Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Probabilistic Knowledge-Based Programs*

Jérôme Lang
CNRS-LAMSADE
Université Paris-Dauphine, France
lang@lamsade.dauphine.fr

Bruno Zanuttini
GREYC
UNICAEN, CNRS, ENSICAEN, France
bruno.zanuttini@unicaen.fr

Abstract

We introduce Probabilistic Knowledge-Based Programs (PKBPs), a new, compact representation of policies for factored partially observable Markov decision processes. PKBPs use branching conditions such as *if the probability of φ is larger than* p, and many more. While similar in spirit to value-based policies, PKBPs leverage the factored representation for more compactness. They also cope with more general goals than standard state-based rewards, such as pure information-gathering goals. Compactness comes at the price of reactivity, since evaluating branching conditions on-line is not polynomial in general. In this sense, PKBPs are complementary to other representations. Our intended application is as a tool for experts to specify policies in a natural, compact language, then have them verified automatically. We study succinctness and the complexity of verification for PKBPs.

1 Introduction

Partially observable Markov decision processes (POMDPs) are probably the most popular framework for planning under partial observability. Planning for a POMDP means computing a *policy*, specifying what action to take at each step at execution time. A key question is how to represent policies. Two well-known families of representations are *history-based* and *value-based* ones. We investigate here a radically different representation: *probabilistic knowledge-based programs*.

Standard knowledge-based programs (KBPs) were introduced by Fagin *et al.* [1995], as high-level protocols describing the actions that agents should perform as a function of their knowledge, such as, typically, *if I know φ then π_1 else* π_2. Knowledge is expressed by $\mathsf{S5}$ modalities. Thus, in a KBP, branching conditions are epistemically interpretable, and deduction tasks are involved at execution time (online).

Some works have considered the use of KBPs in an AI planning context. KBPs are implemented in the situation calculus by Reiter [2001] and Claßen and Lakemeyer [2006]. Son and Baral [2001] and Herzig *et al.* [2003] use epistemic logic for expressing planning problems, with KBPs as output. Laverny and Lang [2005] generalize KBPs to belief-based programs with uncertain action effects and noisy observations. Lang and Zanuttini [2012; 2013] show that KBPs are powerful for expressing policies or plans: epistemic branching conditions allow for exponentially more compact representations than standard plans; the counterpart is an increase of complexity for plan verification and existence. Yet some other approaches that investigate succinct representations of plans, although not under the form of KBPs, are discussed by Bonet *et al.* [2010] and Bäckström and Jonsson [2011].

However, the practical applicability of these works is hampered by the restriction to nonprobabilistic belief states. Hence we consider a probabilistic variant of KBPs, PKBPs for short. These form a representation of policies or contingent plans for standard POMDPs and stochastic planning domains. The novelty is in allowing complex combinations of probabilistic statements as branching conditions, such as *if the probability that a tiger is behind door i is less than 0.1 then open door i, else listen at the door j with the lowest probability (among all doors) of there being a tiger behind.*

PKBPs provide an alternative to standard representations, as a more natural and compact language, but with reasoning involved during execution. We view them as a tool for helping experts to specify policies easily; so we study their succinctness and the complexity of automatically verifying them.

Background is given in Section 2 and related work discussed in Section 3. In Section 4 we define PKBPs and their execution semantics. In Section 5 we show that PKBPs are more succinct than standard POMDP policies or contingent plans in probabilistic planning. We address execution with compact representations in Section 6 and the complexity of plan verification in Section 7. We conclude in Section 8.

2 Preliminaries

A *partially observable Markov decision process* is a tuple $\Pi = (S, \mathbf{b}^0, A, \Omega, T, O, R)$, where S, A, Ω are finite sets of *states*, *actions*, and *observations*, and $\mathbf{b}^0$ is a probability distribution on S (written $\mathbf{b}^0 \in \Delta(S)$). An agent controls Π by taking an action $a^t \in A$ and observing some $o^t \in \Omega$ at each timestep $t = 0, 1, \ldots$. The dynamics are given by T, O, R:

- at each timestep $t \geq 0$, the process is in some state $s^t \in S$, which is not directly observed; s^0 is drawn from $\mathbf{b}^0$.

*Funded by the French Research Agency (ANR-10-BLAN-215).

- $\forall a \in A, s, s' \in S$, $T(s,a,s') \in [0,1]$ gives the probability of $s^{t+1} = s'$ for $s^t = s, a^t = a$.
- for $T(s,a,s') > 0$, $\forall o \in \Omega$, $O(s,a,s',o) \in [0,1]$ gives the probability of $o^t = o$ for $s^t = s, a^t = a, s^{t+1} = s'$.
- $R(\mathbf{b}) \in \mathbb{Q}$ gives the reward obtained by the agent at timestep t if its belief state is $\mathbf{b}$ (see below).

A *purely ontic* action a never gives feedback ($\exists o_\top \in \Omega, \forall s, s', O(s,a,s',o_\top) = 1$), and a *sensing* action a never changes the current state ($\forall s, T(s,a,s) = 1$).

Partial observability implies that the agent cannot monitor the current state s^t at execution time. Rather, it can maintain a *belief state* $\mathbf{b}^t \in \Delta(S)$; $\mathbf{b}^t(s)$ is its belief at timestep t that s^t is s. The belief state can be monitored using *progression* by actions taken and observations received:

$$\text{Prog}(\mathbf{b}^t, a^t, o^t)(s') = \frac{\sum_s \mathbf{b}^t(s) T(s,a^t,s') O(s,a^t,s',o^t)}{\sum_{s,s''} \mathbf{b}^t(s) T(s,a^t,s'') O(s,a^t,s'',o^t)}$$

Note that we define rewards on *belief states*. Since we consider *expectations* of rewards, a state-based reward function $R' : S \to \mathbb{Q}$ can be captured by $R(\mathbf{b}) = \sum_{s \in S} \mathbf{b}(s) R'(s)$.[1] The more general setting however comes for free in our approach (since we do not use the piecewise linearity and convexity of Q-values), and captures more general goals: for instance, *information-gathering goals* [Sabbadin *et al.*, 2007; Araya-López *et al.*, 2010] such as $R(\mathbf{b}) = 1$ if $\exists s \in S, \mathbf{b}(s) > 0.99$ and 0 otherwise (identify the current state), or $\sum_i (1 - P(x_i \mid \mathbf{b}) P(\overline{x}_i \mid \mathbf{b}))$ (identify the value of each x_i, where x_i is a state variable, or equivalently, a set of states).

A *policy* π prescribes an action at each step, given (possibly indirectly) past actions and observations. Its *expected value* at horizon $H < \infty$ is $\mathbb{E}(\sum_{t=0}^{H} R(\mathbf{b}^t))$, with $\mathbf{b}^{t+1} = \text{Prog}(\mathbf{b}^t, a^t, o^t)$, $a^t = \pi(a^0, o^0, \ldots, a^t, o^t)$, and expectation over $s^0 \sim \mathbf{b}^0, s^{t+1} \sim T(s^t, a^t, \cdot), o^t \sim O(s^t, a^t, s^{t+1}, \cdot)$.

Example 2.1. *We consider a variant of the famous tiger example. There are 5 doors, a tiger hidden behind 2 of them, and a princess behind one of the other 3. Initially, all triples (i,j,k) (tigers behind doors i and j, princess behind door k, with i,j,k all different) are equiprobable. We consider (sensing) actions $listen_i$, $i = 1, \ldots, 4$ (note that one cannot listen at door 5), and (ontic) actions $open_i$, $i = 1, \ldots, 5$. When performing $listen_i$, the agent may hear the tiger roaring (r_+) or not (r_-), with probability* 0.5 *each, if a tiger is behind door i, and never hears anything otherwise. If the agent performs $open_i$, it is eaten for sure by the tiger if there is one behind door i (yielding reward -1), and married to the princess for sure if she is there (yielding reward $+1$).*

In a POMDP formulation, $T(s, listen_i, s')$ is 1 *for $s = s'$ and* 0 *otherwise (the state is unchanged) and, for instance: $O(s, listen_i, s, r_+)$ is* 0.5 *if there is a tiger behind door i in s; $T(s, open_i, s')$ is* 1 *if there is a tiger behind door i in s, and s' is the same as s but with door i open and the agent eaten, and $R(s')$ is -1 in such s'; $R(\mathbf{b})$ is defined by $\sum_s \mathbf{b}(s) R(s)$.*

Representation of Policies Representing policies is a crucial issue for POMDPs, given that a policy π maps an exponential number (in the horizon H) of histories to actions. Two main families of approaches are studied in the literature.

[1]The extension to $R(s^t, a^t, s^{t+1})$ or $R(\mathbf{b}^t, a, \mathbf{b}^{t+1})$ is direct.

In *history-based* approaches, when π is executed it is in some internal state q^t at any timestep, and there is a direct mapping from internal states to actions to take. The transition to the next internal state q^{t+1} is directly provoked by the observation o^t received. Basic examples are *policy trees*, with nodes labelled by actions and arcs by observations: q^t is the current node, a^t is its label, and q^{t+1} is obtained by following the arc labelled o^t. A more compact representation is given by *finite-state controllers*, which are generalisations of this to possibly cyclic graphs, typically seen as automata.

In *value-based* approaches, the agent explicitly maintains its belief state $\mathbf{b}^t$, and π gives the expected value of taking each action (assuming that the continuation of the policy will be optimal) as a function of $\mathbf{b}^t$; the action maximizing this value is chosen. A typical example is a set Ξ of *α-vectors*: π is represented by pairs (α_i, a_i) with α_i an $|S|$-dimensional vector of scalars and $a_i \in A$. Such Ξ represents π through $\pi(\mathbf{b}^t) = a_{i^*}$, with $i^* = \text{argmax}_i\, \mathbf{b}^t \cdot \alpha_i$ (inner product).

Factored Representations We use *factored representations* of POMDPs, in which $X = \{x_1, \ldots, x_n\}$ is set of Boolean *state variables* and $S = 2^X$ is the set of assignments to them. $\mathbf{b}^0, T, O, R$ are specified using structured representations. We assume $\mathbf{b}^0$ is represented by a Bayesian network (BN) over X,[2] and T, O by *two-slice temporal BNs* (2TBNs).

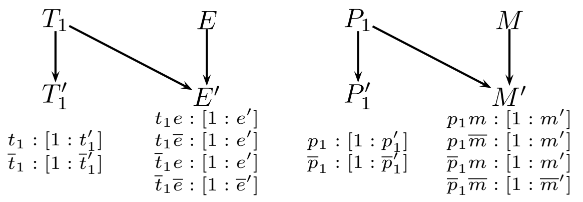
We assume familiarity with BNs (see [Darwiche, 2009]). 2TBNs [Dean and Kanazawa, 1989] have been used in planning since [Boutilier *et al.*, 1999], and are taken into account in probabilistic PDDL [Younes *et al.*, 2005]. A 2TBN over X is a BN over $X \cup X'$, where $X' = \{x'_1, \ldots, x'_n\}$; x_i and x'_i represent x_i at some timestep and at the next one, respectively. Each edge in the underlying digraph of a 2TBN is from some unprimed to some primed variable, or from x'_i to x'_j for $i < j$ in some fixed ordering. Moreover, only primed variables have conditional probability tables (CPTs). A 2TBN hence represents distributions on the assignments to X' conditional on the value of X. An example of representation of T with 2TBNs is given below. The function O is represented similarly: the 2TBN for $O(\cdot, a, \cdot, \cdot)$ has variables $X \cup X'$ and o (with domain Ω), and a CPT only for o, which gives the probability ditribution on Ω for all s, s' with $T(s,a,s') > 0$. We discuss the representation of rewards in Section 7.

Example 2.2 (cont'd). *We use state variables t_i (a tiger is behind door i, $i = 1, \ldots, 5$), p_i (the princess is behind door i, $i = 1, \ldots, 5$), m (married) and e (eaten). The state assigning only t_1, t_3, p_4 to true formalizes the situation where the tigers are behind doors* 1 *and* 3, *and the princess behind door* 4. *Figure 1 shows a part of the 2TBN N_{open_1} representing $T(\cdot, open_1, \cdot)$ (variables t_i and p_i, $i \neq 1$, are not represented), Figure 2 a part of the BN $N_{\mathbf{b}^0}$ for $\mathbf{b}^0$, and Figure 3 a part of the BN for* $\text{Prog}(\mathbf{b}^0, open_1)$ *obtained by progressing $N_{\mathbf{b}^0}$ by N_{open_1}* before receiving the observation.

3 Related Work

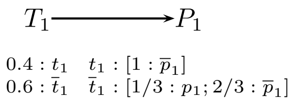
Boutilier and Poole [1996] investigated factored representations for POMDPs, with CPTs represented by decision trees,

[2]Abusing notation, we do not distinguish between state variables and the corresponding random variables.



T_1
E
P_1
M
T'_1
E'
P'_1
M'
$t_1 : [1 : t'_1]$
$\overline{t}_1 : [1 : \overline{t}'_1]$
$t_1 e : [1 : e']$
$t_1\overline{e} : [1 : e']$
$\overline{t}_1 e : [1 : e']$
$\overline{t}_1\overline{e} : [1 : \overline{e}']$
$p_1 : [1 : p'_1]$
$\overline{p}_1 : [1 : \overline{p}'_1]$
$p_1 m : [1 : m']$
$p_1\overline{m} : [1 : m']$
$\overline{p}_1 m : [1 : m']$
$\overline{p}_1\overline{m} : [1 : \overline{m}']$

Figure 1: (A part of) the 2TBN N_{open_1}



T_1
P_1
$0.4 : t_1$
$0.6 : \overline{t}_1$
$t_1 : [1 : \overline{p}_1]$
$\overline{t}_1 : [1/3 : p_1; 2/3 : \overline{p}_1]$

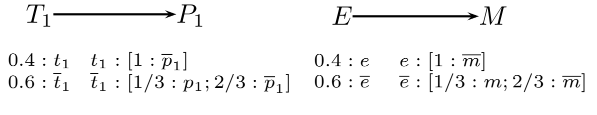
Figure 2: (A part of) the BN $N_{\mathbf{b}^0}$ representing $\mathbf{b}^0$

and derived a condensed representation of α-vectors. This work was extended in particular by Hansen and Feng [2000] to CPT and α-vector representation by ADDs, and by Shani *et al.* [2008] for point-based (nonexact) approaches. *Probabilistic STRIPS operators* [Kushmerick *et al.*, 1995] are another factored representation. They were investigated for POMDPs by Wang and Schmolze [2005], who also derived compact representations of α-vectors. We discuss these works more in depth in Section 5. Poupart [2005, Chapters 3, 4] gives a thorough overview of compact representations up to 2005.

Mundhenk *et al.* [2000] and Madani *et al.* [2003] consider *succinct* and *compressed* representations of POMDPs, where T and O are represented by circuits. They study the decidability and complexity of evaluating a policy and deciding whether there exists one reaching a given value, but do not consider compact representations of (history-dependent) policies. Yet other compact representations studied in the literature are *first order* [Sanner and Boutilier, 2009] and *predictive-state* representations [Hamilton *et al.*, 2014].

The Knowledge Representation and Reasoning community also studied progression (and sometimes regression) of belief states by stochastic actions (including sensing actions) in expressive and compact logical languages, such as first-order logic or the situation calculus [Bacchus *et al.*, 1999; Gabaldon and Lakemeyer, 2007; Belle and Levesque, 2013; Belle and Lakemeyer, 2011]), propositional logic and dynamic BNs [Hajishirzi and Amir, 2010], or propositional action languages [Iocchi *et al.*, 2009]. Extensions to dynamic logic were also developed for reasoning with stochastic actions (associated with modalities) and observations, such as in [Aucher, 2007; van Eijck and Schwarzentruber, 2014; Rens *et al.*, 2014]. Nevertheless, none of these works considers programs which branch on knowledge or belief.

Finally, although we are interested in exact representations of policies here, let us mention the large body of work on approximate solutions to POMDPs, the most prominent of which being point-based approaches [Shani *et al.*, 2013].



T_1
P_1
E
M
$0.4 : t_1$
$0.6 : \overline{t}_1$
$t_1 : [1 : \overline{p}_1]$
$\overline{t}_1 : [1/3 : p_1; 2/3 : \overline{p}_1]$
$0.4 : e$
$0.6 : \overline{e}$
$e : [1 : \overline{m}]$
$\overline{e} : [1/3 : m; 2/3 : \overline{m}]$

Figure 3: (A part of) the BN representing $\mathrm{Prog}(\mathbf{b}^0, open_1)$

4 Probabilistic Knowledge-Based Programs

We now introduce *probabilistic knowledge-based programs* (or *policies* — PKBPs). The representation aims at taking the best of history- and value-based representations: like the former, they do not need to distinguish belief states with different expected values, provided the action to take is the same, and like the latter, they summarize the relevant portion of the history as conditions on the current belief state.

The term *knowledge-based* is borrowed from knowledge-based programs [Fagin *et al.*, 1995], and refers to the fact that PKBPs branch on what the agent *knows* (*believes*) at execution time. Hence they rely on belief state monitoring. As we will see, with PKBPs the next action to take cannot be computed in polynomial time in general, but PKBPs can be exponentially more compact than "reactive" policies. Moreover, they are expressed in a very natural language.

Hence PKBPs complement standard representations. In particular, we have in mind applications in which experts do not want to let an automated system compute decisions, like when human lives (military scenarios) or a lot of money (*e.g.*, spacecrafts) are at stake. Then PKBPs provide a very interesting (compact and natural) language for them to write policies by themselves, while allowing for automated verification.

Moreover, in some applications decisions need not be made instantaneously online: for instance, for systems interacting with humans, when decisions are interleaved with human actions (typically slow as compared to CPU clocks), or for robotic systems in which the execution of one action is very slow (*e.g.*, Curiosity is known to move at 150 m/h on Mars).

Representation We use a language, written $\mathcal{F}^n$, of *probability formulas* over variables $x_1, \dots, x_n$ [Fagin *et al.*, 1990].

The atoms in $\mathcal{F}^n$ are all atoms of the form $P(\varphi)$, where φ is a propositional formula over $x_1, \dots, x_n$. The *value* of $P(\varphi)$ in a belief state $\mathbf{b}$ is the probability for φ to be true in $\mathbf{b}$, namely, $V_{\mathbf{b}}(\varphi) = \sum_{s \models \varphi} \mathbf{b}(s)$. Expressions can be formed using sum and product[3] between atoms and/or rational numbers; *e.g.*, $2P(\varphi)P(\psi) + P(\xi) + 0.5$. We write $\mathcal{E}^n$ for the set of all such expressions, and define $V_{\mathbf{b}}(E)$ in the obvious way for $E \in \mathcal{E}^n$. Finally, $\mathcal{F}^n$ is defined inductively by:

- $\top$ (truth) and $\bot$ (falsity) are probability formulas in $\mathcal{F}^n$,
- for $E \in \mathcal{E}^n, p \in \mathbb{Q}$, $\bowtie \in \{\geq, >, \leq, <\}$, $E \bowtie p$ is in $\mathcal{F}^n$,
- for $\Phi, \Psi \in \mathcal{F}^n$, $\neg\Phi$, $\Phi \vee \Psi$, $\Phi \wedge \Psi$, $\Phi \rightarrow \Psi$ are in $\mathcal{F}^n$.

While an expression $E \in \mathcal{E}^n$ has a *value* $V_{\mathbf{b}}(E)$ in a belief state $\mathbf{b}$, a probability formula Φ is either true or false in $\mathbf{b}$. For instance, the uniform belief state satisfies $P(\varphi) > 0$ for any satisfiable φ, and falsifies $P(x_1) \geq 2P(x_2) \vee P(x_1) \leq 0$. We write $\mathbf{b} \models \Phi$ if $\mathbf{b}$ satisfies Φ, and $\mathbf{b} \not\models \Phi$ otherwise. We sometimes write $\overline{x}_i$ instead of $\neg x_i$.

Example 4.1 (cont'd). *Let* $\mathbf{b}$ *assign probability* $1/30$ *to each state satisfying* $\overline{e}$ *and* $\overline{m}$*, and* 0 *to all other states (the agent is not married nor eaten, and situations are otherwise equally*

[3]Other efficiently computable operators could be used.

likely). Then $V_{\mathbf{b}}(P(t_1 \vee t_2))$ is $1/2$, $V_{\mathbf{b}}(P(t_1) - 1/2 P(t_2 \wedge \bar{t}_3))$ is $1/5$, and $\mathbf{b} \models P(t_1) > 0 \wedge P(t_2) \leq 1/2$ holds.

We can now define PKBPs formally.

Definition 1. *Let $n \in \mathbb{N}$ and A a set of actions.* Probabilistic knowledge-based programs *(PKBPs) are defined by:*

- *λ is a PKBP (empty policy), and every $a \in A$ is a PKBP,*
- *if π_1, π_2 are PKBPs, then $[\pi_1; \pi_2]$ is a PKBP (sequence),*
- *if π_1, π_2 are PKBPs and $\Phi \in \mathcal{F}^n$ is a probability formula, then* **[if** Φ **then** π_1 **else** π_2 **]** *is a PKBP,*[4]
- *if π is a PKBP and $\Phi \in \mathcal{F}^n$ is a probability formula, then* **[while** Φ **do** π **]** *is a PKBP.*

Example 4.2 (cont'd). *The following is a PKBP:*

```
$listen_1$; $listen_2$; $listen_3$; $listen_4$;
while $P(t_1) > 0.1 \wedge \cdots \wedge P(t_5) > 0.1$ do
    if $P(t_1) \leq P(t_2) \wedge \cdots \wedge P(t_1) \leq P(t_5)$ then
        [ $listen_1$; if $P(t_1) \leq 0.1$ then $open_1$ ]
    elseIf $P(t_2) \leq P(t_1) \wedge \cdots \wedge P(t_2) \leq P(t_5)$ then
        [ $listen_2$; if $P(t_2) \leq 0.1$ then $open_2$ ]
    ...
    else [ if $P(t_5) \leq 0.1$ then $open_5$ ]
```

In general, PKBPs represent *nonstationary* policies, which can prescribe a different action for the same belief state at different timesteps. If **while** constructs are omitted, PKBPs can also be seen as *plans*, prescribing a finite sequence of actions (contingent on observations) for a given initial belief state. *Stationary* policies are captured by policies of the form

```
while $\top$ do [ if $\Phi_1$ then $a_1$ elseIf $\Phi_2$ then $a_2$ ... else $a_k$ ] .
```

PKBPs make up a very natural language for specifying policies. In particular, they make explicit what conditions on the belief state imply the choice of a given action, unlike with value-based representations, even for factored POMDPs [Boutilier and Poole, 1996].

Semantics As a representation of a policy, a PKBP gives an action for each timestep t, as a function of actions taken and observations received so far, as summarized in the current belief state $\mathbf{b}^t$. The semantics is straightforward. For instance, for π = **[** a;π_1 **]** , the prescribed action is a, and execution goes on from the next timestep on with π_1. For π = **[if** Φ **then** π_1 **else** π_2 **]** , the agent must first decide $\mathbf{b}^t \models \Phi$: if this is true then the agent acts following π_1 (which may recursively call for other tests), and otherwise it acts following π_2. Finally, if the PKBP boils down to the empty PKBP λ, we define the execution to stop (and no more reward to be obtained).

A PKBP of the form π = **[while** Φ **do** π_1 **]** has exactly the same semantics as the PKBP **[if** Φ **then** π_1 ; π **else** λ**]** . Of course, plans with loops are not guaranteed to stop.[5]

[4]**[if** Φ **then** π_1 **else** λ **]** is abbreviated by **[if** Φ **then** π_1 **]** .

[5]Note that there is an ambiguity about the meaning of the infinitely repeated empty plan (the simplest form of which being of the form **[while** Φ **do** λ **]**): it is not clear whether running an empty plan forever is a plan that stops immediately or runs forever. This is however not important for this paper (we consider finite horizons).

Example 4.3 (cont'd). *Let us denote a belief state $\mathbf{b}$ as the tuple $(P(t_1), \ldots, P(t_5))$ (probability of there being a tiger behind each door), with situations otherwise equally likely. The initial belief state is $\mathbf{b}^0 = (0.4, 0.4, 0.4, 0.4, 0.4)$. We take $a^0 = listen_1$; assume we receive observation r^-, then $\mathbf{b}^1 = (0.25, 0.4375, 0.4375, 0.4375, 0.4375)$. Then we take $listen_2$; assuming $o^1 = r^-$ we get $\mathbf{b}^2 = (0.28, 0.28, 0.48, 0.48, 0.48)$. After taking $listen_3$ and observing r^+ we get $\mathbf{b}^3 \approx (0.17, 0.17, 1.0, 0.33, 0.33)$. After performing $listen_4$ and observing r^- we get $\mathbf{b}^4 \approx (0.2, 0.2, 1.0, 0.2, 0.4)$. Now we evaluate the branching condition and find $\mathbf{b}^4 \models P(t_1) \leq P(t_i)$ for all $i > 1$, therefore we take $a^1 = listen_1$. Assuming $o^1 = r^-$ we get $\mathbf{b}^5 \approx (0.11, 0.22, 1.0, 0.22, 0.44)$, hence $a^5 = listen_1$ again. Assuming we observe r^- once again, we get $\mathbf{b}^6 \approx (0.06, 0.24, 1, 0.24, 0.47)$, hence we execute open$_1$.*

Interestingly, some PKBPs containing a **while** loop (such as the one in Example 4.2) almost surely stop, that is, the probability that they stop after at most t steps tends to 1 with $t \to \infty$. Then the probabilities of all finite histories sum up to 1 and the expected reward at an infinite horizon is well-defined, even without using a discount factor.[6]

5 Succinctness

We now show that PKBPs are always at least as, and possibly exponentially more, succinct than other representations of policies, which makes them appealing. Precisely, we compare them to representations using α-vectors, possibly compactly represented, and to generic "reactive" representations.

Definition 2. *A representation of a policy π is* reactive *if for any POMDP Π, timestep t, history $\vec{h} = (a^0, o^0, \ldots, a^t, o^t)$ with each a^u given by π, the action $a^{t+1} = \pi(\vec{h})$ can be computed in polytime in the size of Π and π.*

Policy trees and finite state controllers are reactive representations, since the action to take can be read off directly from the current state in the representation, which can be updated by simply following the observation received. Similarly, a set Ξ of *flat* α-vectors, that is, α-vectors represented explicitly as vectors of $|S|$ rational numbers, forms a reactive representation, as a flat belief state can be maintained with per-step updating in time polynomial in $|S|$ and hence in the size of Ξ, and deciding the next action to take requires essentially as many inner products as $|\Xi|$. Note that under our definition, a representation can be reactive while of size exponential in the size of the problem (this is typically the case for sets of α-vectors). The picture is more subtle for *compact* α-vectors; we discuss this case at the end of this section.

Recall that P/poly is the (nonuniform) complexity class of problems whose answer can be computed by a polynomial-size circuit for each input size. NP $\not\subseteq$ P/poly is a standard complexity-theoretic assumption; the converse would imply a collapse of the polynomial hierarchy at the second level. The following result says that some PKBPs are exponentially more succinct than *any* equivalent reactive representation.

[6]Of course, some PKBPs stop only with some probability < 1, possibly even 0, such as **[while** $\top$ **do** *toss-coin* **]** .

Proposition 5.1. *If* NP $\not\subseteq$ P/poly *holds, there are policies with polysize PKBP representations, but no polysize reactive representation. This holds even if we forbid the* **while** *construct in PKBPs (but not in reactive policies) and restrict the reward function to be state-based.*

Proof. Building on Lang and Zanuttini's [2013] construction for (standard) KBPs, we define POMDPs $(\Pi_1, \Pi_2, \dots)$ such that $\Pi_m = (n, \mathbf{b}^0, A, \Omega, T, O, R)$ encodes the NP-complete 3-SAT problem over $x_1, \dots, x_m$. Intuitively, the state encodes an "immutable"[7] 3CNF $\varphi = \bigwedge_{i \in I}(L_{i,1} \vee L_{i,2} \vee L_{i,3})$, where each $L_{i,j}$ is a literal x_k or $\overline{x}_k$, and a mutable assignment $\vec{x} = x_1 \dots x_m$ to its variables, together with the constraint $(\vec{x} \not\models \varphi) \rightarrow \overline{x}_s$ for a distinguished, immutable variable x_s. The goal is to set $\vec{x}$ to a model of φ or to detect that φ is unsatisfiable, which amounts to being certain that x_s is false. We only sketch the construction, and refer to Lang and Zanuttini [2013] for details on the encoding.

- $\mathbf{b}_0$ is uniform over all states satisfying $(\vec{x} \not\models \varphi) \rightarrow \overline{x}_s$ (x_s is false if $\vec{x} \not\models \varphi$, with no more informed prior),
- Action $test(L_{i,j})$ reliably reports which literal $L_{i,j}$ is, and action $a_i^\top$ (resp. $a_i^\perp$) always sets x_i to $\top$ (resp. $\perp$),
- R is 1 at states satisfying $\overline{x}_s$ or $\vec{x} \models \varphi$, 0 elsewhere.

It is easy to check that Π_m has a representation of size polynomial in m using BNs for $\mathbf{b}^0$, 2TBNs for actions, and an expression in $\mathcal{E}^n$ for R (see Section 7). Now the following PKBP π has expected value 1 for Π at horizon $O(\binom{2m}{3} + m)$:

```
$test(\ell_{1,1})$ $test(\ell_{1,2})$ ... $test(\ell_{\binom{2m}{3},3})$
**if** $P(x_s) > 0$ **then**
    **if** $P(\varphi \wedge x_1) > 0$ **then** $a_1^\top$ **else** $a_1^\perp$
    **if** $P(\varphi \wedge x_2) > 0$ **then** $a_2^\top$ **else** $a_2^\perp$
    ...
    **if** $P(\varphi \wedge x_m) > 0$ **then** $a_m^\top$ **else** $a_m^\perp$
```

Indeed, $P(x_s) = 0$ holds if and only if $P(\vec{x} \models \varphi)$ is 0, that is, if φ is unsatisfiable since $\vec{x}$ is initially unknown. On the other hand, if φ is satisfiable, then the PKBP builds a model of it over $\vec{x}$ (read "$P(\varphi \wedge x_i)$" as "there is at least one assignment which extends $x_0, \dots, x_{i-1}$, sets x_i to true, and satisfies φ").

Clearly, π has size polynomial in m. Now, towards contradiction, assume there is a polynomial p and *reactive policies* $\pi_1, \pi_2, \dots$ for $\Pi_1, \Pi_2, \dots$, with $|\pi_i| \leq p(m)$ and expected value 1 at the given horizon. Then executing π_m solve the 3-SAT problem over m variables. Since π_m is reactive and the horizon is polynomial in m, executing π_m amounts to running a polytime algorithm for 3-SAT over m variables, hence 3-SAT is in P/poly, a contradiction. $\square$

Proposition 5.2. *For all families* $(\Pi)_n$ *of POMDPs and families* $(\pi)_n$ *of policies for* $(\Pi)_n$*, if* $(\pi)_n$ *has a polysize representation using flat* α*-vectors or* α*-vectors represented by decision trees [Boutilier and Poole, 1996], then it also has a polysize PKBP representation.*

[7]By this we mean that no action can change the variables of φ.

Proof. We show the result for decision trees, which are at least as compact as flat vectors. Given a representation $\Xi_n = \{(\alpha_i, a_i) \mid i = 1, \dots, N\}$, we first represent π_n by:

```
**while** $\top$ **do**
    **if** $\mathbf{b} \cdot \alpha_1 \geq \mathbf{b} \cdot \alpha_2 \wedge \cdots \wedge \mathbf{b} \cdot \alpha_1 \geq \mathbf{b} \cdot \alpha_N$ **then** $a_1$
    **elseIf** $\mathbf{b} \cdot \alpha_2 \geq \mathbf{b} \cdot \alpha_3 \wedge \cdots \wedge \mathbf{b} \cdot \alpha_2 \geq \mathbf{b} \cdot \alpha_N$ **then** $a_2$
    ...
    **else** $a_N$
```

Now we reformulate $\mathbf{b} \cdot \alpha_i$ to $\sum_\beta \alpha_i(\beta) P(\beta)$, where the sum is over all branches β of the decision tree α_i, $\alpha_i(\beta)$ is the value associated to β, and $P(\beta)$ stands for $P(\bigwedge_\beta x_i = v_i)$, i.e., the probability of the conjunction of all decisions which make up β. Clearly, this conversion yields a PKBP representing π_n and of size polynomial in the size of Ξ_n. $\square$

For α-vectors represented by Algebraic Decision Diagrams (ADDs) [Hansen and Feng, 2000; Sim *et al.*, 2008] or with other compact representations [Wang and Schmolze, 2005], the result does not directly hold. Instead, we could obtain the same result using ADDs or other compact representations for formulas φ in atoms $P(\varphi)$. All our results could be lifted to such representations, but we leave the details for future work.

6 Execution with Compact Representations

We now show how to execute a PKBP π for a factored POMDP $\Pi = (n, \mathbf{b}_0, A, \Omega, T, O, R)$, *i.e.*, how to monitor the belief state and evaluate conditions. We give two approaches.

Standard Progression The most natural approach to belief state monitoring is to maintain a BN over $x_1, \dots, x_n$. It is well-known that the progression of $\mathbf{b}^t$ can be realized by connecting together the BN for $\mathbf{b}^t$ and the 2TBNs of $T(\cdot, a^t, \cdot)$ and $O(\cdot, a^t, \cdot, \cdot)$, then conditioning on o^t and marginalizing unprimed variables. For details about algorithms performing this with BNs we refer the reader to Darwiche [2009].

We now consider the evaluation of branching conditions. Recall that PP is the class of problems which are solvable in polytime by a probabilistic algorithm which has probability less than $1/2$ to give the wrong answer, and that an algorithm is in P$^{\text{PP}}$ it it runs in polytime with access to a PP-oracle.

Proposition 6.1. *The problem of deciding, given* π *and a BN for* $\mathbf{b}^t$*, which action to take according to* π *in* $\mathbf{b}^t$*, is in P*$^{\text{PP}}$.

Proof. From the semantics of PKBPs, this amounts to deciding $\mathbf{b}^t \models \Phi$ for a polynomial number of branching conditions $\Phi \in \mathcal{F}^n$. In turn, $\mathbf{b}^t \models \Phi$ can be decided in polynomial time given the value $P_{\mathbf{b}}(\varphi)$ for each atom $P(\varphi)$ occurring in Φ, so it is enough to show that the computation of $P_{\mathbf{b}}(\varphi)$ is in PP.

For this we use a classical trick (see [Darwiche, 2009, Sec. 5.2.1]): we add to $\mathbf{b}^t$ a fresh node for each operator in φ (assumed binary, parenthesizing as necessary). For instance, if φ is of the form $\psi \vee \xi$, we add a node N_φ with parents N_ψ, N_ξ, and CPT entries $\psi\xi : 1.0$, $\overline{\psi\xi} : 0.0$, etc. This results in a BN $\mathbf{b}^+$ of size polynomial in that of $\mathbf{b}^t$ and containing, in particular, a node N_φ for the binary random variable representing the satisfaction of φ. Then evaluating $P_{\mathbf{b}}(\varphi)$ can be done *via* a polynomial number of tests $P_{\mathbf{b}}(\varphi) \geq p$, each of which is in PP [Darwiche, 2009, Section 11.4].[8] □

While this may seem a prohibitive complexity for making the next decision online, one can take advantage of the large body of work about reasoning with BNs, using reasoners as black-boxes, to achieve efficient execution in practice.

Memoryful Progression A generic drawback of belief state monitoring is that the representation of $\mathbf{b}^t$ may grow exponentially with t, since state variables initially uncorrelated can become correlated when conditioning on observations (as E and M on Figure 3). Hence, we propose another progression operator which, at least from the complexity-theoretic point of view, allows to keep the size of $\mathbf{b}^t$ under control. The idea is keep track of the whole history of actions taken and observations *possibly* received, which can be done efficiently, and to condition on the observations *actually* received only when necessary, for evaluating branching conditions.

For $\Pi = (n, \mathbf{b}_0, A, \Omega, T, O, R)$, $\vec{h} = (a^0, o^0, \ldots, a^t, o^t)$, we define the *memoryful representation* of $Prog(\mathbf{b}^0, \vec{h})$ to be the BN obtained as follows. We first rename each x_i to x_i^0 in the BN for $\mathbf{b}^0$. Now for each timestep u, we make a fresh copy of the 2TBN for $T(\cdot, a^u, \cdot)$ and that for $O(\cdot, a^u, \cdot, \cdot)$. In these we rename x_i to x_i^u, x_i' to x_i^{u+1} ($i = 1, \ldots, n$), and the observation variable o to o^u. Finally, we connect all these copies together and to $\mathbf{b}^0$, merging common variables.

In the end, we obtain a representation of the current belief state $\mathbf{b}^t$ *in intension*, as a BN whose size only grows linearly with t. It is easy to check that deciding $\mathbf{b}^t \models \Phi$ for a branching condition Φ can be done by conditioning on $o^0, o^1, \ldots, o^t$ as received in $\vec{h}$, then marginalizing x_i^u for all $u < t$. Of course, a direct algorithm doing so would face an explosion in size, but we can take advantage of any off-the-shelf algorithm for reasoning with BNs (see again [Darwiche, 2009]).

Note that it is certainly not desirable in general to use memoryful instead of standard progression in practice. A reasonable heuristic would certainly be to use memoryful progression and regularly test whether some past variables can be marginalized while keeping the representation small. Anyway, as a theoretical tool, memoryful progression allows to monitor the belief state in polynomial space.

7 Verification

Given that PKBPs provide a language for experts to specify policies in a compact and natural way, an important problem is that of *verifying* that such a specified policy is "correct". Such an automated verification complements the expertise of the user who specified the policy. We restrict here to evaluation at a finite horizon, leaving issues related to termination for future work (see the end of Section 4).

Here we assume rewards are represented by expressions in $\mathcal{E}^n$, like $\sum_i (1 - P(x_i)P(\overline{x}_i))$ (see Section 2). This makes the presentation simpler, but the very same result would hold with another representation of rewards, e.g., by ADDs.[9]

Proposition 7.1. *The following problem is in PSPACE: given a factored POMDP $\Pi = (n, \mathbf{b}^0, A, \Omega, T, A, R)$, a PKBP π, $H \in \mathbb{N}$ written in unary, and $\theta \in \mathbb{Q}$, is the expected value of π (for Π at horizon H) at least θ?*[10]

Proof. The algorithm enumerates all observation histories $\vec{h} = (o^1, o^2, \ldots, o^H) \in O^H$ of size H, and for each one computes $w(\vec{h}) = p(\vec{h}) \times r(\vec{h})$, namely, the reward $r(\vec{h})$ obtained by π along $\vec{h}$ weighted by the probability $p(\vec{h})$ for $\vec{h}$ to occur. Provided this can be computed in PSPACE for each $\vec{h}$, the whole algorithm runs in PSPACE, since it simply sums up all $w(\vec{h})$'s in an accumulator.[11]

For a given $\vec{h} = (o^0, o^1, \ldots, o^H)$, we incrementally build a series of Bayesian networks $B^0, \ldots, B^H$, where B^t is the memoryful progression of $\mathbf{b}^0$ by $(a^0, o^0, \ldots, a^k, o^k)$ (where a^t's are given by π). At each step we use B^t and the construction in Section 6 to decide which action a^t to take next, and we build B^{t+1}. At the same time we update the probability $p(\vec{h})$ to $p(\vec{h}) \times P_{B^{t+1}}(o^{t+1})$, and $r(\vec{h})$ to $r(\vec{h}) + R(\mathbf{b}^{t+1})$ using the tools from Section 6 again. Since memoryful progression requires only polynomial space, and all elementary queries to the BNs are in $\mathrm{P}^{\mathrm{PP}} \subseteq$ PSPACE, we get the result. □

It is worth noting that there is no real increase in complexity for verification when we use PKBPs instead of other representations. Indeed, Mundhenk *et al.* [2000] show that verification is already PSPACE-hard for time-dependent (instead of history-dependent) policies and nonobservable MDPs (instead of POMDPs). While the complexity setting is not exactly the same, this suggests that not much is lost with PKBPs as long as verification is concerned (while a lot may be gained in compactness). Note that membership in PSPACE could not be taken for granted, given that we use factored representations and that observations introduce correlations.

8 Conclusion

We have introduced probabilistic knowledge-based programs (PKBPs), as a new representation of policies for partially observable planning problems. PKBPs are more succinct than standard, reactive policies, and are more intuitive to read and to express by an expert. Moreover, verifying PKBPs at a finite horizon remains in PSPACE. On the other hand, PKBPs require reasoning at execution time and hence cannot be reactive, so they constitute a representation complementary to standard ones. Which one to use should be chosen depending on the domain at hand.

There are numerous issues for further research, such as (obviously) the computation of solution PKBPs for a given planning problem, as well as using advanced algorithms on logics for representing probabilistic beliefs [Fagin *et al.*, 1990], and investigating heuristic and approximate approaches to POMDPs using PKBPs.

[8] Each digit in $P_{\mathbf{b}}(\varphi)$ can be computed in one test $P_{\mathbf{b}}(\varphi) \geq q + 2^{-i}$, and for evaluating $E > p$ we need no more digits than p has.

[9] For the result we only need the problem of computing $R(\mathbf{b})$ given a BN for $\mathbf{b}$ to be in PSPACE.

[10] Requiring that H is given in unary form means that $O(H)$ is considered to be polynomial in the size of the input.

[11] There are essentially $|O|^H$ histories to consider, hence if each $w(\vec{h})$ can be represented in polynomial space $p(|\Pi|)$, their sum can be represented in space $\log(|O|^H \times 2^{|p(\Pi)|}) \in O(H \log(|O|) + |p(|\Pi|))$, which is indeed polynomial in H and the size of Π.

References

[Araya-López *et al.*, 2010] M. Araya-López, O. Buffet, V. Thomas, and F. Charpillet. A POMDP extension with belief-dependent rewards. In *NIPS*, pages 64–72, 2010.

[Aucher, 2007] G. Aucher. Interpreting an action from what we perceive and what we expect. *J. of Applied Non-classical Logics*, 17(1):9–38, 2007.

[Bacchus *et al.*, 1999] F. Bacchus, J. Halpern, and H. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *AIJ*, 111(1-2):171–208, 1999.

[Bäckström and Jonsson, 2011] C. Bäckström and P. Jonsson. Limits for compact representations of plans. In *ICAPS*, pages 146–153, 2011.

[Belle and Lakemeyer, 2011] V. Belle and G. Lakemeyer. A semantical account of progression in the presence of uncertainty. In *AAAI*, 2011.

[Belle and Levesque, 2013] V. Belle and H. Levesque. Reasoning about probabilities in dynamic systems using goal regression. In *UAI*, 2013.

[Bonet *et al.*, 2010] B. Bonet, H. Palacios, and H. Geffner. Automatic derivation of finite-state machines for behavior control. In *AAAI*, 2010.

[Boutilier and Poole, 1996] C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *AAAI*, pages 1168–1175, 1996.

[Boutilier *et al.*, 1999] C. Boutilier, T.. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[Claßen and Lakemeyer, 2006] J. Claßen and G. Lakemeyer. Foundations for knowledge-based programs using ES. In *KR*, pages 318–328, 2006.

[Darwiche, 2009] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge Univ. Press, 2009.

[Dean and Kanazawa, 1989] T. Dean and K. Kanazawa. A Model for Reasoning about Persistence and Causation. *Computational Intelligence*, 5:142–150, 1989.

[Fagin *et al.*, 1990] R. Fagin, J. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1/2):78–128, 1990.

[Fagin *et al.*, 1995] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. MIT Press, 1995.

[Gabaldon and Lakemeyer, 2007] A. Gabaldon and G. Lakemeyer. ESP: A logic of only-knowing, noisy sensing and acting. In *AAAI*, pages 974–979, 2007.

[Hajishirzi and Amir, 2010] H. Hajishirzi and E. Amir. Reasoning about deterministic actions with probabilistic prior and application to stochastic filtering. In *KR*, 2010.

[Hamilton *et al.*, 2014] W. Hamilton, M. Milani Fard, and J. Pineau. Efficient learning and planning with compressed predictive states. *JMLR*, 15:3395–3439, 2014.

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

[Herzig *et al.*, 2003] A. Herzig, J. Lang, and P. Marquis. Action representation and partially observable planning using epistemic logic. In *IJCAI*, pages 1067–1072, 2003.

[Iocchi *et al.*, 2009] L. Iocchi, T. Lukasiewicz, D. Nardi, and R. Rosati. Reasoning about actions with sensing under qualitative and probabilistic uncertainty. *ACM Trans. Comput. Log.*, 10(1), 2009.

[Kushmerick *et al.*, 1995] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. *AIJ*, 76:239–286, 1995.

[Lang and Zanuttini, 2012] J. Lang and B. Zanuttini. Knowledge-based programs as plans – the complexity of plan verification. In *ECAI*, pages 504–509, 2012.

[Lang and Zanuttini, 2013] J. Lang and B. Zanuttini. Knowledge-based programs as plans: Succinctness and the complexity of plan existence. In *TARK*, 2013.

[Laverny and Lang, 2005] N. Laverny and J. Lang. From knowledge-based programs to graded belief-based programs, part II: off-line reasoning. In *IJCAI*, pages 497–502, 2005.

[Madani *et al.*, 2003] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *AIJ*, pages 5–34, 2003.

[Mundhenk *et al.*, 2000] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision process problems. *J. ACM*, 47(4):681–720, 2000.

[Poupart, 2005] P. Poupart. *Exploiting Structure to efficiently solve large scale partially observable Markov decision processes*. PhD thesis, University of Toronto, 2005.

[Reiter, 2001] R. Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT press, 2001.

[Rens *et al.*, 2014] G. Rens, T. Meyer, and G. Lakemeyer. SLAP: specification logic of actions with probability. *J. Applied Logic*, 12(2):128–150, 2014.

[Sabbadin *et al.*, 2007] R. Sabbadin, J. Lang, and N. Ravoanjanahary. Purely epistemic Markov decision processes. In *AAAI*, 2007.

[Sanner and Boutilier, 2009] S. Sanner and C. Boutilier. Practical solution techniques for first-order MDPs. *AIJ*, 173(5):748–788, 2009.

[Shani *et al.*, 2008] G. Shani, R. Brafman, S. Shimony, and P. Poupart. Efficient ADD operations for point-based algorithms. In *ICAPS*, pages 330–337, 2008.

[Shani *et al.*, 2013] G. Shani, J. Pineau, and R. Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[Sim *et al.*, 2008] H. Sim, K.-E. Kim, J. Kim, D.-S. Chang, and M.-W. Koo. Symbolic heuristic search value iteration for factored POMDPs. In *AAAI*, pages 1088–1093, 2008.

[Son and Baral, 2001] T. C. Son and C. Baral. Formalizing sensing actions: A transition function based approach. *AIJ*, 125(1-2):19–91, 2001.

[van Eijck and Schwarzentruber, 2014] J. van Eijck and F. Schwarzentruber. Epistemic probability logic simplified. In *Advances in Modal Logic*, pages 158–177, 2014.

[Wang and Schmolze, 2005] C. Wang and J. Schmolze. Planning with POMDPs using a compact, logic-based representation. In *ICTAI*, pages 523–530, 2005.

[Younes *et al.*, 2005] H. Younes, M. Littman, D. Weissman, and J. Asmuth. The first probabilistic track of the international planning competition. *JAIR*, 24:851–887, 2005.