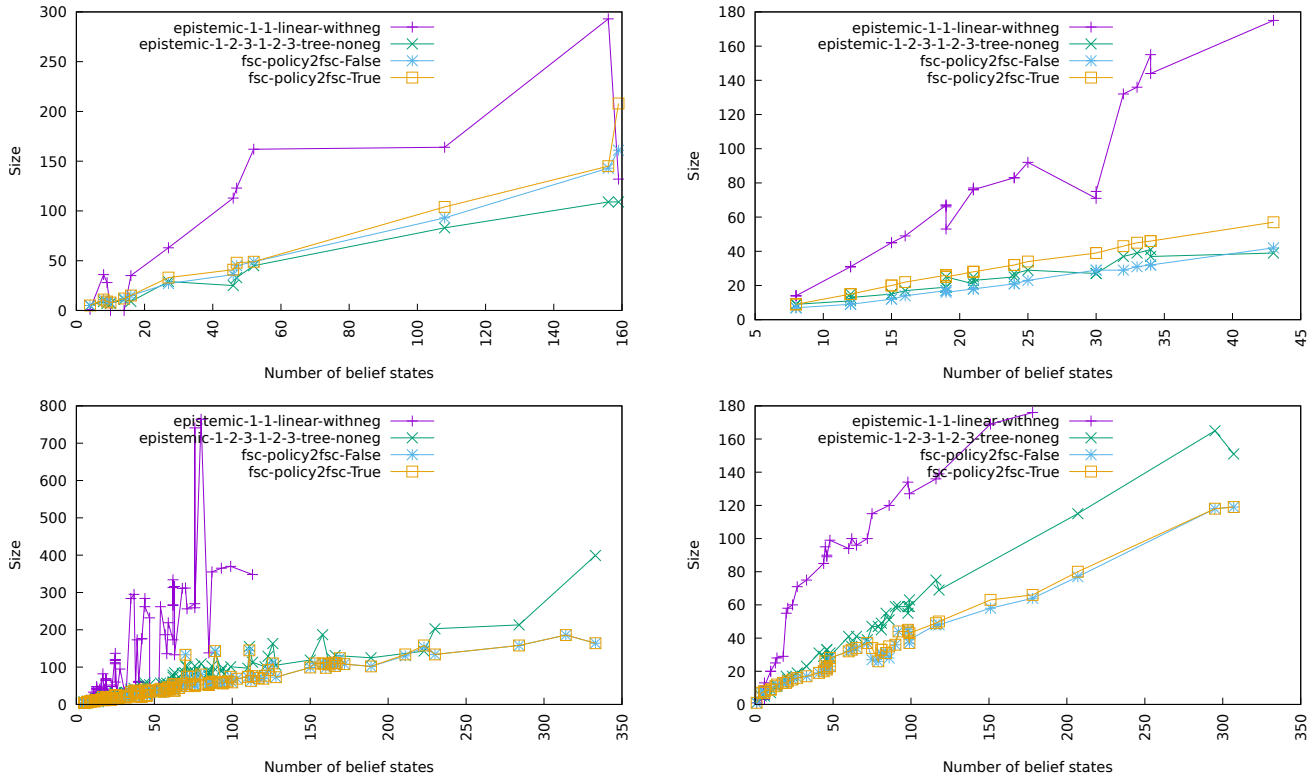

Figure 3: Size of representations for several methods. From top-left to bottom-right: Minesweeper, Rocksample, Mastermind, Wumpus

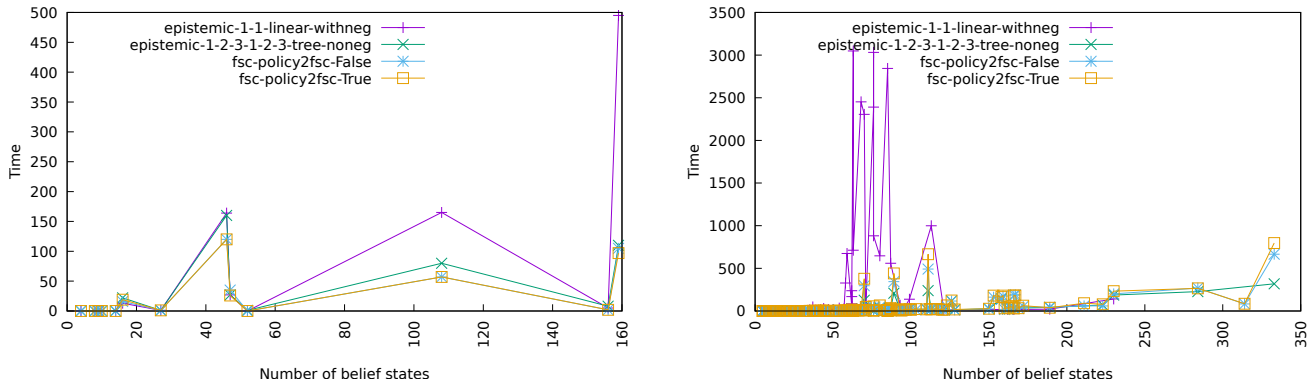

Figure 4: Time (seconds) for computing representations. Left: Minesweeper. Right: Mastermind

policy2fsc-True") or non-strict ("fsc-policy2fsc-False") variant. The x-axis is ordered by the number of belief states in the considered set B.

We observe that the representations computed by all approaches grow linearly in size with the number of belief states. Though the size is not defined in the same manner for all approaches, FSCs seem to yield the most compact models, but decision trees with positive features of size up to 3 seem to be on par, and even better on Minesweeper.

We also investigated the impact of the width of epistemic features (not displayed for lack of space). As the width grows, so does the number of features, and the size of the linear representations grows substantially as well. Moreover, eventually the MILP becomes too large and computation fails, while decision trees and FSCs can still be computed in reasonable time. Moreover, the decision tree representations tend to become smaller with larger width. This is consistent with the fact that they offer a greater expressive power (Proposition 1) and hence, allow for more decisive splits to be found. Finally, whether negative features are included or not does not seem to have much impact on the size (while restricting to positive features obviously makes the representations easier to understand).

As concerns computational time, Figure 4 presents results for two representative families of POMDPs. For both linear and decision tree models, the time needed to compute our

	Width = 1				Width = 1-2				Width = 1-2-3			
	Noneg		Withneg		Noneg		Withneg		Noneg		Withneg	
	np	nr	np	nr	np	nr	np	nr	np	nr	np	nr
M	0.72%	0.72%	0.72%	0.72%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
W	0.0%	2.31%	0.0%	2.31%	0.0%	0.53%	0.0%	0.53%	0.0%	0.53%	0.0%	0.0%

Table 1: Projectability and representability across epistemic settings for Mastermind and Wumpus

epistemic representations is in general of the same order of magnitude as that required for FSCs, though the small size of the problems do not allow to infer a definite trend.
Again, for lack of space, we do not depict results about our methods with different epistemic widths. However, for all families of POMDPs, the computational time required for our representations increases with the epistemic width and thus the number of features. This increase is much steeper for linear representations than for decision trees, as already evoked when analyzing the size. As concerns decision trees, even when the set of epistemic features grows, the trees select only a small subset of them, each used only once. This suggests that while the tree has access to a richer feature space, it does not necessarily require more splits to reach an effective decision boundary. In contrast, when fewer features are available but are used or reused more frequently, the tree requires additional splits to refine its decision boundaries. This, in turn, increases the number of computational rounds needed to build a classifier that fits the data perfectly.

Finally, we have analyzed the proportion of POMDPs that are either not projectable (np) or projectable but not linearly representable (nr). As Table 1 shows, there are very few instances of np/nr, and they appear in only two benchmarks. This indicates that, in most cases, a low epistemic width suffices to project a policy. Moreover, a representation can usually be computed.

7 Conclusion

We proposed an approach for redescribing POMDP policies onto sets of epistemic features, with the goal of improving interpretability. Our experiments show that our approach has the potential to yield representations on par with finite-state controllers in terms of size and computation time.
In the near future, we intend to investigate how it scales to larger problems, in particular by using factored representations of POMDPs. We also plan to investigate how to enrich FSCs with epistemic features, trying to get the best of both. Finally, our long-term objective is to perform reinforcement learning directly on the epistemic descriptions.

Acknowledgements

This work has been funded by the French National Agency for Reasearch (ANR) through grant ANR-22-CE23-0029.

References

[1] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *Formal Methods Syst. Des.*, 10(2/3):171–206, 1997.

[2] B. Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[3] T. Bewley, J. Lawry, and A. Richards. Modelling agent policies with interpretable imitation learning, 2020.

[4] B. Bonet and H. Geffner. Factored probabilistic belief tracking. In S. Kambhampati, editor, *Proc. IJCAI 2016*, pages 3045–3052. IJCAI/AAAI Press, 2016.

[5] T. Chakraborti, A. Kulkarni, S. Sreedharan, D. E. Smith, and S. Kambhampati. Explicability? Legibility? Predictability? Transparency? Privacy? Security? The Emerging Landscape of Interpretable Agent Behavior. In *Proc. ICAPS 2019*, pages 86–96. AAAI Press, 2019.

[6] V. G. Costa, J. Pérez-Aracil, S. Salcedo-Sanz, and C. E. Pedreira. Evolving interpretable decision trees for reinforcement learning. *Artif. Intell.*, 327:104057, 2024.

[7] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning, 2017.

[8] M. Du, N. Liu, and X. Hu. Techniques for interpretable machine learning, 2019.

[9] J. Gajcin, R. Nair, T. Pedapati, R. Marinescu, E. Daly, and I. Dusparic. Contrastive explanations for comparing preferences of reinforcement learning agents, 2021.

[10] L. Gao and L. Guan. Interpretability of machine learning: Recent advances and future prospects, 2023.

[11] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal. Explaining explanations: An approach to evaluating interpretability of machine learning. *CoRR*, abs/1806.00069, 2018.

[12] N. Grandien, Q. Delfosse, and K. Kersting. Interpretable end-to-end neurosymbolic reinforcement learning agents, 2024.

[13] M. Grześ, P. Poupart, X. Yang, and J. Hoey. Energy efficient execution of POMDP policies. *IEEE Transactions on Cybernetics*, 45(11):2484–2497, 2015.

[14] E. Hansen. An improved policy iteration algorithm for partially observable MDPs. In *NIPS'97*, pages 1015–1021. MIT Press, 1998.

[15] E. A. Hansen and Z. Feng. Dynamic programming for POMDPs using a factored state representation. In *Proc. AIPS, 2000*, pages 130–139. AAAI, 2000.

[16] J. Hoey, R. St-Aubin, A. J. Hu, and C. Boutilier. SPUDD: stochastic planning using decision diagrams. In K. B. Laskey and H. Prade, editors, *Proc. UAI 1999*, pages 279–288. Morgan Kaufmann, 1999.

[17] J. Huang, P. P. Angelov, and C. Yin. Interpretable policies for reinforcement learning by empirical fuzzy sets. *Engineering Applications of Artificial Intelligence*, 91:103559, 2020.

[18] T. Huber, K. Weitz, E. André, and O. Amir. Local and global explanations of agent behavior: Integrating strategy summaries with saliency maps. *Artificial Intelligence*, 301:103571, 2021.

[19] H. Kohler, R. Akrour, and P. Preux. Interpretable decision tree search as a Markov decision process, 2024.

[20] H. Kohler, Q. Delfosse, R. Akrour, K. Kersting, and P. Preux. Interpretable and editable programmatic tree policies for reinforcement learning. *CoRR*, abs/2405.14956, 2024.

[21] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems IV 2008*. The MIT Press, 2008.

[22] J. Lang and B. Zanuttini. Probabilistic knowledge-based programs. In *Proc. IJCAI 2015*, IJCAI'15, page 1594–1600. AAAI Press, 2015.

[23] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis. Explainable AI: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021.

[24] C. Lu, R. Shi, Y. Liu, K. Hu, S. S. Du, and H. Xu. Rethinking transformers in solving POMDPs. In *Proc. ICML 2024*. OpenReview.net, 2024.

[25] B. Mazoure, T. Doan, T. Li, V. Makarenkov, J. Pineau, D. Precup, and G. Rabusseau. Representation of reinforcement learning policies in reproducing kernel Hilbert spaces, 2020.

[26] D. Meli, A. Castellini, and A. Farinelli. Learning logic specifications for policy guidance in POMDPs: An inductive logic programming approach. *Journal of Artificial Intelligence Research*, 79:725–776, 2024.

[27] N. Meuleau, L. Peshkin, K.-E. Kim, and L. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proc. UAI 1999*, pages 427–436, 1999.

[28] S. Miura and S. Zilberstein. A unifying framework for observer-aware planning and its complexity. In *Proc. UAI 2021*, volume 161 of *Proceedings of Machine Learning Research*, pages 610–620. AUAI Press, 2021.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[30] T. Ni, B. Eysenbach, and R. Salakhutdinov. Recurrent model-free RL can be a strong baseline for many POMDPs. In *Proc. ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, pages 16691–16723. PMLR, 2022.

[31] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: an anytime algorithm for POMDPs. In *Proc. IJCAI 2003*, page 1025–1030. Morgan Kaufmann Publishers Inc., 2003.

[32] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, University of Toronto, 2005.

[33] J. R. Quinlan. *C4.5: programs for machine learning*. Elsevier, 1993.

[34] L. Saulières, M. C. Cooper, and F. D. de Saint Cyr. Backward explanations via redefinition of predicates, 2024.

[35] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. UAI 200'*, UAI '04, page 520–527, Arlington, Virginia, USA, 2004. AUAI Press.

[36] T. Smith and R. G. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. *CoRR*, abs/1207.1412, 2012.

[37] N. Topin, S. Milani, F. Fang, and M. Veloso. Iterative bounding MDPs: Learning interpretable policies via non-interpretable methods, 2021.

[38] B. Zanuttini, J. Lang, A. Saffidine, and F. Schwarzentruber. Knowledge-based programs as succinct policies for partially observable domains. *Artificial Intelligence*, 288, 2020.