Action representation and partially observable planning using epistemic logic

Andreas Herzig
IRIT/UPS
F-31062 Toulouse Cedex
France
herzig@irit.fr

Jérôme Lang
IRIT/UPS
F-31062 Toulouse Cedex
France
lang@irit.fr

Pierre Marquis
CRIL/Université d'Artois
F-62307 Lens Cedex
France
marquis@cril.univ-artois.fr

Abstract

We propose a purely logical framework for planning in partially observable environments. Knowledge states are expressed in a suitable fragment of the epistemic logic S5. We show how to lift the effects of actions (both physical actions and sensing actions) from the state level to the epistemic level. We show how progression, regression and plan generation can be achieved in our framework.

1 Introduction

Planning under incomplete knowledge and partial observability is a tricky issue in AI, because of its computational (temporal and spatial) hardness. *Partially observable Markov decision processes* (POMDP) – are the mainstream approach to partially observable planning. Nevertheless, the applicability of the POMDP approach is limited from the practical side as soon as the set of states has a strong combinatorial structure, rendering the number of states much too high for an explicit representation of actions, preferences, and policies. On the other hand, *logical approaches to planning* under incomplete knowledge allow for much more compact encodings of planning problems than POMDPs; most of them deal with an incomplete initial state and/or nondeterministic actions, but either they do not handle partial observability, or at least they do it in a very simple way, by assuming for instance that the set of variables is partitioned between (directly) observable and unobservable variables.

To fill the gap between POMDPs and logical approaches, an abstraction of the POMDP model (leaving aside probabilities and expected utility) can be considered. It should account at least for the following elements: a set $\mathcal{S}$ of states; a set $\mathcal{B}$ of belief states built from $\mathcal{S}$; a set $\mathcal{A}$ of actions; each action is associated with a transition model between states and/or belief states; some preference structure (e.g., a simple goal or a utility function); and a set $\mathcal{O}$ of observations, together with some correlation function between states and observations.

While policies for a fully observable MDP map states to actions, the output of such an abstract POMDP is a policy mapping *belief states* to actions; indeed, a POMDP can be viewed as a fully observable MDP over the set of belief states (this is a classical way of understanding POMDPs – and even to solve them).

In this paper, we present a rich *logical* framework that instantiates the abstraction above, views a partially observable process as a fully observable process over belief states, and allows for expressing actions and policies in a compact way. The framework has a fairly good level of generality (since it avoids for instance to commit to a particular action language, see Section 3.1) and is therefore modular enough to be easily adapted or extended.

The simplest and best-known way of making explicit the distinction between facts and beliefs in logic consists in expressing the problem in an epistemic or doxastic logic. To make the exposition simpler, we assume that the agent has *accurate beliefs*, i.e., all she believes is true in the actual world. This means that we identify belief and knowledge (since knowledge is usually viewed as true belief); therefore our framework is based on the logic S5 and instead of *belief* we use the term *knowledge* throughout the paper[1]. S5 is computationally no more complex than classical logic: S5 satisfiability is NP-complete [Ladner, 1977].

In Section 2 we define two notions of knowledge states: simple knowledge states (for on-line plan execution) and complex knowledge states (for off-line reasoning about the effects of a plan). In Section 3 we show how a knowledge state evolves after an action is performed. Then we show in Section 4 how to perform goal regression, and we show in Section 5 how it can be used so as to implement a sound and complete plan generator. Section 6 discusses related work.

2 Knowledge states

The language $\mathcal{L}_K$ of propositional logic S5 is built up from a finite set of propositional variables VAR, the usual connectives, the logical constants $\top$ and $\bot$ and the epistemic modality **K**. S5 formulas are denoted by capital Greek letters Φ, Ψ etc. An S5 formula is *objective* (or modality-free) iff it does not contain any occurrence of **K** (i.e., it is a classical propositional formula). Objective formulas are denoted by small Greek letters φ , ψ etc.

[1]Alternatively, we could have chosen to work with *beliefs*, using the doxastic logic KD45, which is very similar to S5 except that beliefs may be wrong, that is, $\mathbf{K}\varphi \rightarrow \varphi$ is not an axiom. The technical issues developed in this paper would have been almost identical. Now, choosing another logic than S5 or KD45 would induce a lot of complications, including an important complexity gap.

A fundamental property of S5 is that nested modalities collapse, i.e., $\mathbf{KK}\Phi$ is equivalent to $\mathbf{K}\Phi$ and $\mathbf{K}\neg\mathbf{K}\Phi$ to $\neg\mathbf{K}\Phi$; for this reason, we assume without loss of generality that the scope of each occurrence of modality $\mathbf{K}$ in formula Φ is an objective formula.

An *epistemic atom* is an S5 formula of the form $\mathbf{K}\varphi$, where φ is objective. An *epistemic formula* is a formula built up from epistemic atoms and usual connectives: $\mathbf{K}(a \vee b) \vee \neg\mathbf{K}(c \wedge \neg d)$ is an epistemic formula, while $a \wedge \mathbf{K}b$ is not. An epistemic formula is *positive* iff it does not contain any occurrence of $\mathbf{K}$ in the scope of a negation: $\mathbf{K}(a \vee b) \vee \neg\mathbf{K}(c \wedge \neg d)$ is not positive, while $\mathbf{K}(a \vee b) \vee \mathbf{K}(c \wedge \neg d)$ is.

$\mathcal{S} = 2^{VAR}$ is the set of all interpretations of $\mathcal{L}$, also called *states*. States are denoted by s, s' etc. If φ is an objective formula, we let $Mod(\varphi) = \{s \in \mathcal{S} | s \text{ is a model of } \varphi\}$.

A structure for S5 is defined as a nonempty set of states $M \subseteq \mathcal{S}$.[2] Rather than "structure", we call M a *knowledge state* (SKS). Intuitively, it represents all the states the agent considers possible. The satisfaction of an S5 formula by an SKS M for a state $s \in M$ is defined inductively by:

- for φ objective, $(M, s) \models \varphi$ iff $s \models \varphi$;
- for φ objective, $(M, s) \models \mathbf{K}\varphi$ iff $\forall s' \in M$ we have $(M, s') \models \varphi$;
- $(M, s) \models \Phi \wedge \Psi$ iff $(M, s) \models \Phi$ and $(M, s) \models \Psi$;
- $(M, s) \models \Phi \vee \Psi$ iff $(M, s) \models \Phi$ or $(M, s) \models \Psi$;
- $(M, s) \models \neg\Phi$ iff $(M, s) \not\models \Phi$.

Finally, an SKS M satisfies Φ, denoted by $M \models \Phi$, iff for all $s \in M$ we have $(M, s) \models \Phi$. An S5 formula is valid (resp. satisfiable) iff it is satisfied by all SKSs (resp. by at least one SKS). An S5 formula Ψ is a consequence of an S5 formula Φ, noted $\Phi \models \Psi$ iff every SKS that satisfies Φ also satisfies Ψ. Φ and Ψ are equivalent, noted $\Phi \equiv \Psi$ iff $\Phi \models \Psi$ and $\Psi \models \Phi$ holds. Note that $\mathbf{K}(\varphi \wedge \psi)$ is equivalent to $\mathbf{K}\varphi \wedge \mathbf{K}\psi$ but $\mathbf{K}(\varphi \vee \psi)$ is implied by but generally not equivalent to $\mathbf{K}\varphi \vee \mathbf{K}\psi$.

Importantly, *an SKS can be identified with the strongest epistemic atom* $\mathbf{K}\varphi$ *that is satisfied by* M. Such a $\mathbf{K}\varphi$ is unique up to equivalence. Therefore, SKSs will be denoted syntactically, and without ambiguity, by epistemic atoms $\mathbf{K}\varphi$.

A *complex knowledge state* (CKS) is a positive epistemic formula[3], generated by epistemic atoms and the connectives $\wedge$ and $\vee$. For instance, $[\mathbf{K}(a \vee b) \wedge \mathbf{K}\neg c] \vee \mathbf{K}(b \leftrightarrow c)$ is a CKS but $\mathbf{K}(a \vee b) \rightarrow \mathbf{K}c$ is not, $\neg\mathbf{K}a$ neither.

[2] This semantics is equivalent (and simpler for our purpose) to the usual semantics by means of Kripke models $\langle W, val, R\rangle$ where W is a set of worlds, val a valuation function and R an equivalence relation. See for instance [Fagin *et al.*, 1995].

[3] We restrict the syntax of CKS to positive epistemic formulas because for almost all problems, ignorance can already be expressed by the fact that positive knowledge is not provable from the current CKS. This way of generating explicit ignorance from implicit is a kind of Epistemic Closed World Assumption, already at work in [Reiter, 2001] and reminiscent of autoepistemic logic; its principle can be roughly be stated as "if I cannot prove that I know φ then I don't know φ". However, for the purpose of this paper, we do not need this completion because the set of valid plans from a CKS and the set of valid plans from its completion coincide.

Example 1 *In a model-based diagnosis context, let* $\Sigma = \bigwedge_{c \in COMP} \mathbf{K}(\neg ab(c) \leftrightarrow \varphi_c)$ *the formula representing the behaviour of a set of components* $COMP$. *Suppose that the agent is performing a test plan so as to identify the faulty components. The formula characterizing the condition for stopping performing tests and starting reparing the faulty components is* $\Phi = \bigwedge_{c \in COMP}(\mathbf{K}ab(c) \vee \mathbf{K}\neg ab(c))$. *The latter formula is a CKS. It is equivalent to an exponentially large disjunction of SKS:* $\Phi \equiv \bigvee\{\mathbf{K}\delta_I \mid I \subseteq COMP\}$ *where* $\delta_I = (\bigwedge_{c \in I} ab(c)) \bigwedge (\bigwedge_{c \in COMP \setminus I} \neg ab(c))$.

A CKS Φ is said to be in *epistemic disjunctive normal form* (EDNF) iff it has the form $\mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n$, where each φ_i is an objective formula. Clearly, every CKS has an EDNF equivalent.

3 Actions and progression

In general, actions have both physical (or *ontic*) and epistemic effects, i.e., they are meant to change both the state of the world and the agent's knowledge state, but without loss of generality we assume (as usually in AI) that any "mixed" action can be decomposed into two actions, the first one having only ontic effects and the second one only epistemic effects. For instance, the action of tossing a coin is decomposed into a `blind-toss` action followed by a `see` action telling the agent whether the coin has landed on `heads` or on `tails`.

3.1 Ontic actions

Ontic actions are meant to have effects on the outside world, especially *physical* effects such as moving a block, switching the light, moving etc. They are assumed to be described in a *propositional action language* (allowing or not for nondeterminism, for ramifications/causality). Any action language $\mathcal{L}_\mathcal{A}$ can be chosen, provided that it is propositional and that it expresses the effects of an action α within a formula Σ_α involving atoms labelled by t and atoms labelled by $t+1$ (the former for the state before the action is performed, the latter for the state after it is performed). Among candidate languages we find those of the $\mathcal{A}$ family [Gelfond and Lifschitz, 1993], "propositionalized" situation calculus [Lin, 2000] and causality languages [McCain and Turner, 1998][4].

The description Σ_α of an action α allows for computing the successor state of a state s (or the set of successor states if α is nondeterministic). This set is represented by any objective formula $prog(s, \alpha)$ (whose models form the set of successor states). This definition extends to sets of initial states, or equivalently to propositional formulas, by $prog(\varphi, \alpha) = \bigvee_{s \in Mod(\varphi)} prog(s, \alpha)$.

Now, given the description Σ_α of action α and a CKS Φ, the subsequent CKS, denoted by $Prog(\Phi, \alpha)$, is determined by the following definition:

Definition 1 (progression for ontic actions) *Let* α *be an ontic action and* $\Phi = \mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n$ *be a CKS in EDNF.*

$$Prog(\Phi, \alpha) = \mathbf{K}prog(\varphi_1, \alpha) \vee \ldots \vee \mathbf{K}prog(\varphi_n, \alpha).$$

[4] The common core of these languages is the use of explicit or implicit *successor state axioms*. These languages coincide for deterministic, ramification-free actions and differ in the way they treat nondeterminism (using for instance exogeneous variables or `Release` statements), ramifications, etc.

The problem with the latter definition is that the CKS has to be put in EDNF first. This does not induce any loss of expressivity but the transformation may be exponentially large so that we may want to compute the successor CKS directly from a CKS expressed in any form, such as in Example 1.

We now give a more elegant way of computing progression via an extension of *variable forgetting* to S5 formulas. We recall first from [Lin and Reiter, 1994] the inductive definition of $forget(X, \varphi)$ where φ is objective and $X \subseteq VAR$:
(0) $forget(\emptyset, \varphi) = \varphi$;
(1) $forget(\{x\}, \varphi) = \varphi_{x \leftarrow \perp} \vee \varphi_{x \leftarrow \top}$;
(2) $forget(\{X \cup \{x\}, \varphi) = forget(\{x\}, forget(X, \varphi))$ [5].

Definition 2 (forgetting) *Let Φ be an S5 formula and $X \subseteq VAR$. Forget(X, Φ) is the strongest S5 formula (a) entailed by Φ and (b) not mentioning any variable of X.*

Forget$(.,.)$ is well-defined, because the set of formulas satisfying (a) and (b) is closed by conjunction. Thus *Forget*(X, Φ) is unique up to equivalence in S5.

Proposition 1

- *if Φ is a CKS, then Forget(X, Φ) is a CKS.*
- *Forget*$(X, \mathbf{K}\varphi) \equiv \mathbf{K}forget(X, \varphi)$.
- *Forget*$(X, \Phi_1 \vee \Phi_2) \equiv$ *Forget*$(X, \Phi_1) \vee$ *Forget*(X, Φ_2).
- *if Φ_1 and Φ_2 do not share any variable then Forget$(X, \Phi_1 \wedge \Phi_2) \equiv$ Forget$(X, \Phi_1) \wedge$ Forget(X, Φ_2).*

The interest of defining this forgetting operator is made clear by the following result. From every symbol $x \in VAR$ we make two copies x_t, x_{t+1}, where t (resp. $t+1$) represents the time point *before* (resp. *after*) the action is performed; then, for any formula $\varphi \in \mathcal{L}_{VAR}$, φ_t (resp. φ_{t+1}) is the formula from φ obtained by replacing each variable x by x_t (resp. x_{t+1}).

Proposition 2 *Let α be an ontic action and let Φ be a CKS. We have:*

$$Prog(\Phi, \alpha)_{t+1} \equiv \mathit{Forget}(VAR_t, \Phi_t \wedge \mathbf{K}\Sigma_\alpha)$$

where $VAR_t = \{x_t \mid x \in VAR\}$.

3.2 Epistemic actions

An epistemic action β is described wlog by the finite set of its possible outcomes $\{o_1, \ldots, o_p\}$, where each o_i is an objective formula describing a possible observation, and $o_1 \vee \ldots \vee o_p$ is a tautology[6]. Accordingly, we have $\mathcal{O} = \mathcal{L}_{VAR}$ in our framework. The effects of β are induced by the following principles: *non-intrusiveness* (for any objective formula φ, if φ holds before β is performed, then it still holds after), *no-forgetting* (if the agent knows an objective formula φ before performing β, then it still knows it after) and *reliability* (if the agent observes o_j after performing β, then o_j holds, so that it knows o_j after observing it). These three properties imply that the effects of an epistemic action associated with the outcome set $\{o_1, \ldots, o_p\}$ can be represented by the following progression operator:

[5]In (1), $\varphi_{x \leftarrow \top}$ (resp. $\varphi_{x \leftarrow \perp}$) represents the formula obtained by replacing in φ each occurrence of variable x by $\top$ (resp. $\perp$).

[6]Or, more generally, $\Sigma \models o_1 \vee \ldots \vee o_p$, where Σ is the given set of static domain rules.

Definition 3 (progression for epistemic actions) *Let β be an epistemic action associated with the set of outcomes $\{o_1, \ldots, o_p\}$, and let Φ be a CKS. We have:*

$$Prog(\Phi, \beta) = \Phi \wedge (\mathbf{K}o_1 \vee \ldots \vee \mathbf{K}o_p).$$

It follows that for any epistemic actions β, β_1, β_2, progression is indifferent to the order of epistemic actions: $Prog(Prog(\Phi, \beta_1), \beta_2) \equiv Prog(Prog(\Phi, \beta_2), \beta_1)$, that it is idempotent: $Prog(Prog(\Phi, \beta), \beta) \equiv Prog(\Phi, \beta)$. and preserves positive knowledge: $Prog(\Phi, \beta) \models \Phi$.

Epistemic actions can be assumed fully executable without loss of generality (adding new symbols if necessary)[7].

A simple and well-known class of epistemic actions is the class of *binary tests*, also called *truth tests*: if φ is an objective formula, the action $test(\varphi)$ is defined by the outcome set $\{\varphi, \neg\varphi\}$. Therefore we have $Prog(\Phi, test(\varphi)) = \Phi \wedge (\mathbf{K}\varphi \vee \mathbf{K}\neg\varphi)$. In the rest of the paper, we focus on truth tests, which will make things simpler to expose[8].

Note that a straightforward (but important) consequence of Definitions 1 and 3 is that the set of CKS is closed under progression, that is, if Φ is a CKS and α an ontic or epistemic action, then $Prog(\Phi, \alpha)$ is a CKS.

4 Regression

The problem is stated as follows: given a CKS Φ (representing a goal knowledge state) and an (ontic or epistemic) action α, characterize the weakest CKS denoted by $Reg(\Phi, \alpha)$, in which performing α leads to a CKS satisfying Φ.

Definition 4 (regression) *Let α be an (ontic or epistemic) action and let Φ be a CKS. The* regression *of Φ by α, is the unique CKS (up to logical equivalence), denoted by $Reg(\Phi, \alpha)$, such that (a) $Prog(Reg(\Phi, \alpha), \alpha) \models \Phi$ and (b) for any Φ' such that $Prog(\Phi', \alpha) \models \Phi$ we have $\Phi' \models Reg(\Phi, \alpha)$.*

This definition is well-founded: indeed, if $Prog(\Phi_1, \alpha) \models \Phi$ then for any $\Phi_2 \models \Phi_1$ we have $Prog(\Phi_2, \alpha) \models \Phi$ (this is easily verified both for ontic actions and for epistemic actions). Therefore the set $\{\Psi \mid Prog(\Psi, \alpha) \models \Phi\}$ is closed w.r.t. disjunction, which entails that there exists a weakest Ψ such that $Prog(\Psi, \alpha) \models \Phi$, equivalent to $\bigvee\{\Xi \mid Prog(\Xi, \alpha) \models \Phi\}$.

4.1 Regression for ontic actions

For any objective formula φ, let $reg(\varphi, \alpha)$ be the standard (non-epistemic) regression of φ by α, i.e. any objective formula whose set of models consists of all states $s \in \mathcal{A}$ s.t. $Mod(prog(s, \alpha)) \subseteq Mod(\varphi)$.

Definition 5 (sufficient success conditions) *A consistent conjunction of literals γ is a* sufficient success condition *for*

[7]Note that actions that may fail to be informative can be expressed by the occurrence of a tautology in their outcomes. In this case, unfortunately, we get that $Prog(\Phi, \beta) \equiv \Phi$, because $\mathbf{K}o_1 \vee \ldots \vee \mathbf{K}o_p \equiv \top$ as soon as $o_i \equiv \top$ for some i.

[8]This does not induce any loss of generality, because any epistemic action can be rewritten equivalently into a logarithmic sequence of binary tests, together with the addition of some domain constraints.

α and φ if and only if for any pair of states $(s, s') \in \mathcal{S} \times \mathcal{S}$ such that s is a model of γ and s' is a model of $prog(s, \alpha)$, s' is a model of φ. γ is a minimal sufficient success condition *for α and φ if and only if it is a sufficient success condition for α and φ and no proper subconjunction of γ is. We denote by $\Gamma(\alpha, \varphi)$ the disjunction of all minimal sufficient success conditions for α and φ.*

When actions are described using an action language satisfying the properties stated in Section 3.1, such conditions can be computed abductively. Clearly, γ is a sufficient success condition for α and φ if and only if $\gamma_t \wedge \Sigma_\alpha$ is satisfiable, and every model of $\gamma_t \wedge \Sigma_\alpha$ is a model of φ_{t+1}.

Proposition 3 *Let Σ_α be the description of α in a propositional action language $\mathcal{L}_\mathcal{A}$. $reg(\varphi, \alpha)$ is equivalent to*

$$\Gamma(\alpha, \varphi) = \bigvee\{\gamma \mid \gamma_t \in PI^{VAR_t}_{\Sigma_\alpha}(\varphi_{t+1})\}$$

In this proposition, $PI^{VAR_t}_{\Sigma_\alpha}((\varphi)_{t+1})$ denotes the set of *prime implicants* of the formula $\Sigma_\alpha \rightarrow (\varphi)_{t+1}$ that are consistent with Σ_α and built up from variables of VAR_t, only.

The latter abductive characterization of ontic actions is independent of the action language chosen – and it now allows for characterizing the regression of complex epistemic states by an ontic action.

Proposition 4 *Let α be an ontic action, Φ a CKS and let $\mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n$ be an EDNF formula equivalent to Φ. Then*

$$Reg(\Phi, \alpha) \equiv \mathbf{K}reg(\varphi_1, \alpha) \vee \ldots \vee \mathbf{K}reg(\varphi_n, \alpha).$$

4.2 Regression for epistemic actions

Proposition 5 *Let β be an epistemic action whose outcome set is $\{o_1, \ldots, o_p\}$ and let Φ be a CKS. Let $\mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n$ be any EDNF formula equivalent to Φ. Then*

$$Reg(\Phi, \beta) \equiv \bigvee\{\mathbf{K}((o_1 \rightarrow \varphi_{f(1)}) \wedge \ldots \wedge (o_p \rightarrow \varphi_{f(p)})) | f \in \{1..n\}^{\{1..p\}}\}$$

In particular, if $\beta = test(\psi)$ is a binary test then

$$Reg(\Phi, \beta) \equiv \bigvee\{\mathbf{K}((\psi \rightarrow \varphi_i) \wedge (\neg\psi \rightarrow \neg\varphi_j)) \mid i, j \in 1 \ldots n\}$$

Sketch of proof. We give it in the latter case only (the proof for the general case is similar). We abbreviate $Reg(\Phi, \beta)$ by Δ. We first establish (the proof is omitted) that Δ is necessarily a CKS, i.e., $\Delta = \mathbf{K}\delta_1 \vee \ldots \vee \mathbf{K}\delta_q$. Now,

$\models \Delta \wedge (\mathbf{K}\psi \vee \mathbf{K}\neg\psi) \rightarrow \Phi$

iff $\forall k \in 1..q \left\{ \begin{array}{l} \models \mathbf{K}(\delta_k \wedge \psi) \rightarrow \mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n \\ \models \mathbf{K}(\delta_k \wedge \neg\psi) \rightarrow \mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n \end{array} \right. \quad (*)$

At that stage, we make use of the following lemma: for any objective formulas A, B and C, $\mathbf{K}A \rightarrow (\mathbf{K}B \vee \mathbf{K}C)$ is valid (in S5) iff $A \rightarrow B$ is valid or $A \rightarrow C$ is valid. (*) is then equivalent to

$\forall k \in 1..q, \exists i, j \in 1..n$ s.t. $\delta_k \wedge \psi \rightarrow \varphi_i$ and $\delta_k \wedge \neg\psi \rightarrow \varphi_j$ are classically valid, which is equivalent to: $\forall k \in 1..q, \exists i, j \in 1..n$ s.t. $\delta_k \rightarrow ((\psi \rightarrow \varphi_i) \wedge (\neg\psi \rightarrow \varphi_j))$ is valid. The rest of the proof is easy.

Example 2 *Consider the epistemic actions $\alpha = test(u \wedge v)$ and $\beta = test(u \leftrightarrow v)$, and the ontic action γ that switches the value of u, thus described by $(u_{t+1} \leftrightarrow \neg u_t) \wedge (v_{t+1} \leftrightarrow v_t)$. Let $\Phi = \mathbf{K}v \vee \mathbf{K}\neg v$. Applying Proposition 5 we get $Reg(\Phi, \alpha) \equiv \mathbf{K}v \vee \mathbf{K}\neg v \vee \mathbf{K}(v \rightarrow u) \vee \mathbf{K}(v \wedge \neg u) \equiv \mathbf{K}v \vee \mathbf{K}(v \rightarrow u)$; $Reg(\Phi, \beta) \equiv \mathbf{K}v \vee \mathbf{K}\neg v \vee \mathbf{K}u \vee \mathbf{K}\neg u$; $Reg(Reg(\Phi, \alpha), \beta) \equiv \mathbf{K}(u \rightarrow v) \vee \mathbf{K}(v \rightarrow u)$. Applying Proposition 3 we get $Reg(\Phi, \gamma) \equiv \Phi$, $Reg(Reg(\Phi, \alpha), \gamma) \equiv \mathbf{K}v \vee \mathbf{K}(v \rightarrow \neg u)$; $Reg(Reg(\Phi, \beta), \gamma) \equiv Reg(\Phi, \beta)$; $Reg(Reg(Reg(\Phi, \alpha), \beta), \gamma) \equiv \mathbf{K}(u \vee v) \vee \mathbf{K}(\neg u \vee \neg v)$.*

5 Plan generation

Definition 6 (planning problems) *A planning problem $\mathcal{P}$ w.r.t. an action language $\mathcal{L}_A$ consists of an SKS $\Phi = \mathbf{K}\varphi_{init}$ describing the initial knowledge state of the agent, a finite set of actions $\mathcal{A} = \mathcal{A}_O \cup \mathcal{A}_E$ and a CKS Γ describing the goal. Effects of ontic actions are described in a propositional action language.*

The reason why Γ is a CKS is that it is not sufficient to reach the goal, it must also be the case that the goal is *known* to be reached. Γ may be purely epistemic goal such as $\mathbf{K}\varphi \vee \mathbf{K}\neg\varphi$, i.e., an agent may have the ultimate goal to know whether φ holds or not.

Plans are defined inductively as follows:

- the empty plan λ is a plan;
- any action (ontic or epistemic) is a plan;
- if π and π' are plans then $\pi; \pi'$ is a plan;
- if π and π' are plans and Φ a CKS, then $\texttt{if}\,\Phi\,\texttt{then}\,\pi\,\texttt{else}\,\pi'$ is a plan.

Therefore, a plan can be seen as a program without loops, whose branching conditions are *epistemic formulas*: the agent can decide whether she *knows* that a given objective formula is true (whereas she is not always able to decide whether a given objective formula is true in the actual world).

While CKS are relevant for off-line planning, i.e., for *reasoning about the possible effects of a plan*, they are no longer relevant for representing knowledge during plan execution, since at each time step the agent is in exactly one knowledge state.

Definition 7 (progression of a CKS by a plan) *The progression of a CKS by a plan is defined inductively by*

1. *$Prog(\Phi, \lambda) = \Phi$;*
2. *$Prog(\Phi, (\pi; \pi')) = Prog(Prog(\Phi, \pi), \pi')$;*
3. *if Φ is a SKS, i.e., $\Phi = \mathbf{K}\varphi$, then* $Prog(\Phi, \texttt{if}\,\Psi\,\texttt{then}\,\pi\,\texttt{else}\,\pi')$ $= \left\{ \begin{array}{ll} Prog(\Phi, \pi) & \textit{if } \Phi \models \Psi; \\ Prog(\Phi, \pi') & \textit{if } \Phi \not\models \Psi. \end{array} \right.$
4. *if Φ is not a SKS, i.e., $\Phi = \mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n$ for $n \geq 2$, then $Prog(\mathbf{K}\varphi_1 \vee \ldots \vee \mathbf{K}\varphi_n, \texttt{if}\,\Psi\,\texttt{then}\,\pi\,\texttt{else}\,\pi') = \bigvee_{i=1\ldots n} Prog(\mathbf{K}\varphi_i, \texttt{if}\,\Psi\,\texttt{then}\,\pi\,\texttt{else}\,\pi')$*

Definition 8 (valid plans) *A plan π is a* valid plan *for the planning problem $\mathcal{P}$ if and only if $Prog(\Phi, \pi) \models \Gamma$.*

Exemple 2 (cont'd) *Initially, the agent does not know the values of u and v, and her goal is to reach a belief state where she knows the value of v: $\Gamma = \mathbf{K}v \vee \mathbf{K}\neg v$.*

Let $\pi = \beta; \texttt{if}\,\mathbf{K}(u \leftrightarrow v)\,\texttt{then}\,\alpha\,\texttt{else}\,(\gamma; \alpha)$.

$Prog(\mathbf{K}\top, \pi)$

$= Prog(Prog(\mathbf{K}\top, \beta), \texttt{if}\,\mathbf{K}(u \leftrightarrow v)\,\texttt{then}\,\alpha\,\texttt{else}\,(\gamma; \alpha))$

$= Prog(\mathbf{K}(u \leftrightarrow v), \alpha) \vee Prog(\mathbf{K}(u \leftrightarrow \neg v), (\gamma; \alpha))$
$= \mathbf{K}(u \wedge v) \vee \mathbf{K}(\neg u \wedge \neg v)$
Therefore, π is valid.

A valid plan can be computed by the following backward algorithm based on goal regression which is reminiscent of dynamic programming. The current goal Φ_C, expressed in EDNF, is initialized as Γ. Then we nondeterministically pick an action α and compute $Reg(\Phi_C, \alpha)$. The current goal is then updated by $\Phi_C := \Phi_C \vee Reg(\Phi_C, \alpha)$. The process is iterated until $\mathbf{K}\varphi_{init} \models \Phi_C$ or it is not possible to improve Φ_C anymore. Since there is a finite number of possible belief states, the algorithms stop *and returns a valid plan, if such a plan exists*. An *ordered list* L is constantly updated, initialized by $\langle\langle \mathbf{K}G_1, \lambda\rangle, \ldots, \langle \mathbf{K}G_m, \lambda\rangle\rangle$ where $\Gamma \equiv \mathbf{K}G_1 \vee \ldots \vee \mathbf{K}G_m$; each time a new disjunct (i.e., not subsumed by any previous disjunct of Φ_C) $\mathbf{K}\psi$ is added to Φ_C after regressing by action α, the pair $\langle \mathbf{K}\psi, \alpha\rangle$ is added to L.

There are two slightly different possible outputs: (1) either the output is just L, i.e., an *ordered knowledge-based program* (or decision list): at execution time, when observations are made, the new knowledge state is computed, then we look for the leftmost $\langle \mathbf{K}\psi, \alpha\rangle$ in L such that $\mathbf{K}\psi$ is true in the current knowledge state and α is performed; (2) or the output is a *ready-to-use conditional plan* computed by "simulating" possible executions from $\mathbf{K}\varphi_{init}$ to $\mathbf{K}G$ using L.

Exemple 2 (cont'd) *Regressing by α, γ and then β, the algorithm finds a valid knowledge-based program for Γ:*
$L = \langle\langle \mathbf{K}v, \lambda\rangle, \langle \mathbf{K}\neg v, \lambda\rangle, \langle \mathbf{K}(v \rightarrow u), \alpha\rangle, \langle \mathbf{K}(v \rightarrow \neg u), \gamma\rangle, \langle \mathbf{K}\top, \beta\rangle\rangle$. *The associated conditional plan is* $\pi = \beta; \texttt{if}\, \mathbf{K}(u \leftrightarrow v)\, \texttt{then}\, \alpha\, \texttt{else}\, (\gamma; \alpha)$.

6 Related work

Knowledge-based programs

In the planning community, the idea of using explicit knowledge preconditions for actions and plans comes back to [Moore, 1985; Morgenstern, 1987]. Developed in a different perspective (agent design), *knowledge-based programs* [Fagin *et al.*, 1995; Brafman *et al.*, 1998; Herzig *et al.*, 2000; Reiter, 2001] are high-level protocols that describe the actions an agent should perform as a function *of its knowledge*. Thus, in a knowledge-based program, branching conditions are epistemically interpretable, and plans explicitly involve deduction tasks during on-line execution (just like in our framework). Actually, the output of our plan generation process *is a knowledge-based program*. Therefore, our work can be seen as an upstream task that *generates* a valid knowledge-based program from a compact specification of action effects and goals.

Action languages

A number of works have extended action languages so as to handle explcit knowledge and partial observability, especially [Lobo *et al.*, 1997; de Giacomo and Rosati, 1999; Baral and Son, 2001]. Knowledge is represented in all cases by an explicit or implicit epistemic modality (plus a "minimal knowledge" semantics in [de Giacomo and Rosati, 1999]). The line of work most related to ours is of [Baral and Son, 2001]; indeed, not only they represent epistemic actions with an epistemic modality but they also allow for conditional plans with epistemic branching conditions. Our work can be seen as an extension of theirs[9]: (i) our formalism is general enough to accept any propositional action language (including those handling causal rules) for representing the effects of ontic actions); (ii) our syntax is less restricted, since we allow for any and-or combination of SKS (i.e., CKSs) while they consider SKS only; as argued in Section 3, this makes the representation more compact, when reasoning at planning time about the future consequences of actions; (iii) our progression and regression operators have significant computational characterizations (e.g., ontic regression has an abductive characterization); lastly, we have a sound and complete algorithm for plan generation.

Planning under partial observability

There is a number of recent approaches for logic-based plan generation under partial observability.

[Bonet and Geffner, 1998] gives a high-level language for describing action effects on both the world and the agent's beliefs. Their language is a decidable fragment of first-order logic. By describing ontic actions with successor state axioms, they allow for handling the frame problem and ramification problems. After a problem has been represented in their language, its description is automatically translated into a POMDP model and solved by using POMDP algorithms, so that there is no need to define progression and regression directly in the logic, nor to have an explicit knowledge modality: this is the main difference with our approach, where the compact logical representation is kept and propagated throughout the process.

The next three approaches solve the plan generation problem directly in a high-level language but, on the other hand, they all make important restrictions that lead to a loss of expressivity. These restrictions imply that none of these approaches makes use of action languages, while ours can benefit from the huge amount of work in this area and accordingly, can handle the frame problem as well as ramification and causality in the best possible way while maintaining computational complexity at a reasonable level.

[Bacchus and Petrick, 1998; Petrick and Bacchus, 2002], like us, use an epistemic modality. Apart from the representation of ontic actions (less expressive than ours due to the abovementioned point[10]), they restrict the syntax of epistemic formulas (for instance, simple disjunctive beliefs such as $\mathbf{K}(a \vee b)$ cannot be expressed) and consequently, as they notice, their algorithm sometimes fail to discover a valid plan.

The approaches [Bertoli *et al.*, 2001; Rintanen, 2002] do not make use of an epistemic modality, and therefore cannot

[9] Actually, only of the first part of [Baral and Son, 2001], since the second half of their paper gives a detailed study of sound and efficient approximations of their formalism. We plan to integrate similar approximations in our framework.

[10] On the other hand, they use a fragment of first-order logic which allows for expressing some actions (such as value tests) elegantly, and they motivate their expressivity restrictions by efficiency considerations, so that their approach is a good trade-off between efficiency and expressivity.

explicitly express disjunctions of belief states (i.e., CKSs) or complex knowledge-based programs. The representation of belief states in both approaches uses BDDs, which allow for a compact representation but not as efficient as propositional formulas[11]. While the algorithm in [Bertoli *et al.*, 2001] uses progression (based on model checking), [Rintanen, 2002] has a regression operator, and, interestingly enough, his combination operator $\otimes$ between belief states (which aims at computing, given two belief states $B1$ and $B2$, the maximal belief states in which, after observing the values of observable variables, leads to know that the true state is in $B1$ or to know that the true state is in $B2$) can be reformulated using our epistemic regression (Section 4.2) [12] and thus epistemic logic helps understanding how and why this operator works [13].

Situation calculus

[Scherl and Levesque, 1993] represent sensing actions in the situation calculus by means of an explicit accessibility relation between situations (which means that knowledge is treated as an ordinary fluent) which corresponds exactly to the semantics of our epistemic modality (once situations have been identified with states). Our approach expresses the problem at the formula level and enables thus a more concise representation and can benefit from existing complexity and automated deduction results for $\mathsf{S5}$. Levesque [Levesque, 1996] builts on the above framework towards a general theory of planning with sensing, handling complex plans involving, like ours, nondeterminism, observations and branching (and also loops).

Acknowledgements

The third author has been partly supported by the IUT de Lens, the Université d'Artois, the Région Nord / Pas-de-Calais under the TACT-TIC project, and by the European Community FEDER Program.

References

[Bacchus and Petrick, 1998] F. Bacchus and R. Petrick. Modelling an agent's incomplete knowledge during planning and execution. In *KR-98*, pages 432–443, 1998.

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

[Bertoli *et al.*, 2001] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in nondeterministic domains under partial observability via symbolic model checking. In *IJCAI-01*, pages 473–478, 2001.

[Bonet and Geffner, 1998] B. Bonet and H. Geffner. High-level planning and control with incomplete information using POMDPs. In *AAAI Fall Symp. om POMDPs*, 1998.

[Brafman *et al.*, 1998] R. Brafman, J. Halpern, and Y. Shoham. On the knowledge requirements of tasks. *Artificial Intelligence*, 98, 1998.

[de Giacomo and Rosati, 1999] G. de Giacomo and R. Rosati. Minimal knowledge approach to reasoning about actions and sensing. *Electronic Transactions on Artificial Intelligence*, 3 (section C):1–18, 1999.

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

[Gelfond and Lifschitz, 1993] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[Herzig *et al.*, 2000] A. Herzig, J. Lang, D. Longin, and Th. Polacsek. A logic for planning under partial observability. In *AAAI-00*, pages 768–773, 2000.

[Ladner, 1977] R.E. Ladner. The computational complexity of provability in systems of modal propositional logics. *SIAM J. Comput.*, 6:467–480, 1977.

[Levesque, 1996] H. Levesque. What is planning in the presence of sensing? In *AAAI-96*, pages 1139–1146, 1996.

[Lin and Reiter, 1994] F. Lin and R. Reiter. Forget it! In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 154–159, New Orleans, 1994.

[Lin, 2000] F. Lin. From causal theories to successor state axioms and STRIPS-like systems. In *AAAI-00*, pages 786–791, 2000.

[Lobo *et al.*, 1997] J. Lobo, G. Mendez, and S. R. Taylor. Adding knowledge to the action description language A. In *AAAI-97*, pages 454–459, 1997.

[McCain and Turner, 1998] N. McCain and H. Turner. Satisfiability planning with causal theories. In *KR-98*, pages 212–223, 1998.

[Moore, 1985] R.C. Moore. *A formal theory of knowledge and action*, chapter Formal Theories of the Commonsense World. Ablex, 1985.

[Morgenstern, 1987] L. Morgenstern. Knowledge preconditions for actions and plans. In *IJCAI-87*, pages 867–874, 1987.

[Petrick and Bacchus, 2002] R. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. AIPS-02*, pages 212–221, 2002.

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[Rintanen, 2002] J. Rintanen. Backward plan construction for planning with partial observability. In *Proc. AIPS-02*, pages 173–182, 2002.

[Scherl and Levesque, 1993] R. B. Scherl and H. J. Levesque. The frame problem and knowledge-producing actions. In *AAAI-93*, pages 698–695, 1993.

[11] A (RO)BDD can be translated into a DAG propositional formula without exponential increase in size, while the converse is not true in general.

[12] Indeed, introducing the action *observe* performed systematically after any ontic action (Rintanen does not consider real epistemic actions but assumes instead *automatic observability*: there is a set of variables whose value is observed after *any* action) and giving the truth value of each observable variable o_i, $B_1 \otimes B_2$ can be identified with $Reg(observe, \mathbf{K}for(B_1) \vee \mathbf{K}for(B_2))$ where $for(B)$ is a propositional formula such that $Mod(for(B)) = B$. More precisely, if $B \in B_1 \otimes B_2$ then $\mathbf{K}for(B)$ is one of the disjuncts of $Reg(observe, \mathbf{K}for(B_1) \vee \mathbf{K}for(B_2))$ after minimization.

[13] Note also that all approaches discussed in this section pay attention to efficiency (which may explain the restrictions made) and all of them have been implemented and report experimental results.