A Complete Epistemic Planner without the Epistemic Closed World Assumption

Hai Wan[a], Rui Yang[b*], Liangda Fang[b], Yongmei Liu[b], and Huada Xu[a]
[a]School of Software, Sun Yat-sen University, Guangzhou, China
[b]Dept. of Computer Science, Sun Yat-sen University, Guangzhou, China
{wanhai,ymliu}@mail.sysu.edu.cn, {yangr7,fangld,xuhuada}@mail2.sysu.edu.cn

Abstract

Planning with epistemic goals has received attention from both the dynamic logic and planning communities. In the single-agent case, under the epistemic closed-world assumption (ECWA), epistemic planning can be reduced to contingent planning. However, it is inappropriate to make the ECWA in some epistemic planning scenarios, for example, when the agent is not fully introspective, or when the agent wants to devise a generic plan that applies to a wide range of situations. In this paper, we propose a complete single-agent epistemic planner without the ECWA. We identify two normal forms of epistemic formulas: weak minimal epistemic DNF and weak minimal epistemic CNF, and present the progression and entailment algorithms based on these normal forms. We adapt the PrAO algorithm for contingent planning from the literature as the main planning algorithm and develop a complete epistemic planner called EPK. Our experimental results show that EPK can generate solutions effectively for most of the epistemic planning problems we have considered including those without the ECWA.

1 Introduction

Planning with epistemic goals has received attention from both the dynamic logic and planning communities. Bolander and Andersen [2011] formalized multi-agent epistemic planning based on dynamic epistemic logic [van Ditmarsch *et al.*, 2007] and showed that it is undecidable in general. Further, Aucher and Bolander [2013] showed that multi-agent epistemic planning is undecidable in the presence of only purely epistemic actions. Meanwhile, Yu *et al.* [2013] identified two important decidable fragments of multi-agent epistemic planning. Recently, Kominis and Geffner [2015], and Muise *et al.* [2015] showed how to exploit classical planning to solve restricted versions of multi-agent epistemic planning problems.

There are earlier works on single-agent epistemic planning. Herzig *et al.* [2003] proposed a framework for epistemic planning where knowledge bases (KBs) are expressed as positive epistemic formulas, and showed how progression, regression and plan generation can be achieved in their framework. Bienvenu *et al.* [2010] identified two normal forms of epistemic formulas called $\mathbf{S}_5$-$\mathrm{CNF}_{\mathrm{CNF,DNF}}$ and $\mathbf{S}_5$-$\mathrm{DNF}_{\mathrm{DNF,CNF}}$, and showed that progression is tractable when the KB is in $\mathbf{S}_5$-$\mathrm{DNF}_{\mathrm{DNF,CNF}}$, and entailment is tractable when the KB is in $\mathbf{S}_5$-$\mathrm{DNF}_{\mathrm{DNF,CNF}}$ and the query is in $\mathbf{S}_5$-$\mathrm{CNF}_{\mathrm{CNF,DNF}}$. However, the above works are only theoretical studies. Petrick and Bacchus [2002; 2004] presented a first-order epistemic planning system PKS based on the idea of progression. The kinds of knowledge they consider include literal knowledge, knowing-whether knowledge, and exclusive-or knowledge. However, their progression and reasoning algorithms are both incomplete, and hence their planner is incomplete.

In the single-agent case, under the epistemic closed-world assumption (ECWA) [Herzig *et al.*, 2003], that is, if I cannot prove that I know φ, then I don't know φ, epistemic planning can be reduced to contingent planning [Peot and Smith, 1992]. Bertoli *et al.* [2006] used binary decision diagrams [Bryant, 1986] to represent knowledge bases and proposed a planner MBP based on symbolic model checking. To *et al.* [2011] used minimal DNF formulas to represent knowledge bases, and proposed an algorithm PrAO with pruning techniques used in AND/OR forward search to solve contingent planning problems.

However, in some epistemic planning scenarios, it is inappropriate to make the ECWA. To illustrate this, consider the following modified *wolf-sheep-cabbage* example. *Every Monday, a farmer has to take at least two objects of a wolf, a sheep and a basket of cabbages to cross a river. The wolf will eat the sheep, and the sheep will eat the cabbages, if the farmer is not with them. What is a plan for the farmer to cross the river?* The initial knowledge base of the planning problem can be represented as $\mathbf{K}(wi \wedge si) \vee \mathbf{K}(si \wedge ci) \vee \mathbf{K}(wi \wedge ci)$, where wi denotes that a wolf is on the left bank of the river initially. Also, si and ci are similar. Although the initial KB does not entail any of $\mathbf{K}wi$, $\mathbf{K}si$, and $\mathbf{K}ci$, we cannot draw the conclusions $\neg\mathbf{K}wi$, $\neg\mathbf{K}si$, and $\neg\mathbf{K}ci$.

In general, the following are some scenarios where it is inappropriate to make the ECWA. Firstly, the agent is not fully introspective; hence the agent is not able to infer what she does not know from an explicit representation of what she knows. Secondly, the agent wants to devise a generic plan

*Corresponding author

that can be used for a wide range of situations. Thirdly, the agent is doing planning for the purpose of intention recognition; hence the agent is not sure about the observed agent's mental state and can only make reasonable assumptions about what the observed agent knows and does not know.

In this paper, we propose a complete single-agent epistemic planner without the epistemic closed-world assumption. We begin with a theoretical formulation of single-agent epistemic planning without the ECWA. Then we identify two normal forms of epistemic formulas: weak minimal epistemic DNF, in short WM-EDNF, and weak minimal epistemic CNF, in short WM-ECNF. We present the progression and entailment algorithms based on the above representations, and show that the entailment algorithm is tractable when the KB is in WM-EDNF and the query is in WM-ECNF. Next we adapt the PrAO algorithm [To *et al.*, 2011] as the main planning algorithm and develop a complete epistemic planner called EPK. Our experimental results show that EPK can generate solutions effectively for most of the epistemic planning problems we have considered including those without the ECWA.

2 Preliminaries

In this section, we first introduce propositional logic and two notions: minimal DNF/CNF and strong entailment. Then, we extend the above notions to epistemic logic, and identify two forms of epistemic formulas: WM-EDNF which represents the KBs and WM-ECNF which represents the preconditions of actions and the goals of problems. Finally, we give the definitions of ontic actions and sensing actions, and the model-theoretic definition of progression wrt actions.

2.1 Propositional logic

Throughout this paper, we fix $\mathcal{X} = \{x_1, \cdots, x_n\}$ be a finite set of propositions. The propositional language $\mathcal{L}_{prop}$ is generated from $\mathcal{X}$, the usual connectives and two boolean constants $\top$ and $\bot$. A *state* is a subset of $\mathcal{X}$, *e.g.*, $\{x_1, x_2\}$ is a state where both x_1 and x_2 hold and other propositions do not. A *literal* is either a proposition x or its negation $\neg x$. $\overline{l}$ denotes the complement of a literal l. For a set of literals L, $\overline{L} = \{\overline{l} \mid l \in L\}$. We call φ a *term* (resp. *clause*) if it is in the form of $\bigwedge_i l_i$ (resp. $\bigvee_i l_i$) where every l_i is a literal. Term (resp. clause) is often represented as a set of literals. So the union of two terms and the difference of two terms can be considered as terms. For example, suppose that $t_1 = x_1 \wedge \neg x_2$ and $t_2 = \neg x_2 \wedge x_3$. Then $t_1 \cup t_2 = x_1 \wedge \neg x_2 \wedge x_3$ and $t_1 \setminus t_2 = x_1$. A *DNF* (resp. *CNF*) ϕ is form of $\bigvee_i \psi_i$ (resp. $\bigwedge_i \psi_i$) where every ψ_i is a term (resp. clause).

Definition 2.1 (Minimal DNF/CNF) A DNF (resp. CNF) ϕ whose form is $\bigvee_i \psi_i$ (resp. $\bigwedge_i \psi_i$) is *minimal* if for any two distinct terms (resp. clauses) ψ_i and ψ_j, $\psi_i \not\models \psi_j$ and $\psi_j \not\models \psi_i$.

Obviously, we can transform a DNF (resp. CNF) ϕ to an equivalent minimal one, denoted by $min(\phi)$, by removing the redundant terms (resp. clauses).

Definition 2.2 Let ϕ and ϕ' be two DNFs where $\phi = \bigvee_i \psi_i$ and $\phi' = \bigvee_j \psi'_j$. We say that ϕ strongly entails ϕ', denoted by $\phi \mapsto \phi'$, if for all ψ_i, there exists ψ'_j s.t. $\psi_i \models \psi'_j$.

Proposition 2.1 *Let ϕ and ϕ' be two DNFs. If $\phi \mapsto \phi'$, then $\phi \models \phi'$.*

The above proposition says that if ϕ strongly entails ϕ', then ϕ entails ϕ', but not vice versa. For example, let $\phi = a \vee \neg a$ and $\phi' = b \vee \neg b$. Since $\phi \equiv \phi' \equiv \top$, we get that $\phi \models \phi'$. But $\phi \not\mapsto \phi'$ because $a \not\models b$ and $a \not\models \neg b$. Similarly, we can extend the concept to CNFs.

Definition 2.3 Let ϕ and ϕ' be two CNFs where $\phi = \bigwedge_i \psi_i$ and $\phi' = \bigwedge_j \psi'_j$. We say that ϕ strongly entails ϕ', denoted by $\phi \mapsto \phi'$, if for all ψ'_j, there exists ψ_i s.t. $\psi_i \models \psi'_j$.

Proposition 2.2 *Let ϕ and ϕ' be two CNFs. If $\phi \mapsto \phi'$, then $\phi \models \phi'$.*

2.2 Epistemic logic

Objective facts can be represented as propositional formulas. In epistemic planning, we model what the agent knows about the current state, which is an subjective formula, in epistemic logic. We will illustrate the syntax and semantics of epistemic logic $\mathcal{L}_K$ as follows.

Definition 2.4 The language of epistemic logic $\mathcal{L}_K$ is defined as follows using BNF notation:

$$\Phi ::= \mathbf{K}\phi \mid \Phi \wedge \Psi \mid \neg\Phi,$$

where $\phi \in \mathcal{L}_{prop}$, and $\Phi, \Psi \in \mathcal{L}_K$.

For distinguishing two kinds of formulas, we use capital letters to denote the epistemic formulas, *i.e.*, Φ, Ψ etc. The propositional formulas are denoted by lowercase letter, *i.e.*, ϕ, ψ etc.

Definition 2.5 (Epistemic state) An epistemic state, in short e-state, is a finite non-empty set of states.

Intuitively, the agent considers every state in an epistemic state is possible.

Definition 2.6 Let M be an e-state. We interpret formulas in $\mathcal{L}_K$ as follows:

- for $\phi \in \mathcal{L}_{prop}$, $M \models \mathbf{K}\phi$ if for every $s \in M$, $s \models \phi$;
- $M \models \Phi \wedge \Psi$ if $M \models \Phi$ and $M \models \Psi$;
- $M \models \neg\Phi$ if $M \not\models \Phi$.

We let $Mod(\Phi) = \{M \subseteq 2^{\mathcal{X}} \mid M \neq \emptyset \text{ and } M \models \Phi\}$, which is the set of e-states satisfying Φ. We remark that $Mod(\bot) \equiv \emptyset$ since there are no e-states satisfying $\bot$.

Let Φ be an epistemic formula. We say that Φ is an *epistemic literal* if it is in the form of $\mathbf{K}\phi$ or $\hat{\mathbf{K}}\phi$ where $\phi \in \mathcal{L}_{prop}$. We also say that Φ is an *epistemic term* (resp. *clause*) if it is in the form of $\mathbf{K}\psi \wedge \bigwedge_i \hat{\mathbf{K}}\eta_i$ (resp. $\hat{\mathbf{K}}\psi \vee \bigvee_i \mathbf{K}\eta_i$). Throughout this paper, we require ψ and all η_i's are DNF (resp. CNF) for an epistemic term (resp. clause).

Definition 2.7 (Epistemic DNF/CNF) An epistemic formula Φ is an epistemic DNF (resp. CNF), in short EDNF (resp. ECNF) is in the form of $\bigvee_i \Psi_i$ (resp. $\bigwedge_i \Psi_i$) where Ψ_i is an epistemic term (resp. clause).

Definition 2.8 (Separable)

- An epistemic term $\mathbf{K}\psi \wedge \bigwedge_i \hat{\mathbf{K}}\eta_i$ is *separable* if $\eta_i \models \psi$ for each i.

- An epistemic clause $\hat{\mathbf{K}}\psi \vee \bigvee_i \mathbf{K}\eta_i$ is *separable* if $\psi \models \eta_i$ for each i.
- An EDNF $\bigvee_i \Psi_i$ is *separable* if every Ψ_i is separable.
- An ECNF $\bigwedge_i \Psi_i$ is *separable* if every Ψ_i is separable.

We can transform an epistemic formula to an equivalent separable one. For example, $\mathbf{K}\psi \wedge \bigwedge_i \hat{\mathbf{K}}\eta_i$ is equivalent to $\mathbf{K}\psi \wedge \bigwedge_i \hat{\mathbf{K}}(\psi \wedge \eta_i)$. The latter is separable.

Definition 2.9 (Strong entailment) Let Φ and Φ' be two separable epistemic terms where $\Phi = \mathbf{K}\psi \wedge \bigwedge_i \hat{\mathbf{K}}\eta_i$ and $\Phi' = \mathbf{K}\psi' \wedge \bigwedge_j \hat{\mathbf{K}}\eta'_j$. We say that Φ strongly entails Φ', denoted by $\Phi \mapsto \Phi'$ if $\psi \mapsto \psi'$ and for all η'_j, either $\psi \mapsto \eta'_j$ or there exists η_i s.t. $\eta_i \mapsto \eta'_j$.

Similar to propositional logic, we can get the following proposition.

Proposition 2.3 *Let Φ and Φ' be two separable EDNFs. If $\Phi \mapsto \Phi'$, then $\Phi \models \Phi'$.*

Now, we can define weak minimal EDNFs.

Definition 2.10 (Weak minimal EDNF) A separable EDNF Φ whose form is $\bigvee_i \Psi_i$ is *weak minimal* if for any two distinct epistemic terms Ψ_i and Ψ_j, $\Psi_i \not\mapsto \Psi_j$ and $\Psi_j \not\mapsto \Psi_i$.

For space limitations, we omit a similar notion weak minimal ECNF. We use $wmin(\Phi)$ to denote the weak minimal of Φ.

Proposition 2.4 *Every epistemic formula can be transformed to a WM-EDNF (resp. WM-ECNF) equivalently.*

2.3 Actions and Progression

In this paper, we extend the preconditions of actions so as to be represented as epistemic formulas.

A *conditional effect* is a pair $\langle con, lit\rangle$ where con and lit are sets of literals. Intuitively, it means that literals in lit, both positive and negative, become true if a certain action is executed when con holds. The definitions of ontic actions and sensing actions are as follows.

Definition 2.11 (Ontic action) An ontic action a_o is a pair $\langle pre, eff\rangle$, where

- pre: an epistemic formula, which specifies precondition;
- eff: a set of conditional effects.

Throughout this paper, we require the preconditions of actions including sensing actions to be WM-ECNFs. Given a state s and an ontic action a_o, the positive effect $pe(a_o, s)$ of executing a_o in s is defined as

$$\{l \in lit(e) \mid \exists e \in eff(a_o).s \models con(e) \text{ and } l \text{ is positive}\}.$$

The negative effect, denoted by $ne(a_o, s)$, is similar. The new state resulting from s by executing a_o, denoted by $s \circledast a_o$, is $s \setminus ne(a_o, s) \cup pe(a_o, s)$. We also require that ontic actions are not self-contradictory. For an ontic action a_o, there do not exist a proposition x, a state s and two conditional effects $e, e' \in eff(a_o)$ such that the following hold: (1) $x \in lit(e)$; (2) $\neg x \in lit(e')$; (3) $s \models con(e)$; and (4) $s \models con(e')$.

We say an action a is applicable in an e-state M iff $M \models pre(a)$. We say a is executable in Φ iff a is applicable in every e-state of Φ. On performing a_o in M, the new e-state $M \otimes a_o$ is $\{s \circledast a_o \mid s \in M\}$.

Definition 2.12 (Progression wrt ontic actions) Let $\Phi \in \mathcal{L}_K$, a_o be an ontic action executable in Φ. Then Φ', denoted by $prog(\Phi, a_o)$, is a progression of Φ for a_o if

$$Mod(\Phi') = \{M \otimes a_o \mid M \in Mod(\Phi)\}.$$

We give the definition of sensing actions below.

Definition 2.13 (Sensing action) A sensing action a_s is a pair $\langle pre, res\rangle$, where

- pre: an epistemic formula, which specifies precondition;
- res: a formula in $\mathcal{L}_{prop}$, which is the sensing result.

After executing a sensing action a_s, the agent will know whether $res(a_s)$ holds. The applicability and executability of sensing actions are similar to those of ontic actions. Given an e-state M and a propositional formula ϕ, the new epistemic state $M \oslash \phi$ is $\{s \in M \mid s \models \phi\}$, *i.e.*, the set of states of M satisfying ϕ.

Definition 2.14 (Progression wrt sensing actions) Let Φ be an epistemic formula, a_s a sensing action executable in Φ. Then a pair (Φ^+, Φ^-), denoted by $prog(\Phi, a_s)$, is a progression of Φ for a_s where

- $Mod(\Phi^+) = \{M \oslash res(a_s) \mid M \in Mod(\Phi)\} \setminus \{\emptyset\}$;
- $Mod(\Phi^-) = \{M \oslash \neg res(a_s) \mid M \in Mod(\Phi)\} \setminus \{\emptyset\}$.

3 Epistemic planning

In this section, we introduce the definitions of epistemic planning problem and its solution.

Definition 3.1 (Epistemic planning problem) An epistemic planning problem $\mathcal{P}$ is a tuple $\langle \mathcal{O}, \mathcal{S}, \mathcal{I}, \mathcal{G}\rangle$, where

- $\mathcal{O}$: a set of ontic actions;
- $\mathcal{S}$: a set of sensing actions;
- $\mathcal{I}$: an epistemic formula specifying the initial KB;
- $\mathcal{G}$: an epistemic formula specifying the goal formula.

$\mathcal{O}$ and $\mathcal{S}$ are separate, *i.e.*, $\mathcal{O} \cap \mathcal{S} = \emptyset$.

Throughout this paper, we require $\mathcal{I}$ to be a WM-EDNF and $\mathcal{G}$ to be a WM-ECNF.

Example 1 We now formalize the example from the introduction as follows. Here, we only give several ontic actions about that the farmer moving from the right bank to the left bank. The other ontic actions are similar. We let sa denote that the sheep is alive, ce denote that the cabbages are not eaten, d denote that the boat can load two objects (excluding the farmer), and fl, wl, sl and cl denote that the farmer, the wolf, the sheep or the cabbages are on the left bank of the river respectively.

1. $left$: the farmer moves to the left bank without taking anything.
   - $pre(left) \equiv \mathbf{K}\neg fl$
   - $eff(left) = \{\langle \emptyset, \{\neg fl\}\rangle,$
     $\langle \{wi, si, sa, \neg wl, \neg sl\}, \{\neg sa\}\rangle,$
     $\langle \{si, sa, ci, ce, sl, cl\}, \{\neg ce\}\rangle,$
     $\langle \{si, sa, ci, ce, \neg sl, \neg cl\}, \{\neg ce\}\rangle\}$
2. $wLeft$: the farmer tries to take the wolf to the left bank.

- $pre(wLeft) \equiv \mathbf{K}(\neg fl \wedge \neg d)$
- $eff(wLeft) = \{\langle \emptyset, \{fl\}\rangle,$
  $\langle \{wi, \neg wl\}, \{wl\}\rangle,$
  $\langle \{si, sa, ci, ce, sl, cl\}, \{\neg ce\}\rangle,$
  $\langle \{si, sa, ci, ce, \neg sl, \neg cl\}, \{\neg ce\}\rangle\}$

  Intuitively, the farmer will move alone if the wolf is on the left bank or there does not exist a wolf initially.

3. $wsLeft$: the farmer tries to take the wolf and the sheep to the left bank.
   - $pre(wsLeft) \equiv \mathbf{K}(\neg fl \wedge d)$
   - $eff(wsLeft) = \{\langle \emptyset, \{fl\}\rangle,$
     $\langle \{wi, \neg wl\}, \{wl\}\rangle,$
     $\langle \{si, sa, \neg sl\}, \{sl\}\rangle\}$

   Similar to $wLeft$, the farmer will take the wolf (resp. the sheep) if it is on the right bank.
4. $senseD$: the farmer inquires whether the boat can load two objects.
   - $pre(senseD) \equiv \mathbf{K}\top$
   - $res(senseD) \equiv d$
5. $\mathcal{I} = (\mathbf{K}(ci \wedge si) \vee \mathbf{K}(wi \wedge si) \vee (ci \wedge si)) \wedge \mathbf{K}(fl \wedge (ci \rightarrow ce \wedge cl) \wedge (si \rightarrow sa \wedge sl) \wedge (wi \rightarrow wl)) \wedge \hat{\mathbf{K}}d \wedge \hat{\mathbf{K}}\neg d$.
6. $\mathcal{G} = \mathbf{K}((ci \rightarrow ce \wedge \neg cl) \wedge (si \rightarrow sa \wedge \neg sl) \wedge (wi \rightarrow \neg wl))$.

We formalize the notion of conditional plans through *action trees* as follows.

Definition 3.2 (Action tree) Let $\mathcal{P}$ be an epistemic planning problem where $\mathcal{P} = \langle \mathcal{O}, \mathcal{S}, \mathcal{I}, \mathcal{G}\rangle$. An action tree $\mathcal{T}$ is defined inductively as follows:

- ϵ: the empty tree
- $a_o; \mathcal{T}'$, where $a_o \in \mathcal{O}$, and T' is an action tree
- $a_s; (\mathcal{T}^+|\mathcal{T}^-)$, where $a_s \in \mathcal{S}$ and $\mathcal{T}^+$ and $\mathcal{T}^-$ are action trees.

Let us denote *undefined* by $undef$. Now, we define the progression wrt action trees.

Definition 3.3 (Progression wrt action trees) Let Φ be an epistemic formula, and $\mathcal{T}$ an action tree. The progression of Φ wrt $\mathcal{T}$, denoted by $prog(\Phi, \mathcal{T})$, is defined inductively as follows:

- $prog(\Phi, \epsilon) = \{\Phi\}$, if $\Phi \not\equiv \bot$;
- $prog(\Phi, \epsilon) = undef$, otherwise;
- $prog(\Phi, a_o; \mathcal{T}') = \{prog(prog(\Phi, a_o), \mathcal{T}')\}$, if $\Phi \models pre(a_o)$ and $\Phi \not\equiv \bot$;
- $prog(\Phi, a_o; \mathcal{T}') = undef$, otherwise;
- $prog(\Phi, a_s; (\mathcal{T}^+|\mathcal{T}^-)) = prog(\Phi^+, \mathcal{T}^+) \cup prog(\Phi^-, \mathcal{T}^-)$, if $\Phi \models pre(a_s)$ and $\Phi \not\models \mathbf{K}res(a_s)$ and $\Phi \not\models \mathbf{K}\neg res(a_s)$ and $\Phi \not\equiv \bot$;
- $prog(\Phi, a_s; (\mathcal{T}^+|\mathcal{T}^-)) = undef$, otherwise.

We define solutions to epistemic planning problems below, which are typically action trees.

Definition 3.4 (Epistemic planning solution) Let $\mathcal{P}$ be an epistemic planning problem where $\mathcal{P} = \langle \mathcal{O}, \mathcal{S}, \mathcal{I}, \mathcal{G}\rangle$. An action tree $\mathcal{T}$ is a solution to $\mathcal{P}$ if $\mathcal{T}$ is finite, $prog(\mathcal{I}, \mathcal{T})$ is defined, and for every epistemic formulas Φ of $prog(\mathcal{I}, \mathcal{T})$, Φ entails the goal formula $\mathcal{G}$.

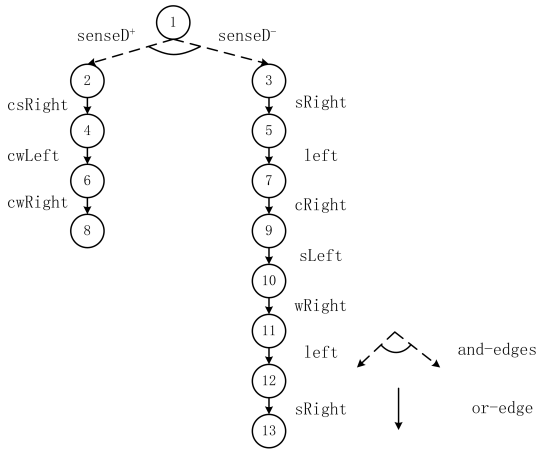

Figure 1: A solution to the example

Example 2 (Example 1 continued) The action tree, depicted in Figure 1, is a solution to this example. It states as follows. Firstly, the farmer senses whether a boat can load two objects. If the boat can load two objects, then the farmer will try to take the sheep and the cabbages to cross the river. Because the cabbages are eaten by the sheep if the farmer is not around, he tries to take the cabbages back to the left bank. Finally, he tries to take the wolf and the cabbages to the right bank. Otherwise, the farmer should obey the "eat rule", and has to take only one object to cross the river until all the objects are on the right bank of the river.

4 Progression and Reasoning

In this section, we introduce three core parts of EPK: (1) progression wrt ontic actions and sensing actions; (2) determining whether a KB entails the precondition of an action; (3) checking whether two KBs are strongly equivalent.

4.1 Progression

To update KBs according to the effects of ontic actions, we firstly transform a DNF ϕ to an equivalent one ϕ' such that for every conditional effect and every term of ϕ', either the term entails the condition con of the conditional effect, or it entails $\neg con$.

Definition 4.1 Let t and t' be two terms. The extension of t wrt t', denoted by $ext(t, t')$, is defined as

$$ext(t, t') = \begin{cases} t, & \text{if } t \models t' \text{ or } t \wedge t' \equiv \bot; \\ t \wedge t' \vee \bigvee_{t \not\models l \text{ and } t' \models l} (t \wedge \neg l), & \text{otherwise.} \end{cases}$$

Intuitively, we keep t if it entails t' or it entails the negation of t'. Otherwise, we transform t to a DNF $ext(t, t')$ such that for each term of $ext(t, t')$, either it entails t' or it entails the negation of t'.

Proposition 4.1 *Let t and t' be two terms, and $\phi = min(ext(t, t'))$. Then $\phi \equiv t$ and for every term t^* of ϕ, either $t^* \models t'$ or $t^* \models \neg t'$ holds.*

For a DNF $\phi = \bigvee_i t_i$, we can define a similar operation: $ext(\phi, t') = min(\bigvee_i ext(t_i, t'))$.

Definition 4.2 Let a_o an ontic action. We say a term t is *enabling for* a_o if for every conditional effect $\langle con, lit\rangle \in eff(a_o)$, either $t \models con$ or $t \models \neg con$ holds. We say a DNF ϕ is *enabling for* a_o if every term t of ϕ is enabling for a_o.

Proposition 4.2 *Let ϕ be a DNF, a_o an ontic action where $eff(a_o) = \{\langle con_i, lit_i\rangle\}_{i=1}^k$, and $\phi' = ext((ext(\phi', con_1), \cdots), con_k)$. Then $\phi' \equiv \phi$ and ϕ' is enabling for a_o.*

The above proposition shows how to transform a DNF to an equivalent one enabling for ontic actions.

Definition 4.3 Let t be a term, a_o an ontic action which t is enabling for. The effect of a_o in t, denoted by $e(t, a_o)$, is $\{l \mid \exists \langle con, lit\rangle \in eff(a_o).t \models con \text{ and } l \in lit\}$.

Definition 4.4 Let a_o be an ontic action, and ϕ a DNF enabling for a_o. The update of ϕ wrt a_o, denoted by $update(\phi, a_o)$, is $min(\bigvee_{t\in\phi}(t \setminus \overline{e(t, a_o)} \cup e(t, a_o)))$.

The above proposition tells us how to update a DNF according to the effect of an ontic actions. We are now ready to give the computation of the progression wrt ontic actions.

Proposition 4.3 *Let $\Psi = \boldsymbol{K}\psi \wedge \bigwedge_i \hat{\boldsymbol{K}}\eta_i$ be a separate epistemic term, and a_o an ontic action executable in Ψ, and $\Psi' = \boldsymbol{K}\psi' \wedge \bigwedge_i \hat{\boldsymbol{K}}\eta_i'$ where $\psi' = update(\psi, a_o)$ and $\eta_i' = update(\eta_i, a_o)$ for each i. Ψ' is a progression of Ψ wrt a_o.*

Proposition 4.4 *Let $\Phi = \bigvee_i \Psi_i$ be a separable EDNF, and a_o an ontic action executable in Φ. Then $prog(\Phi, a_o) \equiv \bigvee_i prog(\Psi_i, a_o)$.*

Even if Φ is a WM-EDNF, the progressed KB $prog(\Phi, a_o)$ may not be a WM-EDNF. But we can transform it to an equivalent WM-EDNF.

Next, we introduce the progression of separable epistemic terms wrt observation.

Definition 4.5 Let $\Psi = \mathbf{K}\psi \wedge \bigwedge_i \hat{\mathbf{K}}\eta_i$ be a separate epistemic term, and $\phi \in \mathcal{L}_{prop}$. The update of Ψ wrt ϕ, denoted by $obs(\Psi, \phi)$, is $\mathbf{K}(\psi \wedge \phi) \wedge \bigwedge_{\eta_i \models \phi} \hat{\mathbf{K}}\eta_i$.

After observing ϕ, an agent remains her knowledge of ψ and keeps considering η_i possible if $\eta_i \models \phi$. The latter is counterintuitive, but it is valid. This is because the possible state satisfying η_i may not satisfy ϕ when we do not make the ECWA.

Proposition 4.5 Let $\Phi = \bigvee_i \Psi_i$ be an EDNF, and a_s a sensing action executable in Φ. A pair (Φ^+, Φ^-) is a progression of Φ wrt a_s where $\Phi^+ \equiv \bigvee_i obs(\Psi_i, res(a_s))$ and $\Phi^- \equiv \bigvee_i obs(\Psi_i, \neg res(a_s))$.

By Proposition 4.5, we can implement progression wrt sensing actions. Similarly, we transform Φ^+ (resp. Φ^-) to a WM-EDNF when it is not a WM-EDNF.

4.2 Reasoning

Firstly, we introduce a simple property of propositional logic.

Proposition 4.6 *Let $\phi = \bigvee_i \psi_i$ be a DNF and $\phi' = \bigwedge_j \psi_j'$ a CNF. Then $\phi \models \phi'$ iff for every i, j, we have $\psi_i \models \psi_j'$.*

Proposition 4.7 *Let Ψ be a separable epistemic term whose form is $\boldsymbol{K}\psi \wedge \bigwedge_i \hat{\boldsymbol{K}}\eta_i$ and Ψ' a separable epistemic clause whose form is $\hat{\boldsymbol{K}}\psi' \vee \bigvee_j \boldsymbol{K}\eta_j'$. $\Psi \models \Psi'$ iff one of the following conditions holds*

1. *$\psi \wedge \neg\psi'$ is inconsistent;*
2. *there exists i s.t. $\neg\psi' \wedge \eta_i$ is inconsistent;*
3. *there exists j s.t. $\psi \wedge \neg\eta_j'$ is inconsistent.*

Proposition 4.8 *Let $\Phi = \bigvee_i \Psi_i$ be an EDNF and $\Phi' = \bigwedge_j \Psi_j$ an ECNF. $\Phi \models \Phi'$ iff for every i, j, $\Psi_i \models \Psi_j$.*

By Proposition 4.8, we can determine whether an action is executable in the current KB in polynomial time.

Theorem 4.1 *Let Φ be an EDNF and Ψ be an ECNF. Deciding whether $\Phi \models \Psi$ is tractable.*

Next, we show how to judge the equivalence of two KBs.

Definition 4.6 (Strong equivalence) Let $\Phi = \bigvee_i \Psi_i$ and $\Phi' = \bigvee_j \Psi_j'$ be two WM-EDNFs. We say $\Phi \cong \Phi'$ if the following two conditions all hold:

1. for every i, there exists j s.t. $\Psi_i \mapsto \Psi_j'$;
2. for every j, there exists i s.t. $\Psi_j' \mapsto \Psi_i$.

Proposition 4.9 Let Φ and Φ' be two WM-EDNFs. If $\Phi \cong \Phi'$, then $\Phi \equiv \Phi'$.

By Proposition 4.9, we can determine whether two WM-ENDFs are strongly equivalent. Here, we only give a sufficient condition, but not a sufficient and necessary condition. So, sometimes, two WM-ENDFs are equivalent, but not strongly equivalent.

5 Planning Algorithm

In this section, we briefly describe PrAO algorithm [To *et al.*, 2011], which is the main planning algorithm of EPK. The algorithm is based on progression and entailment which have been introduced in the last section.

A node n in a search graph is a knowledge base Φ_n with a state $state(n)$ including four tags: $goal$, $dead$, $unexplored$, $explored$ and an additional tag: $connected$. As depicted in Algorithm 1, Lines 1 to 5 check whether $\mathcal{I}$ entails $\mathcal{G}$, if so, then return the empty plan; otherwise, initialize the search graph. Lines 6 to 15 construct the search graph until it finds a solution or guarantees that there are no solutions. Lines 7 to 10 select an $unexplored$ and $connected$ node n in $\mathcal{N}$ for exploration. If no such node exists then return no solution. Line 11 expands the selected node and updates the graph accordingly. Lines 12 to 15 check the state of the initial node. If its state is $goal$, return $BuildPlan(\mathcal{N}, \mathcal{T}, n_0)$ to generate a solution; if $dead$, return no solution. Otherwise, continue the loop.

For space limitations, we briefly introduce subroutine $Explore$ and the heuristic function used in Algorithm 1.

Algorithm 1: $Plan(\mathcal{O}, \mathcal{S}, \mathcal{I}, \mathcal{G})$

Input : An epistemic problem $\mathcal{P} = \langle \mathcal{O}, \mathcal{S}, \mathcal{I}, \mathcal{G} \rangle$
Output: A solution if it guarantees goal achievement or NULL otherwise.

```
if I ⊨ G then
    return an empty plan ϵ
else
    Let n_0 = I, state(n_0) = unexplored,
        connected(n_0) = true, T = ∅ and N = {n_0}
while true do
    if {n ∈ N | state(n) = unexplored and
                connected(n) = true} = ∅ then
        return NULL
    Let n be the unexplored and connected node with a
    heuristic function in N
    Explore(n)
    if state(n_0) = goal then
        return BuildPlan(N, T, n_0)
    if state(n_0) = dead then
        return NULL
```

$Explore(n)$ can generate the children of n for actions executable in Φ_n with loop detection. The application of an ontic action starting from n results in a new node, while applying sensing actions produces two new nodes, *i.e.*, n is split amongst the observations. It is a simple idea that the design of heuristic function for choosing nodes in $\mathcal{N}$ to explore. For each epistemic clause of goal in WM-ECNF, we count how many epistemic term which can entail that epistemic clause in current KB, and sum them up. Then we can get the heuristic value of current node via dividing total times by the number of epistemic clause of goal.

Finally, we have the following theorem stating that Algorithm 1 is sound and complete.

Theorem 5.1 *Let $\mathcal{P} = \langle \mathcal{O}, \mathcal{S}, \mathcal{I}, \mathcal{G} \rangle$ be an epistemic planning problem, and $\mathcal{T}_s$ an action tree constructed by the $Plan(\mathcal{O}, \mathcal{S}, \mathcal{I}, \mathcal{G})$. We have:*

1. *If $\mathcal{T}_s$ is NULL, then $\mathcal{P}$ has no solution; otherwise, $\mathcal{T}_s$ is a solution tree for $\mathcal{P}$, and $\mathcal{P}$ has a solution;*
2. *If $\mathcal{P}$ has a solution, then $\mathcal{T}_s$ is a solution tree for $\mathcal{P}$; otherwise, $\mathcal{T}_s$ is NULL.*

6 Implementation and Experiments

Our planner EPK is implemented in C++, and our experiments were done on a PC with Intel i7-4700MQ (2.4GHz) CPU and 4GB RAM on Linux.[1] Results are in the form of $t(d/s)$, where t, d and s denote the overall execution time, the depth of the solution tree and the number of nodes in the solution respectively. Usually, d and s are criteria for evaluation of the quality of a solution. "–" denotes out-of-memory or time-out (20 minutes limit). Heuristic, BFS, DFS and IDS stand for search with heuristic function, breadth-first search, depth-first search, and iterative-deepening search respectively.

We experimentally compare EPK with PKS, and use a testbed of five domains, including two domains taken from PKS, other domains taken from contingent planning. Some domains in PKS contain functions and numerical representation, which EPK can not handle. Two domains in PKS are as follows: btc and $unix$ series. In addition, other domains, $doors$, $push$ and $dispose$ are from classical contingent planning benchmarks.

Table 1 reports the comparative experimental results. We can observe that, generally speaking, our planner EPK are better than PKS in the performance except btc. In detail, the heuristic function is not suitable for these domains, since this search method is slower than BFS and DFS in EPK. For domains btc, $unix$ series, PKS is faster than EPK, while the quality of epistemic planning solutions of EPK are better than PKS using BFS or DFS. For domains such as $push$, $doors$, $dispose$ series, EPK is faster than PKS using BFS and PKS even can not solve $push$ and $doors$ series using DFS.

Table 2 shows that domains without the ECWA. Although heuristic search in domains such as $block$, $farmer$, $medpks$-2 and $medpks$-3 is slower than BFS and DFS, it is obvious that heuristic function is very important especially for $cube$ series and $ebtcs$-2-5.

Table 1: Experimental results between PKS and EPK

Problem	EPK			PKS		
	Heuristic	BFS	DFS	BFS	DFS	IDS
btc-20	0.549 (39/40)	—	0.033 (39/40)	—	0.005 (40/41)	—
btc-30	3.976 (59/60)	—	0.169 (59/60)	—	0.008 (60/61)	—
btc-50	56.826 (99/100)	—	1.203 (99/100)	—	0.009 (100/101)	—
unix-1	0.030 (7/8)	0.006 (3/4)	0.015 (3/4)	0.003 (3/4)	0.003 (81/82)	0.003 (3/4)
unix-2	0.470 (4/6)	0.104 (3/4)	0.199 (3/4)	0.026 (3/4)	0.212 (>100/>100)	0.025 (3/4)
doors-2	0.010 (9/16)	0.011 (5/9)	0.010 (8/14)	0.419 (5/9)	—	0.035 (5/10)
doors-5	34.518 (71/72)	5.495 (6/7)	0.346 (8/9)	67.386 (6/7)	—	75.125 (8/9)
push-3-1	0.009 (3/4)	0.052 (3/4)	0.296 (3/4)	0.094 (3/4)	—	0.100 (3/4)
push-3-2	163.478 (8/9)	1.889 (4/5)	—	2.354 (4/5)	—	2.477 (4/5)
dispose-2-1	0.001 (2/3)	0.001 (2/3)	0.002 (2/3)	0.001 (2/3)	0.023 (>100/>100)	0.001 (2/3)
dispose-2-2	0.094 (13/18)	0.087 (7/11)	0.176 (7/11)	0.491 (7/11)	—	0.516 (7/13)
dispose-3-1	0.009 (4/5)	0.166 (4/5)	15.170 (4/5)	0.868 (4/5)	8.537 (>1K/>1K)	0.875 (4/5)

Table 2: Experimental results in EPK without the ECWA

Problem	Heuristic	BFS	DFS
farmer	0.073(8/13)	0.043(8/13)	0.084(8/13)
block-3	1.777(12/44)	0.006(5/9)	0.019(5/9)
inroom	0.001(3/6)	0.001(3/6)	0.001(3/6)
doors-2	0.132(5/6)	10.091(3/7)	—
medpks-2	0.008(3/6)	0.001(2/3)	0.001(2/3)
medpks-3	0.013(2/3)	0.004(2/3)	0.005(2/3)
medpks-6	0.136(2/3)	0.168(2/3)	0.163(2/3)
ebtcs-2-5	0.014(3/4)	8.155(3/4)	—
ebtcs-2-6	—	28.671(3/7)	—
cube-3	0.714(15/18)	0.986(13/16)	1.070(13/16)
cube-5	8.424(48/51)	—	—
cube-6	124.528(98/101)	—	—

[1] http://ss.sysu.edu.cn/%7ewh/epk.html

7 Conclusions

We have introduced a general and effective approach to single-agent epistemic planning with deterministic actions without making the ECWA. We identify two compact representations: WM-EDNF and WM-ECNF. Based on the representations, we give the progression and entailment algorithms which are core modules in planning. Moreover, entailment is tractable when the KB is in WM-EDNF and the query is in WM-ECNF. We adapt the PrAO algorithm as the main planning algorithm, and our planner is complete. Our empirical evaluation shows that our planner is better than PKS in most of the epistemic planning problems we have tested. Also, EPK generates solutions effectively for most of the epistemic planning problems we have considered, especially those without the ECWA.

There are several possible directions for future study. Firstly, we want to improve the efficiency of our planner via designing a compact yet efficient and scalable representation and developing better heuristic functions, such as graph heuristics. Secondly, we want to extend the definition of ontic actions so that conditional effects can contain not only propositional literals but also epistemic literals, and give the semantics for these actions. Thirdly, we would also like to extend this work to multi-agent cases.

8 Acknowledgments

The authors would like to thank Andreas Herzig for his useful suggestions. Hai Wan thanks Guangzhou Science and Technology Project (No. 2013J4100058) for the support of this research. Liangda Fang and Yongmei Liu acknowledge the support of the National Natural Science Foundation of China under grant No. 61073053.

References

[Aucher and Bolander, 2013] Guillaume Aucher and Thomas Bolander. Undecidability in epistemic planning. In Francesca Rossi, editor, *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-2013)*, pages 27–33, 2013.

[Bertoli *et al.*, 2006] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Strong planning under partial observability. *Artificial Intelligence*, 170(4):337–384, 2006.

[Bienvenu *et al.*, 2010] Meghyn Bienvenu, Hélène Fargier, and Pierre Marquis. Knowledge compilation in the modal logic S5. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI-2010)*, pages 261–265, 2010.

[Bolander and Andersen, 2011] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single- and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011.

[Bryant, 1986] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 100(8):677–691, 1986.

[Herzig *et al.*, 2003] Andreas Herzig, Jérôme Lang, and Pierre Marquis. Action representation and partially observable planning using epistemic logic. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pages 1067–1072, 2003.

[Kominis and Geffner, 2015] Filippos Kominis and Hector Geffner. Beliefs in multiagent planning: From one agent to many. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS-2015)*, 2015.

[Muise *et al.*, 2015] Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-2015)*, pages 3327–3334, 2015.

[Peot and Smith, 1992] Mark A. Peot and David E. Smith. Conditional nonlinear planning. In *Proceedings of the First International Conference on Artificial Intelligence Planning Systems (AIPS-1992)*, pages 189–197, 1992.

[Petrick and Bacchus, 2002] Ronald P.A. Petrick and Fahiem Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS-2002)*, pages 212–222, 2002.

[Petrick and Bacchus, 2004] Ronald P.A. Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-2004)*, pages 613–622, 2004.

[To *et al.*, 2011] Son Thanh To, Tran Cao Son, and Enrico Pontelli. Contingent planning as and/or forward search with disjunctive representation. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS-2011)*, pages 258–265, 2011.

[van Ditmarsch *et al.*, 2007] Hans van Ditmarsch, Wiebe van Der Hoek, and Barteld P. Kooi. *Dynamic epistemic logic*. Springer, 2007.

[Yu *et al.*, 2013] Quan Yu, Ximing Wen, and Yongmei Liu. Multi-agent epistemic explanatory diagnosis via reasoning about actions. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-2013)*, pages 1183–1190, 2013.