A simple account of multiagent epistemic planning

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

A simple account of multiagent epistemic planning

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

IRIT - CNRS UMR 5505 – University of Toulouse, France
cooper@irit.fr, herzig@irit.fr, maffre@irit.fr,
maris@irit.fr, regnier@irit.fr

Abstract A realistic model of multiagent planning must allow us to model notions which are absent in classical planning such as communication and knowledge. We investigate multiagent planning based on a simple logic of action and knowledge that is based on the visibility of propositional variables. Using such a formal logic allows us to deduce the validity of a plan from the validity of the individual actions which compose it. We present a coding of multiagent planning problems expressed in this logic into the classical planning language PDDL. Feeding the resulting problem into a PDDL planner provides a provably correct plan for the original multiagent planning problem. We use the gossip problem as a running example.

Keywords: Planning, multiagent system, epistemic logic

1 Introduction

Suppose there are n people, and each of them knows some item of gossip not known to the others. They communicate by telephone, 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 can be viewed as a multiagent planning problem where contrarily to classical planning it is not the world that evolves but the agents' knowledge. In this paper we provide a simple multiagent epistemic logic allowing to reason about such problems. We also study how they can be encoded into PDDL, the classical Planning Domain Definition Language first introduced by (Mc Dermott *et al.*, 1998).

2 A simple epistemic logic

We start by defining a language and semantics of a simple logic of action and knowledge that is based on the visibility of propositional variables. Our logic is a dialect of Dynamic Logic of Propositional Assignments DL-PA (Balbiani *et al.*, 2013).

2.1 Language

Let $\mathsf{PVar} = \{p_1, p_2, \ldots\}$ be a finite set of *propositional variables* and $\mathsf{Agt} = \{i_1, i_2, \ldots\}$ a finite set of *agents*. To each $p \in \mathsf{PVar}$ and each agent $i \in \mathsf{Agt}$ we associate one more propositional variable $\mathbf{Kw}_i p$, which reads "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. The set of *atomic formulas* of our language is then

$$\mathsf{Fml}_0 = \mathsf{PVar} \cup \{\mathbf{Kw}_i p \ : \ i \in \mathsf{Agt} \text{ and } p \in \mathsf{PVar}\}.$$

We remark that our framework can be further extended to also contain information of the form $\mathbf{Kw}_i \mathbf{Kw}_j p$ ('i knows whether j knows whether p'). Note that the latter does not entail that i himself knows whether p.

Our language has two syntactical entities, formulas and programs. They are defined by mutual recursion as follows :

$$\varphi ::= \alpha \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\pi\rangle\varphi$$
$$\pi ::= +\alpha \mid -\alpha \mid \pi;\pi \mid \pi \sqcup \pi \mid \varphi? \mid \pi^*$$

where α ranges over the set of atomic formulas Fml_0.

$\langle\pi\rangle\varphi$ is the formula which results when the program π is applied to the formula φ. It reads "there exists an execution of π after which φ is true". The other boolean operators $\bot$, $\top$, $\vee$, $\rightarrow$ and $\leftrightarrow$ and the dual operator $[\pi]\varphi = \neg\langle\pi\rangle\neg\varphi$ (which reads "after every execution of π, φ is true") are defined as usual.

Programs are composed of instructions which are either assignments ($+\alpha$ or $-\alpha$), sequences of instructions ($\pi;\pi$), non-deterministic choice between instructions ($\pi \sqcup \pi$), tests ($\varphi?$) or repetitions (π^*).

We define the more standard modal operator 'knowing that' ranging over literals as follows :

$$\mathbf{K}_i p \stackrel{\mathsf{def}}{=} p \wedge \mathbf{Kw}_i p$$
$$\mathbf{K}_i \neg p \stackrel{\mathsf{def}}{=} \neg p \wedge \mathbf{Kw}_i p$$

A *boolean formula* is a formula without the dynamic operator $\langle . \rangle$. The set of all boolean formulas is noted $\mathsf{Fml}_{\mathsf{Bool}}$. We note $\mathsf{var}(\varphi)$ the set of atomic formulas appearing in the boolean formula φ. For example, $\mathsf{var}(q \wedge \mathbf{Kw}_i p) = \{q, \mathbf{Kw}_i p\}$. (Note that $p \notin \mathsf{var}(q \wedge \mathbf{Kw}_i p)$.) The set of *atomic programs* is

$$\mathsf{Pgm}_0 = \{+\alpha \ : \ \alpha \in \mathsf{Fml}_0\} \cup \{-\alpha \ : \ \alpha \in \mathsf{Fml}_0\}.$$

Two elements of Pgm_0 are *in conflict* if they assign different truth values to the same variable.

2.2 Semantics

A state is a subset of the set of atomic formulas (corresponding to exactly those atomic formulas that are true in this state). Formulas are interpreted as sets of states (those states in which the formula is true) and programs are interpreted as relations on states.

$$\begin{aligned}
||\alpha|| &= \{s \ : \ \alpha \in s\} \\
||\neg\varphi|| &= 2^{\mathsf{Fml}_0} \setminus ||\varphi|| \\
||\varphi \wedge \psi|| &= ||\varphi|| \cap ||\psi|| \\
||\langle\pi\rangle\varphi|| &= \{s \ : \ \text{there is a } s' \in ||\varphi|| \text{ such that } \langle s, s'\rangle \in ||\pi||\} \\
||+\alpha|| &= \{\langle s, s'\rangle \ : \ s' = s \cup \{\alpha\}\} \\
||-\alpha|| &= \{\langle s, s'\rangle \ : \ s' = s \setminus \{\alpha\}\} \\
||\pi_1;\pi_2|| &= ||\pi_1|| \circ ||\pi_2|| \\
&= \{\langle s, s'\rangle \ : \ \text{there is } s'' \text{ such that } \langle s, s''\rangle \in ||\pi_1|| \text{ and } \langle s'', s'\rangle \in ||\pi_2||\} \\
||\pi_1 \sqcup \pi_2|| &= ||\pi_1|| \cup ||\pi_2|| \\
||\varphi?|| &= \{\langle s, s\rangle \ : \ s \in ||\varphi||\} \\
||\pi^*|| &= (||\pi||)^*
\end{aligned}$$

In words, a propositional variable α is true in s if it belongs to it ; the negation of a formula $\neg\varphi$ is true in all worlds where φ is not true ; the conjunction $\varphi \wedge \psi$ is true in states where both φ and ψ are true ; and $\langle\pi\rangle\varphi$ is true if there is a state reachable by executing π where φ is true. Concerning programs, assignments update the state by adding or removing the corresponding variable ; the sequence $\pi_1;\pi_2$ reaches a state by executing π_1 and executes π_2 from this state ; the non-deterministic choice $\pi_1 \sqcup \pi_2$ represents the union of the relations for π_1 and for π_2 ; the test $\varphi?$ stays in the same state if it verifies φ (otherwise it fails and the program stops) ; and the repetition π^* reaches any state attainable if we repeat π any number of times by making the reflexive transitive closure of the relation $||\pi||$.

3 Action theories and planning tasks

A conditional action is a couple $a = \langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ where $\mathsf{pre}(a)$ is a boolean formula (the *precondition* of a) and $\mathsf{eff}(a)$ is a set of *conditional effects* : couples of the form $\langle \mathsf{cond}_i, \pi_i\rangle$ such that $\mathsf{cond}_i \in \mathsf{Fml}_{\mathsf{Bool}}$ is a boolean formula and $\pi_i \in \mathsf{Pgm}_0$ is an assignment.

As an example, let us consider the conditional action $\mathsf{toggle}(p)$ of flipping the truth value of the propositional variable p. It is described as $\mathsf{toggle}_p = \langle \mathsf{pre}(\mathsf{toggle}_p), \mathsf{eff}(\mathsf{toggle}_p)\rangle$ with $\mathsf{pre}(\mathsf{toggle}_p) = \top$ and $\mathsf{eff}(\mathsf{toggle}_p) = \{\langle p, -p\rangle, \langle \neg p, +p\rangle\}$.

Example 1
The action call_{ij} *corresponds to agents* i *and* j *telling each other every secret they know among all* n *secrets. The secret of agent* i *is modelled by a propositional variable* s_i *and agent* j*'s knowledge of this secret is modelled by* $\mathbf{Kw}_j s_i$*. We have* $\mathsf{call}_{ij} = \langle \mathsf{pre}(\mathsf{call}_{ij}), \mathsf{eff}(\mathsf{call}_{ij}) \rangle$ *with* $\mathsf{pre}(\mathsf{call}_{ij}) = \top$ *and*

$$\begin{aligned}\mathsf{eff}(\mathsf{call}_{ij}) = \{&\langle \mathbf{Kw}_i s_1, +\mathbf{Kw}_j s_1 \rangle, \ldots, \langle \mathbf{Kw}_i s_n, +\mathbf{Kw}_j s_n \rangle,\\ &\langle \mathbf{Kw}_j s_1, +\mathbf{Kw}_i s_1 \rangle, \ldots, \langle \mathbf{Kw}_j s_n, +\mathbf{Kw}_i s_n \rangle\}.\end{aligned}$$

Intuitively, we check knowledge of both agents about every secret and add knowledge of a secret to one agent only if it is known to the other.

A given conditional action a determines a relation between states :

$$\begin{aligned}||a|| = \big\{\langle s, s' \rangle \ : \ & s \in ||\mathsf{pre}(a)|| \text{ and there is no } \langle \mathsf{cond}_1, \pi_1 \rangle, \langle \mathsf{cond}_2, \pi_2 \rangle \in \mathsf{eff}(a)\\ & \text{such that } \pi_1 \text{ and } \pi_2 \text{ are in conflict and } s \in ||\mathsf{cond}_1 \wedge \mathsf{cond}_2||\\ & \text{and } s' = \big(s \setminus \{\alpha \ : \ \text{there is } \langle \mathsf{cond}, -\alpha \rangle \in \mathsf{eff}(a) \text{ and } s \in ||\mathsf{cond}||\}\big)\\ & \qquad \cup \{\alpha \ : \ \text{there is } \langle \mathsf{cond}, +\alpha \rangle \in \mathsf{eff}(a) \text{ and } s \in ||\mathsf{cond}||\}\big\}\end{aligned}$$

Thus an action is executable if its precondition is currently verified and if there cannot be any conflict. When executed, for every conditional effect such that the condition is true, the action will remove every α such that the effect is $-\alpha$ and add every α such that the effect is $+\alpha$.

We say that a state s is *reachable* from a state s_0 via a set of actions Act if there is a sequence of actions $\langle a_1, \ldots, a_n \rangle$ and a sequence of states $\langle s_0, \ldots, s_n \rangle$ with $n \geq 0$ such that $s_n = s$ and $\langle s_{k-1}, s_k \rangle \in ||a_k||$ for every k such that $1 \leq k \leq n$.

A *planning task* is a triple $\langle \mathsf{Act}, s_0, \mathsf{Goal} \rangle$ where Act is a finite set of actions, $s_0 \subseteq \mathsf{Fml}_0$ is a state (the initial state) and $\mathsf{Goal} \in \mathsf{Fml}_{\mathsf{Bool}}$ is a boolean formula (the goal). It is solvable if at least one of the goal states in $||\mathsf{Goal}||$ is reachable from s_0 via Act ; else it is unsolvable.

Example 2 (Example 1, ctd.)
The planning task corresponding to the gossip problem is $\langle \mathsf{Act}, s_0, \mathsf{Goal} \rangle$ *with* $\mathsf{Act} = \{\mathsf{call}_{ij} : i, j \in \mathsf{Agt}$ *and* $i \neq j\}$*,* $s_0 = \{\mathbf{Kw}_i s_i : 1 \leq i \leq n\}$ *(every agent knows its own secret) and* $\mathsf{Goal} = \bigwedge_{i,j \in \mathsf{Agt}} \mathbf{Kw}_i s_j$ *(every agent knows every secret).*

4 Encoding into DL-PA

Consider an action $a = \langle \mathsf{pre}(a), \{\langle \mathsf{cond}_1, \pi_1 \rangle, \ldots, \langle \mathsf{cond}_n, \pi_n \rangle\} \rangle$. Action a can be directly coded into PDDL as we will show in Section 5. However, some extra work is required to code actions in DL-PA.

Before executing an action, we have to check that its effects are not in conflict ; if two effects are, the action can only be executed if these effects cannot both happen. The formula

$$\mathsf{conflict}_a = \bigvee_{\substack{\langle \mathsf{cond}_i, \pi_i \rangle, \langle \mathsf{cond}_j, \pi_j \rangle \in \mathsf{eff}(a),\\ \pi_i \text{ and } \pi_j \text{ are in conflict}}} \mathsf{cond}_i \wedge \mathsf{cond}_j$$

is true when there is a couple of effects of a whose effects are conflicting and whose conditions are both true. We suppose that the disjunction of an empty set is $\bot$. In our running example there are no conflicts : we have $\mathsf{conflict}_{\mathsf{call}_{ij}} = \bot$.

We also have to take care that the truth value of conditions cannot change because of effects. To achieve this, we copy our conditions at the beginning of the action and evaluate copies. Note that this technical problem is specific to DL-PA ; when coding a planning problem in PDDL this problem does not arise since in PDDL all conditions are checked before any effects are produced. We use copies of atomic variables α, noted α'. We suppose that α' is syntactically different from every $\beta \in \mathsf{Fml}_0$. Then the copy of a formula φ is noted φ' and is defined such that every α occurring in φ is replaced by its copy α'. Furthermore, we define the program assigning the value of α to its copy α' by

$$\mathsf{copy}(\alpha) = (\alpha?; +\alpha') \sqcup (\neg\alpha?; -\alpha').$$

Given a set of atoms $A = \{\alpha_1, \ldots, \alpha_m\}$, we define the program $\mathsf{copy}(A) = \mathsf{copy}(\alpha_1); \ldots; \mathsf{copy}(\alpha_m)$ storing the the truth values of the α_i by copying them. (Note that the order does not matter.) Remark that $\varphi \leftrightarrow [\mathsf{copy}(\mathsf{var}(\varphi))]\varphi'$ is valid.

We can finally associate to a the DL-PA program :

$$\mathsf{prog}_a = \mathsf{pre}(a)?; \neg\mathsf{conflict}_a?; \mathsf{copy}\Big(\bigcup_{\langle \mathsf{cond}_i, \pi_i\rangle \in \mathsf{eff}(a)} \mathsf{var}(\mathsf{cond}_i)\Big); \\ \big((\mathsf{cond}_1{}'?; \pi_1) \sqcup \neg\mathsf{cond}_1{}'?\big); \ldots; \big((\mathsf{cond}_n{}'?; \pi_n) \sqcup \neg\mathsf{cond}_n{}'?\big).$$

Proposition 1
The program prog_a *is well-defined ; in other words,* $||a|| = ||\mathsf{prog}_a||$.

Proof 1
Take a state s.

If $s \notin ||\mathsf{pre}(a)||$, *then the program* prog_a *fails because of the first test* $\mathsf{pre}(a)?$, *thus is not related to any state, as specified by* $||a||$. *The next test* $\mathsf{conflict}_a?$ *exactly corresponds to the constraint on conflict in* $||a||$ *: if* $\mathsf{conflict}_a$ *is false in* s *then no couple of conditional effects of* a *is in conflict and the program continues.*

Furthermore, the ordering of the sequence prog_a *does not matter because we test copies of the atoms occurring in the conditions (thus we test the truth of every condition in* s*). By the semantics of the non-deterministic choice operator* $\sqcup$, *each instruction* $(\mathsf{cond}_i{}'?; \pi_i) \sqcup \neg\mathsf{cond}_i{}'?$ *is equivalent to an if-then control sequence : if the guard* $\mathsf{cond}_i{}'$ *is true then* π_i *is executed, whereas if* $\neg\mathsf{cond}_i{}'$ *is true then no instruction is executed and the program continues. Therefore, if the condition is verified in* s, *then, due to the semantics of* $+\alpha$ *and* $-\alpha$, *if* π_i *is of the form* $+\alpha$ *then* α *is added to* s, *while if* π_i *is of the form* $-\alpha$ *then* α *is removed from* s, *as specified in* $||a||$.

It is interesting to note that coding epistemic planning into a formal logic has allowed us to identify the problem of conflicting effects. This highlights the fact that an automatic planner designed to solve planning problems expressed in PDDL and consisting of actions with conditional effects must take into account the possibility of conflicting effects : if two conditions of an action a are true and their respective effects are conflicting then action a cannot be executed. The possibility of conflicting effects is not explicitly prohibited in PDDL and this potential problem must therefore be managed within the automatic planner.

Example 3 (Example 1, ctd.)
The action call_{ij} *is associated to the program :*

$$\mathsf{prog}_{\mathsf{call}_{ij}} = \top?; \neg\bot?; \mathsf{copy}\big(\{\mathbf{Kw}_i s_k \ : \ 1 \leq k \leq n\} \cup \{\mathbf{Kw}_j s_k \ : \ 1 \leq k \leq n\}\big); \\ \big((\mathbf{Kw}_i s_1{}'?; +\mathbf{Kw}_j s_1) \sqcup \neg\mathbf{Kw}_i s_1{}'?\big); \\ \ldots; \\ \big((\mathbf{Kw}_i s_n{}'?; +\mathbf{Kw}_j s_n) \sqcup \neg\mathbf{Kw}_i s_n{}'?\big); \\ \big((\mathbf{Kw}_j s_1{}'?; +\mathbf{Kw}_i s_1) \sqcup \neg\mathbf{Kw}_j s_1{}'?\big); \\ \ldots; \\ \big((\mathbf{Kw}_j s_n{}'?; +\mathbf{Kw}_i s_n) \sqcup \neg\mathbf{Kw}_j s_n{}'?\big)$$

Note that in this case, both $\mathsf{pre}(a)?$ *and* $\neg\mathsf{conflict}_a?$ *(which are respectively* $\top?$ *and* $\neg\bot?$*) are always true, and hence could clearly be omitted.*

Proposition 2
A planning task $\langle \mathsf{Act}, s_0, \mathsf{Goal}\rangle$ *is solvable if and only if* s_0 *satisfies* $\big\langle (\bigsqcup_{a\in\mathsf{Act}} \mathsf{prog}_a)^* \big\rangle \mathsf{Goal}$.

Proof 2
We know by Proposition 1 that prog_a *behaves correctly and produces the same effects as action* a.

We can read the formula $\big\langle (\bigsqcup_{a\in\mathsf{Act}} \mathsf{prog}_a)^* \big\rangle \mathsf{Goal}$ *as follows :* $\bigsqcup_{a\in\mathsf{Act}} \mathsf{prog}_a$ *means choosing one action from* Act, *then applying the star operator to this means a sequence of arbitrary length of arbitrary actions, i.e., a plan* π.

For a planning task to be solvable, at least one state satisfying the goal must be reachable from s_0 by actions from Act*, that is, there is a sequence of actions such that executing them leads to a state in* $||\mathsf{Goal}||$*. By the semantics, s_0 satisfies $\langle\pi\rangle\mathsf{Goal}$ if and only if there exists an execution of π starting at s_0 such that we end in a state satisfying* Goal *; this corresponds to our definition of reachable. Therefore the task is solvable if and only if the formula is satisfied at the initial state.*

5 Encoding into PDDL

In this section we present a method for automatically encoding planning problems defined in our logic into PDDL. As already observed, in PDDL we do not need to make copies of variables as we were obliged to do in DL-PA. We also make the assumption that the automatic planner does not allow the execution of an action with conflicting effects : this means that we do not explicitly code the test for conflicts as we did in DL-PA. Under this assumption, Proposition 2 guarantees that the planner will find a solution-plan if and only if one exists.

Consider the planning task $\langle\mathsf{Act}, s_0, \mathsf{Goal}\rangle$. In order to encode this planning problem into PDDL, some requirement flags should be set depending on the form of the formulas cond_i used in the effects $\langle\mathsf{cond}_i, \pi_i\rangle$ of actions from Act, and on the formula Goal. Since all these formulas are in $\mathsf{Fml}_{\mathsf{Bool}}$, they have to be encoded into PDDL using two boolean operators :

— `and` directly with default requirement flag `:strips`,

— `not` using the requirement flags `:negative-preconditions` if used only on atomic propositions, and `:disjunctive-preconditions` in addition if used on conjunctions.

Moreover, the operator `or` could also be used, when this latter requirement flag is set, in order to simplify writing. Given a boolean formula $\varphi \in \mathsf{Fml}_{\mathsf{Bool}}$, we define a recursive function $tr_{PDDL}(\varphi)$ which returns the encoding of φ into PDDL :

$$\begin{aligned}
tr_{PDDL}(p) &::= \texttt{(p)}\\
tr_{PDDL}(\mathbf{Kw}_i p) &::= \texttt{(Kw i p)}\\
tr_{PDDL}(\neg\varphi) &::= \texttt{(not } tr_{PDDL}(\varphi)\texttt{)}\\
tr_{PDDL}(\varphi_1 \wedge \varphi_2) &::= \texttt{(and } tr_{PDDL}(\varphi_1)\ tr_{PDDL}(\varphi_2)\texttt{)}\\
tr_{PDDL}(\varphi_1 \vee \varphi_2) &::= \texttt{(or } tr_{PDDL}(\varphi_1)\ tr_{PDDL}(\varphi_2)\texttt{)}
\end{aligned}$$

An atomic proposition $p \in \mathsf{PVar}$ is encoded as a fluent without any parameters by its name `(p)`, whereas $\mathbf{Kw}_i p$ with $i \in \mathsf{Agt}$ and $p \in \mathsf{PVar}$ is encoded by a special fluent with i and p as parameters `(Kw i p)`. The initial state s_0 is then trivially encoded, and the Goal and the preconditions of all actions in Act can be encoded using the function above. Moreover, the requirement flag `:conditional-effects` must be set, unless all the effects of the actions in Act are of the form $\langle\top, \pi\rangle$ (recall that $\pi \in \mathsf{Pgm}_0$ is an assignment). Consider an action $a = \langle\mathsf{pre}(a), \{\langle\mathsf{cond}_1, \pi_1\rangle, \ldots, \langle\mathsf{cond}_n, \pi_n\rangle\}\rangle$. For every $\langle\mathsf{cond}_i, \pi_i\rangle \in \mathsf{eff}(a)$, we add the following conditional effects to `:effect` of the action a in PDDL :

— if $\pi_k = +p$, then we add `(when` $tr_{PDDL}(\mathsf{cond}_k)$ `(p))` ;

— if $\pi_k = -p$, then we add `(when` $tr_{PDDL}(\mathsf{cond}_k)$ `(not p))` ;

— if $\pi_k = +\mathbf{Kw}_i p$, then we add `(when` $tr_{PDDL}(\mathsf{cond}_k)$ `(Kw i p))` ;

— if $\pi_k = -\mathbf{Kw}_i p$, then we add `(when` $tr_{PDDL}(\mathsf{cond}_k)$ `(not (Kw i p)))` ;

Example 4 (Example 1, ctd.)
The action call_{ij} *is coded in PDDL, using conditional effects as follows :*

```
(:action call
     :parameters (?i - gossip ?j - gossip)
     :effect (and (when (Kw ?i s1) (Kw ?j s1))
                  ...
                  (when (Kw ?i sn) (Kw ?j sn))
                  (when (Kw ?j s1) (Kw ?i s1))
                  ...
                  (when (Kw ?j sn) (Kw ?i sn))))
```

Almost all planners from last International Planning Competition (IPC 2014) [1] handle conditional effects and negative preconditions, and most of them handle disjunctive preconditions. Moreover, most of them also handle universal effects which allows us to encode problems more succinctly, and to model actions from a planning domain independently of the actual problem as we can see in the following example.

Example 5 (Example 1, ctd.)
Here is the action call_{ij} *redefined in PDDL using universal effects :*

```
(:action call
     :parameters (?i - gossip ?j - gossip)
     :effect (and (forall (?s - secret)
                    (and (when (Kw ?i ?s) (Kw ?j ?s))
                         (when (Kw ?j ?s) (Kw ?i ?s))))))
```

6 Variants

Several variants of the Gossip Problem have been studied in the literature, and a survey of these alternatives and the associated results was published by Hedetniemi *et al.* (1988). We enumerate some of them and give an intuition on how they could be encoded as actions.

A first variant is to consider a non-complete graph instead of a complete one, that is, some people cannot call each other. We can add a variable $linked_{ij}$ such that the precondition of the action call_{ij} is now $linked_{ij} \wedge linked_{ji}$. In this case, if the graph is connected, the minimal number of calls to obtain the solution is either $2n-4$ or $2n-3$, depending on the structure of the graph (Harary & Schwenk, 1974).

We can also restrict calls to one-way communications, such as mails, and thus considering directed graphs. This can be achieved by only providing information from i in a call, thus call_{ij} is cut in half and becomes $\langle \top, \{\langle \mathbf{Kw}_i s_1, +\mathbf{Kw}_j s_1\rangle, \ldots, \langle \mathbf{Kw}_i s_n, +\mathbf{Kw}_j s_n\rangle\}\rangle$. (Thus call_{ij} is not equivalent to call_{ji} anymore.) If the directed graph is strongly connected, the minimal number of calls is $2n-2$ (Harary & Schwenk, 1974).

A well-known variant of the gossip problem is the NOHO ("No One Hears Own") version (Baker & Shostak, 1972) where nobody should hear their own secret. In this version, assuming agents cannot hide secrets, a call can only take place between i and j if the precondition $\neg\mathbf{Kw}_j s_i \wedge \neg\mathbf{Kw}_i s_j$ is satisfied, that is, j cannot tell i the secret of agent i and vice versa. Such a problem can only have a solution if n is even, in which case the minimal number of calls is still $2n-4$ (West, 1982).

An interesting variant for temporal planning is to consider time instead of calls, and thus suppose that several calls are executed in parallel. If the number of participants n is even, the time taken is $\lceil \log_2 n \rceil$, and if n is odd, it is $\lceil \log_2 n \rceil + 1$ (Bavelas, 1950; Landau, 1954; Knödel, 1975).

Agents can be interested in learning at least k secrets without caring which secrets they access. This can be modelled by changing the goal into a disjunction of all possible outcomes (conjunctions of $\mathbf{Kw}_i s_j$) such that each agent learns k secrets. A complete study of the problem (called the *partial gossiping problem* Richards & Liestman (1988)) and results depending on k can be found in (Chang & Tsay, 1996).

Finally, a last variant could be to find a sequence of calls such that no agent is allowed to lie by omission (at each call, she always tells all the secrets she knows), but in the end, some people should not know some secrets. This can actually be pretty easily achieved by only changing the goal (actions remain the same) to allow negative literals in the conjunctions. As an example, take $n=3$, a goal could be $\mathbf{Kw}_1 s_1 \wedge \mathbf{Kw}_1 s_2 \wedge \mathbf{Kw}_1 s_3 \wedge \mathbf{Kw}_2 s_1 \wedge \mathbf{Kw}_2 s_2 \wedge \mathbf{Kw}_2 s_3 \wedge \mathbf{Kw}_3 s_1 \wedge \neg\mathbf{Kw}_3 s_2 \wedge \mathbf{Kw}_3 s_3$ (everyone knows every secret except 3 that does not know the secret of 2). This can be achieved by the sequence of actions $\mathsf{call}_{13}, \mathsf{call}_{12}$. To the best of our knowledge, this variant has not been studied. This problem is not always solvable as some goals may not be reachable. For example, $\mathbf{Kw}_1 s_1 \wedge \mathbf{Kw}_1 s_2 \wedge \mathbf{Kw}_1 s_3 \wedge \mathbf{Kw}_2 s_1 \wedge \mathbf{Kw}_2 s_2 \wedge \neg\mathbf{Kw}_2 s_3 \wedge \mathbf{Kw}_3 s_1 \wedge \neg\mathbf{Kw}_3 s_2 \wedge \mathbf{Kw}_3 s_3$ (2 does not know the secret of 3 and conversely) is not attainable since call_{23} must never happen, so if 1 calls 2, then calls 3, then 3 will know the secret of 2 and if 1 calls 3 then 2, 2 will know the secret of 3.

What we observe is that all these variants are easy to model by modifying planning tasks, and hence can be directly encoded as DL-PA programs and PDDL tasks. This gives a sound and efficient method to fully solve this problem.

1. http ://helios.hud.ac.uk/scommv/IPC-14/planners.html

7 Conclusion and future work

In this article we have made a first step towards a realistic and provably-correct method for multiagent 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 intend to continue this line of research by incorporating other important aspects of multiagent 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), along with epistemic aspects such as higher-order observation (visibility on visibility of agents). In the long term, we also aim to generalise this approach to temporal planning in which actions are durative and may overlap.

We can note that, although we have only mentioned PDDL in this paper, 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 (Rintanen *et al.*, 2006).

Références

BAKER B. & SHOSTAK R. (1972). Gossips and telephones. *Discrete Mathematics*, **2**(3), 191–193.

BALBIANI P., HERZIG A. & TROQUARD N. (2013). Dynamic logic of propositional assignments : a well-behaved variant of PDL. In O. KUPFERMAN, Ed., *Proceedings of the 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS)*, p. 143–152.

BAVELAS A. (1950). Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, **22**(6), 725–730.

CHANG G. J. & TSAY Y.-J. (1996). The partial gossiping problem. *Discrete Mathematics*, **148**(1-3), 9–14.

HARARY F. & SCHWENK A. J. (1974). The communication problem on graphs and digraphs. *Journal of the Franklin Institute*, **297**(6), 491–495.

HEDETNIEMI S. M., HEDETNIEMI S. T. & LIESTMAN A. L. (1988). A survey of gossiping and broadcasting in communication networks. *Networks*, **18**(4), 319–349.

KNÖDEL W. (1975). New gossips and telephones. *Discrete Mathematics*, **13**(1), 95.

LANDAU H. G. (1954). The distribution of completion times for random communication in a task-oriented group. *The Bulletin of Mathematical Biophysics*, **16**(3), 187–201.

MC DERMOTT D., GHALLAB M., HOWE A., KNOBLOCK C., RAM A., VELOSO M., WELD D. & WILKINS D. (1998). *PDDL—The Planning Domain Definition Language*. Rapport interne CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

RICHARDS D. & LIESTMAN A. L. (1988). Generalizations of broadcasting and gossiping. *Networks*, **18**(2), 125–138.

RINTANEN J., HELJANKO K. & NIEMELÄ I. (2006). Planning as satisfiability : parallel plans and algorithms for plan search. *Artif. Intell.*, **170**(12-13), 1031–1080.

WEST D. B. (1982). A class of solutions to the gossip problem, part I. *Discrete Mathematics*, **39**(3), 307–326.