A simple account of multi-agent epistemic planning

Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, Pierre Régnier

Abstract. A realistic model of multi-agent planning must allow us to formalize notions which are absent in classical planning, such as communication and knowledge. We investigate multi-agent planning based on a simple logic of knowledge that is grounded on the visibility of propositional variables. Using such a formal logic allows us to prove the existence of a plan given the description of the individual actions. We present an encoding of multi-agent planning problems expressed in this logic into the classical planning language PDDL. The solvability of a planning task is reduced to a model checking problem in a dynamic extension of our logic, proving its complexity. Feeding the resulting problem into a PDDL planner provides a provably correct plan for the original multi-agent planning problem. We apply our method on several examples such as the gossip problem.

1 Introduction

Suppose there are \( n \) agents each of which knows some secret: a piece of information that is not known to the others. They communicate by phone calls, and whenever one person calls another they tell each other all they know at that time. How many calls are required before everyone else knows what they know too? This gossip problem can be viewed as perhaps the simplest multi-agent planning problem: it is only the agents’ knowledge that evolves, while the facts of the world remain unchanged. We develop a formal framework in which it is possible to express some interesting generalizations of this problem.

Dynamic Epistemic Logic DEL [23] provides a formal framework for the representation of knowledge and update of knowledge, and several recent approaches to multi-agent planning are based on it, starting with [4, 16]. While DEL provides a very expressive framework, it was unfortunately proven to be undecidable even for rather simple fragments of the language [1, 7]. Some decidable fragments were studied, most of which focused on public events [16, 24]. However, the gossip problem requires private communication. There exist other approaches on planning with uncertainty that do not use DEL. The framework presented in [15] allows us to reason about knowledge on literals in a multi-agent setting. A similar approach with beliefs can be found in [18]. While restricted to a single agent, the framework of [19] also deals with ‘knowing whether’ formulas (i.e., knowing \( p \) or knowing \( \neg p \)).

In this paper we provide a simple multi-agent epistemic logic that we call EL-O\(^8\) (Epistemic Logic of Observation), allowing us to model actions and epistemic planning tasks such as the gossip problem. Our logic provides special variables describing what agents can see. These variables determine indistinguishability relations, which allow us to interpret arbitrary formulas containing epistemic operators in the standard way and to reduce them to boolean formulas. By extending EL-O\(^8\) with dynamic operators, we are able to formalize the existence of a plan, giving the complexity result. We also study an encoding of actions into PDDL, the classical Planning Domain Definition Language [17]. This allows us to find a plan efficiently with a PDDL planner, which we do with extensions of the gossip problem and with the ‘exam problem’ where truth values of facts can also evolve.

The paper is organized as follows. In Section 2 we introduce our epistemic logic EL-O\(^8\). In Section 3 we give a formal definition of actions and planning tasks within our framework. In Section 4 we show how the existence of a plan can be encoded in the extension of EL-O\(^8\) with dynamic operators, and give the complexity result. In Section 5 we present the encoding into PDDL. In Section 6 we apply our framework to examples. We conclude in Section 7.

2 A simple epistemic logic

The logic EL-O\(^8\) is a fragment of Dynamic Epistemic Logic of Propositional Assignments and Observation DEL-PAO [12], which is a dialect of Dynamic Logic of Propositional Assignments DL-PA [13, 3]. We start by defining its language and semantics.

The basic ingredient of DEL-PAO are atoms of the form \( S_i p \), to be read as "agent \( i \) sees \( p \)" or "agent \( i \) knows whether \( p \)." We understand this as follows: when \( i \) knows whether \( p \) then either \( p \) is true and \( i \) knows that, or \( p \) is false and \( i \) knows that. We also allow for higher-order visibility with atoms of the form \( S_j S_i p \) (\( j \) sees whether \( i \) sees \( p \)), \( S_i S_j S_k p \) (\( k \) sees whether \( j \) sees whether \( i \) sees \( p \)), and so on. Along with visibility, DEL-PAO includes joint visibility and deals with a relation of ‘introspective consequence’ between agents. Here we simplify things and only consider individual visibility. We call the resulting logic EL-O\(^8\).

2.1 Language

Let \( Prop = \{ p_1, p_2, \ldots \} \) be a countable set of propositional variables and \( Agt = \{ 1, \ldots, n \} \) a finite set of agents. The set of visibility operators is \( OBS = \{ S_i : i \in Agt \} \). The set of all sequences of visibility operators is noted \( OBS^* \); elements of \( OBS^* \) are noted \( \sigma, \sigma' \), etc. An atom is any sequence of visibility operators \( S_i \), followed by a propositional variable. Formally,

\[
ATM = \{ \sigma p : \sigma \in OBS^*, p \in Prop \}
\]

Elements of \( ATM \) are noted \( \alpha, \alpha', \beta, \beta' \), etc. The depth of an atom is the number of visibility operators composing it.

Then the language of EL-O\(^8\) is defined by the following grammar:

\[
\varphi ::= \alpha | \neg \varphi | \varphi \land \varphi | K_i \varphi
\]
where \( \alpha \) ranges over ATM and \( i \) over Agt.

The formula \( K_i\varphi \) reads “agent \( i \) knows that \( \varphi \), based on what she observes.” The other boolean operators \( \bot, \top, \lor, \land \) and \( \leftrightarrow \) are defined as usual.

A boolean formula is a formula without the epistemic operator \( K_i \).

The set of all boolean formulas is noted \( \text{Fml} \). We note ATM\((\varphi)\) the set of atoms appearing in the boolean formula \( \varphi \). For example, ATM\((q \land S_i p)\) = \{\(q, S_i p\)\}. (Note that \( p \notin \text{ATM}(q \land S_i p)\).)

The visibility operator \( S_i \) can only be applied to atoms: it means that agent \( i \) sees the value of this atom (she sees whether it is true or false). On the other hand, the epistemic operator \( K_i \) can be applied to any formula and means that \( i \) knows that this formula is true. So \( K_i \neg p \) is a well-formed formula but \( S_i \neg p \) is not. In Section 2.2 we will show that our logic nevertheless allows us to reason about knowledge of complex formulas.

### 2.2 States and indistinguishability relations

Our worlds, alias states, are simply subsets of atoms, the ones that are currently true. Therefore the set of all worlds is \( 2^{\text{ATM}} \). We denote worlds by \( s, s', \ell, \) etc.

We interpret knowledge operators thanks to the visibility information contained in states. Intuitively, for an atom \( \alpha \), we have:

\[
\alpha \land S_i \alpha \leftrightarrow K_i \alpha \\
\neg \alpha \land S_i \alpha \leftrightarrow K_i \neg \alpha
\]

For complex formulas \( K_i \varphi \), we rely on accessible worlds just as in standard epistemic logic [10]: \( i \) knows that \( \varphi \) if \( \varphi \) is true in all worlds indistinguishable from the current one. Unlike standard epistemic logic however, the indistinguishability relation is not primitive but is constructed from visibility information: two worlds \( s \) and \( s' \) are indistinguishable for agent \( i \), noted \( s \sim_i s' \), if all atoms that \( i \) sees in \( s \) keep the same value in \( s' \). Formally:

\[
s \sim_i s' \iff \forall \alpha \in \text{ATM}, \text{if } S_i \alpha \in s \text{ then } s(\alpha) = s'(\alpha)
\]

where \( s(\alpha) = s'(\alpha) \) if and only if either \( \alpha \in s \) and \( \alpha \in s' \) or \( \alpha \notin s \) and \( \alpha \notin s' \). It is this construction that allows us to model knowledge in a more compact and natural way: all the information about indistinguishability is contained in the actual state and there is no need to explicitly give possible worlds and how they relate.

In epistemic logic, indistinguishability relations are usually assumed to be equivalence relations. While ours are clearly reflexive, they are neither transitive nor symmetric.\(^2\) To ensure transitivity and symmetry, we impose introspection: agents must always know what they know and what they do not know. In terms of visibility, this means that atoms of the form \( S_i S_i \alpha \) should always be true. Moreover, every agent should be aware of these facts. Generally: atoms of the form \( \sigma S_i S_i \alpha \) with \( \sigma \) a sequence of visibility operators, possibly empty, must be true. Let us define the (infinite) set of introspectively valid atoms as:

\[
\text{ATM}^{\text{INTR}} = \{ \alpha : \alpha \in \text{ATM} \land \alpha = \sigma S_i S_i \alpha' \}.
\]

We say that a state containing all these atoms is introspective and denote the set of all introspective states by INTR. Formally,

\[
\text{INTR} = \{ s \in 2^{\text{ATM}} : \text{ATM}^{\text{INTR}} \subseteq s \}.
\]

Clearly, \( s \cup \text{ATM}^{\text{INTR}} \in \text{INTR} \) for every state \( s \).

### Proposition 1 ([12]).

Relations \( \sim_i \) are equivalence relations on INTR.

\(^2\) For example, \( \varnothing \sim_i \{S_i p, p\} \) while \( \{S_i p, p\} \varphi \not\in \varnothing \) since \( \{S_i p, p\}(p) \neq \varnothing(p) \).

### Proposition 2 ([12]).

Let \( s \in \text{INTR} \) and \( s' \in 2^{\text{ATM}} \). If \( s \sim_i s' \) then \( s' \in \text{INTR} \).

We say that a boolean formula \( \varphi \) is in normal form if and only if \( \varphi \) does not contain an introspectively valid atom, i.e., no \( \alpha \in \text{ATM}(\varphi) \) belongs to \( \text{ATM}^{\text{INTR}} \).

### 2.3 Semantics

Formulas are interpreted over states \( s \in 2^{\text{ATM}} \). The truth conditions are as follows:

\[
\begin{align*}
\alpha &\quad \text{iff } \alpha \in s \\
\neg \alpha &\quad \text{iff } s \not\models \alpha
\end{align*}
\]

\[
s \models \varphi \quad \text{iff } s \models \varphi
\]

\[
s \models K_i\varphi \quad \text{iff } \text{for every } s' \in 2^{\text{ATM}} \text{ such that } s \sim_i s', s' \models \varphi
\]

Let us stress that truth conditions are defined on any state \( s \in 2^{\text{ATM}} \), even if we have seen that the \( \sim_i \) are equivalence relations only on introspective states. Moreover, in the truth condition of \( K_i\varphi \) we do not require the \( s' \) to be introspective: by Proposition 2, \( s' \) will belong to \( \text{INTR} \) if \( s \) is introspective. A formula \( \varphi \) is valid if and only if \( s \models \varphi \) for every \( s \in 2^{\text{ATM}} \); it is valid in \( \text{INTR} \) if and only if \( s \models \varphi \) for every \( s \in \text{INTR} \).

### Proposition 3.

For every EL-O\(^\text{R} \) formula \( \varphi \), there exists a formula \( \varphi' \) in normal form such that \( \varphi \leftrightarrow \varphi' \) is valid in \( \text{INTR} \).

This formula can be obtained by replacing every introspectively valid atom of \( \varphi \) by \( \top \).

### Proposition 4.

For every state \( s \in 2^{\text{ATM}} \), every boolean formula \( \varphi \) in normal form, and every \( \alpha \in \text{ATM}^{\text{INTR}} \), we have:

\[
s \setminus \{\alpha\} \models \varphi \quad \text{iff and only if } s \cup \{\alpha\} \models \varphi.
\]

By Proposition 4, the truth value of a formula in normal form is the same in \( s \) and in its introspective, but infinite counterpart \( s \cup \text{ATM}^{\text{INTR}} \).

### 2.4 From visibility to knowledge

We now show how to reduce EL-O\(^\text{R} \) formulas to boolean formulas. This will allow us to reduce multiagent planning problems that are expressible in EL-O\(^\text{R} \) to classical planning problems.

### Proposition 5 ([12]).

The following equivalences are valid.

\[
\begin{align*}
K_i\alpha &\leftrightarrow S_i \alpha \land \alpha \\
K_i \neg \alpha &\leftrightarrow S_i \alpha \land \neg \alpha
\end{align*}
\]

\[
K_i(\varphi \land \varphi') \leftrightarrow K_i\varphi \land K_i\varphi'
\]

\[
K_i\bigg( \bigvee_{\alpha \in A^+} \alpha \lor \bigvee_{\alpha \in A^-} \neg \alpha \bigg) \leftrightarrow \begin{cases} 
\bigvee_{\alpha \in A^+} K_i\alpha \
\bigvee_{\alpha \in A^-} K_i \neg \alpha
\end{cases}
\]

Moreover, the rule of replacement of equivalents preserves validity.

The last equivalence may appear curious to one familiar with epistemic logic. It is actually inherent to the notion of visibility: if an agent knows that \( p \) or \( q \) is true by looking at them, she immediately knows which one is true. This is discussed in [12] and [6]; the latter proposes an extension of DEL-PAO where the equivalence is invalid.

With the help of Proposition 5, starting with the innermost operator \( K_i \) (thanks to the rule of replacement of equivalents) we can...
reduce any EL-$Q^S$ formula to a boolean formula. Let us focus on reducing formulas of the form $K_{i_1} \ldots K_{i_m} \alpha$ to a conjunction of atoms. For example, we have by Proposition 5:

\[ K_{i_1} K_{i_2} \Leftrightarrow K_{i_1}(S_{i_2} \land p) \Leftrightarrow K_{i_1} S_{i_2} \land K_{i_2} \Leftrightarrow S_{i_1} S_{i_2} \land S_{i_1} p \land S_{i_2} p. \]

We generalize this: define the set of ‘epistemic atoms’ of an epistemic formula $\phi$ of the form $K_{i_1} \ldots K_{i_m} \alpha$ with $m \geq 0$ such that $\alpha$ is not introspectively valid as follows:

\[
EATM(\alpha) = \{ \alpha \} \\
EATM(K_{i_j} \phi) = EATM(\phi) \cup \{ \alpha : \alpha \in EATM(\phi) \land \alpha \text{ is not of the form } S_{i_0} \alpha \}
\]

The last line ensures that $EATM(\phi)$ does not contain any introspectively valid atom as we will be interested in formulas in normal form. Denote the conjunction of all these atoms by $\bigwedge_{\alpha \in EATM(\phi)} \alpha$.

**Proposition 6.** The following equivalence is valid in INTR.

\[ K_{i_1} \ldots K_{i_m} \alpha \Leftrightarrow \bigwedge_{\alpha \in \text{EATM}(K_{i_1} \ldots K_{i_m})} \alpha \]

**Lemma 1.** Let $p \geq 0$. Let $r_1, \ldots, r_p$ be such that $1 \leq r_1 < \ldots < r_p \leq m$. Then $EATM(K_{i_1} \ldots K_{i_{r_p}} \alpha) \subseteq EATM(K_{i_1} \ldots K_{i_m} \alpha)$.

In words, the set of epistemic atoms of $K_{i_1} \ldots K_{i_m} \alpha$ includes every epistemic atom of a formula composed of epistemic operators on a subsequence of $i_1, \ldots, i_m$.

We extend $EATM(.)$ to a conjunction of formulas as expected:

\[
EATM(\bigwedge_{i_1 \ldots i_m \in \text{Agt}} \bigwedge_{\alpha \in A} K_{i_1} \ldots K_{i_m} \alpha) = \bigwedge_{i_1 \ldots i_m \in \text{Agt}} \bigwedge_{\alpha \in A} EATM(K_{i_1} \ldots K_{i_m} \alpha)
\]

where $A \subseteq ATM$ is a set of atoms.

We will use these epistemic atoms in applications.

### 3 Epistemic planning with conditional effects

In this section, we formally define actions and planning tasks within our framework EL-$Q^S$. We assume that we perform planning tasks in fully observable, deterministic domains.

An **conditional action** is a pair $a = (\pre(a), \eff(a))$ where:

- $\pre(a) \subseteq \text{Fml}_{\text{bool}}$ is a boolean formula: the precondition of $a$;
- $\eff(a) \subseteq \text{Fml}_{\text{bool}} \times 2^{\text{ATM}} \times 2^{\text{ATM}}$ is a set of triples $ce$ of the form $(\text{cond}(ce), \text{eff}^+(ce), \text{eff}^-(ce))$: the conditional effects of $a$, where $\text{cond}(ce)$ is a boolean formula (the condition) and $\text{eff}^+(ce)$ and $\text{eff}^-(ce)$ are sets of added and deleted atoms respectively.

We impose that there is no conflicting effects: for every $ce_1, ce_2 \in \eff(a)$ with $\text{cond}(ce_1)$ and $\text{cond}(ce_2)$ consistent, $\text{eff}^+(ce_1) \cap \text{eff}^-(ce_2) = \emptyset$.

We are interested in actions whose executions are equivalent on introspective and non-introspective states. This requires some constraints. We say that the action $a$ is in *normal form* if and only if (1) the formulas $\pre(a)$ and $\text{cond}(ce)$ for every $ce \in \eff(a)$ are in normal form, and (2) for every $ce \in \eff(a)$, if $\alpha \in \text{eff}^+(ce) \cup \text{eff}^-(ce)$ then $\alpha$ is not introspectively valid.

For example, consider the conditional action $\pre(a)$ of flipping the truth value of the propositional variable $p$. It is described as $\pre(a) = (\pre(toggle), \eff(toggle))$ with $\pre(toggle) = \top$ and $\eff(toggle) = \{ (p, \neg p), (\neg p, p), \emptyset \}$. The conditions $p$ and $\neg p$ are inconsistent, thus not leading to conflict, and the action is in normal form since it does not involve higher-order knowledge.

**Example 1** (The gossip problem). Let $Agt = \{ 1, \ldots, n \}$ and $\text{Prop} = \{ s_i : i \in Agt \}$. Each propositional variable $s_i$ represents the secret of agent $i$. We are not interested in its value, but only in the knowledge of its value. (We suppose each $s_i$ is true.)

During the action $\text{call}_i^j$, agents $i$ and $j$ tell each other every secret they know among all $n$ secrets. We have $\text{call}_i^j = \langle \pre(\text{call}_i^j), \eff(\text{call}_i^j) \rangle$ with $\pre(\text{call}_i^j) = \top$ and $\eff(\text{call}_i^j) = \{ (S_i s_i \lor S_j s_j, \{ S_i s_i, S_j s_j \}, \emptyset), \ldots, (S_i s_j \lor S_j s_i, \{ S_i s_i, S_j s_j \}, \emptyset) \}$.

Intuitively, we add visibility of a secret to both agents if at least one knows it. (So we add variables that are already true; in this case there will be no effect.)

There is no possible conflict since call$_i^j$ has no negative effects, and is in normal form since it does not involve higher-order knowledge.

A conditional action $a$ (that may not be in normal form) determines a relation between states that is a partial function:

$$s R_a s'$$

iff (1) $1 \models \pre(a)$, and

(2) for every $ce \in \eff(a)$ such that $(\text{eff}^+(ce) \cup \text{eff}^-(ce)) \cap \text{ATM}_{\text{INTR}} \neq \emptyset$, $s \not\models \text{cond}(ce)$, and

(3) $s' = \left(s \setminus \bigcup_{ce \in \eff(a)} \text{cond}(ce) \right) \cup \bigcup_{ce \in \eff(a)} \text{eff}^+(ce)$.

In words, an action adds and removes atoms as expected if its pre-condition is satisfied and none of its conditional effects involving an introspective atom can be triggered.

**Proposition 7.** For every action $a$, there exists an action $a'$ in normal form such that for every $s, t \in \text{INTR}$, we have:

$$s R_{a'} t \text{ if and only if } s R_a t.$$

Actions in normal form can be obtained by the following modification. (1) for every conditional effect $ce \in \eff(a)$ such that $(\text{eff}^+(ce) \cup \text{eff}^-(ce)) \cap \text{ATM}_{\text{INTR}} \neq \emptyset$, replace $\pre(a)$ by $\pre(a) \land \neg \text{cond}(ce)$ and remove $ce$ from $\eff(a)$, and (2) put the resulting $\pre(a)$ and $\text{cond}(ce)$, for every $ce \in \eff(a)$, in normal form.

**Proposition 8.** For every $s, t \in 2^{\text{ATM}}$, every action $a$ in normal form, and every $\alpha \in \text{ATM}_{\text{INTR}}$, we have:

$$s \setminus \{ \alpha \} R_a t \cup \{ \alpha \} \text{ if and only if } s \cup \{ \alpha \} R_a t \cup \{ \alpha \}.$$

As with Proposition 4 for formulas, Proposition 8 implies that if there exists an execution of a from $s$ that leads to $t$, then there exists an execution of the same action from $s \cup \text{ATM}_{\text{INTR}} \cup \text{ATM}_{\text{INTR}}$. This is ensured by the fact that actions in normal form neither test nor add nor remove introspectively valid atoms.

We say that a state $s$ is *reachable* from a state $s_0$ via a set of conditional actions $\Ac$ if there is a sequence of actions $\{a_1, \ldots, a_m\}$ from $\Ac$ and a sequence of states $(s_0, s_1, \ldots, s_m)$ with $m \geq 0$ such that $s_0 = s_0$, $s = s_m$ and $s_{i-1} R_{a_i} s_i$ for every $s_k$ such that $1 \leq k \leq m$.

A simple epistemic planning task is a triple $(\Ac, s_0, \text{Goal})$ where $\Ac$ is a finite set of actions, $s_0 \subseteq \text{ATM}$ is a finite state (the initial state), and $\text{Goal} \in \text{Fml}_{\text{bool}}$ is a boolean formula. It is solvable if at least one state $s$ such that $s \models \text{Goal}$ is reachable from $s_0$ via $\Ac$; otherwise it is unsolvable. It is in normal form if and only if (1) every action $a \in \Ac$ is in normal form, and (1) the formula $\text{Goal}$ is in normal form.

**Example 2** (Example 1, ctd.). The planning task corresponding to the gossip problem is $G_j = \langle \text{Act}^{G_j}, s_0^{G_j}, \text{Goal}^{G_j} \rangle$ with
• \( \text{Act}^{G_i} = \{ \text{call}_i', i, j \in \text{Agt} \text{ and } i \neq j \}; \)
• \( S^{G_i}_0 = (S_i, S_j) : i \in \text{Agt} \cup \{S_i : i \in \text{Agt}\}; \)
• \( \text{Goal}^{G_i} = \bigwedge_{i,j \in \text{Agt}} K_i S_j. \)

In the initial state, every agent knows her own secret and none of the other secrets. Secrets are also true initially, so that, since no action can change the truth value of \( s \), \( \text{Goal}^{G_i} \) is equivalent to \( \bigwedge_{i,j \in \text{Agt}} K_i S_j. \)

This planning task is trivially in normal form.

4. Dynamic extension and complexity results

Consider the planning task \( \langle \text{Act}, s_0, \text{Goal} \rangle \). In this section, we introduce an extension of EL-O\(^S\) with dynamic operators, which we call DEL-PAO\(^S\) (Dynamic Epistemic Logic of Propositional Assignments and Observation without common knowledge). We show how actions from \( \text{Act} \) can be encoded into DEL-PAO\(^S\) programs. Then we prove that the solvability of \( \langle \text{Act}, s_0, \text{Goal} \rangle \) is in PSPACE by showing that it can be polynomially reduced to a DEL-PAO\(^S\) model checking problem.

4.1 A simple dynamic epistemic logic

The language of DEL-PAO\(^S\) extends the language of EL-O\(^S\) with the dynamic operator \( \pi \), with \( \pi \) a program: the formula \( \langle \pi \rangle \phi \) reads “there exists an execution of \( \pi \) after which \( \phi \) is true.”

The syntax of programs is defined by the following grammar:

\[
\pi ::= +\alpha | -\alpha | \pi \pi | \pi \mid \pi | \pi \pi | \phi
\]

where \( \alpha \) ranges over the set of atoms.

Atomic programs \( +\alpha \) and \( -\alpha \) are assignments; they respectively set the value of the atom \( \alpha \) to true and false. Complex programs are composed of sequences of instructions \( \pi \), non-deterministic choice between instructions \( \pi \mid \pi \), repetitions \( \pi \pi \) and tests \( \phi \).

The dual operator \( [\pi] \phi \equiv -[\pi] \neg \phi \) (“after every execution of \( \pi \), \( \phi \) is true”) is defined as usual. Moreover, if \( \phi \) then \( \pi \) abbreviates \( \neg \phi \pi \mid \neg \phi \) and if \( \phi \) then \( \pi \) else \( \pi' \) abbreviates \( (\phi?; \pi) \mid (\neg \phi?; \pi'). \)

Semantically, a program is interpreted as a binary relation \( R_\pi \) on states, such that:

\[
R_\pi = \{(s, s') : \text{if } s' = s \cup \{\alpha\} \text{ then } R_\pi s' \text{ if } s' \in \{\alpha\} \text{ and } \alpha \notin \text{ATM}_{\text{INTR}} \}
\]

\[
R_\pi = \{(s, s') : \text{if } s' = s \setminus \{\alpha\} \text{ then } R_\pi s' \text{ if } s' \notin \{\alpha\} \}\]

\[
R_\pi = \{(s, s') : \text{if } s' = s \cup \{\alpha\} \text{ then } R_\pi s' \text{ if } s' \in \{\alpha\} \text{ and } \alpha \notin \text{ATM}_{\text{INTR}} \}
\]

\[
R_\pi = \{(s, s') : \text{if } s \text{ and } s' \mid \phi \text{ then } R_\pi s' \}
\]

The truth condition of the new operator is then:

\[
\langle \pi \rangle \phi \iff \exists s' \in 2^{\text{ATM}} \text{ such that } R_\pi s' \text{ and } s' \models \phi.
\]

In words, \( \langle \pi \rangle \phi \) is true if there is a state reachable by executing \( \pi \) where \( \phi \) is true. An assignment \( +\alpha \) or \( -\alpha \) updates the state by adding or (unless introspectively valid) removing \( \alpha \); a sequential composition \( \pi_1 \pi_2 \) executes first \( \pi_1 \) and then \( \pi_2 \); a nondeterministic composition \( \pi_1 \mid \pi_2 \) takes the union of relations for \( \pi_1 \) and for \( \pi_2 \); an iteration \( \pi^* \) reaches any state attainable if we repeat \( \pi \) an arbitrary number of times; a test \( \phi? \) stays in the same state if \( \phi \) is true there (otherwise the program fails and produces no result world).

Observe that unlike the epistemic operators \( K_i \), the evaluation of dynamic operators may terminate in a non introspective state. However, trying to remove an introspectively valid atom (e.g. by executing \( -S_i \neg \phi \)) will lead to a failure of the program because of the definition of \( R_{\text{INTR}} \): a program starting in \( \text{INTR} \) will never exit \( \text{INTR} \).

4.2 Storing variables

The conditional effects of the actions that we have defined in Section 3 are produced in parallel. We have to simulate this in DEL-PAO\(^S\) by sequential composition. We therefore have to take care that the truth value of no condition is modified by an effect. To achieve this, we store the values of our conditions before executing our action, and evaluate such values. This problem does not arise in PDDL where all conditions are checked before any effects are produced.

We use new atomic variables noted \( c \), called storage variables, which we suppose do not appear in the planning task under concern. Then the program storing the value of a formula is defined as:

\[
\text{str}(\{c_1, \ldots, c_m\}) = -c_1; \ldots; -c_m.
\]

4.3 Encoding of actions

Intuitively, an action is a DEL-PAO\(^S\) program, only executed if the precondition is fulfilled, applying each conditional effect whose condition is satisfied. For example, the action \( \text{toggle} \) (flipping the value of the variable \( p \)) corresponds to the program \( \text{str}(p, c_1); \text{str}(-p, c_2); \text{if } c_1 \text{ then } -p; \text{if } c_2 \text{ then } +p \).

We first show how to perform one conditional effect \( e \) whose condition’s value was stored in \( e \).

\[
\text{exeAct}(\text{ceff}(e), e) = \text{if } e \text{ then } +\alpha_1; \ldots; +\alpha_m; -\beta_1; \ldots; -\beta_l \text{ where ceff}(e) = \{\alpha_1, \ldots, \alpha_m\} \text{ and ceff}(e) = \{\beta_1, \ldots, \beta_l\}. \]

Note that the ordering of atoms is not important since \( \text{ceff}(e) \cap \text{ceff}(e) = \emptyset. \) Then we can associate to action \( e \) the DEL-PAO\(^S\) program \( \text{exeAct}(a) \):

\[
\text{exeAct}(a) = \text{pre}(a)?; \text{str}(\text{cmd}(e_1), c_1); \ldots; \text{str}(\text{cmd}(e_m), c_m); \text{exeCE}(e_1, c_1); \ldots; \text{exeCE}(e_m, c_m); \text{rst}(\{c_1, \ldots, c_m\});
\]

with \( \text{eff}(a) = \{e_1, \ldots, e_m\}. \) The ordering of effects is not important since we test values of storage variables.

Proposition 10. For every \( s, t \in 2^{\text{ATM}} \) such that \( s \) does not contain any storage variable, and every action \( a \) in normal form, the program \( \text{exeAct}(a) \) behaves like \( a \):

\[
s R_a t \iff \text{if and only if } s R_{\text{exeAct}(a)} t.
\]

Intuitively, our program \( \text{exeAct}(a) \) is divided in four parts which are executed in sequence:
Example 3 (Example 1, ctd.). The action \textit{call}$_j$, for any \(i, j \in \text{Agt}t\), is associated to the program:

\[
\text{exeAct} (\textit{call}$_j$) = \top; \\
\text{str}(S; S_0 \vee S_i; c_1); \ldots; \text{str}(S; S_n \vee S_j; s_n; c_m); \\
\text{if } c_1 \text{ then } +S; s_1; +S; s_1; \\
\ldots; \\
\text{if } c_m \text{ then } +S; s_n; +S; s_n; \\
\text{rst}((c_1, \ldots, c_m))
\]

Note that in this case, \textit{pre}(\textit{call}$_j$)? can clearly be dropped.

### 4.4 Solvability of a planning task

Now that we have defined the encoding of actions, we can capture the solvability of a planning task in DEL-PAO\(^5\).

**Proposition 11.** A planning task \((\text{Act}, s_0, \text{Goal})\) in normal form such that \(s_0\) does not contain any storage variable is solvable if and only if:

\[
s_0 \models (\bigcup_{a \in \text{Act}} \text{exeAct} (a))^* \text{Goal}.
\]

Intuitively, our formula reads “there exists an execution of \(\bigcup_{a \in \text{Act}} \text{exeAct} (a)^*\) after which \text{Goal} is true.” The program \(\bigcup_{a \in \text{Act}} \text{exeAct} (a)^*\) non-deterministically chooses an action \(a\) from \text{Act} and executes the corresponding program \text{exeAct}(a), then repeats this a finite number of times. This produces a sequence of actions, i.e., a plan.

We do not impose that \(s_0\) is introspective as it would make it infinite; this is not necessary by Proposition 8 since the planning task is in normal form: if there is an execution of \(\bigcup_{a \in \text{Act}} \text{exeAct} (a)^*\) starting from the introspective state \(s_0 \cup \text{ATM}_{\text{INT}}\) and leading to a state satisfying \text{Goal}, then there is one starting from the non-introspective state \(s_0\) and leading to a state satisfying \text{Goal}.

**Proposition 12.** Deciding the solvability of a planning task with DEL-PAO\(^5\) is \text{PSPACE}-complete.

The lower bound comes from classical planning \([5\); the upper bound is given by Proposition 11, where the problem is reduced to a model checking problem of DEL-PAO\(^5\), a fragment of DEL-PAO whose model checking problem is in \text{PSPACE}.

This result compares favorably to DEL-based epistemic planning, which is undecidable even for simple fragments \([1, 7\). The difference is due to the simplicity of our underlying epistemic logic (cf. Proposition 5) as well as to the limited expressivity of our actions: we can basically model private announcements, while DEL has more general event models.

## 5 Encoding into PDDL

In this section, we present a method for encoding planning problems defined in DEL-PAO\(^5\) into PDDL. As already observed, in PDDL, we do not need to store conditions as we were obliged to do in DEL-PAO\(^5\). Consider a planning task \((\text{Act}, s_0, \text{Goal})\). We show how to encode boolean formulas and actions in PDDL.

### 5.1 Translation of formulas

Some PDDL requirement flags should be set depending on the form of conditions \(cnd(ce)\) of conditional effects \(ce\) of actions and of the formula \text{Goal}:

- the default flag :strips for conjunctions;
- the flag :negative-preconditions for negations;
- the flag :disjunctive-preconditions for disjunctions, and disjunctions, if used to simplify writing.

Given a boolean formula \(\varphi \in \text{Prop}_{\text{bool}}\), we define a recursive function \(\text{trPDDL}(\varphi)\) which returns the encoding of \(\varphi\) into PDDL:

\[
\text{trPDDL}(S_1, \ldots, S_m, p) := \begin{cases} 
(p) & \text{if } m = 0 \\
(S\neg m & i_1 \ldots i_m & p) & \text{otherwise} 
\end{cases}
\]

\[
\text{trPDDL}(\neg \varphi) := (\text{not PDDL}(\varphi)) \\
\text{trPDDL}(\varphi \land \varphi_2) := (\text{and PDDL}(\varphi_1, \varphi_2)) \\
\text{trPDDL}(\varphi \lor \varphi_2) := (\text{or PDDL}(\varphi_1, \varphi_2))
\]

with \(p \in \text{Prop}, m \geq 0\), and \(i_1, \ldots, i_m \in \text{Agt}\).

In words, a visibility atom \(\alpha = S_1, \ldots, S_m, p\) is encoded by a special fluent with \(m+1\) parameters. If \(m = 0\), then the propositional variable \(p\) is encoded as a fluent without parameters. We note \(\text{trPDDL}(\alpha)\) the translation of an atom \(\alpha\) in the general case (\(p\) or \(S_1, \ldots, S_m, p\)). Other boolean operators are encoded as expected.

The initial state \(s_0\) is trivially encoded as a set of fluents thanks to \(\text{trPDDL}(\alpha)\). \text{Goal} and the preconditions of every action can be encoded using \(\text{trPDDL}(\varphi)\) since they are all boolean formulas.

### 5.2 Encoding of actions

The requirement flag :conditional-effects must be set.

Consider an action \(a\). For every \(ce \in \text{eff} (a)\) with \(\text{ceff}^+(ce) = \{\alpha_1, \ldots, \alpha_m\}\) and \(\text{ceff}^-(ce) = \{\beta_1, \ldots, \beta_l\}\), we add the conditional effect:

\[
\{\begin{align*} 
\text{when } \text{trPDDL}(\text{cnd}(ce)) \& \text{and PDDL}(\alpha_1) \ldots \text{and PDDL}(\alpha_m) \\
\text{not PDDL}(\beta_1) \ldots \text{not PDDL}(\beta_l) \nonumber 
\end{align*} \}
\]

Note that, again, the ordering is not important.

---

3 The version of DEL-PAO presented in [12] does not include the iteration (represented by the star ‘\^\*’) in the language of programs. However, a more general result, including the star and with a \text{PSPACE} model checking, can be found in [8].
Example 4 (Example 1, ctd.). The action call is coded in PDDL as follows:

\(\text{::action call-1-2}\\n\text{:effect (and}\\n\text{(\{when (or (\text{s-1 1 sl}) (\text{s-1 2 sl})) (and (\text{s-1 1 sl}) (\text{s-1 2 sl}))))}\\n\text{...}\\n\text{(when (or (\text{s-1 1 sn}) (\text{s-1 2 sn})) (and (\text{s-1 1 sn}) (\text{s-1 2 sn}))))}\\n\text{)}\)

This is the direct encoding of a call into PDDL. Remark that we could generalize it to any \(i\) and \(j\) by adding the line ‘:parameters’ (\(!? i j)’ and replacing every ‘(\text{s-1 1 .}’ by ‘(\text{s-1 ?i .}’ and every ‘(\text{s-1 2 .}’ by ‘(\text{s-1 ?j .}’. We will use the latter in experiments because of its succinctness.

Almost all planners from last International Planning Competition (IPC 2014) handle conditional effects and negative preconditions, and most of them handle disjunctive preconditions. For experiments, we chose to use the planner FDSS-2014 [21] that was satisfying all these preconditions.

6 Applications

In this section, we first study the ‘exam problem’ (a simple illustrative example concerning privacy of information), then generalizations of the gossip problem. We sometimes write simply ‘a’ for ‘execute(a)’ when used within dynamic operators (\(\langle\) and [\(\rangle\).

6.1 The exam problem

Suppose we have two agents: a teacher and a student. The teacher has prepared the exam and keeps it in her office; the goal of the student is to know the exam topic, but without the teacher seeing her doing this. To achieve this goal, the student must enter the teacher’s office, read the exam while the teacher is not inside, and exit the office.

Let the corresponding planning task be \(\text{Exam} = \langle\text{Act}_\text{Exam}, \text{Prop}_\text{Exam}, \text{Goal}_\text{Exam}\rangle\). Let \(\text{Agt} = \{t,s\}\) and \(\text{Prop} = \{\text{exam, open, in}_i, \text{in}_t\}\). Agent \(t\) is the teacher and agent \(s\) is the student. The variable exam represents the topic of the exam. Like secrets in the gossip problem, its value is not relevant and we only reason about the knowledge of it (we will assume it is true). The variable open reads “the teacher’s office is open”, and \(\text{in}_i\) for \(i\) an agent, “agent \(i\) is in the teacher’s office”.

Initially, we assume the office is empty and the door is closed:

\(s_{\text{Exam}} = \{\text{exam}\}\).

As we said, the goal for the student is to know the exam’s topic without being caught by the teacher. The goal is \(s_\text{exam} \land \neg K_t(s\text{, exam} \land \neg \text{in}_s\). In terms of visibility atoms, this becomes:

\(\text{Goal}_\text{Exam} = s_\text{exam} \land \neg s_t(s\text{, exam} \land \neg \text{in}_s\).

We study two variants of this problem with different actions.

Vigilant teacher. In this first version, we suppose the teacher always closes her office door when leaving. The set of actions is:

\(\text{Act}_\text{Exam} = \{\text{openAndGoln}_1, \text{goOutAndClose}_1, \text{goIn}_i, \text{goOut}_i, \text{readExam}_i\}\),

where

\(\text{openAndGoln}_1 = \langle \text{\{in}_1 \land \neg s_t(s\text{, exam}\}, \{\langle t, \{\text{open}, \text{in}_i, s_t(s\text{, exam}\}, \emptyset\}\}\rangle\).

goOutAndClose\(_i = \langle \text{\{in}_i \land \neg s_t(s\text{, exam}\}, \{\langle t, \{\text{\{open}, \text{in}_i, s_t(s\text{, exam}\}, \emptyset\}\}\rangle\).

goIn\(_1 = \langle \text{\{in}_1 \land \neg s_t(s\text{, exam}\}, \{\langle t, \{\text{\{open}, \text{in}_i, s_t(s\text{, exam}\}, \emptyset\}\}\rangle\).

goOut\(_1 = \langle \text{\{in}_1 \land \neg s_t(s\text{, exam}\}, \{\langle t, \{\text{\{open}, \text{in}_i, s_t(s\text{, exam}\}, \emptyset\}\}\rangle\).

Action openAndGoln\(_1\) makes the teacher open and enter the room, and thus watch the exam. Action goOutAndClose\(_i\) makes her leave and close the room; she cannot watch the exam anymore. We add the precondition \(\neg s_t(s\text{, exam}\) to ensure that the teacher cannot leave if she has witnessed the student see the exam, so that she cannot forget this fact. For the student, goIn\(_i\) and goOut\(_i\) makes her enter and leave the office, with the precondition that it is open; readExam\(_i\) makes her see the exam topic, acquiring the knowledge on its value.

In this case, no plan exists reaching the goal. Indeed, the student can only enter the room if the door is open, which can only happen when the teacher is inside the room. Therefore the student cannot read the exam’s topic without the teacher knowing it: \(s_\text{exam} \rightarrow K_s(s_\text{exam}\). This was confirmed by experiments: FDSS-2014 cannot find a plan.

Inattentive teacher. Now we assume that the teacher can leave the room without closing the door. This is done by dividing actions openAndGoln\(_i\), and goOutAndClose\(_i\) in two parts:

- we replace openAndGoln\(_i\) by:

\(\text{open}_i = \langle \neg \text{open}, \{\langle t, \{\text{open}\}, \emptyset\}\rangle\).

\(\text{goIn}_i = \langle \text{\{open}, \{\langle t, \{\text{in}_i, s_t(s\text{, exam}\}, \emptyset\}\}\rangle\).

- we replace goOutAndClose\(_i\) by:

\(\text{goOut}_1 = \langle \text{\{in}_i \land \neg s_t(s\text{, exam}\}, \{\langle t, \{\text{\{open}, \text{in}_i, s_t(s\text{, exam}\}, \emptyset\}\}\rangle\).

\(\text{close}_i = \langle \text{\{open}, \{\langle t, \{\text{\{open}\}\}\rangle\}\rangle\).

Thus the set of actions becomes:

\(\text{Act}_\text{Exam} = \{\text{open}_i, \text{close}_i, \text{goIn}_i, \text{goOut}_i, \text{goOut}_i, \text{readExam}_i\}\).

In this setting, the problem becomes solvable: for example, the plan \(\text{open}_1, \text{goIn}_1; \text{goIn}_1, \text{goOut}_1, \text{readExam}_1; \text{goOut}_1\) is a solution plan. More mundanely, the planner finds the shortest plan:

\(\text{open}_1, \text{goIn}_1, \text{readExam}_1, \text{goOut}_1\).

In these two examples, we are more interested in the existence of a plan than in the plan itself: the first variant is safe for the teacher, while the second is not.

6.2 The generalized gossip problem

In this section, we present a formalisation of a generalisation of the gossip problem in our framework. A study of this problem and its variants can be found in [9].

The generalized gossip problem. We model the generalized gossip problem, introduced in [14], as a planning task \(G_D = \langle\text{Act}_D, \text{Prop}_D, \text{Goal}_D\rangle\). In this generalization, the goal is not only for every agent to know every secret, but also every agent must know this fact, and every agent must know that, and so on until a given depth \(D \geq 1\). Let \(\text{Agt} = \{1, \ldots, n\}\) and \(\text{Prop} = \{s_i : i \in \text{Agt}\}\). In terms of knowledge, the goal of the generalized gossip problem is: \(\varphi_D = \bigwedge_{i \in \text{Agt}} K_{i_1} \cdots K_{i_D} \bigwedge_{i \in \text{Prop}} s_i\).
We have seen in Section 2.4 how to express this with a boolean formula, thanks to our epistemic atoms:

$$\text{Goal}^{G_D} = \bigwedge \text{EATM}(\varphi_{G_0})$$

Recall that introspectively valid atoms are not included in $\text{EATM}(\varphi)$, thus the goal is in normal form.

The initial state and the set of actions stay the same:

$$G^0_{s_0} = \{ S_i : i \in Agt \} \cup \{ S_i : i \notin Agt \}$$

$$\text{Act} = \{ \text{call}_i : i, j \in \text{Agt}, i \neq j \}$$

The preconditions of calls also remain unchanged: $\text{pre}(\text{call}_i) = T$.

However, their effects are different. Agents will not only transmit secrets but also knowledge of secrets. They will also learn the higher-order knowledge we need in the gossip problem when exchanging secrets. For example, if $i$ knows the secret of $\ell$ and $j$ calls $i$, $j$ will learn the secret of $\ell$, but also that $i$ knows it; $j$ will learn that $j$ knows that $\ell$ and so on until depth $D$. Moreover, if $i$ knows that $i_j$ knows the secret of $\ell$, then $j$ learns that $i_j$ knows the secret of $\ell$, but also that $i$ knows that, and so on until depth $D$.

As an example, suppose $D = 4$ and we have $K(4)$. Then after the call between $i$ and $j$, we will have, e.g., $K(K_i K_i K_i s)$, $K(K_i K_i K_i s)$, $K K K_i K_i K_i K_i s$, and so on, that is, any combination of $K_i$ and $K_j$ followed by $K_j$, for a maximum depth of $D$.

For a given integer $m$ and two agents $i$ and $j$, we note $\{S_i, S_j\} \leq m$ the set of non-empty non-intrusive subsets of visibility operators $S_i$ and $S_j$ of length at most $m$. For instance, $\{S_i, S_j\} \leq 2 = \{S_i, S_j S_i S_j S_i S_j S_i \}$.

Thus we have that, during a call between $i$ and $j$, if $i$ or $j$ knows that $K_{i_1} \cdots K_{i_m} s$, i.e., if $K_{i_1} \cdots K_{i_m} s \cap K_{i_1} \cdots K_{i_m} s$ is true, then $\sigma S_{i_1} \cdots S_{i_m} s$ for every $\sigma \in \{S_i, S_j\} \leq m$ becomes true formally:

$$\text{eff}(\text{call}_i) = \{ \bigwedge \text{EATM}(K_{i_1} \cdots K_{i_m} s) \} \cup \bigwedge \text{EATM}(K_{i_1} \cdots K_{i_m} s), \{\sigma S_{i_1} \cdots S_{i_m} s : \sigma \in \{S_i, S_j\} \leq D-m, \emptyset \} : 0 \leq m < D \land i_1, \ldots, i_m, \ell \in \text{Agt} \text{ such that for every } 1 \leq k < m, i_k \neq i_{k-1}, \text{ and } i \neq i_1 \land j \neq i \}$$

Consecutive agents in $S_i, S_j, S_k$ are required to be different so that we do not involve any introspectively valid atom and we obtain an action in normal form. If we take $D = 1$, we retrieve our definition of $\text{call}_i$ from Example 1 (with tests of secrets that could be omitted).

We require knowledge instead of visibility, i.e., $\bigwedge \text{EATM}(K_{i_1} \cdots K_{i_m} s)$ instead of just $S_i S_j S_k$, so that agents only know what they know. For example, we do not want 1 to see whether 2 knows the secret of 3 without 2 knowing the secret of 3; it would imply that 1 watches 2, and that if 2 learns the secret of 3 during a call, 1 will know this even if she did not participate in this call.

### Proposition 13.

The equivalence $\text{call}_i \varphi \leftrightarrow \neg \text{call}_i \neg \varphi$ is valid.

This is due to calls being deterministic: executing a call always leads to exactly one state.

### Lemma 2.

The following formulas are valid.

1. $S_{i_1} \cdots S_{i_m} s \leftrightarrow [\text{call}_i] S_{i_1} \cdots S_{i_m} s$ if $i \neq i_1$ and $j \neq i_1$.
2. $S_{i_1} \cdots S_{i_m} s \rightarrow [\text{call}_i] S_{i_1} \cdots S_{i_m} s$ for any $i, j$.

(1) means that when $i_1$ is not involved in a call, her knowledge does not evolve. Indeed, along with Proposition 13, it implies:

- $S_{i_1} \cdots S_{i_m} s \rightarrow [\text{call}_i] S_{i_1} \cdots S_{i_m} s$
- $S_{i_1} \cdots S_{i_m} s \rightarrow \neg [\text{call}_i] \neg S_{i_1} \cdots S_{i_m} s$

if $i \neq j$ and $j \neq i$. (2) means that knowledge of agents cannot decrease with a call. Both lines are deduced from the definition of calls.

While the original gossip problem with $n \geq 4$ agents can be solved in $2n - 4$ calls [2, 22, 11], the generalized gossip problem can be solved in at most $(D + 1)(n - 2)$ calls [14]. For instance, suppose $D = 2$ and $n = 5$, then the sequence $\text{call}_1; \text{call}_2; \text{call}_3; \text{call}_1; \text{call}_2; \text{call}_3; \text{call}_1; \text{call}_2; \text{call}_3$ is a solution with $3 \times 3 = 9$ calls. Our experiments have confirmed that the protocol given in [14] is optimal for $D = 2$ and $n \leq 5$.

### Negative goals.

We now introduce an extension of the generalized gossip problem where goals can be ‘negative’. We write it $G_{\neg \text{neg}} = \langle \text{Act}, G_{\neg \text{neg}}, \text{Goal}^{G_{\neg \text{neg}}}, \text{s}_0^{G_{\neg \text{neg}}}, \text{G}_{\neg \text{neg}} \rangle$. In this variant, we change the goal and impose that some agents do not know some secrets, or some knowledge of secrets, at the end of the sequence of calls. For example, we want 1 not to know the secret of 2, or 1 not to know that 2 knows the secret of 3. The action set, the calls, and the initial state remain the same: $\text{Act}^{G_{\neg \text{neg}}} = \text{Act}^{G_D}$ and $\text{s}_0^{G_{\neg \text{neg}}} = \text{s}_0^{G_D}$.

We note $\text{Goal}^{G_{\neg \text{neg}}}$ the goal of the generalized gossip problem where only atoms from $A$, such that $A \cap \text{ATM}_{\text{INTRA}} = \emptyset$, are false.

Formally:

$$\text{Goal}^{G_{\neg \text{neg}}} = \left( \bigwedge_{\alpha \in \text{EATM}(\varphi_{G_I}) \cap A} \alpha \right) \land \left( \bigwedge_{\alpha \in A} \neg \alpha \right)$$

We present several properties of the gossip problem that will be useful in deciding solvability of $G_{\neg \text{neg}}$.

### Lemma 3.

Let $m \geq 2$ be an integer. Let $D \geq m$. Take $m+1$ agents $i_1, i_2, i_3, \ldots, i_m, \ell \in \text{Agt}$ such that $i_1, i_2$ and $i_3$ are distinct. We have:

$$\neg S_{i_1} S_{i_2} S_{i_3} \cdots S_{i_m} s \land [\text{call}_{i_1}] S_{i_1} S_{i_2} S_{i_3} \cdots S_{i_m} s \rightarrow \left\{ \begin{array}{ll}
\bigwedge \text{EATM}(K_{i_1} K_{i_3} \cdots K_{i_m} s) \land \bigwedge \text{EATM}(K_{i_2} K_{i_3} \cdots K_{i_m} s) & \text{if } j = i_2 \\
\bigwedge \text{EATM}(K_{i_1} K_{i_3} \cdots K_{i_m} s) & \text{otherwise}
\end{array} \right.$$
Proof. We prove it by induction on the sequence of calls. We are only interested in cases where $S_1, S_2, S_3, \ldots, S_{m+1}$ is not Introspective.

Base case: initial situation. We prove:

$$G^{G_{neg}}_0 \models S_1 S_2 S_3 \ldots S_{m+1} \rightarrow S_1 S_2 \ldots S_{m+1}.$$

This is trivially true because only atoms of the form $S_i$ are true initially.

Inductive case. Suppose:

$$G^{G_{neg}}_0 \models \square([S_1 S_2 S_3 \ldots S_{m+1} \rightarrow S_1 S_2 \ldots S_{m+1}]).$$

We prove that for an arbitrary $s$:

$$s \models (S_1 S_2 S_3 \ldots S_{m+1} \rightarrow S_1 S_2 \ldots S_{m+1}) \rightarrow$$

$$\vdash \diamond\square([S_1 S_2 S_3 \ldots S_{m+1} \rightarrow S_1 S_2 \ldots S_{m+1}]).$$

First suppose $i_1$ is not involved in the new call, that is, $i_1 \neq i$ and $i_1 \neq j$. We know by (1) of Lemma 2 that her knowledge (every atom beginning with $S_i1$) does not evolve. Thus the implication stays true.

Now suppose $i_1$ is involved in the new call; without loss of generality, suppose $i = i_1$. By (2) of Lemma 2, we know that a true atom stays true after a call. Then (3) is equivalent to:

$$s \models \neg S_1 S_2 S_3 \ldots S_{m+1} \wedge \vdash\square([S_1 S_2 S_3 \ldots S_{m+1} \rightarrow S_1 S_2 \ldots S_{m+1}]).$$

In words, if call$^i$ makes $S_1 S_2 S_3 \ldots S_{m+1}$ true, then either $S_1 S_2 S_3 \ldots S_{m+1}$ was true or it becomes true.

By Lemma 3, we know that the premise of (4) implies either $\lor EATOM(K_1, K_2 \ldots K_{m+1})$ if $j = i_2$, or $\lor EATOM(K_1, K_2 \ldots K_{m+1})$ otherwise. It is possible prove that each of these three statements implies either $S_1 S_2 S_3 \ldots S_{m+1}$ or $\square([S_1 S_2 S_3 \ldots S_{m+1} \rightarrow S_1 S_2 \ldots S_{m+1}]).$

Therefore $S_1 S_2 S_3 \ldots S_{m+1} \rightarrow S_1 S_2 \ldots S_{m+1}$ is preserved by call$^i$, hence the result. □

With this in mind, we look at some specific examples of goals.

The goal $Goal^{(G_{neg})}_{(S_{i_1} S_{i_2})}$, which only 1 does not know the secret of 2, will always be reachable for $D = 1$ and $n \geq 3$. For example, FDSS-2014 returns the plan call$^1$; call$^2$; call$^3$; call$^4$; for $n = 5$. More generally, the following protocol gives a solution:

1. call$^i$ for every $i \in Agt \setminus \{2\}$
2. solve $GD$ for $D = 1$ and $Agt = \{2, \ldots, n\}$

However, it will never be reachable for $D \geq 2$ and $n \geq 3$: by contraposition of Proposition 14, if $S_{i_1}$ is false then $S_1 S_2 S_3$ is false, and we cannot reach the goal where only $S_1 S_2$ is false. FDSS-2014 indeed cannot find any plan for $D = 2$ and $n \leq 4$. (It is obviously unsolvable for any depth when $n = 2$, since the only available action, call$^1$, establishes $S_{i_1}$.)

Now suppose we have $D \geq 2$ and we want 1 to not know whether 2 knows the secret of 3 (but we do want 2 to know the secret of 3): our goal is $Goal^{(G_{neg})}_{(S_{i_1} S_{i_2})}$. The following protocol produces a solution for $D \geq 3$:

1. call$^2$ for every $i \in Agt \setminus \{3\}$
2. solve $GD$ for $D = 2$ and $Agt = \{1, 3, 4, \ldots, n\}$
3. call$^i$ for every $i \in Agt \setminus \{1\}$

One of the plans FDSS-2014 finds is call$^1$; call$^2$; call$^3$; call$^4$; call$^1$; call$^2$; call$^3$; for $n = 5$. Again by Proposition 14, we know that if $S_1 S_2 S_3$ is false then $S_1 S_2 S_3$ is also false. Therefore this goal is always unreachable for $D \geq 3$ and $n \geq 4$. We can generalize this result: we have that $Goal^{(G_{neg})}_{(S_{i_1} S_{i_2} S_{i_3})}$ is never reachable for $D \geq m+1$ and $n \geq m+2$.

Now consider the goal $Goal^{(G_{neg})}_{(S_{i_1} S_{i_2} S_{i_3})}$, where 1 must not know the secret of 2 and 2 must not know the secret of 3. For $D = 1$ and $n \geq 4$, the protocol for $Goal^{(G_{neg})}_{(S_{i_1} S_{i_2} S_{i_3})}$ generalizes as follows:

1. call$^i$ for every $i \in Agt \setminus \{2, 3\}$ ending with $i = n$.
2. call$^i$; call$^i$; call$^i$; call$^i$; call$^i$; for $n = 5$.
3. solve $GD$ for $D = 1$ and $Agt = \{3, \ldots, n\}$.

For $n = 5$, FDSS-2014 returns call$^1$; call$^2$; call$^3$; call$^4$; call$^5$; Immediately, and again by Proposition 14, we know that $Goal^{(G_{neg})}_{(S_{i_1} S_{i_2} S_{i_3})}$ will never be reachable for $D \geq 2$ and $n \geq 3$ (since, e.g., $S_1 S_2$ will also be false if $S_1 S_2$ is false).

7 Conclusion

In this article we have made a first step towards a realistic and provably-correct method for multi-agent epistemic planning. Our use of a logic of action and knowledge, which is known to be sound, together with an efficient automatic planner (which is assumed to be correct in the case of classical planning with conditional effects) provides an efficient method for producing plans which are guaranteed to be correct.

We believe that DEL-based epistemic planning being undecidable even for simple fragments is problematic. For example, if actions make factual changes to the world, then the problem is undecidable whenever epistemic operators are allowed in preconditions; if actions are purely epistemic, then it is undecidable whenever two agents are involved or the epistemic depth exceeds 2 [1, 7]. Of course, the low complexity of DEL-PAO comes with the price of expressivity. We have seen that our epistemic logic EL-O has more validities than standard epistemic logic. We have also seen in the exam problem that considering knowledge instead of belief is a restriction leading to counter-intuitive design of actions (the teacher must not exit the room if she has seen the student see the exam). While relaxing knowledge in DEL is simple, this is not easy in DEL-PAO. However, our framework at least allows us to update knowledge along with facts of the world and to specify epistemic preconditions of any form. Since any epistemic formula can be reduced to a boolean formula, the translation to PDDL is immediate.

We intend to continue this line of research by incorporating other important aspects of multi-agent planning, namely control (i.e., which agents are allowed to change the value of which variables) and mutual exclusion (to guarantee that at most one agent has control of a variable at any instant). In the long term, we also aim to generalize this approach to temporal planning where actions are durative and may overlap; flexible planning, where actions may happen between intervals of time; and contingent planning, with uncertainty on the initial state or the effects of actions (and the presence of sensing actions). Another perspective is to encode DEL-PAO or even full DEL into PDDL. This would allow us to perform model checking with optimized PDDL planners.

We can note that, although we have mentioned only PDDL here, alternative approaches exist. For example, it is possible to code a planning problem containing actions with conditional effects directly into SAT and then use an efficient SAT solver to find a plan [20].

Acknowledgements

We would like to thank the anonymous reviewers for their thoughtful reading and comments.
REFERENCES


