ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-193

A Simple Account of Multi-Agent Epistemic Planning

Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, Pierre Régnier [1]

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 standard 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 each item of gossip is known to everyone? 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 possible to express some interesting generalizations of this problem.

Dynamic Epistemic Logic DEL [24] 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 [5, 17]. While DEL provides a very expressive framework, it was unfortunately proven to be undecidable even for rather simple fragments of the language [1, 8]. Some decidable fragments were studied, most of which focused on public events [17, 25]. However, the gossip problem requires private communication. There exist other approaches on planning with uncertainty that do not use DEL. The framework presented in [16] allows us to reason about knowledge on literals in a multi-agent setting. A similar approach with beliefs can be found in [19]. While restricted to a single agent, the framework of [20] 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$^{\mathbf{S}}$ (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$^{\mathbf{S}}$ 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 standard Planning Domain Definition Language [18]. 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$^{\mathbf{S}}$. 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$^{\mathbf{S}}$ 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$^{\mathbf{S}}$ is a fragment of Dynamic Epistemic Logic of Propositional Assignments and Observation DEL-PAO [13], which is a dialect of Dynamic Logic of Propositional Assignments DL-PA [14, 4, 3]. We start by defining its language and semantics.

The basic ingredient of DEL-PAO are atoms of the form $\mathbf{S}_i p$, 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 $\mathbf{S}_j\mathbf{S}_i p$ (j sees whether i sees p), $\mathbf{S}_k\mathbf{S}_j\mathbf{S}_i p$ (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 atoms. Here we simplify things and only consider individual visibility. We call the resulting logic EL-O$^{\mathbf{S}}$.

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 $OBS = \{\mathbf{S}_i : i \in Agt\}$. The set of all sequences of visibility operators is noted OBS^*; elements of OBS^* are noted σ, σ', etc. An *atom* is any sequence of visibility operators $\mathbf{S}_i$ followed by a propositional variable. Formally,

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

Elements of ATM are noted α, α', β, β', etc. The depth of an atom is the number of visibility operators composing it.

Then the language of EL-O$^{\mathbf{S}}$ is defined by the following grammar:

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi$$

[1] University of Toulouse, IRIT, France.
http://www.irit.fr/?lang=en

where α ranges over ATM and i over Agt.

The formula $K_i\varphi$ reads "agent i knows that φ, based on what she observes." The other boolean operators $\bot$, $\top$, $\vee$, $\rightarrow$ 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 Fml_{bool}. We note $ATM(\varphi)$ the set of atoms appearing in the boolean formula φ. For example, $ATM(q \wedge \mathbf{S}_i p) = \{q, \mathbf{S}_i p\}$. (Note that $p \notin ATM(q \wedge \mathbf{S}_i p)$.)

The visibility operator $\mathbf{S}_i$ 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 $\mathbf{S}_i\neg p$ 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^{ATM}. We denote worlds by s, s', t, etc.

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

$$\alpha \wedge \mathbf{S}_i\alpha \leftrightarrow K_i\alpha$$
$$\neg\alpha \wedge \mathbf{S}_i\alpha \leftrightarrow K_i\neg\alpha$$

For complex formulas $K_i\varphi$, we rely on accessible worlds just as in standard epistemic logic [11]: i knows that φ if φ 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:

$$s \sim_i s' \text{ iff for every } \alpha \in ATM, \text{ if } \mathbf{S}_i\alpha \in s \text{ then } s(\alpha) = s'(\alpha)$$

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 $\mathbf{S}_i\mathbf{S}_i\alpha$ should always be true. Moreover, every agent should be aware of these facts. Generally: atoms of the form $\sigma\mathbf{S}_i\mathbf{S}_i\alpha$ with σ a sequence of visibility operators, possibly empty, must be true. Let us define the (infinite) set of introspectively valid atoms as:

$$ATM_{INTR} = \{\alpha : \alpha \in ATM \text{ and } \alpha = \sigma\mathbf{S}_i\mathbf{S}_i\alpha'\}.$$

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

$$INTR = \{s \in 2^{ATM} : ATM_{INTR} \subseteq s\}.$$

Clearly, $s \cup ATM_{INTR} \in INTR$ for every state s.

Proposition 1 ([13]). *Relations $\sim_i$ are equivalence relations on $INTR$.*

Proposition 2 ([13]). *Let $s \in INTR$ and $s' \in 2^{ATM}$. If $s \sim_i s'$ then $s' \in INTR$.*

[2] For example, $\emptyset \sim_i \{\mathbf{S}_i p, p\}$ while $\{\mathbf{S}_i p, p\} \not\sim_i \emptyset$ since $\{\mathbf{S}_i p, p\}(p) \neq \emptyset(p)$.

2.3 Semantics

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

$s \models \alpha$ iff $\alpha \in s$

$s \models \neg\varphi$ iff $s \not\models \varphi$

$s \models \varphi \wedge \varphi'$ iff $s \models \varphi$ and $s \models \varphi'$

$s \models K_i\varphi$ iff for every $s' \in 2^{ATM}$ such that $s \sim_i s'$, $s' \models \varphi$

Let us stress that truth conditions are defined on any state $s \in 2^{ATM}$, 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 $INTR$ if s is introspective. A formula φ is valid if and only if $s \models \varphi$ for every $s \in 2^{ATM}$; it is valid in $INTR$ if and only if $s \models \varphi$ for every $s \in INTR$.

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

Proposition 3. *For every EL-O$^{\mathbf{S}}$ formula φ, there exists a formula φ' in normal form such that $\varphi \leftrightarrow \varphi'$ is valid in $INTR$.*

This formula can be obtained by replacing every introspectively valid atom of φ by $\top$.

Proposition 4. *For every state $s \in 2^{ATM}$, every boolean formula φ in normal form, and every $\alpha \in ATM_{INTR}$, we have:*

$$s\backslash\{\alpha\} \models \varphi \text{ if and only if } s\cup\{\alpha\} \models \varphi.$$

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

2.4 From visibility to knowledge

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

Proposition 5 ([13]). *The following equivalences are valid.*

$$K_i\alpha \leftrightarrow \mathbf{S}_i\alpha \wedge \alpha$$
$$K_i\neg\alpha \leftrightarrow \mathbf{S}_i\alpha \wedge \neg\alpha$$
$$K_i(\varphi \wedge \varphi') \leftrightarrow K_i\varphi \wedge K_i\varphi'$$
$$K_i\Big(\bigvee_{\alpha\in A^+}\alpha \vee \bigvee_{\alpha\in A^-}\neg\alpha\Big) \leftrightarrow \begin{cases} \Big(\bigvee\limits_{\alpha\in A^+} K_i\alpha\Big) \vee \Big(\bigvee\limits_{\alpha\in A^-} K_i\neg\alpha\Big) & \text{if } A^+ \cap A^- = \emptyset \\ \top & \text{otherwise} \end{cases}$$

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 [13] and [7]; 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-O$^{\mathbf{S}}$ formula to a boolean formula. Let us focus on reducing formulas of the form $K_{i_1}\dots K_{i_m}\alpha$ to a conjunction of atoms. For example, we have by Proposition 5:

$$K_iK_jp \leftrightarrow K_i(\mathbf{S}_jp \wedge p) \leftrightarrow K_i\mathbf{S}_jp \wedge K_ip$$

$$\leftrightarrow \mathbf{S}_i\mathbf{S}_j p \wedge \mathbf{S}_j p \wedge \mathbf{S}_i p \wedge p.$$

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

$$EATM(\alpha) = \{\alpha\}$$
$$EATM(K_i\varphi) = EATM(\varphi) \cup \{\mathbf{S}_i\alpha : \alpha \in EATM(\varphi) \text{ and } \alpha \text{ is not of the form } \mathbf{S}_i\alpha'\}$$

The last line ensures that $EATM(\varphi)$ 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 $\bigwedge EATM(\varphi) = \bigwedge_{\alpha \in EATM(\varphi)} \alpha$.

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

$$K_{i_1} \dots K_{i_m}\alpha \leftrightarrow \bigwedge EATM(K_{i_1} \dots K_{i_m}\alpha)$$

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

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

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

$$EATM\Big(\bigwedge_{\substack{i_1,\dots,i_m \in Agt \\ \text{and } \alpha \in A}} K_{i_1} \dots K_{i_m}\alpha\Big) = \bigcup_{\substack{i_1,\dots,i_m \in Agt \\ \text{and } \alpha \in A}} EATM\Big(K_{i_1} \dots K_{i_m}\alpha\Big)$$

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-O$^{\mathbf{S}}$. We assume that we perform planning tasks in fully observable, deterministic domains.

3.1 Actions with conditional effects

An *conditional action* is a pair $\mathsf{a} = \langle pre(\mathsf{a}), \mathit{eff}(\mathsf{a})\rangle$ where:

- $pre(\mathsf{a}) \in Fml_{bool}$ is a boolean formula: the *precondition* of a;
- $\mathit{eff}(\mathsf{a}) \subseteq Fml_{bool} \times 2^{ATM} \times 2^{ATM}$ is a set of triples ce of the form $\langle cnd(ce), \mathit{ceff}^+(ce), \mathit{ceff}^-(ce)\rangle$: the *conditional effects* of a, where $cnd(ce)$ is a boolean formula (the condition) and $\mathit{ceff}^+(ce)$ and $\mathit{ceff}^-(ce)$ are sets of atoms (added and deleted atoms respectively).

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

For example, consider the conditional action toggle_p of flipping the truth value of the propositional variable p. It is described as $\mathsf{toggle}_p = \langle pre(\mathsf{toggle}_p), \mathit{eff}(\mathsf{toggle}_p)\rangle$ with $pre(\mathsf{toggle}_p) = \top$ and $\mathit{eff}(\mathsf{toggle}_p) = \{\langle p, \emptyset, \{p\}\rangle, \langle \neg p, \{p\}, \emptyset\rangle\}$. The conditions p and $\neg p$ are inconsistent, thus not leading to conflict.

Example 1 (The gossip problem)**.** Let $Agt = \{1, \dots, n\}$ and $Prop = \{s_i : i \in Agt\}$. 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 call^i_j, agents i and j tell each other every secret they know among all n secrets. We have $\mathsf{call}^i_j = \langle pre(\mathsf{call}^i_j), \mathit{eff}(\mathsf{call}^i_j)\rangle$ with $pre(\mathsf{call}^i_j) = \top$ and

$$\mathit{eff}(\mathsf{call}^i_j) = \{\langle \mathbf{S}_i s_1 \vee \mathbf{S}_j s_1, \{\mathbf{S}_i s_1, \mathbf{S}_j s_1\}, \emptyset\rangle,$$
$$\dots,$$
$$\langle \mathbf{S}_i s_n \vee \mathbf{S}_j s_n, \{\mathbf{S}_i s_n, \mathbf{S}_j s_n\}, \emptyset\rangle\}.$$

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.

A conditional action a determines a relation between states that is a partial function:

$$s R_{\mathsf{a}} s' \text{ iff } (1)\ s \models pre(\mathsf{a}), \text{ and}$$
$$(2) \text{ for every } ce \in \mathit{eff}(\mathsf{a}) \text{ such that}$$
$$(\mathit{ceff}^+(ce) \cup \mathit{ceff}^-(ce)) \cap ATM_{INTR} \neq \emptyset,$$
$$s \not\models cnd(ce), \text{ and}$$
$$(3)\ s' = \Big(s \setminus \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} \mathit{ceff}^-(ce)\Big) \cup \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} \mathit{ceff}^+(ce).$$

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

We say that the action a is in *normal form* if and only if (1) the formulas $pre(\mathsf{a})$ and $cnd(ce)$ for every $ce \in \mathit{eff}(\mathsf{a})$ are in normal form, and (2) for every $ce \in \mathit{eff}(\mathsf{a})$, if $\alpha \in \mathit{ceff}^+(ce) \cup \mathit{ceff}^-(ce)$ then α is not introspectively valid.

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

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

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

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

$$s\backslash\{\alpha\}\ R_{\mathsf{a}}\ t\backslash\{\alpha\} \text{ if and only if } s\cup\{\alpha\}\ R_{\mathsf{a}}\ t\cup\{\alpha\}.$$

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 $s \cup ATM_{INTR}$ to $t \cup ATM_{INTR}$. This is ensured by the fact that actions in normal form neither test nor add nor remove introspectively valid atoms.

3.2 Simple epistemic planning tasks

We say that a state s is *reachable* from a state s_0 via a set of conditional actions Act if there is a sequence of actions $\langle \mathsf{a}_1, \dots, \mathsf{a}_m\rangle$ from Act and a sequence of states $\langle u_0, \dots, u_m\rangle$ with $m \geq 0$ such that $s_0 = u_0$, $s = u_m$ and $u_{k-1} R_{\mathsf{a}_k} u_k$ for every k such that $1 \leq k \leq m$.

A simple epistemic *planning task* is a triple $\langle Act, s_0, Goal\rangle$ where Act is a finite set of actions, $s_0 \subseteq ATM$ is a finite state (the initial state) and $Goal \in Fml_{bool}$ is a boolean formula. It is *solvable* if at least one state s such that $s \models Goal$ is reachable from s_0 via Act; otherwise it is unsolvable.

We say that the planning task $\langle Act, s_0, Goal\rangle$ is in *normal form* if and only if (1) every action $\mathsf{a} \in Act$ is in normal form, and (2) the formula $Goal$ is in normal form.

Example 2 (Example 1, ctd.)**.** The planning task corresponding to the gossip problem is $G_1 = \langle Act^{G_1}, s_0^{G_1}, Goal^{G_1}\rangle$ with

- $Act^{G_I} = \{\mathsf{call}^i_j : i, j \in Agt \text{ and } i \neq j\}$;
- $s_0^{G_I} = \{\mathbf{S}_i s_i : i \in Agt\} \cup \{s_i : i \in Agt\}$;
- $Goal^{G_I} = \bigwedge_{i,j \in Agt} \mathbf{S}_i s_j$.

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_i, $Goal^{G_I}$ is equivalent to $\bigwedge_{i,j \in Agt} K_i s_j$.

This planning task is in normal form since every action call^i_j is trivially in normal form.

4 Dynamic extension and complexity results

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

4.1 A simple dynamic epistemic logic

The language of DEL-PAO$^{\mathbf{S}}$ extends the language of EL-O$^{\mathbf{S}}$ with the dynamic operator $\langle\pi\rangle$, with π a program: the formula $\langle\pi\rangle\varphi$ reads "there exists an execution of π after which φ is true."

The syntax of programs is defined by the following grammar:

$$\pi ::= +\alpha \mid -\alpha \mid \pi;\pi \mid \pi \sqcup \pi \mid \pi^* \mid \varphi?$$

where α ranges over the set of atomic formulas ATM and φ over the set of formulas.

Atomic programs $+\alpha$ and $-\alpha$ are assignments: they respectively set the value of the atom α to true and to false. Complex programs are composed of sequences of instructions ($\pi;\pi$), non-deterministic choice between instructions ($\pi \sqcup \pi$), repetitions (π^*) and tests ($\varphi?$).

The dual operator $[\pi]\varphi = \neg\langle\pi\rangle\neg\varphi$ ("after every execution of π, φ is true") is defined as usual. Moreover, **if** φ **then** π abbreviates $(\varphi?;\pi) \sqcup \neg\varphi?$ and **if** φ **then** π **else** π' abbreviates $(\varphi?;\pi) \sqcup (\neg\varphi?;\pi')$.

Semantically, a program is interpreted as a binary relation R_π on states, such that:

$sR_{+\alpha}s'$ iff $s' = s \cup \{\alpha\}$

$sR_{-\alpha}s'$ iff $s' = s \setminus \{\alpha\}$ and $\alpha \notin ATM_{INTR}$

$sR_{\pi_1;\pi_2}s'$ iff there exists $u \in 2^{ATM}$ such that $sR_{\pi_1}u$ and $uR_{\pi_2}s'$

$sR_{\pi_1\sqcup\pi_2}s'$ iff $sR_{\pi_1}s'$ or $sR_{\pi_2}s'$

$sR_{\pi^*}s'$ iff there exist $u_0, \ldots, u_m \in 2^{ATM}$ with $m \geq 0$ such that $s = u_0$, $s' = u_m$ and $u_{k-1}R_\pi u_k$ for every $1 \leq k \leq m$

$sR_{\varphi?}s'$ iff $s = s'$ and $s \models \varphi$

The truth condition of the new operator is then:

$s \models \langle\pi\rangle\varphi$ iff there exists $s' \in 2^{ATM}$ such that $sR_\pi s'$ and $s' \models \varphi$

In words, $\langle\pi\rangle\varphi$ is true if there is a state reachable by executing π where φ is true. An assignment $+\alpha$ or $-\alpha$ updates the state by adding or (unless introspectively valid) removing α; a sequential composition $\pi_1;\pi_2$ executes first π_1 and then π_2; a nondeterministic composition $\pi_1 \sqcup \pi_2$ takes the union of relations for π_1 and for π_2; an iteration π^* reaches any state attainable if we repeat π an arbitrary number of times; a test $\varphi?$ stays in the same state if φ 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 $-\mathbf{S}_i\mathbf{S}_i p$) will lead to a failure of the program because of the definition of $R_{-\alpha}$: a program starting in $INTR$ will never exit $INTR$.

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$^{\mathbf{S}}$ 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:

$$\mathsf{str}(\varphi, c) = \textbf{if } \varphi \textbf{ then } +c \textbf{ else } -c.$$

Proposition 9. *If c does not occur in φ then the equivalence $\varphi \leftrightarrow [\mathsf{str}(\varphi, c)]c$ is valid.*

We will see that after the execution of our program, we will make all the storage variables false so that we do not have to worry about them. The program resetting the value of a given set of storage variables is simply defined as:

$$\mathsf{rst}(\{c_1, \ldots, c_m\}) = -c_1; \ldots; -c_m.$$

4.3 Encoding of actions

Intuitively, an action is a DEL-PAO$^{\mathbf{S}}$ program, only executed if the precondition is fulfilled, applying each conditional effect whose condition is satisfied. For example, the action toggle_p (flipping the value of the variable p) corresponds to the program $\mathsf{str}(p, c_1); \mathsf{str}(\neg p, c_2);$ **if** c_1 **then** $-p$; **if** c_2 **then** $+p$. This highlights the importance of storing values of conditions: the program **if** p **then** $-p$; **if** $\neg p$ **then** $+p$ would actually always make p true.

We first show how to perform one conditional effect ce whose condition's value was stored in c:

$$\mathsf{exeCE}(ce, c) = \textbf{if } c \textbf{ then } +\alpha_1; \ldots; +\alpha_m; -\beta_1; \ldots; -\beta_\ell$$

where $\mathit{ceff}^+(ce) = \{\alpha_1, \ldots, \alpha_m\}$ and $\mathit{ceff}^-(ce) = \{\beta_1, \ldots, \beta_\ell\}$. Note that the ordering of atoms is not important since $\mathit{ceff}^+(ce) \cap \mathit{ceff}^-(ce) = \emptyset$. Then we can associate to action a the DEL-PAO$^{\mathbf{S}}$ program $\mathsf{exeAct}(\mathsf{a})$:

$$\begin{aligned}\mathsf{exeAct}(\mathsf{a}) = \; & \mathit{pre}(\mathsf{a})?; \\ & \mathsf{str}(\mathit{cnd}(ce_1), c_1); \ldots; \mathsf{str}(\mathit{cnd}(ce_m), c_m); \\ & \mathsf{exeCE}(ce_1, c_1); \ldots; \mathsf{exeCE}(ce_m, c_m); \\ & \mathsf{rst}(\{c_1, \ldots, c_m\}),\end{aligned}$$

with $\mathit{eff}(a) = \{ce_1, \ldots, ce_m\}$. The ordering of effects is not important since we test values of storage variables.

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

$$s \; R_{\mathsf{a}} \; t \;\; \textit{if and only if} \;\; s \; R_{\mathsf{exeAct(a)}} \; t.$$

Intuitively, our program $\mathsf{exeAct}(\mathsf{a})$ is divided in four parts which are executed in sequence:

1. $pre(\mathsf{a})?$: we test the precondition of the action. Semantically, if $s \models pre(\mathsf{a})$ then we stay in the same state s and continue to execute the program, and if $s \not\models pre(\mathsf{a})$ then the program fails (no world is related to s by $\mathsf{exeAct}(\mathsf{a})$). This corresponds to the requirement that $s \models pre(\mathsf{a})$ in R_{a}.
2. $\mathsf{str}(cnd(ce_1), c_1); \ldots; \mathsf{str}(cnd(ce_m), c_m)$: we store every condition. Recall that $\varphi \leftrightarrow [\mathsf{str}(\varphi, c)]c$ is valid, ensuring that testing each c_i after $\mathsf{str}(cnd(ce_i), c_i)$ is equivalent to testing $cnd(ce_i)$ before effects are executed. Observe that after the execution of this part, each atom has kept the same value it had in s; only storage variables have been modified.
3. $\mathsf{exeCE}(ce_1, c_1); \ldots; \mathsf{exeCE}(ce_m, c_m)$: we apply effects based on the truth values of storage variables. For each program $\mathsf{exeCE}(ce_i, c_i)$, if $s \models c_i$, then every positive effect from $\mathit{ceff}^+(ce_i)$ is added to s, while every negative effect from $\mathit{ceff}^-(ce_i)$ is removed from s, as specified in R_{a}. The action being in normal form ensures that $\mathsf{exeCE}(ce_i, c_i)$ will not remove an introspectively valid atom, preventing any failure.
4. $\mathsf{rst}(\{c_1, \ldots, c_m\})$: all storage variables are put to false, like they were in s, so that the execution of $\mathsf{exeAct}(\mathsf{a})$ is exactly equivalent to the execution of a (where storage variables do not appear).

Example 3 (Example 1, ctd.)**.** The action call^i_j, for any $i, j \in Agt$, is associated to the program:

$$\begin{aligned}\mathsf{exeAct}(\mathsf{call}^i_j) = {} & \top?; \\ & \mathsf{str}(\mathbf{S}_i s_1 \vee \mathbf{S}_j s_1, c_1); \ldots; \mathsf{str}(\mathbf{S}_i s_n \vee \mathbf{S}_j s_n, c_n); \\ & \textbf{if } c_1 \textbf{ then } +\mathbf{S}_i s_1; +\mathbf{S}_j s_1; \\ & \ldots; \\ & \textbf{if } c_n \textbf{ then } +\mathbf{S}_i s_n; +\mathbf{S}_j s_n; \\ & \mathsf{rst}(\{c_1, \ldots, c_n\})\end{aligned}$$

Note that in this case, $pre(\mathsf{call}^i_j)?$ 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$^{\mathbf{S}}$.

Proposition 11. *A planning task $\langle Act, s_0, Goal\rangle$ in normal form such that s_0 does not contain any storage variable is solvable if and only if:*

$$s_0 \models \Big\langle \Big(\bigsqcup_{\mathsf{a} \in Act} \mathsf{exeAct}(\mathsf{a}) \Big)^* \Big\rangle Goal.$$

Intuitively, our formula reads "there exists an execution of $\big(\bigsqcup_{\mathsf{a}\in Act} \mathsf{exeAct}(\mathsf{a})\big)^*$ after which $Goal$ is true." The program $\big(\bigsqcup_{\mathsf{a}\in Act} \mathsf{exeAct}(\mathsf{a})\big)^*$ non-deterministically chooses an action a from Act and executes the corresponding program $\mathsf{exeAct}(\mathsf{a})$, 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 $\big(\bigsqcup_{\mathsf{a}\in Act} \mathsf{exeAct}(\mathsf{a})\big)^*$ starting from the introspective state $s_0 \cup ATM_{INTR}$ and leading to a state satisfying $Goal$, then there is one starting from the non-introspective state s_0 and leading to a state satisfying $Goal$.

Proposition 12. *Deciding the solvability of a planning task with DEL-PAO$^{\mathbf{S}}$ is* PSPACE*-complete.*

The lower bound comes from classical planning [6]; the upper bound is given by Proposition 11, where the problem is reduced to a model checking problem of DEL-PAO$^{\mathbf{S}}$, a fragment of DEL-PAO whose model checking problem is in PSPACE.[3]

This result compares favorably to DEL-based epistemic planning, which is undecidable even for simple fragments [1, 8]. 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$^{\mathbf{S}}$ into PDDL. As already observed, in PDDL we do not need to store conditions as we were obliged to do in DEL-PAO$^{\mathbf{S}}$. Consider a planning task $\langle Act, s_0, Goal\rangle$. 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 $Goal$:

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

Given a boolean formula $\varphi \in Fml_{bool}$, we define a recursive function $tr_{PDDL}(\varphi)$ which returns the encoding of φ into PDDL:

$$tr_{PDDL}(\mathbf{S}_{i_1} \ldots \mathbf{S}_{i_m} p) ::= \begin{cases} \texttt{(p)} & \text{if } m = 0 \\ \texttt{(S-m i1 ... im p)} & \text{otherwise} \end{cases}$$

$$tr_{PDDL}(\neg\varphi) ::= \texttt{(not } tr_{PDDL}(\varphi)\texttt{)}$$

$$tr_{PDDL}(\varphi_1 \wedge \varphi_2) ::= \texttt{(and } tr_{PDDL}(\varphi_1)\ tr_{PDDL}(\varphi_2)\texttt{)}$$

with $p \in Prop$, $m \geq 0$, and $i_1, \ldots, i_m \in Agt$.

In words, a visibility atom $\alpha = \mathbf{S}_{i_1} \ldots \mathbf{S}_{i_m} p$ 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 $tr_{PDDL}(\alpha)$ the translation of an atom α in the general case (p or $\mathbf{S}_{i_1} \ldots \mathbf{S}_{i_m} p$). Other boolean operators are encoded as expected.

The initial state s_0 is trivially encoded as a set of fluents thanks to $tr_{PDDL}(\alpha)$. $Goal$ and the preconditions of every action can be encoded using $tr_{PDDL}(\varphi)$ 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 $ce \in \mathit{eff}(\mathsf{a})$ with $\mathit{ceff}^+(ce) = \{\alpha_1, \ldots, \alpha_m\}$ and $\mathit{ceff}^-(ce) = \{\beta_1, \ldots, \beta_\ell\}$, we add the conditional effect:

$$\begin{aligned}&\texttt{(when } tr_{PDDL}(cnd(ce)) \\ &\quad \texttt{(and } tr_{PDDL}(\alpha_1) \texttt{ ... } tr_{PDDL}(\alpha_m) \\ &\qquad \texttt{(not } tr_{PDDL}(\beta_1)\texttt{) ... (not } tr_{PDDL}(\beta_\ell)\texttt{)))}\end{aligned}$$

Note that, again, the ordering is not important.

[3] The version of DEL-PAO presented in [13] 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 PSPACE model checking, can be found in [9].

Example 4 (Example 1, ctd.)**.** The action call_2^1 is coded in PDDL as follows:

```
(:action call-1-2
  :effect (and
    (when (or (S-1 1 s1) (S-1 2 s1))
      (and (S-1 1 s1) (S-1 2 s1)))
    ...
    (when (or (S-1 1 sn) (S-1 2 sn))
      (and (S-1 1 sn) (S-1 2 sn)))))
```

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 '`(S-1 1 .)`" by "`(S-1 ?i .)`' and every '`(S-1 2 .)`" by "`(S-1 ?j .)`'. We will use the latter in experiments because of its succinctness.

Almost all planners from last International Planning Competition (IPC 2014)[4] handle conditional effects and negative preconditions, and most of them handle disjunctive preconditions. For experiments, we chose to use the planner FDSS-2014 [22] 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.[5] We sometimes write simply 'a' for '$\mathsf{exeAct(a)}$' when used within dynamic operators $\langle . \rangle$ and $[.]$.

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 $Exam = \langle Act^{Exam}, s_0^{Exam}, Goal^{Exam} \rangle$. Let $Agt = \{t, s\}$ and $Prop = \{exam, open, in_t, in_s\}$. 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 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:

$$s_0^{Exam} = \{exam\}.$$

As we said, the goal for the student is to know the exam's topic without being caught by the teacher. The goal is $\mathbf{S}_s exam \wedge \neg K_t \mathbf{S}_s exam \wedge \neg in_s$. In terms of visibility atoms, this becomes:

$$Goal^{Exam} = \mathbf{S}_s exam \wedge \neg \mathbf{S}_t \mathbf{S}_s exam \wedge \neg in_s.$$

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:

$$Act^{Exam} = \{\mathsf{openAndGoIn}_t, \mathsf{goOutAndClose}_t, \mathsf{goIn}_s, \mathsf{goOut}_s, \mathsf{readExam}_s\},$$

where

$$\mathsf{openAndGoIn}_t = \langle \neg in_t, \{\langle \top, \{open, in_t, \mathbf{S}_t \mathbf{S}_s exam\}, \emptyset \rangle\} \rangle$$

$$\mathsf{goOutAndClose}_t = \langle in_t \wedge \neg \mathbf{S}_s exam, \{\langle \top, \emptyset, \{\mathbf{S}_t \mathbf{S}_s exam, in_t, open\}\rangle\}\rangle$$

$$\mathsf{goIn}_s = \langle open \wedge \neg in_s, \{\langle \top, \{in_s\}, \emptyset\rangle\}\rangle$$

$$\mathsf{goOut}_s = \langle open \wedge in_s, \{\langle \top, \emptyset, \{in_s\}\rangle\}\rangle$$

$$\mathsf{readExam}_s = \langle in_s, \{\langle \top, \{\mathbf{S}_s exam\}, \emptyset\rangle\}\rangle$$

Action $\mathsf{openAndGoIn}_t$ makes the teacher open and enter the room, and thus watch the exam. Action $\mathsf{goOutAndClose}_t$ makes her leave and close the room; she cannot watch the exam anymore. We add the precondition $\neg \mathbf{S}_s exam$ 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_s and goOut_s makes her enter and leave the office, with the precondition that it is open; $\mathsf{readExam}_s$ 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: $\mathbf{S}_s exam \rightarrow K_t \mathbf{S}_s exam$. 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 $\mathsf{openAndGoIn}_t$ and $\mathsf{goOutAndClose}_t$ each in two parts:

- we replace $\mathsf{openAndGoIn}_t$ by:
$$\mathsf{open}_t = \langle \neg open, \{\langle \top, \{open\}, \emptyset\rangle\}\rangle$$
$$\mathsf{goIn}_t = \langle open \wedge \neg in_t, \{\langle \top, \{in_t, \mathbf{S}_t \mathbf{S}_s exam\}, \emptyset\rangle\}\rangle,$$
- we replace $\mathsf{goOutAndClose}_t$ by:
$$\mathsf{goOut}_t = \langle open \wedge in_t \wedge \neg \mathbf{S}_s exam, \{\langle \top, \emptyset, \{\mathbf{S}_t \mathbf{S}_s exam, in_t\}\rangle\}\rangle$$
$$\mathsf{close}_t = \langle open, \{\langle \top, \emptyset, \{open\}\rangle\}\rangle.$$

Thus the set of actions becomes:

$$Act^{Exam} = \{\mathsf{open}_t, \mathsf{close}_t, \mathsf{goIn}_t, \mathsf{goOut}_t, \mathsf{goIn}_s, \mathsf{goOut}_s, \mathsf{readExam}_s\}.$$

In this setting, the problem becomes solvable: for example, the plan $\mathsf{open}_t; \mathsf{goIn}_t; \mathsf{goIn}_s; \mathsf{goOut}_t; \mathsf{readExam}_s; \mathsf{goOut}_s$ is a solution plan. More mundanely, the planner finds the shortest plan: $\mathsf{open}_t; \mathsf{goIn}_s; \mathsf{readExam}_s; \mathsf{goOut}_s$.

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 [10].

The generalized gossip problem. We model the generalized gossip problem, introduced in [15], as a planning task $G_D = \langle Act^{G_D}, s_0^{G_D}, Goal^{G_D} \rangle$. 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 $Agt = \{1, \ldots, n\}$ and $Prop = \{s_i : i \in Agt\}$. In terms of knowledge, the goal of the generalized gossip problem is:

$$\varphi_{G_D} = \underbrace{\bigwedge_{i_1 \in Agt} K_{i_1} \ldots \bigwedge_{i_D \in Agt} K_{i_D}}_{D \text{ times}} \bigwedge_{\ell \in Agt} s_\ell.$$

[4] http://helios.hud.ac.uk/scommv/IPC-14/planners.html

[5] All resources and PDDL files we used for experiments are available at http://www.irit.fr/%7EAndreas.Herzig/P/Ecai16.html.

We have seen in Section 2.4 how to express this with a boolean formula, thanks to our epistemic atoms:

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

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

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

$$s_0^{G_D} = \{\mathbf{S}_i s_i : i \in Agt\} \cup \{s_i : i \in Agt\},$$
$$Act^{G_D} = \{\mathsf{call}_j^i : i, j \in Agt, i \neq j\}.$$

The preconditions of calls also remain unchanged: $pre(\mathsf{call}_j^i) = \top$. 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 ℓ and i calls j, j will learn the secret of ℓ, but also that i knows it; i will learn that j knows that she knows it; and so on until depth D. Moreover, if i knows that i_1 knows the secret of ℓ, then j learns that i_1 knows the secret of ℓ, but also that i knows that, an so on until depth D. As an example, suppose $D = 4$ and we have $K_i K_{i_1} s_\ell$. Then after the call between i and j, we will have, e.g., $K_j K_{i_1} s_\ell$, $K_i K_j K_{i_1} s_\ell$, $K_j K_i K_{i_1} s_\ell$, $K_j K_i K_j K_{i_1} s_\ell$, $K_i K_j K_i K_{i_1} s_\ell$, and so on, that is, any combination of K_i and K_j followed by $K_{i_1} s_\ell$, for a maximum depth of D.

For a given integer m and two agents i and j, we note $\{\mathbf{S}_i, \mathbf{S}_j\}^{\leq m}$ the set all non-empty non-introspective sequences of visibility operators $\mathbf{S}_i$ and $\mathbf{S}_j$ of length at most m. For instance, $\{\mathbf{S}_i, \mathbf{S}_j\}^{\leq 2} = \{\mathbf{S}_i, \mathbf{S}_j, \mathbf{S}_i\mathbf{S}_j, \mathbf{S}_j\mathbf{S}_i\}$.

Thus we have that, during a call between i and j, if i or j knows that $K_{i_1} \dots K_{i_m} s_\ell$, i.e., if $K_i K_{i_1} \dots K_{i_m} s_\ell \vee K_j K_{i_1} \dots K_{i_m} s_\ell$ is true, then $\sigma \mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell$ for every $\sigma \in \{\mathbf{S}_i, \mathbf{S}_j\}^{\leq D-m}$ becomes true. Formally:

$$\begin{aligned} eff(\mathsf{call}_j^i) = \{\langle & \bigwedge EATM(K_i K_{i_1} \dots K_{i_m} s_\ell) \vee \\ & \bigwedge EATM(K_j K_{i_1} \dots K_{i_m} s_\ell), \\ & \{\sigma \mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell : \sigma \in \{\mathbf{S}_i, \mathbf{S}_j\}^{\leq D-m}\}, \emptyset\rangle : \\ & 0 \leq m < D \text{ and } i_1, \dots, i_m, \ell \in Agt \text{ such that} \\ & \text{for every } 1 \leq k < m, i_k \neq i_{k+1}, \text{ and } i \neq i_1 \text{ and } j \neq i_1\} \end{aligned}$$

Consecutive agents in $\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell$ 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 call_j^i from Example 1 (with tests of secrets that could be omitted).

We require knowledge instead of visibility, i.e., $\bigwedge EATM(K_i K_{i_1} \dots K_{i_m} s_\ell)$ instead of just $\mathbf{S}_i \mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell$, so that agents only exchange 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* $[\mathsf{call}_j^i]\neg\varphi \leftrightarrow \neg[\mathsf{call}_j^i]\varphi$ *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.*

$$\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell \leftrightarrow [\mathsf{call}_j^i]\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell \quad \text{if } i \neq i_1 \text{ and } j \neq i_1 \quad (1)$$
$$\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell \rightarrow [\mathsf{call}_j^i]\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell \quad \text{for any } i, j \quad (2)$$

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

$$\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell \rightarrow [\mathsf{call}_j^i]\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell$$
$$\neg\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell \rightarrow [\mathsf{call}_j^i]\neg\mathbf{S}_{i_1} \dots \mathbf{S}_{i_m} s_\ell$$

if $i \neq i_1$ and $j \neq i_1$. (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, 23, 12], the generalized gossip problem can be solved in at most $(D+1)(n-2)$ calls [15]. For instance, suppose $D = 2$ and $n = 5$, then the sequence $\mathsf{call}_3^1; \mathsf{call}_4^1; \mathsf{call}_5^2; \mathsf{call}_5^1; \mathsf{call}_3^1; \mathsf{call}_4^2; \mathsf{call}_4^1; \mathsf{call}_5^1; \mathsf{call}_3^2$ is a solution with $3 \times 3 = 9$ calls. Our experiments have confirmed that the protocol given in [15] 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 $G\text{-}neg_D = \langle Act^{G\text{-}neg_D}, s_0^{G\text{-}neg_D}, Goal^{G\text{-}neg_D}\rangle$. 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: $Act^{G\text{-}neg_D} = Act^{G_D}$ and $s_0^{G\text{-}neg_D} = s_0^{G_D}$.

We note $Goal_A^{G\text{-}neg_D}$ the goal of the generalized gossip problem where only atoms from A, such that $A \cap ATM_{INTR} = \emptyset$, are false. Formally:

$$Goal_A^{G\text{-}neg_D} = \Big(\bigwedge_{\alpha \in EATM(\varphi_{G_D}) \setminus A} \alpha\Big) \wedge \Big(\bigwedge_{\alpha \in A} \neg\alpha\Big).$$

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

Lemma 3. *Let $m \geq 2$ be an integer. Let $D \geq m$. Take $m+1$ agents $i_1, i_2, i_3, \dots, i_m, \ell \in Agt$ such that i_1, i_2 and i_3 are distinct. We have:*

$$(\neg\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell \wedge [\mathsf{call}_j^{i_1}]\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell) \rightarrow$$
$$\begin{cases} \bigwedge EATM(K_{i_1} K_{i_3} \dots K_{i_m} s_\ell) \vee \bigwedge EATM(K_{i_2} K_{i_3} \dots K_{i_m} s_\ell) & \\ \qquad\qquad \text{if } j = i_2 & \\ \bigwedge EATM(K_j K_{i_2} K_{i_3} \dots K_{i_m} s_\ell) \quad \text{otherwise} & \end{cases}$$

Proof. In words, we are looking for conditions that make the atom $\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell$ true after a call. We are only interested in cases where $\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell$ is not introspective; when it is, it will never be added by an action and thus the implication is trivially true. We examine the two cases.

First case: $j = i_2$. Remember that by the definition of calls, $\mathsf{call}_{i_2}^{i_1}$ only produces atoms beginning with a sequence σ of $\mathbf{S}_{i_1}$ and $\mathbf{S}_{i_2}$. Since i_3 is distinct from i_1 and i_2 and we avoid introspective atoms, $\mathsf{call}_{i_2}^{i_1}$ can only add our atom by adding $\sigma\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell$ with $\sigma = \mathbf{S}_{i_1}\mathbf{S}_{i_2}$. Then either $\bigwedge EATM(K_{i_1} K_{i_3} \dots K_{i_m} s_\ell)$ or $\bigwedge EATM(K_{i_2} K_{i_3} \dots K_{i_m} s_\ell)$ must be true (before the call), which corresponds to the right side of the implication.

Second case: $j \neq i_2$. Following the same reasoning, $\mathsf{call}_j^{i_1}$ can only add our atom by setting to true $\sigma\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell$ with $\sigma = \mathbf{S}_{i_1}$; then either $\bigwedge EATM(K_{i_1} K_{i_2} K_{i_3} \dots K_{i_m} s_\ell)$ must be true or $\bigwedge EATM(K_j K_{i_2} K_{i_3} \dots K_{i_m} s_\ell)$ must be true. We cannot have the former since $\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell$ is false. The latter corresponds to the right side of the implication. □

Proposition 14. *Let $m \geq 2$ be an integer. Let $D \geq m$. Take $m+1$ agents $i_1, i_2, i_3, \dots, i_m, \ell \in Agt$ such that i_1, i_2 and i_3 are distinct. Then after a sequence of calls $\mathcal{C} = \mathsf{call}_{j_{r_1}}^{i_{r_1}}; \dots; \mathsf{call}_{j_{r_p}}^{i_{r_p}}$, if $\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell$ is true then $\mathbf{S}_{i_1}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell$ is true. Formally:*

$$s_0^{G\text{-}neg_D} \models [\mathcal{C}](\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell \rightarrow \mathbf{S}_{i_1}\mathbf{S}_{i_3} \dots \mathbf{S}_{i_m} s_\ell).$$

Proof. We prove it by induction on the sequence of calls. We are only interested in cases where $\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell$ is not introspective.

Base case: initial situation. We prove:

$$s_0^{G\text{-}neg_D} \models \mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell \rightarrow \mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell.$$

This is trivially true because only atoms of the form $\mathbf{S}_i s_i$ are true initially.

Inductive case. Suppose:

$$s_0^{G\text{-}neg_D} \models [\mathcal{C}](\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell \rightarrow \mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell).$$

We prove that for an arbitrary s:

$$s \models (\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell \rightarrow \mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell) \rightarrow [\mathsf{call}_j^i](\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell \rightarrow \mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell). \quad (3)$$

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 $\mathbf{S}_{i_1}$) 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:

$$s \models (\neg\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell \wedge [\mathsf{call}_j^{i_1}]\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell) \rightarrow (\mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell \vee [\mathsf{call}_j^{i_1}]\mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell). \quad (4)$$

In words, if $\mathsf{call}_j^{i_1}$ makes $\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell$ true, then either $\mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell$ was true or it becomes true.

By Lemma 3, we know that the premise of (4) implies either $\bigwedge EATM(K_{i_1}K_{i_3}\dots K_{i_m}s_\ell)$ or $\bigwedge EATM(K_{i_2}K_{i_3}\dots K_{i_m}s_\ell)$ if $j = i_2$, or $\bigwedge EATM(K_jK_{i_2}K_{i_3}\dots K_{i_m}s_\ell)$ otherwise. It is possible prove that each of these three statements implies either $\mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell$ or $[\mathsf{call}_j^{i_1}]\mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell$, using the definition of calls and Lemma 1.

Therefore $\mathbf{S}_{i_1}\mathbf{S}_{i_2}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell \rightarrow \mathbf{S}_{i_1}\mathbf{S}_{i_3}\dots\mathbf{S}_{i_m}s_\ell$ is preserved by call_j^i, hence the result. □

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

The goal $Goal^{G\text{-}neg_D}_{\{\mathbf{S}_1 s_2\}}$, where 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 $\mathsf{call}_3^1; \mathsf{call}_4^1; \mathsf{call}_5^1; \mathsf{call}_5^2; \mathsf{call}_5^3; \mathsf{call}_5^4$ for $n = 5$. More generally, the following protocol gives a solution:

1. call_i^1 for every $i \in Agt \setminus \{2\}$;
2. solve G_D for $D = 1$ and $Agt = \{2, \dots, n\}$.

However, it will never be reachable for $D \geq 2$ and $n \geq 3$: by contraposition of Proposition 14, if $\mathbf{S}_1 s_2$ is false then $\mathbf{S}_1\mathbf{S}_3 s_2$ is false, thus we cannot reach the goal where only $\mathbf{S}_1 s_2$ 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_2^1, establishes $\mathbf{S}_1 s_2$.)

Now suppose we have $D \geq 2$ and we want 1 not to know whether 2 knows the secret of 3 (but we do want 2 to know the secret of 3): our goal is $Goal^{G\text{-}neg_D}_{\{\mathbf{S}_1\mathbf{S}_2 s_3\}}$. The following protocol produces a solution for $D = 2$ and $n \geq 3$:

1. call_i^2 for every $i \in Agt \setminus \{3\}$;
2. solve G_D for $D = 2$ and $Agt = \{1, 3, 4, \dots, n\}$;
3. call_i^2 for every $i \in Agt \setminus \{1\}$.

One of the plans FDSS-2014 finds is $\mathsf{call}_2^1; \mathsf{call}_4^2; \mathsf{call}_5^2; \mathsf{call}_5^3; \mathsf{call}_3^1; \mathsf{call}_4^1; \mathsf{call}_4^2; \mathsf{call}_5^2; \mathsf{call}_3^2$ for $n = 5$. Again by Proposition 14, we know that if $\mathbf{S}_1\mathbf{S}_2 s_3$ is false then $\mathbf{S}_1\mathbf{S}_4\mathbf{S}_2 s_3$ 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 $Goal^{G\text{-}neg_D}_{\{\mathbf{S}_{i_1}\dots\mathbf{S}_{i_m}s_\ell\}}$ is never reachable for $D \geq m{+}1$ and $n \geq m{+}2$.

Now consider the goal $Goal^{G\text{-}neg_D}_{\{\mathbf{S}_1 s_2, \mathbf{S}_2 s_3\}}$, 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 $Goal^{G\text{-}neg_D}_{\mathbf{S}_1 s_2}$ generalizes as follows:

1. call_i^1 for every $i \in Agt \setminus \{2, 3\}$ ending with $i = n$;
2. $\mathsf{call}_n^2; \mathsf{call}_3^1$;
3. solve G_D for $D = 1$ and $Agt = \{3, \dots, n\}$.

For $n = 5$, FDSS-2014 returns $\mathsf{call}_4^1; \mathsf{call}_5^1; \mathsf{call}_3^1; \mathsf{call}_5^2; \mathsf{call}_5^3; \mathsf{call}_5^4$. However, and again by Proposition 14, we know that $Goal^{G\text{-}neg_D}_{\{\mathbf{S}_1 s_2, \mathbf{S}_2 s_3\}}$ will never be reachable for $D \geq 2$ and $n \geq 3$ (since, e.g., $\mathbf{S}_1\mathbf{S}_3 s_2$ will also be false if $\mathbf{S}_1 s_2$ 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 together with an state of the art automatic planner (which is assumed to be correct in the case of classical planning with conditional effects) provides a method for producing plans which are guaranteed to be correct.

Our approach contrasts with the undecidability of DEL-based epistemic planning which occurs even for simple fragments. 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, 8]. Of course, the low complexity of DEL-PAO$^{\mathsf{S}}$ comes at the price of expressivity. We have seen that our epistemic logic EL-O$^{\mathsf{S}}$ 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$^{\mathsf{S}}$ or even full DEL-PAO 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 [21].

Acknowledgements

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

REFERENCES

[1] Guillaume Aucher and Thomas Bolander, 'Undecidability in epistemic planning', in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, ed., Francesca Rossi, pp. 27–33. AAAI Press, (2013).

[2] Brenda Baker and Robert Shostak, 'Gossips and telephones', *Discrete Mathematics*, **2**(3), 191–193, (1972).

[3] Philippe Balbiani, Andreas Herzig, François Schwarzentruber, and Nicolas Troquard, 'DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE', *CoRR*, **abs/1411.7**, (2014).

[4] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard, 'Dynamic logic of propositional assignments: a well-behaved variant of PDL', in *Proceedings of the 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS)*, ed., Orna Kupferman, pp. 143–152, (2013).

[5] 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).

[6] Tom Bylander, 'The computational complexity of propositional STRIPS planning', *Artificial Intelligence*, **69**, 165–204, (1994).

[7] Tristan Charrier, Emiliano Lorini, Andreas Herzig, Faustine Maffre, and François Schwarzentruber, 'Building epistemic logic from observations and public announcements', in *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016)*, (2016).

[8] Tristan Charrier, Bastien Maubert, and Francois Schwarzentruber, 'On the impact of modal depth in epistemic planning', in *Proc. IJCAI 2016*. AAAI Press, (2016).

[9] Tristan Charrier, Sophie Pinchinat, and François Schwarzentruber, 'Mental programs and arbitrary public announcement logic: relevance and complexity'. Unpublished manuscript, 2016.

[10] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier, 'Simple epistemic planning: generalised gossiping', *ArXiv e-prints*, **abs/1606.0**, (2016).

[11] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi, *Reasoning about Knowledge*, MIT Press, 1995.

[12] Andras Hajnal, Eric C. B. Milner, and Endre Szemerédi, 'A cure for the telephone disease', *Canadian Mathematical Bulletin*, **15**(3), 447–450, (1972).

[13] Andreas Herzig, Emiliano Lorini, and Faustine Maffre, 'A poor man's epistemic logic based on propositional assignment and higher-order observation', in *Proceedings of the 5th International Conference on Logic, Rationality and Interaction (LORI)*, eds., Wiebe van der Hoek, Wesley H. Holliday, and Wen-fang Wang, pp. 156–168. Springer Verlag, (2015).

[14] Andreas Herzig, Emiliano Lorini, Nicolas Troquard, and Frédéric Moisan, 'A dynamic logic of normative systems', in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 228–233, (2011).

[15] Andreas Herzig and Faustine Maffre, 'How to share knowledge by gossiping', in *Proceedings of the 3rd International Conference on Agreement Technologies (AT)*. Springer-Verlag, (2016).

[16] Filippos Kominis and Hector Geffner, 'Beliefs in multiagent planning: from one agent to many', in *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*, eds., Ronen I. Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, pp. 147–155. AAAI Press, (2015).

[17] Benedikt Löwe, Eric Pacuit, and Andreas Witzel, 'DEL planning and some tractable cases', in *Proceedings of the 3rd International International Workshop on Logic, Rationality and Interaction*, pp. 179–192. Springer Berlin Heidelberg, (2011).

[18] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins, 'PDDL – The Planning Domain Definition Language', Technical report, Yale Center for Computational Vision and Control, (1998).

[19] Christian Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg, 'Planning over multi-agent epistemic states: A classical planning approach', in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pp. 3327–3334. AAAI Press, (2015).

[20] 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)*, pp. 2–11, (2004).

[21] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä, 'Planning as satisfiability: parallel plans and algorithms for plan search', *Artif. Intell.*, **170**(12-13), 1031–1080, (2006).

[22] Gabriele Röger, Florian Pommerening, and Jendrik Seipp, 'Fast downward stone soup 2014', in *The 2014 International Planning Competition*, (2014).

[23] Robert Tijdeman, 'On a telephone problem', *Nieuw Archief voor Wiskunde*, **19**(3), 188–192, (1971).

[24] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi, *Dynamic Epistemic Logic*, Springer Publishing Company, Incorporated, 1st edn., 2007.

[25] Quan Yu, Yanjun Li, and Yanjing Wang, 'A dynamic epistemic framework for conformant planning', *Proceedings of TARK*, **15**, 249–259, (2015).