Ignorance is bliss : observability-based dynamic epistemic logics and their applications

Faustine Maffre

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue par :
Faustine MAFFRE

le vendredi 23 septembre 2016

Titre :

Le bonheur est dans l'ignorance :
logiques épistémiques dynamiques basées sur l'observabilité
et leurs applications

École doctorale et discipline ou spécialité :

ED MITT : Domaine STIC : Intelligence Artificielle

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (UMR 5505)

Directeur/trice(s) de Thèse :

Andreas HERZIG, Directeur de recherche, CNRS
Emiliano LORINI, Chargé de recherche, CNRS

Jury :

Martin COOPER, Professeur, UT3 Paul Sabatier, examinateur
Sylvie DOUTRE, Maître de conférences, UT1 Capitole, examinatrice
Jan van EIJCK, Professeur, Université d'Amsterdam, rapporteur
Andreas HERZIG, Directeur de recherche, CNRS, directeur de thèse
Emiliano LORINI, Chargé de recherche, CNRS, co-directeur de thèse
Mark RYAN, Professeur, Université de Birmingham, rapporteur
Nicolas TROQUARD, ATER, Université Paris-Est Créteil, examinateur
Bruno ZANUTTINI, Maître de conférences, Université de Caen Basse-Normandie, examinateur

Ignorance is bliss: observability-based dynamic epistemic logics and their applications

Le bonheur est dans l'ignorance : logiques épistémiques dynamiques basées sur l'observabilité et leurs applications

Faustine Maffre

Abstract

A recent trend in epistemic logic consists in studying construction of knowledge from the agents' observational abilities. It is based on the intuition that an agent's knowledge comes from three possible sources: her observations, communication with other agents, and inference. The approaches mainly focus on the first and suppose that the object of observations are propositional variables. This allows us to model knowledge in a more compact and intuitive way than with Hintikka's possible worlds semantics, where ignorance leads to an increase of the size of models.

The present thesis is part of that line of work. Its aim is to demonstrate that visibility-based epistemic logics provide a suitable tool for several important applications in the field of artificial intelligence. This requires to overcome some difficulties that seem to be inherent to the approach. At the heart of these problems are some counter-intuitive validities such as common knowledge of visibilities or the knowledge operator distributing over disjunctions. The first aim of the thesis is to solve these two issues and to illustrate logics of visibility with well-known toy examples of epistemic logic as well as with more general applications such as epistemic planning and boolean games.

We first introduce a dynamic epistemic logic that is based on what an agent can observe, including joint observation and observation of what other agents observe. This generalizes current visibility-based epistemic logics by solving the first issue: visibilities are not common knowledge, allowing us to reason about higher-order knowledge and furthermore, higher-order ignorance. We show how epistemic operators can be interpreted in this framework and identify the conditions under which the principles of positive and negative introspection are valid. We include dynamic operators: facts of the world and their observability can both be modified by assignment programs. We also provide a sound and complete axiomatization and prove that the model checking problem is PSPACE-complete.

We illustrate our framework by the gossip problem. Given n agents each of which has a secret (a fact not known to anybody else), the goal of the classical version of the gossip problem is to achieve shared knowledge of all secrets in a minimal number of phone calls. We generalize that problem and focus on higher-order shared knowledge: how many calls does it take to obtain shared knowledge of order k? This requires not only the communication of secrets, but also the communication of *knowledge about secrets*. We give a protocol and prove that it is correct and optimal.

An important application of our work is multi-agent epistemic plan-

ning. Given the description of the individual actions, using our formal logic allows us to reduce the solvability of a planning task to a model checking problem. This proves its low complexity compared to standard approaches to epistemic planning. We present an encoding of planning problems expressed in our logic into the standard planning language PDDL. Feeding the resulting problem into a PDDL planner provides a provably correct plan for the original planning problem.

We then extend our logic with control and strategic operators and study epistemic boolean games in this extension. We provide an axiomatization of the logic and establish that the model checking problem is PSPACE-complete. We show how one can reason about equilibria in epistemic boolean games, generalizing current approaches while keeping the same complexity.

We finally extend the original framework by adding public announcements and more generally publicly executed programs. This solves the second issue of visibility-based epistemic logics: epistemic operators do not distribute over disjunction any more. With this framework, we are now able to model most of the classical epistemic problems, starting with the muddy children puzzle. We establish that the model checking problem is PSPACE-complete.

Keywords: dynamic epistemic logic, higher-order observation, gossip problem, epistemic planning, epistemic boolean games, public announcements

Résumé

Une tendance récente en logique épistémique consiste à étudier comment construire la connaissance à partir des capacités d'observation des agents. Cela est basé sur l'intuition que la connaissance d'un agent provient de trois sources possibles : ses observations, la communication avec d'autres agents, et l'inférence. Les approches se concentrent principalement sur les observations et supposent que l'objet de ces observations sont des variables propositionnelles. Cela permet de modéliser les connaissances d'une manière plus compacte et intuitive qu'avec la sémantique des mondes possibles d'Hintikka, où l'ignorance conduit à une augmentation de la taille des modèles.

Cette thèse suit cette ligne de travail. Son objectif est de démontrer que les logiques épistémiques basées sur la visibilité constituent un outil approprié pour plusieurs applications importantes dans le domaine de l'intelligence artificielle. Cela exige de surmonter certaines difficultés qui semblent être inhérentes à l'approche : ces logiques incluent des validités contre-intuitives telles que la connaissance commune des visibilités ou l'opérateur de connaissance distribuant sur la disjonction. Le but premier de la thèse est de résoudre ces deux problèmes et d'illustrer les logiques de visibilité obtenues sur des exemples bien connus de la logique épistémique ainsi que sur des applications plus générales comme la planification épistémique et les jeux booléens épistémiques.

Nous présentons d'abord une logique épistémique dynamique qui est basée sur ce qu'un agent peut observer, y compris l'observation jointe et l'observation de ce que les autres agents observent. Cela généralise les logiques basées sur la visibilité actuelles en résolvant le premier problème : les visibilités ne sont plus connaissance commune, ce qui permet de raisonner sur des connaissances d'ordre supérieur, mais surtout sur l'ignorance d'ordre supérieur. Nous montrons comment les opérateurs épistémiques peuvent être interprétés dans ce cadre et identifions les conditions sous lesquelles les principes d'introspection positive et négative sont valides. Nous incluons des opérateurs dynamiques : les valeurs de vérité des faits du monde et de la visibilité des agents sur ces faits peuvent être modifiées par des affectations. Nous fournissons également une axiomatisation adéquate et complète et prouvons que le problème de model checking est PSPACE-complet.

Nous illustrons notre logique par le problème du bavardage. Étant donnés n agents avec chacun un secret (un fait non connu par les autres), le but de la version classique du problème du bavardage est de parvenir à ce chaque agent connaisse tous les secrets en un nombre minimal d'appels téléphoniques. Nous généralisons ce problème et nous concentrons sur le partage des connaissances d'ordre supérieur : combien d'appels

sont nécessaires pour obtenir la connaissance partagée d'ordre k ? Cela nécessite non seulement la communication des secrets, mais aussi la communication de la connaissance à propos de secrets. Nous exposons un protocole et prouvons qu'il est correct et optimal.

Une application importante est la planification épistémique multi-agents. Compte tenu de la description des actions individuelles, utiliser notre logique formelle nous permet de réduire la solvabilité d'une tâche de planification à un problème de model checking. Cela prouve sa faible complexité par rapport aux approches standards. Nous présentons un encodage des problèmes de planification exprimés dans notre logique vers PDDL, le langage classique de planification. Donner le problème résultant à un planificateur PDDL fournit un plan qui résout le problème de planification original.

Nous étendons ensuite notre logique en ajoutant le contrôle des variables et des opérateurs stratégiques et étudions les jeux booléens épistémiques dans cette extension. Nous fournissons une axiomatisation de la logique et établissons que le problème de model checking est PSPACE-complet. Nous montrons comment il est possible de raisonner sur les équilibres dans les jeux booléens épistémiques, généralisant les approches actuelles tout en gardant la même complexité.

Finalement, nous étendons la logique originale en ajoutant des annonces publiques et plus généralement des programmes exécutés publiquement. Cela résout le second problème des logiques épistémiques basées sur la visibilité : les opérateurs épistémiques ne distribuent plus sur la disjonction. Grâce à cette nouvelle logique, nous sommes capables de modéliser la plupart des problèmes épistémiques classiques, notamment le problème des enfants sales. Nous établissons que le problème model checking est PSPACE-complet.

Mots-clés : logique épistémique dynamique, observation d'ordre supérieur, problème du bavardage, planification épistémique, jeux booléens épistémiques, annonces publiques

Remerciements

Il m'a été dit de ne pas trop attendre pour rédiger les remerciements. Ce type d'exercice n'étant pas mon domaine de prédilection, j'ai choisi de les limiter à un nombre restreint de personnes. Je suis cependant persuadée que l'aboutissement d'un projet tel qu'une thèse passe par un ensemble de rencontres et d'environnements propices, d'abord à son amorçage, puis à son développement. En tout cas, cela a été le cas pour moi : l'université Champollion d'Albi, l'IRIT et en particulier l'équipe LILaC dans laquelle j'ai travaillée, la communauté scientifique de mon domaine, mes parents et mes amis, sont tout autant à citer que les personnes qui vont suivre.

Je commence bien sûr par remercier mes encadrants : Andreas Herzig et Emiliano Lorini. Pendant toute la thèse et même avant, lors de mon stage de fin de master, ils ont su m'inspirer dans ma façon de faire de la recherche et je les considère comme les meilleurs exemples que j'ai pu suivre dans ce domaine. Ils ont su m'aiguiller tout en me laissant travailler de façon autonome, corriger mes erreurs et critiquer mon travail avec indulgence et bienveillance. J'ai beaucoup apprécié de travailler avec eux et j'ai énormément appris pendant ces trois ans.

Je remercie Jan van Eijck et Mark Ryan, pour avoir relu la thèse avec attention et y avoir apporté des retours pertinents. Je remercie aussi Martin Cooper, pour avoir diminué de façon significative la quantité de fautes de grammaire anglaise dans la thèse, ainsi que Frédéric Maris, pour avoir organisé les journées MAFTEC auxquelles j'ai eu le plaisir de participer et lors desquelles j'ai pu présenter mes travaux de thèse. De plus, j'ai été ravie de travailler avec Martin, Frédéric et Pierre Régnier sur les applications de notre logique à la planification et d'avoir pu tenter avec eux de réduire la frontière entre ces deux domaines. Merci aussi à Sylvie Doutre, Nicolas Troquard et Bruno Zanuttini d'avoir accepté de faire partie de mon jury de soutenance. À tout le jury, merci pour leurs questions et remarques pertinentes pendant la soutenance et pour l'ambiance d'échange qu'ils ont su créer lors de celle-ci. Quoiqu'un peu stressée au départ, j'ai trouvé ce moment et la

journée en général très agréable.

J'adresse des remerciements particuliers à Martin Strecker, pour m'avoir offert l'occasion de découvrir l'IRIT et la recherche avant même le début du master, et pour avoir suivi mes progrès depuis ce moment jusqu'à la fin de la thèse. Je remercie aussi Thierry Montaut, pour avoir été un si bon enseignant durant mes années de licence à Albi. J'espère avoir l'opportunité de retourner dans cette université, peut-être pour y enseigner quelques heures, dans laquelle je me suis tant plu pendant trois ans.

Enfin, je voudrais remercier Armelle Bonenfant, avec qui j'ai passé de très bons moments de travail, lors de l'organisation des projets de programmation, ou de détente, autour d'un chocolat ou d'un jeu de société.

Je remercie aussi le Centre International de Mathématiques et d'Informatique de Toulouse (CIMI) pour avoir financé ce travail de thèse.

Contents

1 Introduction: state of the art

This chapter introduces several notions on which we will rely in the thesis, then gives the general outline. It also implicitly presents notation that will be adopted in the following.

Throughout the whole thesis we assume $Prop = \{p, q, \ldots\}$ is a countable non-empty set of propositional variables and $Agt = \{1, \ldots, n\}$ is a finite non-empty set of agents.

Contents

Résumé du chapitre

Ce chapitre présente les notions sur lesquelles nous allons nous appuyer dans la thèse, puis en donne les grandes lignes. Implicitement, il présente aussi les notations qui seront adoptées dans ce qui suit.

Le sujet principal de la thèse concerne les logiques épistémiques basées sur la visibilité, c'est-à-dire les logiques dans lesquelles la connaissance d'un agent est construite à partir de ce qu'elle voit. La première section est consacrée aux logiques existantes de ce type.

La deuxième section concerne les logiques dynamiques, et plus principalement la logique DL-PA [Herzig et al., 2011; Balbiani et al., 2013b], dans laquelle les programmes atomiques sont des affectations de variables à vrai ou faux. Les logiques présentées dans la thèse incluront toutes des programmes de ce type, qui nous permettront de modifier de façon simple les connaissances des agents.

La troisième section donne quelques logiques incluant une notion de contrôle : si un agent contrôle une variable, elle peut changer sa valeur de vérité. Des logiques comme CL-PC [van der Hoek and Wooldridge, 2005] utilisent ce contrôle pour exprimer les capacités stratégiques des agents, un peu de la même façon que la visibilité par rapport à la connaissance. Nous allons voir dans la thèse qu'intégrer le contrôle à notre logique permet de raisonner sur les jeux booléens, dans lesquels les agents peuvent modifier les variables sous leur contrôle pour atteindre leur but, et même sur les jeux booléens épistémiques, dans lesquels ces buts peuvent concerner la connaissance.

La quatrième section donne des informations sur la planification ainsi que sur la planification épistémique, où les actions et les buts peuvent concerner les connaissances des agents. Souvent, la planification épistémique est basée sur DEL [Bolander and Andersen, 2011; Löwe et al., 2011], qui offre un cadre très expressif, mais très rapidement indécidable. D'autres méthodes plus modestes ont été développées pour palier à ce problème. Dans ce même but, nous proposons dans la thèse d'appliquer notre formalisme à la planification épistémique.

Enfin, la dernière section explicite le but de la thèse et son organisation, ainsi que les contributions produites durant la thèse mais non détaillées dans celle-ci.

1.1 Epistemic logics and visibility

In his seminal book [Hintikka, 1962], Jaakko Hintikka proposed to give truth conditions to epistemic operators in terms of *possible worlds*: agent i knows that φ if, from the perspective of the actual world, this statement is true in all worlds that are possible, or indistinguishable, for i. Such epistemic logics resort to Kripke's possible worlds semantics and were popularized in computer science in the 90's by Fagin et al. [Fagin et al., 1995], who were also the first to advocate the use of S5 as *the* logic of knowledge (while Hintikka had settled for S4).

While these semantics are natural for modalities such as temporal

or dynamic operators, there is a bigger gap between knowledge and possible worlds models. The problem is that the number of possible worlds entertained by an agent is inversely proportional to her knowledge. Possible worlds are therefore not very plausible candidates for cognitive models. Furthermore, when one tries to model things such as distributed systems, such models are typically way too big to be stored and analysed on a computer. Possible worlds models as they stand are therefore not a good basis for system verification. This lead theoretical computer scientists to investigate other, more compact ways to represent knowledge, with the aim of making things more feasible.

A first attempt based on interpreted systems can be found in [Lomuscio and Raimondi, 2006], using the model checker MOCHA. In interpreted systems, agents are associated with private local states, which, put together, make up the global state. Their idea is to construct knowledge from these local states: i cannot distinguish two global states if her local states in these global states are equal.

Taking this work as an inspiration, several authors [van der Hoek et al., 2011, 2012] investigated how epistemic logic could be grounded on the notion of visibility (or observability) of propositional variables. The basic idea is that the sentence "agent i knows that the atomic fact p is true" can be reduced to:

- p is true and
- agent i observes the truth value of p.

Similarly, "agent i knows that p is false" can be reduced to:

- p is false and
- agent i observes the truth value of p.

It is therefore supposed that every agent has a set of propositional variables that she can observe, in the sense that she knows their truth values. The other way round, any combination of truth values of the non-observable variables is possible for the agent. Such observability information allows us to reconstruct Hintikka's semantics: two worlds are indistinguishable for agent i if and only if every variable observed by i has the same value in both worlds. The motivation of van der Hoek et al. was not only to decrease the complexity of the model checking problem, but also to conceptually study and axiomatize these *logics of visibility*.

1.1.1 A logic of visibility: ECL-PC

As in standard epistemic logic, the language of logics of visibility extends the language of propositional logic with the unary epistemic operator K_i.

In this section, we present observability-based epistemic logics as done in [van der Hoek et al., 2011]. In this paper, van der Hoek et al. introduce the logic ECL-PC (Epistemic Coalition Logic of Propositional Control), which extends the logic of control and strategy CL-PC [van der Hoek and Wooldridge, 2005] (Coalition Logic of Propositional Control) with epistemic operators.

We focus here on the epistemic part: we will come back to the strategic operator $\Diamond_i$ in Section 1.3. On the other hand, we include the operator of common knowledge CK (that is not studied in [van der Hoek et al., 2011]) in the standard sense: $CK\varphi$ reads "it is common knowledge among all the agents that φ is true" (that is, "everybody knows φ is true and everybody knows that everybody knows that φ is true, and so on *ad infinitum*"). We propose an axiomatization of the epistemic fragment of ECL-PC extended by common knowledge.

In ECL-PC, each agent is associated to a (fixed) set of propositional variables, which are the variables she sees. We choose to encode visibilities not in the model, but by atomic propositions of the form $S_i\, p$, as we will reuse them in the rest of the thesis. The atomic formula $S_i\, p$ reads "agent i sees the value of the propositional variable p." We call these, along with propositional variables, *atomic formulas*, or, for short, *atoms*. Formulas are interpreted over a *valuation*, which is a set of atoms, and a *visibility model*, which contains every possible valuation (every combination of atoms). To comply with ECL-PC semantics, we constrain the indistinguishability relation so that visibilities stay the same in every related valuation. (It is a hypothesis that we will relax in the following chapters.)

Language of ECL-PC

From the set of propositional variables $Prop$ we define the set of *atoms* by

$$ATM = Prop \cup \{S_i\, p : i \in Agt, p \in Prop\}.$$

We denote atoms by α, α', β, β', etc.

Then the language we study is defined by the following grammar:

$$\varphi ::= \alpha \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi \mid CK\varphi$$

where α ranges over ATM and i over Agt. The formula $K_i\varphi$ reads "i knows that φ on the basis of what she observes" and $CK\varphi$ reads "all agents commonly know that φ on the basis of what they observe." We define the operator of shared knowledge EK_J, for $J \subseteq Agt$ a non-empty set of agents, by

$$EK_J\varphi = \bigwedge_{i \in J} K_i\varphi.$$

The formula $EK_J\varphi$ reads "every agent in J knows that φ."

Semantics of ECL-PC

In order to prepare the ground for the following chapters, we give the semantics of ECL-PC in a form that slightly differs from the original presentation, while being equivalent.

A *valuation* $V \in 2^{ATM}$ is a subset of the set of atoms ATM: the atoms that are currently true. For an atom α, we write $V(\alpha) = V'(\alpha)$ if and only if α has the same value in V and in V', i.e., either $\alpha \in V$ and $\alpha \in V'$, or $\alpha \notin V$ and $\alpha \notin V'$. The (unique) visibility model $M_{vis} = 2^{ATM}$ contains every possible valuation.

As said above, the indistinguishability relations of M_{vis} are constructed from visibility information. We denote by $V \sim_i V'$ the fact that valuation V is indistinguishable from valuation V' for i. Formally:

$$V \sim_i V' \text{ iff } S_i\, p \in V \text{ implies } V(p) = V'(p), \text{ and}$$
$$\text{for every } j \in Agt, V(S_j\, p) = V'(S_j\, p).$$

The first part is the formal counterpart of the intuition behind visibility: two valuations V and V' are related for i if every atom α that i sees at V has the same value in V and V'. The second part ensures that visibilities of all agents on every propositional variable are identical in all related valuations; remember that this is a hypothesis of ECL-PC. We will see how this implies complete information about visibility.

As the reader may observe, each relation $\sim_i$ is reflexive, transitive and euclidean, and is therefore an equivalence relation. So each $\sim_i$ divides the visibility model M_{vis} into equivalence classes of valuations in which visibilities remain constant.

From this relation, we define the relation for common knowledge, noted $\sim_{Agt}$, as:

$$\sim_{Agt} = \Big(\bigcup_{i \in Agt} \sim_i \Big)^*$$

where, for a relation R, R^* is the reflexive and transitive closure of R. This is the standard way [Fagin et al., 1995]: a statement φ must be true in any reachable world by any agent in any number of steps for it to be common knowledge.

Example 1.1. Suppose $Prop = \{p, q\}$ and $Agt = \{1, 2\}$. Then $ATM = \{p, q, S_1\, p, S_1\, q, S_2\, p, S_2\, q\}$ and thus the visibility model contains $2^6 = 64$ valuations. Let us only consider the subset of this model where 1 only sees p and 2 only sees q. It is made up of the following four valuations:

$$\{S_1\, p, S_2\, q\}, \{S_1\, p, S_2\, q, p\}, \{S_1\, p, S_2\, q, q\}, \{S_1\, p, S_2\, q, p, q\}.$$

These valuations are related such that 1 cannot distinguish valuations differing only in the value of q and 2 cannot distinguish valuations differing only in the value of p. This is illustrated in Figure 1.1.

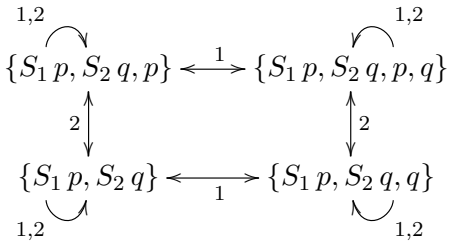


Figure 1.1: Part of the visibility model where agent 1 only sees p and agent 2 only sees q.

Example 1.2. Consider the same sets of propositional variables $Prop$ and agents Agt as in Example 1.1. The subset where 1 sees p and q while 2 only sees q is depicted in Figure 1.2.

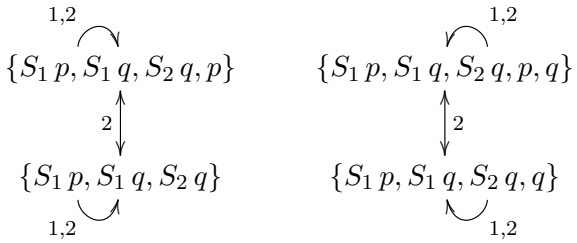


Figure 1.2: Part of the visibility model where agent 1 has full visibility but agent 2 only sees q.

Formulas are interpreted at a valuation of M_{vis}. As the visibility model is unique we keep it implicit and only mention the current valuation V in the truth conditions. They are defined as follows:

$V \models \alpha$	iff $\alpha \in V$
$V \models \neg\varphi$	iff not $(V \models \varphi)$
$V \models \varphi \wedge \varphi'$	iff $V \models \varphi$ and $V \models \varphi'$
$V \models K_i\varphi$	iff $V' \models \varphi$ for every V' such that $V \sim_i V'$
$V \models CK\varphi$	iff $V' \models \varphi$ for every V' such that $V \sim_{Agt} V'$

Therefore agent i knows that φ if φ holds in every valuation i considers possible, based on what i sees; and φ is common knowledge if it holds in any valuation accessible by a path over any indistinguishability relation.

When $V \models \varphi$ we say that V is a *model* of φ. The set of models of φ is noted $\|\varphi\|$. A formula φ is *satisfiable* if it has a model, i.e., if $\|\varphi\| \neq \emptyset$; it is *valid* if φ is true in all models, i.e., if $\|\varphi\| = 2^{ATM}$.

Example 1.3. In Example 1.2, no valuation other than the current one is possible for agent 1; so she always knows the state of the valuation:

$$V \models (K_1 p \vee K_1 \neg p) \wedge (K_1 q \vee K_1 \neg q)$$

for every V of Figure 1.2. In contrast, agent 2 only knows the value of q, but knows that 1 knows the value of p:

$$V \models (\neg K_2 p \wedge \neg K_2 \neg p) \wedge (K_2 q \vee K_2 \neg q) \wedge K_2(K_1 p \vee K_1 \neg p)$$

for every V of Figure 1.2. Finally, we have

$$\{S_1\, p, S_1\, q, S_2\, q, q\} \models CKq.$$

The reader may observe that the interpretations of CK and EK_{Agt} actually coincide. This does not only happen in this example, as we shall see in the next section.

Axiomatization of ECL-PC

Thanks to the visibility atoms we can axiomatize the epistemic fragment of ECL-PC by means of reduction axioms, in a way that is a bit simpler than that of [van der Hoek et al., 2011]. This also simplifies the completeness proof. Our axiomatization of ECL-PC without strategic operators is as follows:

- some axiomatization of $\mathsf{S5}^n_C$ (epistemic logic S5 with n agents and with common knowledge), e.g. that of [Fagin et al., 1995];
- axioms reducing literal knowledge to visibility:

$$K_i p \leftrightarrow p \wedge S_i\, p \qquad (Red_{K,p})$$
$$K_i \neg p \leftrightarrow \neg p \wedge S_i\, p \qquad (Red_{K,\neg p})$$

- positive and negative mutual introspection of visibility:

$$S_i\, p \rightarrow K_j S_i\, p \qquad (PI_S)$$
$$\neg S_i\, p \rightarrow K_j \neg S_i\, p \qquad (NI_S)$$

- distribution of knowledge over disjunction of literals:

$$K_i\Big(\bigvee_{\alpha\in A^+} \alpha \vee \bigvee_{\alpha\in A^-} \neg\alpha\Big) \leftrightarrow \Big(\bigvee_{\alpha\in A^+} K_i\alpha\Big) \vee \Big(\bigvee_{\alpha\in A^-} K_i\neg\alpha\Big) \qquad (Red_{K,\vee})$$

  where A^+ and A^- are sets of atoms such that $A^+ \cap A^- = \emptyset$.

Observe that Axiom ($Red_{K,\vee}$) is not valid when the clause contains contradictory literals, i.e., when $A^+ \cap A^- \neq \emptyset$.

Proposition 1.1. *Our axiomatization of* ECL-PC *without strategic operators is sound.*

Proof. The axiomatization of $\mathsf{S5}^n_C$ is sound because, as we will see in Section 1.1.2, the model of ECL-PC is a particular Kripke model. Therefore the axioms of $\mathsf{S5}^n_C$ are valid and its inference rules preserve validity.

- Axioms ($Red_{K,p}$), ($Red_{K,\neg p}$) and ($Red_{K,\vee}$) will still be true in the logics of visibility that we will present in the next chapters; their proofs are similar to Proposition 2.10 on page 44.
- On the other hand, we will render invalid (PI_S) and (NI_S) in the following; we detail their proof here.

  Consider an arbitrary valuation $V \in 2^{ATM}$. By the definition of $\sim_j$, we have $V(S_i\, p) = V'(S_i\, p)$ for every V' such that $V \sim_j V'$.

  Suppose $V \models S_i\, p$, then for every V' such that $V \sim_j V'$ we have $V' \models S_i\, p$ and therefore $V \models K_j S_i\, p$. Now suppose $V \not\models S_i\, p$, then for every V' such that $V \sim_j V'$ we have $V' \not\models S_i\, p$ and therefore $V \models K_j \neg S_i\, p$.

Therefore all axioms are valid. □

Proposition 1.2. *The following formulas are theorems of* ECL-PC.

- *Reduction of visibility to knowing whether:*

$$K_i p \vee K_i \neg p \leftrightarrow S_i\, p \tag{1.1}$$

- *Reduction of common knowledge to shared knowledge:*

$$CK p \leftrightarrow EK_{Agt} p \tag{1.2}$$
$$CK \neg p \leftrightarrow EK_{Agt} \neg p \tag{1.3}$$

- *Common knowledge of visibility:*

$$S_i\, p \rightarrow CK S_i\, p \tag{1.4}$$
$$\neg S_i\, p \rightarrow CK \neg S_i\, p \tag{1.5}$$

- *Distribution of common knowledge over disjunction of literals:*

$$CK\Big(\bigvee_{\alpha\in A^+} \alpha \vee \bigvee_{\alpha\in A^-} \neg\alpha\Big) \leftrightarrow \Big(\bigvee_{\alpha\in A^+} CK\alpha\Big) \vee \Big(\bigvee_{\alpha\in A^-} CK\neg\alpha\Big) \tag{1.6}$$

  where A^+ and A^- are sets of atoms such that $A^+ \cap A^- = \emptyset$.

Proof. We derive formulas from axioms.

- (1.1). It follows from the reduction axioms ($Red_{K,p}$) and ($Red_{K,\neg p}$) that $K_i p \vee K_i \neg p$ is equivalent to $(p \wedge S_i\, p) \vee (\neg p \wedge S_i\, p)$. The latter is equivalent to $S_i\, p$ by propositional logic.

- (1.2). The left-to-right sense is a theorem of $\mathsf{S5}^n_C$. For the right-to-left sense, we have $EK_{Agt}p \rightarrow K_j p$ by propositional reasoning. Moreover, we have $EK_{Agt}p \rightarrow K_j \bigwedge_{i \in Agt} S_i\, p$ by ($Red_{K,p}$) and (PI_S). Thus

$$EK_{Agt}p \rightarrow K_j\Big(p \wedge \bigwedge_{i \in Agt} S_i\, p\Big)$$

  which by ($Red_{K,p}$) is nothing but $EK_{Agt}p \rightarrow K_j EK_{Agt}p$, for arbitrary j. It follows that $EK_{Agt}p \rightarrow EK_{Agt}EK_{Agt}p$. From the latter we obtain $CK(EK_{Agt}p \rightarrow EK_{Agt}EK_{Agt}p)$ by necessitation. Now we are ready to apply the induction axiom for common knowledge, which tells us that

$$CK(EK_{Agt}p \rightarrow EK_{Agt}EK_{Agt}p) \rightarrow (EK_{Agt}p \rightarrow CKp),$$

  whence the result.
  The proof for (1.3) is similar.

- (1.4). From the positive introspection axiom (PI_S) it follows that $S_i\, p \rightarrow EK_{Agt}S_i\, p$, from which we obtain $CK(S_i\, p \rightarrow EK_{Agt}S_i\, p)$ by necessitation. The induction axiom for common knowledge then tells us that

$$CK(S_i\, p \rightarrow EK_{Agt}S_i\, p) \rightarrow (S_i\, p \rightarrow CK S_i\, p),$$

  hence the result.
  The proof for (1.5) is similar.

- (1.6). The right-to-left direction is a theorem of $\mathsf{S5}^n_C$. For the left-to-right direction we only state the proof for $m = 2$ and propositional variables; the generalisation is straightforward. Using axioms (1.2) and ($Red_{K,p}$) and the definition of EK_{Agt}, it suffices to prove that

$$CK(p \vee q) \rightarrow (p \wedge S_i\, p \wedge S_j\, p) \vee (q \wedge S_i\, q \wedge S_j\, q).$$

  We distribute $\wedge$ over $\vee$ in the right side and prove that $CK(p \vee q)$ implies each of the nine disjuncts.

  1. $CK(p \vee q) \rightarrow (p \vee q)$. This is a theorem of $\mathsf{S5}^n_C$.

2. $CK(p \vee q) \rightarrow (p \vee S_i\, q)$. First, $CK(p \vee q)$ implies $K_i(p \vee q)$ in $\mathsf{S5}^n_C$. Second, $K_i(p \vee q)$ implies $K_i p \vee K_i q$ by $(Red_{K,\vee})$. Third, $K_i p \vee K_i q$ implies $p \vee S_i\, q$ by $(Red_{K,p})$.
3. $CK(p \vee q) \rightarrow (p \vee S_j\, q)$. The proof is similar to case (2).
4. $CK(p \vee q) \rightarrow (S_i\, p \vee q)$. The proof is similar to case (2).
5. $CK(p \vee q) \rightarrow (S_i\, p \vee S_i\, q)$. First, $CK(p \vee q)$ implies $K_i(p \vee q)$ in $\mathsf{S5}^n_C$. Second, $K_i(p \vee q)$ implies $K_i p \vee K_i q$ by $(Red_{K,\vee})$. Third, $K_i p \vee K_i q$ implies $S_i\, p \vee S_i\, q$ by $(Red_{K,p})$.
6. $CK(p \vee q) \rightarrow (S_i\, p \vee S_j\, q)$. First, $CK(p \vee q)$ implies $K_i K_j(p \vee q)$ in $\mathsf{S5}^n_C$. Second, the latter is equivalent to $K_i(K_j p \vee K_j q)$ by $(Red_{K,\vee})$. Third, $K_i(K_j p \vee K_j q)$ implies $K_i(p \vee S_j\, q)$ by the truth axiom of $\mathsf{S5}^n_C$. Fourth, the latter is equivalent to $K_i p \vee K_i S_j\, q$ by $(Red_{K,\vee})$. Finally, the latter implies $S_i\, p \vee S_j\, q$ by $(Red_{K,p})$ and the truth axiom of $\mathsf{S5}^n_C$.
7. $CK(p \vee q) \rightarrow (S_j\, p \vee q)$. Similar to (2).
8. $CK(p \vee q) \rightarrow (S_j\, p \vee S_i\, q)$. Similar to (6).
9. $CK(p \vee q) \rightarrow (S_j\, p \vee S_j\, q)$. Similar to (5).

Putting together these cases we obtain the result.

Therefore all the formulas are valid. □

Proposition 1.3. *Our axiomatization of* ECL-PC *without strategic operators is complete.*

Proof. Completeness follows from the fact that our axiomatics allows us to derive a complete set of reduction axioms for the epistemic operator K_i and CK. We start by showing that these operators can be eliminated when they face a boolean formula.

- When K_i faces a literal then it can be eliminated: first, when that literal is a propositional literal (a variable or a negation thereof) then the reduction axioms $(Red_{K,p})$ and $(Red_{K,\neg p})$ apply; second, when the literal is a visibility atom then we have $K_j S_i\, p \leftrightarrow S_i\, p$ by the axiom of positive introspection (PI_S) and the truth axiom of $\mathsf{S5}^n_C$; third, when the literal is the negation of a visibility atom then we have $K_j \neg S_i\, p \leftrightarrow \neg S_i\, p$ by the axiom of negative introspection (NI_S) and the truth axiom of $\mathsf{S5}^n_C$.
- When CK faces a literal then it can be eliminated in a way similar to the above: we apply equivalences (1.2) and (1.3) of Proposition 1.2 for variables, and we apply implications (1.4) and (1.5) and the truth axiom of $\mathsf{S5}^n_C$ for visibility atoms.

- When K_i faces a clause then it can be replaced by $\top$ if the clause is tautological (by standard principles of normal modal logics); otherwise it can be distributed by the implication ($Red_{K,\vee}$) and its converse (which is a principle of $\mathsf{S5}^n_C$).
- When CK faces a clause then it can be replaced by $\top$ if the clause is tautological (by standard principles of normal modal logics); otherwise it can be distributed by the equivalence (1.6) of Proposition 1.2.
- When K_i or CK face a conjunction or a double negation then they can be simplified by standard principles of normal modal logics.

Now that we can eliminate epistemic operators facing boolean formulas, we can start by innermost such operators and—applying the rule of replacement of equivalents of $\mathsf{S5}^n_C$—eliminate all epistemic operators. The result is a boolean combination of atoms. Such a formula is satisfiable in ECL-PC if and only if it is satisfiable in propositional logic. □

Complexity of model checking of ECL-PC

The model checking of ECL-PC is defined as:

- **Input:** a couple $\langle V, \varphi \rangle$ where φ is a ECL-PC formula and V is finite a valuation;
- **Output:** yes if $V \models \varphi$, no otherwise.

Its complexity was proven to be PSPACE-complete [van der Hoek et al., 2011].

1.1.2 Relation with Kripke models

We have seen that logics of visibility-based knowledge still rely on possible worlds and indistinguishability relations that are constructed from what agents can see. In this section, we examine the link between visibility-based models and standard Kripke models in more detail.

The language of standard epistemic logic (with common knowledge) is the language of ECL-PC but without visibility atoms. It can therefore be defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi \mid CK\varphi$$

where p ranges over $Prop$ and i over Agt.

In the semantics, a Kripke model for epistemic logic is usually defined as a tuple $M = \langle W, \sim_1, \ldots, \sim_n, v \rangle$, where W is a set of worlds, $\sim_1, \ldots, \sim_n$ are equivalence relations, one per agent, and $v : Prop \to 2^W$

is a valuation function mapping every propositional variable to a set of worlds where it is true. With $M = \langle W, \sim_1, \ldots, \sim_n, v\rangle$ a Kripke model and $w \in W$ a world:

$$\begin{array}{ll} M, w \models p & \text{iff } w \in v(p) \\ M, w \models \neg\varphi & \text{iff not } (M, w \models \varphi) \\ M, w \models \varphi \wedge \varphi' & \text{iff } M, w \models \varphi \text{ and } M, w \models \varphi' \\ M, w \models K_i\varphi & \text{iff } M, w' \models \varphi \text{ for every } w' \text{ such that } w \sim_i w' \\ M, w \models CK\varphi & \text{iff } M, w' \models \varphi \text{ for every } w' \text{ such that } w \sim_{Agt} w' \end{array}$$

where $\sim_{Agt} = \left(\bigcup_{i\in Agt} \sim_i\right)^*$ as in ECL-PC.

Again, we write $w(p) = w'(p)$ if and only if p has the same value in w and in w'.

From visibilities to Kripke models

In logics of visibility, every subset of the set of propositional variables is a possible world. This means that a visibility model is always made up of $2^{|ATM|}$ worlds (with $|ATM|$ the cardinality of ATM).

Formally, given a valuation $V_0 \in 2^{ATM}$, we are going to build a pointed Kripke model (M_{V_0}, w_{V_0}) that is bisimilar to (M_{vis}, V_0) w.r.t. the standard language of epistemic logic of the beginning of this section. We define $M_{V_0} = \langle W, \sim_1, \ldots, \sim_n, v\rangle$ as follows:

$$\begin{aligned} W &= 2^{Prop} \\ \sim_i &= \{(w, w') : w, w' \in W \text{ and} \\ &\qquad \text{for every } p \in Prop, S_i\, p \in V_0 \text{ implies } w(p) = w'(p)\} \\ v(p) &= \{w : w \in W \text{ and } p \in w\} \end{aligned}$$

and $w_{V_0} = V_0 \cap Prop$. Therefore only a part of our unique visibility model M_{vis} remains in the Kripke model: all those equivalence classes where the agents' knowledge complies with the visibility information contained in V_0.

Proposition 1.4. *The visibility model (M_{vis}, V_0) and the pointed Kripke model (M_{V_0}, w_{V_0}) are bisimilar w.r.t. $Prop$.*

Proof. Consider the relation $Z \subseteq 2^{ATM} \times W$ such that

$$\begin{aligned} Z = \{(V, w) : &\text{ for every } p \in Prop,\ V(p) = w(p) \text{ and} \\ &\text{for every } p \in Prop,\ i \in Agt, V(S_i\, p) = V_0(S_i\, p)\}. \end{aligned}$$

We check the three conditions for Z to be a bisimulation.

1. **Atomicity.** $(V, w) \in Z$ implies $V(p) = w(p)$ holds for all $p \in Prop$ by definition of Z.

2. **Forth.** Suppose $(V, w) \in Z$ and $V \sim_i V'$ and let $w' = V' \cap \mathit{Prop}$. Then:

   - We have $(V', w') \in Z$ because, first, $V'(p) = w'(p)$ for all p by definition of w'; and second, because for all p and j: $V'(S_j\, p) = V(S_j\, p)$ by definition of $\sim_i$ and $V(S_j\, p) = V_0(S_j\, p)$ because $(V, w) \in Z$.
   - We have $w \sim_i w'$ because we have: first, $S_i\, p \in V$ implies $V(p) = V'(p)$ by definition of $\sim_i$; second, as $(V, w) \in Z$, we have $V(S_i\, p) = V_0(S_i\, p)$, so $S_i\, p \in V_0$ implies $V(p) = V'(p)$; third, again as $(V, w) \in Z$ we have $V(p) = w(p)$, so $S_i\, p \in V_0$ implies $w(p) = V'(p)$. Hence $S_i\, p \in V_0$ implies $w(p) = w'(p)$ by definition of w'.

3. **Back.** Suppose $(V, w) \in Z$ and $w \sim_i w'$. Let $V' = w \cup (V_0 \setminus \mathit{Prop})$. The proof that "$V \sim_i V'$ and $(V', w') \in Z$" follows the same steps as the proof of the "forth" condition, in reverse order.

Moreover, we have $(V_0, w_{V_0}) \in Z$. Therefore (M_{vis}, V_0) and (M_{V_0}, w_{V_0}) are bisimilar w.r.t. the standard language of epistemic logic. □

The pointed Kripke model and the valuation being bisimilar, they satisfy the same formulas: we have $V_0 \models \varphi$ if and only if $M_{V_0}, w_{V_0} \models \varphi$.

Example 1.4. Consider Prop and Agt of examples 1.1 and 1.2. Consider the valuation $V_0 = \{p, S_1\, p, S_2\, q\}$. Then $M_{V_0} = \langle W, \sim_1, \sim_2, v\rangle$ with $W = \{\emptyset, \{p\}, \{q\}, \{p, q\}\}$, where the indistinguishability relations are as depicted in Figure 1.3 and where $w_{V_0} = \{p\}$. Observe that this corresponds to the subset of Example 1.1.

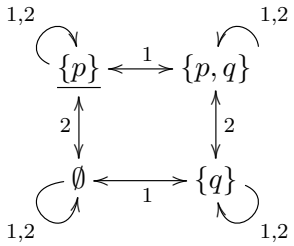


Figure 1.3: Kripke model M_{V_0} for $V_0 = \{p, S_1\, p, S_2\, q\}$.

From Kripke models to visibilities

We have seen that every visibility model can be transformed into a to Kripke model. The other way round, it is in general not possible to transform a Kripke model to a visibility model. There are two reasons for that.

1. First, visibilities stay the same across the indistinguishability relations. This implies that who sees what is common knowledge:

$$S_i\, p \to CK S_i\, p$$
$$\neg S_i\, p \to CK \neg S_i\, p$$

which, along with validity (1.1) of Proposition 1.2, gives:

$$(K_i p \vee K_i \neg p) \to CK(K_i p \vee K_i \neg p)$$
$$(\neg K_i p \wedge \neg K_i \neg p) \to CK(\neg K_i p \wedge \neg K_i \neg p)$$

We have also seen there that this leads to the validity of $CKp \leftrightarrow EK_{Agt}p$, which is invalid in standard epistemic logic.

This is annoying because theory of mind is "flattened": it becomes impossible to reason about higher-order knowledge. This is however fundamental in the gossip problem (generalized to higher-order knowledge) or in any other reasoning task requiring theory of mind.

2. Second, as we have seen in the previous section, a visibility model always contains $2^{|ATM|}$ worlds: every possible subset of ATM is contemplated. This implies the distribution of knowledge over disjunction of literals:

$$K_i\Big(\bigvee_{\alpha\in A^+}\alpha \vee \bigvee_{\alpha\in A^-}\neg\alpha\Big) \leftrightarrow \Big(\bigvee_{\alpha\in A^+}K_i\alpha\Big) \vee \Big(\bigvee_{\alpha\in A^-}K_i\neg\alpha\Big)$$

when $A^+ \cap A^- = \emptyset$. This problem is 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 forbids to model things such as the muddy children puzzle, where each child knows that at least one of the children is muddy without knowing which.

In this thesis, we will propose solutions to these two issues.

1.1.3 Other observation-based logics

Besides ECL-PC, other logics of visibility were also studied in recent years.

The Logic of Revelation and Concealment LRC [van der Hoek et al., 2012] allows us to express, as programs, that a variable is revealed to an agent or concealed from her, modifying the visibilities. However, like in ECL-PC, these visibilities also stay the same across the indistinguishability relations, and therefore who sees what is still common knowledge

among all agents. In the following, we will see how to include dynamic modalities that will change visibilities of agents.

In Flatland Logic [Balbiani et al., 2013a], visibility is grounded on geometry in order to give semantics to epistemic operators: an agent can (or cannot) observe the positions of other agents and can reason about what they observe. Visibility can be higher-order and is also fully determined by geometric constraints such as the position of agents and their angle of view. A notable difference with other visibility-based epistemic logics is that agents see other agents instead of propositional variables. A successor of Flatland Logic is Big Brother Logic [Gasquet et al., 2014] where agents have a fixed position and angle of view, but can rotate freely. We will see how to fully relax these types of constraints.

1.2 Dynamic logics

Inspired from [Engeler, 1967; Hoare, 1969; Yanov, 1959], Algorithmic Logic AL [Salwicki, 1970], and then Dynamic Logic DL [Pratt, 1976] were designed as "logics of programs": to each computer program is associated a modality expressing the fact that when the program terminates, the system is in a state satisfying a statement. Propositional Dynamic Logic PDL [Fischer and Ladner, 1979] is the propositional variant of DL.

In dynamic logics, complex programs are built from abstract atomic programs by sequential composition, non-deterministic composition, iteration and test. Other operators can be considered, like converse or intersection, and iteration is sometimes dropped in so-called "star-free" fragments. Formulas are interpreted over labelled transition systems, where each transition between states is labelled with the name of an atomic program and indicates a possible execution of this atomic program.

1.2.1 Logic of propositional assignments

A recent contribution, Dynamic Logic of Propositional Assignments DL-PA [Herzig et al., 2011; Balbiani et al., 2013b], instantiates PDL atomic programs: they are now assignments of propositional variables to true or to false. We recall the syntax and semantics of this logic as we heavily rely on it in the following.

Language of DL-PA

The language of formulas and programs of DL-PA is defined by the following grammar:

$$\pi ::= +p \mid -p \mid (\pi;\pi) \mid (\pi \sqcup \pi) \mid \pi^* \mid \varphi?$$
$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid [\pi]\varphi$$

where p ranges over $Prop$.

The formula $[\pi]\varphi$ reads "after every execution of π, φ is true" as in standard dynamic logics. As for programs, assignments $+p$ and $-p$ make p true or false; $\pi;\pi'$ is sequential composition: it executes π followed by π'; $\pi \sqcup \pi'$ is non-deterministic choice: it executes either π or π', choosing non-deterministically; π^* is unbounded iteration: it executes π a finite, but non-deterministically determined number of times; $\varphi?$ is test: it checks if φ is currently true and continue if this is the case, or fails and ends the execution otherwise.

Semantics of DL-PA

Formulas are interpreted on valuations $V \in 2^{ATM}$, like the ones we have seen in ECL-PC in Section 1.1.1. Again, the whole model is unique and we keep it implicit in the truth conditions. They are as follows:

$V \models p$	iff $p \in V$
$V \models \neg\varphi$	iff not $(V \models \varphi)$
$V \models \varphi \wedge \varphi'$	iff $V \models \varphi$ and $V \models \varphi'$
$V \models [\pi]\varphi$	iff $V' \models \varphi$ for every V' such that $VR_\pi V'$

where R_π is a binary relation on valuations defined by:

$VR_{+p}V'$	iff $V' = V \cup \{p\}$
$VR_{-p}V'$	iff $V' = V \setminus \{p\}$
$VR_{\pi;\pi'}V'$	iff $V(R_\pi \circ R_{\pi'})V'$
$VR_{\pi\sqcup\pi'}V'$	iff $V(R_\pi \cup R_{\pi'})V'$
$VR_{\pi^*}V'$	iff $V(\bigcup_{k\in\mathbb{N}_0} (R_\pi)^k)V'$
$VR_{\varphi?}V'$	iff $V = V'$ and $V \models \varphi$

The semantics are standard except that atomic programs, being concrete, have always the same effect of updating the current valuation by adding or removing a given variable.

Complexity of model checking of DL-PA

The model checking of DL-PA is defined as:

- **Input:** a couple $\langle V, \varphi \rangle$ where φ is a DL-PA formula and V is a finite valuation;
- **Output:** yes if $V \models \varphi$, no otherwise.

Its complexity was proven to be PSPACE-complete [Balbiani et al., 2014].

1.2.2 On the relation with visibility logics

As we have glimpsed, observation-based logics are often related with dynamics, like LRC [van der Hoek et al., 2012], where programs modify visibilities, and hence agents' knowledge.

Note that the new semantics that we presented for ECL-PC in Section 1.1.1 fit well with the ones of DL-PA. While relaxing the constraint on visibilities of ECL-PC (which stay constant across indistinguishability relations), one of the main contributions of this thesis will be to study a dynamic epistemic logic based on observation, where the truth values of visibilities, but also facts of the world, can evolve using assignments. We will also show that the complexity of the model checking in this new setting does not increase compared to ECL-PC and DL-PA.

1.3 Control, strategies and boolean games

Logics of control, such as Coalition Logic of Propositional Control CL-PC [van der Hoek and Wooldridge, 2005], have been proposed in recent years to model capabilities of agents in a rather simple and elegant way. They allow us to reason about the strategic abilities of agents, i.e., the states an agent is able to reach by modifying the variables she controls. This can be extended to coalitions (sets of agents) by considering variables at least one of them controls.

1.3.1 Control

In CL-PC [van der Hoek and Wooldridge, 2005], each agent controls a set of variables, much like visibilities of ECL-PC (which is an extension of CL-PC as we have mentioned). Control is assumed to be exclusive and exhaustive: a variable is controlled by exactly one agent. With control, it is possible to define the *ceteris paribus* operator $\Diamond_J$, with J a coalition, such that $\Diamond_J \varphi$ reads "agents in J can change values of variables they control while the other agents do not act so that φ is true." It is true at V if and only if there exists a valuation V' that satisfies φ, such that

only variables controlled by at least one agent in J are modified between V and V'.

The *ceteris paribus* operator $\Diamond_J$ allows us to reconstruct the "coalition logic-like" strategic operator $\langle\langle J \rangle\rangle$ [Pauly, 2001, 2002; Alur et al., 2002], such that $\langle\langle J \rangle\rangle \varphi$ reads "agents in J are able to achieve φ by modifying variables they control, whatever the other agents do": $\langle\langle J \rangle\rangle \varphi = \Diamond_J \Box_{Agt \setminus J} \varphi$, with $\Box_J$ the dual operator of $\Diamond_J$ reading that "agents in J cannot avoid φ by changing values of variables they control."

The connection between CL-PC and DL-PA was studied in [Herzig et al., 2011], where it was shown that the former can be polynomially embedded in the star-free fragment of the latter.

Another work on control is [Grossi et al., 2015] in which the authors present a *ceteris paribus* logic, called CP, whose formulas are interpreted on the equivalence classes induced by finite sets of propositional atoms. This logic has modal operators of the form $[X]$ where X is a finite set of propositional variables. The formula $[X]\varphi$ is read "φ is the case in all states which are X-equivalent to the current state," where two states are X-equivalent if they assign the same truth values to the variables in X. The authors show that CL-PC as well as the star-free fragment of DL-PA can be embedded in CP.

1.3.2 Boolean games and epistemic boolean games

Boolean games [Harrenstein et al., 2001; Bonzon et al., 2006; Dunne et al., 2008] are games in which each player wants to achieve a certain goal that is represented by a propositional formula. They correspond to the specific subclass of normal-form games in which agents have binary preferences (i.e., payoffs are either 0 or 1) and are widely accepted as a useful and natural abstraction for reasoning about social interaction in multi-agent systems.

Boolean games

A boolean game is a tuple $(Agt, Prop^{\mathsf{f}}, (\Psi_i)_{i \in Agt}, (\gamma_i)_{i \in Agt})$ where $Prop^{\mathsf{f}}$ is a finite subset of $Prop$, each $\Psi_i \subseteq Prop^{\mathsf{f}}$ is the set of variables agent i controls, and γ_i is a boolean formula such that $ATM(\gamma_i) \subseteq Prop^{\mathsf{f}}$. The latter expresses i's personal goal, i.e., the state of affairs i wants to achieve. Exclusive and exhaustive control is assumed: the sets Ψ_i partition the set of variables $Prop^{\mathsf{f}}$.

A strategy for an agent i, noted s_i, is an interpretation of variables controlled by i, that is to say, $s_i \subseteq \Psi_i$. Therefore the set of all possible strategies of agent i is 2^{Ψ_i}. Given a strategy s_i for each member of a coalition J, the induced strategy for J is $s_J = \bigcup_{i \in J} s_i$. A strategy s_{Agt} for Agt is called a strategy profile. It can be seen as an interpretation of

the set of variables $Prop^{\mathsf{f}}$. Agent i's utility function maps every strategy profile to i's reward; this reward depends only on the truth value of i's goal in s_{Agt}:

$$U_i(s_{Agt}) = \begin{cases} 1 & \text{if } s_{Agt} \models \gamma_i \\ 0 & \text{otherwise.} \end{cases}$$

Let $\langle s_i, s_{Agt\setminus\{i\}} \rangle$ be the strategy profile composed of the strategy s_i of i and of the strategy $s_{Agt\setminus\{i\}}$ of i's opponents. We say that s_i is a best response to $s_{Agt\setminus\{i\}}$ if and only if, for every strategy $s'_i \in 2^{\Psi_i}$,

$$U_i(\langle s'_i, s_{Agt\setminus\{i\}} \rangle) \leq U_i(\langle s_i, s_{Agt\setminus\{i\}} \rangle)$$

holds, i.e., every other strategy of i against the same strategy of opponents would not increase i's reward.

A strategy profile s_{Agt} is a Nash equilibrium if and only if s_i is a best response to $s_{Agt\setminus\{i\}}$ for every $i \in Agt$. This means that every agent cannot get a bigger reward by choosing another strategy if the others do not change theirs.

Epistemic boolean games

In [Ågotnes et al., 2013], epistemic boolean games are studied. In these games, agents' goals are no longer restricted to boolean formulas, but may contain epistemic operators. Semantics, like in ECL-PC, are based on valuations and visibility sets of variables associated to agents, with the drawbacks that we saw in Section 1.1. In particular, agents can affect the truth value of propositional variables and, indirectly, the knowledge of those agents' who can see these variables; but agents cannot modify the visibility conditions of propositional variables.

It is shown in [Ågotnes et al., 2013] that for their class of epistemic boolean games, membership and existence of Nash equilibria (i.e., deciding if the current strategy is a Nash equilibrium and deciding if there exists one) are PSPACE-complete. Using atoms of control in addition to atoms of visibility, we will show in Chapter 5 how to encode a more general class of epistemic boolean games, where agents can affect the truth value of propositional variables but also the visibilities of conditions of propositional variables, including higher-order visibility. We will also show that the complexity of deciding membership and existence of Nash equilibria is not affected by this generalisation.

1.4 Automated planning

Automated planning [Ghallab et al., 2004] is a field of artificial intelligence that aims to, given some actions, generate a plan that can be

executed by an agent and that leads her to her goal.

1.4.1 Classical planning

In its simplest form, a planning task is composed of the initial state, a set of actions, and the goal; a plan is a sequence of actions. It is assumed that the initial state is unique and known, that actions are instantaneous, deterministic and fully observable.

The three components can be described with the help of propositional logic: the initial state being a set of propositional variables, actions adding or removing these variables, possibly with preconditions to be executed, and the goal being a boolean formula. The planning task is solvable if there exists a sequence of actions which, executed from the initial state, leads to a state where the goal is satisfied.

Planning is furthermore interested in the design of algorithms that solve planning tasks, i.e., that find a solution plan. Several languages for representing such tasks were developed; the most commonly known are STRIPS [Fikes and Nilsson, 1971] (Stanford Research Institute Problem Solver) and PDDL [McDermott et al., 1998] (Planning Domain Definition Language), the latter including the former. New planners are regularly proposed and evaluated at competitions such as the International Planning Competition.[1]

1.4.2 Epistemic planning

Epistemic planning focuses on planning under uncertainty: agents may be uncertain about the state of the world, about the actions performed by other agents or about the effects of actions. Goals may also concern the knowledge of agents in addition to facts of the world.

Dynamic Epistemic Logic DEL [van Ditmarsch et al., 2007] provides a formal framework for the representation of knowledge and update of knowledge. Its dynamics rely on event models which modify Kripke models with an operation called product update. These models allow us to represent actions together with the agents' perception of their occurrence.

Several recent approaches to multi-agent epistemic planning are based on DEL, starting with [Bolander and Andersen, 2011; Löwe et al., 2011]. While DEL provides a very expressive framework, it was unfortunately proven to be undecidable even for rather simple fragments of the language. For example, if actions make factual changes to the world, then the problem is undecidable whenever epistemic operators are allowed in preconditions of actions; if actions are purely epistemic, then it is undecidable whenever two agents are involved or the epistemic depth

[1] `http://www.icaps-conference.org/index.php/Main/Competitions`

exceeds 2 [Aucher and Bolander, 2013; Charrier et al., 2016a]. Some decidable fragments were studied, most of which focused on public events [Löwe et al., 2011; Yu et al., 2015]. However, we will see that some tasks such as the gossip problem requires private communication.

There exist other approaches on planning with uncertainty that do not use DEL. The framework presented in [Kominis and Geffner, 2015] allows us to reason about knowledge on literals in a multi-agent setting. A similar approach with beliefs can be found in [Muise et al., 2015]. While restricted to a single agent, the framework of [Petrick and Bacchus, 2004] deals with "knowing that" but also "*knowing whether*" formulas (i.e., knowing p or knowing $\neg p$).

We will show in Chapter 4 how to encode more general multi-agent epistemic planning problems within the framework of our logic of visibility, keeping the complexity low in comparison with DEL-based approaches.

1.5 Aim and contributions of the thesis

1.5.1 Aim of the thesis

The aim of the present thesis is to demonstrate that visibility-based epistemic logics provide a suitable tool for important applications in the field of artificial intelligence. This requires to solve the two issues we identified in Section 1.1.2: common knowledge of visibilities and the epistemic operator distributing over disjunctions. We illustrate the result with well-known toy examples of epistemic logic as well as with more general applications such as epistemic planning and boolean games.

A higher-level motivation behind logics of visibility and the generalisations we propose is also notable: we consider that, given a particular problem, the design and use of an epistemic model is easier with them. As an example, designing the Kripke model which describes the initial situation the muddy children problem may be a difficult task, especially for someone not fully familiar with modal logic. As we will see in Chapter 6, writing down the same situation in terms of visibility is obvious. One could argue hat models do not have to be constructed from scratch: there exist techniques to infer a model from a formula, such as the tableaux method [Fitting, 1983]. The resulting models may still be complex to handle and to work with for non-logicians. Moreover, adding actions to standard epistemic logic leads to DEL. While being highly expressive, the design of a DEL event model (representing an action) is even more complex than of an epistemic Kripke model. This hardness has led to the research of fragments such as Public Announcement Logic PAL. We have also seen that in the field of epistemic planning, where the

standard approach relies on DEL, alternative methods are proposed to avoid its arduousness. We will see that the inclusion of actions in our logics of visibility is straightforward and that they are easy to handle. We hope that this encourage the use of formal methods for concrete applications involving reasoning about knowledge such as in cyber security, autonomous vehicles or video games.

1.5.2 Outline of the thesis

In Chapter 2, we add higher-order knowledge to observability-based logics and allow for situations where, e.g., agent i sees variable p but agent j does not see whether i sees p. This logic, which we call DEL-PAO, solves the first issue: observations are not common knowledge any more. This chapter is based on [Herzig et al., 2015].

DEL-PAO allows us to model in a natural way scenarios such as a generalisation of the gossip problem. In the original gossip problem, friends exchange secrets via telephone calls until everyone knows all secrets. In the novel generalisation studied in Chapter 3, we want that everyone knows all secrets, but also that everyone knows this, and so on until a fixed depth. This chapter is based mostly on [Herzig and Maffre, 2016], with the exception of the proof of optimality that can be found in [Cooper et al., 2016b].

In Chapter 4, we apply DEL-PAO to epistemic planning. We show how our framework allows us to express many problems of epistemic planning (such as the gossip problem) while avoiding the undecidability of current methods based on DEL. This chapter is based on [Cooper et al., 2016a].

The framework of DEL-PAO can also be easily extended with atoms of control, modelling strategic abilities and epistemic boolean games as introduced in Section 1.3. This extension is presented in Chapter 5. This chapter is based on [Herzig et al., 2016].

We show in Chapter 6 how to take into account public announcements, as popularized in dynamic epistemic logics and more generally publicly executed programs. Knowledge may now originate from two sources: observations and communication. This solves the second issue: knowledge operators do not distribute over disjunctions any more. This chapter is based on [Charrier et al., 2016a].

Finally, Chapter 7 settles the complexity of the model checking problem of all presented logics. It introduces a framework into which every given logic can be translated and presents an algorithm solving the model checking problem for this framework that runs in PSPACE. This chapter is based on a section of [Charrier et al., 2016a].

Every paper this thesis is based on is, for most parts, the result of collaborative work between all its authors. Notable parts in which the

author of the thesis has less participated in are: the optimality proof of Chapter 3; the translation into PDDL of Chapter 4; the application to epistemic boolean games of Chapter 5; the model checking algorithms of Chapter 7.

Moreover, several changes have been made by the author of the thesis to some parts of the papers, including corrections of errors and new results, the most important being the inclusion of the Kleene star into the language of programs. They are signalled by footnotes indicating what was altered. Explanations and proofs have also been globally expanded. Finally, drawings have been added, with the sole aim to decorate examples given throughout the thesis. These illustrations were made by the author and are free to use.

1.5.3 Contributions not detailed in this thesis

Formal properties of the generalized gossip problem

Other interesting properties of the generalized gossip problem, introduced in Chapter 3, and not included in this thesis can be found in [Cooper et al., 2016b] (with a long version in [Cooper et al., 2016c]). In this paper, we more generally consider the case where graphs are not complete, i.e., where some agents cannot call each other. We study the necessary number of calls and the complexity of the generalised gossip problem and its variant with ignorance goals, as well as:

- the one-way gossip, where calls are replaced by e-mails;
- the parallel gossip, where we are interested in the number of steps required rather than in the number of calls;
- the gossip with variable secrets, where agents can switch the value of their secrets.

An important result of this paper, namely, the proof of optimality of the algorithm for the standard generalized gossip problem, is nevertheless included in Chapter 3.

Alternating-time temporal logic with commitments

In [Herzig et al., 2013], the authors introduce Alternating-time Temporal Logic with Explicit Actions ATLEA, which extends the language of Alternating-time Temporal Logic ATL with commitments on actions. These commitments specify the next action the agent will perform. This is a first step towards a full specification of strategies that is not included in ATL: operators $\langle\langle J \rangle\rangle$ only indicates whether the coalition of agents J has a strategy to ensure some state or not, without giving any hint on the actions composing this strategy.

In [Herzig et al., 2014], we followed this line of work and introduced Alternating-time Temporal Logic with Explicit Programs ATLEP. In ATLEP, atomic commitments are ATLEA commitments, and complex commitments are constructed using dynamic logic operators. Agents may not only commit to the next action, but also on the ones following the next action. They can also commit on different actions depending on the current state, by using tests, and even commit infinitely thanks to a special operator of infinite iteration.

This work was set aside because we were only able to specify the semantics of ATLEP in a very complicated way, making the study of properties such as axiomatization or complexity difficult, and decreasing the chances of building interesting work over it. One lead might be to not consider PDL but instead the less complex language of DL-PA for actions, but this was not investigated. We do not include this work in this thesis because of its lack of connection with the more recent contributions on logics of visibility.

2 A simple dynamic epistemic logic based on observation

A drawback of current logics of visibility is that what each agent can see is common knowledge to all agents. This is a strong hypothesis that we are going to relax in the present chapter.

In existing frameworks, visibility information is in terms of propositional variables associated to agents. We here consider propositional variables associated to sequences *of agents, giving the central logic of this thesis:* DEL-PAO.

Syntactically, we represent visibility information by means of atomic formulas that we call *visibility atoms*. They take the form $S_{i_1} S_{i_2} \dots S_{i_n} p$, where p is a propositional variable and $i_1, i_2, \dots, i_n$ are agents. When $n{=}0$ then we have nothing but a propositional variable. For $n{=}1$, the atom $S_{i_1} p$ reads "agent i_1 sees the value of the variable p," and for $n{=}2$, the second-order observation $S_{i_1} S_{i_2} p$ reads "agent i_1 sees whether i_2 sees the value of p," and so on.

Our models are valuations as presented in the introduction, i.e., sets of atoms. In order to guarantee positive and negative introspection we have to ensure that agents are always aware of what they see: for every agent i and propositional variable p, we require $S_i S_i p$ to be in every valuation. We say that a valuation V is *introspective* when it contains every visibility atom having two consecutive S_i, such as $S_j S_i S_i S_k p$.

Visibility information allows us to interpret epistemic operators as in other visibility-based logics; the truth condition for $K_i\varphi$ is based on a relation between valuations that can be defined from our visibility atoms: $V \sim_i V'$ if every atom that i sees in V has the same truth value in V and in V'. While the relations $\sim_i$ are reflexive on the set of all valuations, they are symmetric and transitive—and therefore equivalence relations—on the set of introspective valuations only. The positive and negative introspection axioms $K_i\varphi \to K_i K_i \varphi$ and $\neg K_i\varphi \to K_i \neg K_i \varphi$ are valid in the set of introspective valuations.

A further novelty of our approach as compared to existing visibility-based epistemic logics is that we also account for common knowledge: our language includes a special atomic formula for joint attention of the form $JS\,p$ that reads "all agents jointly see the value of p." Metaphorically, joint attention about a propositional variable p can be understood as eye contact between the agents when observing p. Just as individual visibility, we generalize our account to higher-order joint visibility, adding a constraint on valuations that guarantees introspection of common knowledge. We moreover require that joint visibility implies individual visibility by imposing that $S_i\,p \in V$ whenever $JS\,p \in V$. We can then interpret a modal operator of common knowledge CK in the same way as the modal operator of individual knowledge.

Just as several existing proposals, we take inspiration from dynamic epistemic logics DEL [van Ditmarsch et al., 2007] and add dynamics to our observation-based epistemic logic. This will allow us to model systems where knowledge, but also facts of the world, change over time. Specifically, we adapt van der Hoek et al.'s logic LRC which has two update operations modifying visibility: reveal and conceal the value of a variable to some agent. These two primitives can however not be taken over as they stand because the naive update of a valuation may no longer be introspective. We exclude this by an appropriate definition of update. We relate our assignment programs to DL-PA [Herzig et al., 2011; Balbiani et al., 2013b]. We show how visibility updates can capture private announcements of literals, conjunctions and knowledge of literals.

We call our logic DEL-PAO: Dynamic Epistemic Logic of Propositional Assignment and Observation.

Contents

Résumé du chapitre

Un inconvénient des logiques actuelles de visibilité est que ce que chaque agent voit est connaissance commune pour tous les agents. Ceci est une hypothèse forte que nous allons relâcher dans ce chapitre. Dans les logiques existantes, les informations de visibilité sont sous forme d'ensembles de variables propositionnelles associés aux agents. Nous considérons ici des variables propositionnelles associées à des séquences d'agents, ce qui donne la logique centrale de la thèse : DEL-PAO.

Syntaxiquement, nous représentons les informations de visibilité au moyen de formules atomiques que nous appelons *atomes de visibilité*. Ils sont de la forme $S_{i_1} S_{i_2} \dots S_{i_n} p$, où p est une variable propositionnelle et $i_1, i_2, \dots, i_n$ sont des agents. Pour $n{=}0$ nous obtenons simplement variable propositionnelle. Pour $n{=}1$, l'atome $S_{i_1} p$ se lit "l'agent i_1 voit la valeur de la variable p", et pour $n{=}2$, l'observation du second ordre $S_{i_1} S_{i_2} p$ se lit "l'agent i_1 voit si i_2 voit la valeur de p", etc.

Nos modèles sont les valuations comme celles présentées dans l'introduction, c'est-à-dire, des ensembles d'atomes. Afin de garantir l'introspection positive et négative, nous devons nous assurer que les agents sont toujours conscients de ce qu'ils voient : pour chaque agent i et variable propositionnelle p, chaque valuation doit contenir $S_i S_i p$. Une valuation V est dite introspective si elle contient tout atome de visibilité ayant deux S_i consécutifs, comme par exemple $S_j S_i S_i S_k p$.

Ces informations de visibilité nous permettent d'interpréter les opérateurs épistémiques de la même façon que dans les autres logiques basées sur la visibilité ; la condition de vérité pour $K_i\varphi$ est basée sur une relation entre les valuations qui peut être définie à partir de nos atomes de visibilité : $V \sim_i V'$ si chaque atome que i voit dans V a la même valeur de vérité dans V et dans V'. Alors que les relations $\sim_i$ sont réflexives sur l'ensemble des valuations, elles sont symétriques et transitives, et donc des relations d'équivalence, sur l'ensemble des valuations introspectives seulement. Les axiomes d'introspection positive et négative $K_i\varphi \to K_i K_i \varphi$ et $\neg K_i\varphi \to K_i \neg K_i \varphi$ sont valables dans l'ensemble des évaluations introspectives.

Une autre nouveauté de notre approche par rapport aux logiques épistémiques basées sur la visibilité existantes est que nous considérons la connaissance commune : notre langage inclut une formule atomique spéciale pour l'attention jointe, de la forme $JS\,p$, lue "tous les agents voient conjointement la valeur de p". Métaphoriquement, l'attention jointe sur une variable propositionnelle p peut être comprise comme un contact visuel entre les agents lors de l'observation p. Tout comme la visibilité individuelle, nous prenons en compte l'ordre supérieur en ajoutant une contrainte sur les valuations qui garantit l'introspection de la connaissance commune. De plus, nous exigeons que la visibilité commune implique la visibilité individuelle en imposant que $S_i\,p \in V$ lorsque $JS\,p \in V$. On peut alors interpréter un opérateur modal CK de connaissance commune de la même manière que l'opérateur modal de connaissance individuelle.

Tout comme plusieurs propositions existantes, nous nous inspirons des logiques épistémiques dynamiques DEL [van Ditmarsch et al., 2007] et ajoutons des opérateurs dynamiques à notre logique épistémique basée sur l'observation. Cela nous permettra de modéliser des systèmes où la connaissance, mais aussi les propositions, evoluent au fil du temps. Plus précisément, nous adaptons la logique LRC de van der Hoek et al., qui propose deux opérations de mise à jour modifiant la visibilité : révéler et cacher la valeur d'une variable à un agent. Ces deux primitives ne peuvent toutefois pas être utilisées en l'état, car la mise à jour naïve d'une valuation peut ne plus être introspective. Nous excluons ce cas par une définition appropriée de l'opération de mise à jour. Nos programmes d'affectation sont similaires à ceux de DL-PA [Herzig et al., 2011; Balbiani et al., 2013b]. Nous montrons comment les mises à jour de visibilité peuvent capturer des annonces privées de littéraux, de conjonctions et de connaissance de littéraux.

Nous appelons notre logique DEL-PAO : une logique épistémique dynamique des affectations propositionnelles et de l'observation.

2.1 Language of DEL-PAO

Remember that $Prop$ is a countable non-empty set of propositional variables and Agt is a finite non-empty set of agents.

In our language, an atomic formula is a sequence of visibility operators, possibly empty, followed by a propositional variable. The formal definition is as follows.

The set of *observability operators* is

$$OBS = \{S_i : i \in Agt\} \cup \{JS\},$$

where S_i stands for individual visibility of agent i and JS stands for joint visibility of all agents. The set of all sequences of visibility operators is

noted OBS^* and the set of all non-empty sequences is noted OBS^+. We use σ, σ', etc. for elements of OBS^*. The *depth* of a sequence of operators σ, noted $depth(\sigma)$, is the number of operators composing it. Moreover, we note $\sigma[k{:}]$, for $0 \leq k \leq depth(\sigma)$, the sub-sequence starting at the k-th element of σ, such that the first element's index is 0. For example, $JS\, S_1\, S_2[1{:}] = S_1\, S_2$ and $JS\, S_1\, S_2[2{:}] = S_2$. Therefore $\sigma[0{:}] = \sigma$ and we suppose $\sigma[depth(\sigma){:}]$ is the empty sequence.

The set of atomic formulas is

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

Elements of that set are called *visibility atoms*, or atoms for short. Here are some examples:

- $S_1\, p$ reads "1 sees the value of p." It means that 1 knows whether p is true or false.
- $JS\, S_2\, q$ reads "all agents jointly see whether agent 2 sees the value of q." In other words, there is *joint attention* in the group of all agents concerning 2's observation of q. This does not imply that 2 sees the value of q.
- $S_1\, S_2\, S_3\, p$ reads "1 sees whether 2 sees whether 3 sees p."
- ... and so on.

We use α, α', β, β', etc. for elements of ATM.

The language of DEL-PAO is then defined by the following grammar:

$$\pi ::= +\alpha \mid -\alpha \mid (\pi;\pi) \mid (\pi \sqcup \pi) \mid \pi^* \mid \varphi?$$
$$\varphi ::= \alpha \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi \mid CK\varphi \mid [\pi]\varphi$$

where α ranges over ATM and i over Agt.[2]

Programs have the same meaning as in DL-PA: $+\alpha$ makes α true and $-\alpha$ makes α false, $\pi;\pi'$ is sequential composition, $\pi \sqcup \pi'$ is non-deterministic choice, π^* is unbounded iteration, and $\varphi?$ is test. Just as in dynamic logic, the formula $[\pi]\varphi$ reads "after every execution of π, φ is true."

As in other visibility logics, the formula $K_i\varphi$ reads "i knows that φ on the basis of what she observes." Moreover, $CK\varphi$ reads "all agents commonly know that φ on the basis of what they *jointly observe*." Our epistemic operators account for forms of individual and common knowledge that are respectively obtained via individual observation and joint

[2] As mentioned in the introduction, the Kleene star * was not included in the original language of DEL-PAO [Herzig et al., 2015].

observation of facts. As we have seen, this differs therefore conceptually from the classical operators of individual and common knowledge as studied in the area of epistemic logic [Fagin et al., 1995]. We will come back to this in Section 2.2.6.

The other boolean operators $\top$, $\bot$, $\vee$, $\rightarrow$ and $\leftrightarrow$ are defined in the standard way. Dual operators $\widehat{K}_i\varphi$ ("φ is compatible with i's knowledge") abbreviates $\neg K_i \neg\varphi$ and $\langle\pi\rangle\varphi$ ("there exists an execution of π after which φ is true") abbreviates $\neg[\pi]\neg\varphi$. The program $skip$ abbreviates $\top?$ and $fail$ abbreviates $\bot?$.

We also use π^m ("π is repeated m times"), with $m \geq 0$, inductively defined by $\pi^0 = skip$ and $\pi^{m+1} = \pi;\pi^m$, and $\pi^{\leq m}$ ("π is repeated m at most times") for $\bigsqcup_{0\leq k\leq m} \pi^k$. Finally, **if** φ **then** π abbreviates $(\varphi?;\pi) \sqcup \neg\varphi?$ and **if** φ **then** π **else** π' abbreviates $(\varphi?;\pi) \sqcup (\neg\varphi?;\pi')$.

The set of atomic formulas from ATM occurring in the formula φ is noted $ATM(\varphi)$. It is inductively defined as:

$$\begin{aligned} ATM(\alpha) &= \alpha \\ ATM(\neg\varphi) &= ATM(\varphi) \\ ATM(\varphi \wedge \varphi') &= ATM(\varphi) \cup ATM(\varphi') \\ ATM(K_i\varphi) &= ATM(\varphi) \\ ATM(CK\varphi) &= ATM(\varphi) \\ ATM([\pi]\varphi) &= ATM(\pi) \cup ATM(\varphi) \end{aligned}$$

with $ATM(\pi)$ the set of atoms occurring is the program π, defined as:

$$\begin{aligned} ATM(+\alpha) &= \alpha \\ ATM(-\alpha) &= \alpha \\ ATM(\pi;\pi') &= ATM(\pi) \cup ATM(\pi') \\ ATM(\pi \sqcup \pi') &= ATM(\pi) \cup ATM(\pi') \\ ATM(\pi^*) &= ATM(\pi) \\ ATM(\varphi?) &= ATM(\varphi) \end{aligned}$$

For example, let $\pi = q?;+S_2\,p$ and $\varphi = [\pi]S_1\,JS\,p$. Then $ATM(\pi) = \{q, S_2\,p\}$ and $ATM(\varphi) = \{q, S_2\,p, S_1\,JS\,p\}$. Note that $JS\,p$ is not an atom of φ.

2.2 Semantics of DEL-PAO

We denote valuations by V, V', etc. Remember that a valuation is simply a subset of the set of atoms ATM. The set of all valuations is again 2^{ATM}.

In this section, we stipulate constraints that are motivated by the requirement that visibility information should be introspective and that

joint visibility should imply individual visibility. We then define indistinguishability relations between valuations and show how to interpret formulas and programs.

2.2.1 Introspective valuations

A valuation $V \in 2^{ATM}$ is *introspective* if and only if the five following constraints hold, for every $\alpha \in ATM$ and $i \in Agt$:

$$S_i\, S_i\, \alpha \in V \tag{C1}$$
$$JS\; JS\, \alpha \in V \tag{C2}$$
$$JS\; S_i\, S_i\, \alpha \in V \tag{C3}$$
$$\text{if } JS\, \alpha \in V, \text{ then } S_i\, \alpha \in V \tag{C4}$$
$$\text{if } JS\, \alpha \in V, \text{ then } JS\, S_i\, \alpha \in V \tag{C5}$$

The set of all introspective valuations is noted $INTR$.

In words, (C1) impose introspection of individual sight: an agent always sees whether she sees the value of an atom. (C2) requires the same for joint sight; indeed, if $JS\, \alpha$ is true then $JS\; JS\, \alpha$ should be true by introspection, and if $JS\, \alpha$ is false then all agents jointly see that at least one of them has broken eye contact. (C3) forces the first to be common knowledge. (C4) guarantees that joint visibility implies individual visibility. The constraints (C4) and (C5) ensure that $JS\, \alpha \in V$ implies $\sigma\, \alpha \in V$ for $\sigma \in OBS^+$.

This motivates the following relation of *introspective consequence* between atoms:

$$\alpha \Rightarrow_I \beta \text{ iff either } \alpha = \beta, \text{ or } \alpha = JS\, \alpha' \text{ and } \beta = \sigma\, \alpha' \text{ for some } \sigma \in OBS^+.$$

When $\alpha \Rightarrow_I \beta$, we say that α is an *introspective cause* of β and that β is an *introspective consequence* of α. For a given α, we note $\alpha^{\Leftarrow}$ the set of introspective causes of α, and $\alpha^{\Rightarrow}$ the set of its introspective consequences.

Proposition 2.1. *A valuation $V \subseteq ATM$ is introspective if and only if, for every $\alpha, \beta \in ATM$ and $i \in Agt$:*

$$\sigma\, S_i\, S_i\, \alpha \in V \textit{ for every } \sigma \in OBS^* \tag{P1}$$
$$\sigma\, JS\, \alpha \in V \textit{ for every } \sigma \in OBS^+ \tag{P2}$$
$$\textit{if } \alpha \in V \textit{ and } \alpha \Rightarrow_I \beta \textit{ then } \beta \in V \tag{P3}$$

Proof. We prove that, for a valuation V, the constraints (C1)-(C5) are satisfied if and only if (P1), (P2) and (P3) hold.

For the left-to-right direction, we begin with (P3). The interesting case is when $\alpha = JS\, \alpha'$ (otherwise the only introspective consequence of α is itself). Suppose $JS\, \alpha' \in V$. By constraint (C5), we know

that $JS\, S_{i_1}\, \alpha' \in V$, and also that $JS\, S_{i_2}\, S_{i_1}\, \alpha' \in V$ for any i_1 and i_2, and so on. By repeating the application of (C5), we can generate any $JS\, S_{i_m} \dots S_{i_1}\, \alpha'$ and then, by (C4), we can obtain $S_i\, S_{i_m} \dots S_{i_1}\, \alpha'$ for any agents $i_1, \dots i_m$ and i. Moreover, we have $JS\, JS\, \alpha' \in V$ by (C2), and in the same way, we can generate $S_i\, S_{i_m} \dots S_{i_1}\, JS\, \alpha'$ for any agents $i_1, \dots i_m$ and i. By choosing α' appropriately, we can therefore obtain any sequence containing any S_i and JS, that is, all $\sigma\, \alpha'$ for any $\sigma \in OBS^+$.

Using the same technique, we obtain $\sigma\, S_i\, S_i\, \alpha$ and $\sigma\, JS\, \alpha$, the first with (C1) (for σ empty) and (C3) (for σ non-empty), and the second with (C2).

The other way round, constraints (C1), (C2) and (C3) are respectively instances of (P1), (P2) and (P1); and constraints (C4) and (C5) are both guaranteed by (P3). □

An atom $\alpha \in ATM$ is called *valid in* $INTR$, or *introspectively valid*, if and only if α belongs to every valuation in $INTR$.

Proposition 2.2. *The atom α is valid in $INTR$ if and only if α is:*

- *either of the form $\sigma\, S_i\, S_i\, \alpha'$ with $\sigma \in OBS^*$;*
- *or of the form $\sigma\, JS\, \alpha'$ with $\sigma \in OBS^+$.*

Proof. The right-to-left direction is obvious by Proposition 2.1.

For the left-to-right direction, we prove that every atom in $INTR$, i.e., every atom specified in Proposition 2.1, is of one of the given forms.

The first two items of Proposition 2.1 are already in the right form; we only need to prove that for every β such that either $\sigma\, S_i\, S_i\, \alpha' \Rightarrow_I \beta$ or $\sigma\, JS\, \alpha' \Rightarrow_I \beta$, β is of the form $\sigma\, S_i\, S_i\, \alpha'$ or $\sigma\, JS\, \alpha'$.

- Take $\alpha = \sigma\, S_i\, S_i\, \alpha'$ with $\sigma \in OBS^*$. Suppose that σ contains at least one observability operator and starts with a JS (otherwise, the only consequence of α is itself). Thus $\alpha = JS\, \sigma'\, S_i\, S_i\, \alpha'$ with $\sigma' \in OBS^*$. Then by the definition of introspective consequence, $\alpha \Rightarrow_I \beta$ if and only if $\beta = \sigma''\, \sigma'\, S_i\, S_i\, \alpha'$ with $\sigma' \in OBS^*$ and $\sigma'' \in OBS^+$, that is, β is captured by the form (P1) of Proposition 2.1.
- Now take $\alpha = \sigma\, JS\, \alpha'$ with $\sigma \in OBS^+$, and suppose that σ starts with a JS: $\alpha = JS\, \sigma'\, JS\, \alpha'$ with $\sigma' \in OBS^*$. Thus $\alpha \Rightarrow_I \beta$ if and only if $\beta = \sigma''\, \sigma'\, JS\, \alpha'$ with $\sigma' \in OBS^*$ and $\sigma'' \in OBS^+$, that is, β is captured by the form (P2) of Proposition 2.1.

Hence closure under introspective consequence characterizes introspective valuations.

Therefore valid atoms are exactly of the two given forms. □

In words, any atom containing a sequence $S_i\,S_i$ or containing (but not starting with) a JS is introspectively valid.

Observe that we do not impose the constraint "if $\sigma\,\alpha \in V$ for every $\sigma \in OBS^*$ then $JS\,\alpha \in V$," which corresponds to the greatest fixed point definition of the operator of common knowledge from shared knowledge. We will comment on this in Section 2.2.6.

2.2.2 Introspective causes and consequences

In this section, we study in more detail the relation of dependence between atoms that we defined in the previous section:

$$\alpha \Rightarrow_I \beta \text{ iff either } \alpha = \beta, \text{ or } \alpha = JS\,\alpha' \text{ and } \beta = \sigma\,\alpha' \text{ for some } \sigma \in OBS^+.$$

As we will see, this relation is omnipresent in DEL-PAO, especially when dealing with assignments. Remember that we note $\alpha^{\Leftarrow}$ the causes of α and $\alpha^{\Rightarrow}$ its consequences. We characterize them depending on the form of α:

- $\alpha = p$ with $p \in Prop$:
  - $\alpha^{\Leftarrow} = \{p\}$;
  - $\alpha^{\Rightarrow} = \{p\}$.
- $\alpha = S_i\,\sigma\,p$ with $\sigma \in OBS^*$ and $p \in Prop$:
  - $\alpha^{\Leftarrow} = \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\} \cup \{S_i\,\sigma\,p\}$;
  - $\alpha^{\Rightarrow} = \{S_i\,\sigma\,p\}$.
- $\alpha = JS\,\sigma\,p$ with $\sigma \in OBS^*$ and $p \in Prop$:
  - $\alpha^{\Leftarrow} = \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\}$;
  - $\alpha^{\Rightarrow} = \{\sigma'\,\sigma\,p : \sigma' \in OBS^+\}$.

Consequences are obvious from the definition of $\Rightarrow_I$: only atoms starting with a JS may have consequences different from themselves; they are all atoms obtained by replacing the JS operator by any sequence of visibility operators.

Causes are less straightforward. The first case is easy: p has no cause but itself. For the other cases, we actually are in two similar situations: the only difference is that when α starts with a JS, then it is itself contained in the set $\{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\}$ (case $k = 0$).

This latter set may seem complicated so let us illustrate it with an example. Suppose we want the causes of $JS\,S_1\,S_2\,p$. We are looking for every atom starting with a JS operator such that we can replace the JS by any sequence of depth at least 1 to obtain $JS\,S_1\,S_2\,p$. There are three of them:

1. $JS\ S_1\ S_2\ p$ itself (replace JS by JS);
2. $JS\ S_2\ p$ (replace JS by $JS\ S_1$);
3. $JS\ p$ (replace JS by $JS\ S_1\ S_2$).

$JS\ JS\ S_1\ S_2\ p$, for example, would not be eligible because the first JS cannot be replaced by a sequence of depth at least 1. (This implies that the depths of causes of α are always smaller or equal to the depth of α.) Therefore the causes of $S_i\,\sigma\,p$ or $JS\,\sigma\,p$ are every atom starting with a JS operator and followed by any postfix of σ (including the empty sequence), hence the result. Note the special case of $JS\,p$: its only cause is itself.

We will use these definitions and the following properties to demonstrate equivalences between sequences of assignments in Section 2.3.

Proposition 2.3. *Let α be an atom. Then $\alpha^{\Leftarrow} \cap \alpha^{\Rightarrow} = \{\alpha\}$.*

Proof. With the above definitions, the cases where $\alpha = p$ and $\alpha = S_i\,\sigma\,p$ are obvious. The last case, where $\alpha = JS\,\sigma\,p$, derives from the fact that all other causes of α have a depth strictly lower than α and all other consequences of α starting with a JS have depth strictly higher than α. □

Proposition 2.3 implies that if $\alpha \Rightarrow_I \beta$ and $\beta \Rightarrow_I \alpha$, then $\alpha = \beta$.

Proposition 2.4. *Let α, β, γ be three atoms. If $\alpha \Rightarrow_I \gamma$ and $\gamma \Rightarrow_I \beta$, then $\alpha \Rightarrow_I \beta$.*

Proof. Suppose $\gamma \neq \alpha$ and $\gamma \neq \beta$, otherwise the result is obvious. Since $\alpha \Rightarrow_I \gamma$, we have $\alpha = JS\,\alpha'$ and $\gamma = \sigma_1\,\alpha'$ with $\sigma_1 \in OBS^+$. Moreover, because $\gamma \Rightarrow_I \beta$, we have $\gamma = JS\,\gamma'$ and $\beta = \sigma_2\,\gamma'$ with $\sigma_2 \in OBS^+$.

Thus $\gamma' = \sigma_3\,\alpha'$ with $\sigma_3 \in OBS^*$ and hence $\beta = \sigma_2\,\sigma_3\,\alpha'$ (where $\sigma_2 \in OBS^+$) which is of the form $\sigma_1\,\alpha'$, with $\sigma_1 \in OBS^+$, and therefore a consequence of α. Therefore $\alpha \Rightarrow_I \beta$. □

Proposition 2.4 implies that if $\alpha \Rightarrow_I \beta$, then $\alpha^{\Leftarrow} \subseteq \beta^{\Leftarrow}$ and $\beta^{\Rightarrow} \subseteq \alpha^{\Rightarrow}$.

Proposition 2.5. *Let α and β be two atoms. If $\alpha \not\Rightarrow_I \beta$ and $\beta \not\Rightarrow_I \alpha$, then $\alpha^{\Leftarrow} \cap \beta^{\Rightarrow} = \emptyset$ and $\alpha^{\Rightarrow} \cap \beta^{\Leftarrow} = \emptyset$.*

Proof. To prove that $\alpha^{\Leftarrow} \cap \beta^{\Rightarrow} = \emptyset$, suppose there exists an atom γ such that $\gamma \in \alpha^{\Leftarrow}$ and $\gamma \in \beta^{\Rightarrow}$. This implies that $\gamma \Rightarrow_I \alpha$ and that $\beta \Rightarrow_I \gamma$. Hence by Proposition 2.4, $\beta \Rightarrow_I \alpha$, which contradicts our hypothesis.

The reasoning for $\alpha^{\Rightarrow} \cap \beta^{\Leftarrow} = \emptyset$ is similar. □

2.2.3 Indistinguishability relations

As usual in visibility logics, two valuations are related by the indistinguishability relation for agent i, noted $\sim_i$, if every α that i sees has the same value. The relation $\sim_{Agt}$ acts similarly for joint indistinguishability. They are defined as follows:

$$V \sim_i V' \text{ iff } S_i\,\alpha \in V \text{ implies } V(\alpha) = V'(\alpha)$$
$$V \sim_{Agt} V' \text{ iff } JS\,\alpha \in V \text{ implies } V(\alpha) = V'(\alpha)$$

with, as in the introduction, $V(\alpha) = V'(\alpha)$ when either $\alpha \in V$ and $\alpha \in V'$, or $\alpha \notin V$ and $\alpha \notin V'$. In the following, let us write $V =_A V'$, with $A \subseteq ATM$, in place of "$V(\alpha) = V'(\alpha)$ for every $\alpha \in A$." Thus $V \sim_i V'$ if and only if $V =_{\{\alpha : S_i\,\alpha \in V\}} V'$ and $V \sim_{Agt} V'$ if and only if $V =_{\{\alpha : JS\,\alpha \in V\}} V'$.

Now that we account for higher-order knowledge and that visibilities are not constant across $\sim_i$ and $\sim_{Agt}$, the two binary relations are not equivalence relations. Indeed, while being reflexive, they are neither symmetric nor transitive in the general case. For example, $\emptyset \sim_i \{p, S_i\,p\}$ while $\{p, S_i\,p\} \not\sim_i \emptyset$ since $p \notin \emptyset$. However, both properties hold on valuations satisfying the introspection constraints (C1) and (C2).

Proposition 2.6. *The relation $\sim_{Agt}$ and every $\sim_i$ are equivalence relations on $INTR$.*

Proof. Reflexivity is obvious from the definition of relations. We prove symmetry and transitivity of $\sim_i$ for an arbitrary agent i. The proof for $\sim_{Agt}$ is analogous, but with (C2) instead of (C1).

- **Symmetry.** Take two valuations $V, V' \in INTR$. Suppose $V \sim_i V'$ and $V' \not\sim_i V$. By the former, for every α, if $S_i\,\alpha \in V$ then $V(\alpha) = V'(\alpha)$. By the latter, there exists a β such that $S_i\,\beta \in V'$ and $V(\beta) \neq V'(\beta)$. Thus this β is such that $S_i\,\beta \notin V$. Therefore there exists a β such that $S_i\,\beta \in V'$ but $S_i\,\beta \notin V$, that is, $V(S_i\,\beta) \neq V'(S_i\,\beta)$. Since $V \in INTR$, we have $S_i\,S_i\,\alpha \in V$ for every α by constraint (C1). This means that for every α, $V(S_i\,\alpha) = V'(S_i\,\alpha)$. We obtain a contradiction, thus $V' \sim_i V$.

- **Transitivity.** Take three valuations $V, V', V'' \in INTR$. Suppose $V \sim_i V'$, $V' \sim_i V''$ and $V \not\sim_i V''$. By the first two hypotheses, for every α, if $S_i\,\alpha \in V$ then $V(\alpha) = V'(\alpha)$ and if $S_i\,\alpha \in V'$ then $V'(\alpha) = V''(\alpha)$. As observed above, since $V \in INTR$, $V(S_i\,\alpha) = V'(S_i\,\alpha)$ for every α. Therefore if $S_i\,\alpha \in V$, then $S_i\,\alpha \in V'$, that is, if $S_i\,\alpha \in V$, $V(\alpha) = V'(\alpha)$ and $V'(\alpha) = V''(\alpha)$, thus $V(\alpha) = V''(\alpha)$, for every α. This contradicts the third hypothesis since the latter implies that there exists a β such that $S_i\,\beta \in V$ and $V(\beta) \neq V''(\beta)$. Therefore $V \sim_i V''$.

Therefore $\sim_{Agt}$ and $\sim_i$ are equivalence relations on $INTR$. □

Lemma 2.1. *Let $V \in INTR$ and $V' \in 2^{ATM}$. If $V \sim_i V'$ or $V \sim_{Agt} V'$ then $V' \in INTR$.*

Proof. Suppose that $V \sim_i V'$ for an arbitrary agent i and that $V \in INTR$. We prove that V' satisfies constraints (C1)-(C5). Recall that the five constraints are equivalent to the three properties (P1)-(P3) of Proposition 2.1. The proof for $\sim_{Agt}$ is analogous.

Take an arbitrary agent j.

- **(C1).** By (P1), $S_i\, S_j\, S_j\, \alpha \in V$ and $S_j\, S_j\, \alpha \in V$. Thus $S_j\, S_j\, \alpha \in V'$.
- **(C2).** By (P2), $S_i\, JS\, JS\, \alpha \in V$ and $JS\, JS\, \alpha \in V$. Thus $JS\, JS\, \alpha \in V'$.
- **(C3).** By (P1), $S_i\, JS\, S_j\, S_j\, \alpha \in V$ and $JS\, S_j\, S_j\, \alpha \in V$. Thus $JS\, S_j\, S_j\, \alpha \in V'$.
- **(C4).** Suppose $JS\, \alpha \in V'$. By (P2), $S_i\, JS\, \alpha \in V$, thus $V(JS\, \alpha) = V'(JS\, \alpha)$ and thus $JS\, \alpha \in V$. By (P3), this implies that both $S_i\, S_j\, \alpha \in V$ and $S_j\, \alpha \in V$. Thus $S_j\, \alpha \in V'$.
- **(C5).** Suppose $JS\, \alpha \in V'$. As observed with the previous item, this implies that $JS\, \alpha \in V$. Thus by (P3), both $S_i\, JS\, S_j\, \alpha \in V$ and $JS\, S_j\, \alpha \in V$. Thus $JS\, S_j\, \alpha \in V'$.

Therefore $V' \in INTR$. □

Together, Proposition 2.6 and Lemma 2.1 ensure that $\sim_i$ and $\sim_{Agt}$ have a "proper behaviour" on introspective valuations and that they cannot exit them. While only (C1) and (C2) are necessary to ensure the former, for the latter all constraints are required.

2.2.4 Valuation updates

Given an introspective valuation V, our update operations add or remove atoms from V. This requires some care: we want the resulting valuation to be introspective. For example, removing $S_i\, S_i\, p$ should be impossible. Another example is when V does not contain $S_i\, p$: then $V \cup \{JS\, p\}$ would violate (C4).

Thus, when adding an atom to V, we also have to add all its introspective consequences; symmetrically, when removing an atom we also have to remove its introspective causes. To that end, let us define the following update operations on valuations:

$$V{+}\alpha = V \cup \alpha^{\Rightarrow}$$
$$V{-}\alpha = V \setminus \alpha^{\Leftarrow}$$

Proposition 2.7. *Let $V \in INTR$ and $\alpha \in ATM$. Then $V{+}\alpha \in INTR$. Moreover, if α is not valid in $INTR$, then $V{-}\alpha \in INTR$.*

Proof. Doing either $V{+}\alpha$, or $V{-}\alpha$ such that α is not valid in $INTR$, implies that we are not deleting any valid atom in the process. Therefore we only have to take care that property (P3) of Proposition 2.1 (introspective consequence) is preserved:

- when adding α, adding every β such that $\alpha \Rightarrow_I \beta$ ensures that it is;
- when removing α, removing every β such that $\beta \Rightarrow_I \alpha$ ensures that no atom can still introspectively imply α, thus preserving introspective consequence.

Therefore $V{+}\alpha$ and $V{-}\alpha$ are introspective. □

$V{+}\alpha$ and $V{-}\alpha$ will be the updated worlds resulting from adding α to V or removing α (if not valid) from V. Proposition 2.7 ensures that corresponding atomic programs $+\alpha$ and $-\alpha$ will preserve introspection.

2.2.5 Truth conditions and validity

Truth conditions are as follows:

$$
\begin{array}{ll}
V \models \alpha & \text{iff } \alpha \in V \\
V \models \neg\varphi & \text{iff not } (V \models \varphi) \\
V \models \varphi \wedge \varphi' & \text{iff } V \models \varphi \text{ and } V \models \varphi' \\
V \models K_i\varphi & \text{iff } V' \models \varphi \text{ for every } V' \text{ such that } V \sim_i V' \\
V \models CK\varphi & \text{iff } V' \models \varphi \text{ for every } V' \text{ such that } V \sim_{Agt} V' \\
V \models [\pi]\varphi & \text{iff } V' \models \varphi \text{ for every } V' \text{ such that } V R_\pi V'
\end{array}
$$

where R_π is a binary relation on valuations defined by:

$$
\begin{array}{ll}
V R_{+\alpha} V' & \text{iff } V' = V{+}\alpha \\
V R_{-\alpha} V' & \text{iff } V' = V{-}\alpha \text{ and } \alpha \text{ is not valid in } INTR \\
V R_{\pi;\pi'} V' & \text{iff } V(R_\pi \circ R_{\pi'})V' \\
V R_{\pi \sqcup \pi'} V' & \text{iff } V(R_\pi \cup R_{\pi'})V' \\
V R_{\pi^*} V' & \text{iff } V(\bigcup_{k \in \mathbb{N}_0} (R_\pi)^k)V' \\
V R_{\varphi?} V' & \text{iff } V = V' \text{ and } V \models \varphi
\end{array}
$$

Truth conditions are defined on any valuation $V \in 2^{ATM}$. However, we have seen that indistinguishability relations $\sim_i$ and $\sim_{Agt}$ will not be equivalence relations outside of introspective valuations. Therefore knowledge operators may have unexpected behaviour in that case.

The relation R_π is defined like in dynamic logic for the program operators ;, $\sqcup$, * and $?$. As noted in the previous section, the interpretation of assignments is designed in a way such that we stay in $INTR$: the program $+\alpha$ adds all the consequences of α; the program $-\alpha$ fails if α is valid in $INTR$ and otherwise removes all the causes of α. For example, we never have $V R_{-S_1\,S_1\,p} V'$, i.e., the program $-S_1\,S_1\,p$ always fails (thus the formula $[-S_1\,S_1\,p]\varphi$ is always true, whatever φ). In contrast, the program $-S_1\,S_2\,p$ always succeeds, and we have $V R_{-S_1\,S_2\,p}\,(V \setminus \{S_1\,S_2\,p, JS\,S_2\,p, JS\,p\})$ because as we have seen in Section 2.2.2, the only atoms—beyond $S_1\,S_2\,p$ itself—whose consequence is $S_1\,S_2\,p$ are $JS\,S_2\,p$ and $JS\,p$. Therefore $V \models [-S_1\,S_2\,p]\neg JS\,p$ for every V.

We say that two programs π_1 and π_2 are equivalent, noted $\pi_1 \equiv \pi_2$, if and only if R_{π_1} equals R_{π_2}, i.e., for every $V, V' \in 2^{ATM}$ we have $V R_{\pi_1} V'$ if and only if $V R_{\pi_2} V'$.

Lemma 2.2. *Let $V \in INTR$ and $V R_\pi V'$. Then $V' \in INTR$.*

Proof. By Proposition 2.7, we know that $R_{+\alpha}$ and $R_{-\alpha}$ preserve introspection. $R_{\pi;\pi'}$, $R_{\pi\sqcup\pi'}$ and R_{π^*} make unions and compositions and thus preserve introspection, assuming R_π and $R_{\pi'}$ do. Finally, $R_{\varphi?}$ relates the current valuation to itself and thus trivially stays in $INTR$. □

Proposition 2.8. *For every $V \in INTR$, $i \in Agt$ and program π, V is related by $\sim_i$, $\sim_{Agt}$ and R_π only to valuations in $INTR$.*

Proof. This is a direct consequence of lemmas 2.1 and 2.2. □

When $V \models \varphi$ we say that V is a *model* of φ. The set of (not necessarily introspective) models of φ is noted $\|\varphi\|$.

A formula φ is *satisfiable in $INTR$* if φ has an introspective model, i.e., if $\|\varphi\| \cap INTR \neq \emptyset$; it is *valid in $INTR$* if $INTR \subseteq \|\varphi\|$. A formula φ is *plainly satisfiable* if it has a model, i.e., if $\|\varphi\| \neq \emptyset$; it is *plainly valid* if φ is true in all models, i.e., if $\|\varphi\| = 2^{ATM}$.

For example, $JS\,p \wedge \neg S_i\,p$ is plainly satisfiable but not satisfiable in $INTR$. On the other hand, $[-S_1\,S_2\,p]\neg JS\,p$ is valid in $INTR$ (and even plainly valid).

Proposition 2.9. *Let φ be a formula without epistemic operators. Let $V, V' \in 2^{ATM}$ be such that $V =_{ATM(\varphi)} V'$. Then $V \models \varphi$ if and only if $V' \models \varphi$.*

Remember that $V =_A V'$ means that $V(\alpha) = V'(\alpha)$ for every $\alpha \in A$.

Proof. We prove by mutual recursion on formulas and programs the two following hypotheses:

- $H_{\mathtt{f}}(\varphi)$: if $V =_{ATM(\varphi)} V'$, then $V \models \varphi$ if and only if $V' \models \varphi$;

- $H_{\mathtt{p}}(\pi)$: for every $A \subseteq ATM$: if $V =_{ATM(\pi) \cup A} V'$, then for every $U \in 2^{ATM}$ such that $V R_\pi U$, there exists $U' \in 2^{ATM}$ such that $V' R_\pi U'$ and $U =_A U'$.

Take two arbitrary valuations $V, V' \in 2^{ATM}$.

We begin with $H_{\mathtt{f}}(\varphi)$. Using the definition of $ATM(\varphi)$, the proof is straightforward for atoms and boolean operators. We only examine the case where $\varphi = [\pi]\varphi'$.

By definition of $ATM(\varphi)$, we have $ATM(\varphi) = ATM(\pi) \cup ATM(\varphi')$, hence suppose $V =_{ATM(\pi) \cup ATM(\varphi')} V'$. We prove that for every U such that $V R_\pi U$, $U \models \varphi'$ if and only if for every U' such that $V' R_\pi U'$, $U' \models \varphi'$.

By $H_{\mathtt{p}}(\pi)$, we have:

> for every U such that $V R_\pi U$,
> there exists U' such that $V' R_\pi U'$ and $U =_{ATM(\varphi')} U'$.

We also have:

> for every U' such that $V' R_\pi U'$,
> there exists U such that $V R_\pi U$ and $U =_{ATM(\varphi')} U'$.

By $H_{\mathtt{f}}(\varphi')$, they become:

> for every U such that $V R_\pi U$,
> there exists U' such that $V' R_\pi U'$
> and $U \models \varphi'$ if and only if $U' \models \varphi'$ (2.1)
> and for every U' such that $V' R_\pi U'$,
> there exists U such that $V R_\pi U$
> and $U \models \varphi'$ if and only if $U' \models \varphi'$. (2.2)

For the left-to-right direction, suppose that $U \models \varphi'$ for every U such that $V R_\pi U$, but that there exists U' such that $V' R_\pi U'$ and $U' \not\models \varphi'$. By (2.2), the latter implies that there exists U such that $V R_\pi U$ and $U \not\models \varphi'$, which contradicts the former. The right-to-left direction is analogous but using (2.1).

Therefore for every U such that $V R_\pi U$, $U \models \varphi'$ if and only if for every U' such that $V' R_\pi U'$, $U' \models \varphi'$.

We now move to $H_{\mathtt{p}}(\pi)$. We suppose that in all cases, π does not fail; otherwise, $H_{\mathtt{p}}(\pi)$ is trivially true since there is no U such that $V R_\pi U$. Moreover, as we will see in the axiomatization (Section 2.3), π^* can be reduced to $\pi^{\leq 2^{2|ATM(\pi)|}}$, which is a non-deterministic composition of sequences of π. Therefore we skip this case.

Take an arbitrary set of atoms $A \subseteq ATM$.

- $\boldsymbol{\pi = +\alpha}$. We have $ATM(\pi) = \{\alpha\}$, thus suppose $V =_{\{\alpha\} \cup A} V'$. The program is deterministic: $U = V{+}\alpha$ and $U' = V'{+}\alpha$. Take an arbitrary atom $\beta \in A$.

  - Suppose $\alpha \Rightarrow_I \beta$: then β was added by $+\alpha$ and thus $\beta \in U$ and $\beta \in U'$.
  - Now suppose $\alpha \not\Rightarrow_I \beta$: then β was not modified by $+\alpha$. Thus $U(\beta) = V(\beta)$ and $U'(\beta) = V'(\beta)$. Since $V(\beta) = V'(\beta)$ (because $\beta \in A$), then $U(\beta) = U'(\beta)$.

  In both cases, β has the same value in U and in U': $U =_A U'$.

- $\boldsymbol{\pi = -\alpha}$. The proof is analogous to the case $\pi = +\alpha$.

- $\boldsymbol{\pi = \pi_1; \pi_2}$. We have $ATM(\pi) = ATM(\pi_1) \cup ATM(\pi_2)$, thus suppose $V =_{ATM(\pi_1) \cup ATM(\pi_2) \cup A} V'$. By $H_{\mathrm{p}}(\pi_1)$, this implies that:

  for every U_1 such that $V R_{\pi_1} U_1$,
  there exists U_1' such that $V' R_{\pi_1} U_1'$
  and $U_1 =_{ATM(\pi_2) \cup A} U_1'$.

  This time by $H_{\mathrm{p}}(\pi_2)$, this gives:

  for every U_1 such that $V R_{\pi_1} U_1$,
  there exists U_1' such that $V' R_{\pi_1} U_1'$
  and for every U such that $U_1 R_{\pi_2} U$,
  there exists U' such that $U_1' R_{\pi_2} U'$
  and $U =_A U'$.

  Since π does not fail:

  for every U such that there exists U_1 such that $V R_{\pi_1} U_1$ and $U_1 R_{\pi_2} U$,
  there exists U' and U_1' such that $V' R_{\pi_1} U_1'$ and $U_1' R_{\pi_2} U'$
  and $U =_A U'$.

  This is equivalent to:

  for every U such that $V R_{\pi_1;\pi_2} U$,
  there exists U' such that $V' R_{\pi_1;\pi_2} U'$
  and $U =_A U'$,

  which is our result.

- $\boldsymbol{\pi = \pi_1 \sqcup \pi_2}$. We have $ATM(\pi) = ATM(\pi_1) \cup ATM(\pi_2)$, thus suppose $V =_{ATM(\pi_1) \cup ATM(\pi_2) \cup A} V'$. By applying $H_{\mathrm{p}}(\pi_1)$ and $H_{\mathrm{p}}(\pi_2)$, we obtain:

  for every U_1 such that $V R_{\pi_1} U_1$,
  there exists U_1' such that $V' R_{\pi_1} U_1'$ and $U_1 =_{ATM(\pi_2) \cup A} U_1'$
  and for every U_2 such that $V R_{\pi_2} U_2$,
  there exists U_2' such that $V' R_{\pi_2} U_2'$ and $U_2 =_{ATM(\pi_1) \cup A} U_2'$.

  Note that if $V =_{B \cup A} V'$, then $V =_A V'$, thus:

  for every U_1 such that $V R_{\pi_1} U_1$,
  there exists U_1' such that $V' R_{\pi_1} U_1'$ and $U_1 =_A U_1'$
  and for every U_2 such that $V R_{\pi_2} U_2$,
  there exists U_2' such that $V' R_{\pi_2} U_2'$ and $U_2 =_A U_2'$.

  This implies, by addition:

  for every U_1 such that $V R_{\pi_1} U_1$,
  there exists U_1' such that $V' R_{\pi_1} U_1'$ and $U_1 =_A U_1'$
  or there exists U_2' such that $V' R_{\pi_2} U_2'$ and $U_1 =_A U_2'$
  and for every U_2 such that $V R_{\pi_2} U_2$,
  there exists U_2' such that $V' R_{\pi_2} U_2'$ and $U_2 =_A U_2'$
  or there exists U_1' such that $V' R_{\pi_1} U_1'$ and $U_2 =_A U_1'$.

  Thus:

  for every U_1 such that $V R_{\pi_1} U_1$,
  there exists U' such that $V' R_{\pi_1} U'$ or $V' R_{\pi_2} U'$
  and $U_1 =_A U'$
  and for every U_2 such that $V R_{\pi_2} U_2$,
  there exists U' such that $V' R_{\pi_2} U'$ or $V' R_{\pi_1} U'$
  and $U_2 =_A U'$,

  that is:

  for every U such that $V R_{\pi_1} U$ or $V R_{\pi_2} U$,
  there exists U' such that $V' R_{\pi_1} U'$ or $V' R_{\pi_2} U'$
  and $U =_A U'$.

  This is equivalent to:

  for every U such that $V R_{\pi_1 \sqcup \pi_2} U$,
  there exists U' such that $V' R_{\pi_1 \sqcup \pi_2} U'$
  and $U =_A U'$,

  which is our result.

- $\pi = \chi?$. We have $ATM(\pi) = ATM(\chi)$, hence suppose $V =_{ATM(\chi)\cup A} V'$. As we have seen, this implies $V =_{ATM(\chi)} V'$, and by $H_{\mathtt{f}}(\chi)$, we thus have $V \models \chi$ if and only if $V' \models \chi$. We have assumed that π does not fail; this implies that $V \models \chi$ and thus that $V' \models \chi$, hence $U = V$ and $U' = V'$. Thus $U =_{ATM(\chi)\cup A} U'$ and therefore $U =_A U'$.

Therefore if $V =_{ATM(\varphi)} V'$, $V \models \varphi$ if and only if $V' \models \varphi$. □

This proposition will be instrumental in many places across the thesis. Observe that it does not hold when φ contains epistemic operators. For example, the truth value of $K_i p$ depends on that of $S_i\, p$, which however does not occur in $ATM(K_i p)$. We will see in Section 2.4 how to account for these operators.

2.2.6 Discussion

Both the operators of individual knowledge and the operator of common knowledge of DEL-PAO satisfy all the principles of the standard epistemic logic S5.

We have seen in the introduction that there are also some further validities. For example the distribution of knowledge over disjunction of literals is still valid in DEL-PAO:

$$K_i(p \vee q) \to (K_i p \vee K_i q),$$

cf. the axiom ($Red_{\mathrm{K},\vee}$) below. This is a strong principle: to give an example, if one knows that the butler or the gardener was the murderer then one knows which of them it was. We will see how to relax it in Chapter 6.

Our common knowledge operator obeys the fixed point axiom:

$$CKp \to p \wedge \Big(\bigwedge_{i\in Agt} K_i CKp \Big).$$

This is ensured by constraints (C2) and (C4) that make the formula

$$\bigwedge_{i\in Agt} S_i\, JS\, p$$

valid in $INTR$. Our notion of common knowledge is however weaker than standard common knowledge because the induction axiom

$$\Big(\varphi \wedge CK\big(\varphi \to \bigwedge_{i\in Agt} K_i\varphi\big)\Big) \to CK\varphi$$

is invalid in $INTR$. To see this, take as an example $Agt = \{1, 2\}$ and $Prop = p$. Take the valuation $V = \{p\} \cup \{\sigma\, p : \sigma \in OBS^+ \setminus \{JS\}\}$, i.e.,

everybody sees p and everybody knows that, but there is no joint visibility of p. This valuation is introspective since every introspectively valid atom belongs to V and $JS\,p$, which is absent from V, has no introspective causes but itself. Then

$$\Big(p \wedge CK\big(p \to (K_1 p \wedge K_2 p)\big)\Big) \to CKp$$

is not true at V. Indeed, given an arbitrary $V' \in 2^{ATM}$, we have:

$$\begin{aligned}
& V' \models CK\big(p \to (K_1 p \wedge K_2 p)\big) \\
\Leftrightarrow\ & V' \models CK\big(\neg p \vee (K_1 p \wedge K_2 p)\big) \\
\Leftrightarrow\ & V' \models CK\big(\neg p \vee (S_1\,p \wedge S_2\,p)\big) \\
\Leftrightarrow\ & V' \models CK\neg p \vee (CK\,S_1\,p \wedge CK\,S_2\,p) \\
\Leftrightarrow\ & V' \models CK\neg p \vee (JS\,S_1\,p \wedge S_1\,p \wedge JS\,S_2\,p \wedge S_2\,p)
\end{aligned}$$

by the axioms of Proposition 2.10. While $V \not\models CK\neg p$, the atoms $JS\,S_1\,p$, $S_1\,p$, $JS\,S_2\,p$ and $S_2\,p$ belong to V. Therefore $V \models p \wedge CK\big(p \to (K_1 p \wedge K_2 p)\big)$. On the other hand, $V \not\models CKp$ since this is equivalent to $JS\,p \wedge p$, and the former is false in V. Therefore the induction axiom is invalid in $INTR$.

Observe that defining the indistinguishability relation for common knowledge $\sim_{Agt}$ as $(\bigcup_{i \in Agt} \sim_i)^*$ instead of relying on the JS operator would make the induction axiom valid. Beyond the technical reason for that choice (the corresponding constraint is infinite and thus cannot be axiomatized by formula built from visibility atoms) we follow [Lorini and Herzig, 2014; Herzig, 2014] and assume that such a principle is too strong for a logic of common knowledge.

2.3 Axiomatization

Our axiomatization relies on reduction axioms. With them, we are able to reduce any DEL-PAO formula to a propositional formula.

2.3.1 Reduction axioms for epistemic operators

Proposition 2.10. *The following formulas are plainly valid, where α is an atom and A^+ and A^- are sets of atoms:*

$$K_i\alpha \leftrightarrow S_i\,\alpha \wedge \alpha \qquad (Red_{K,\alpha})$$

$$CK\,\alpha \leftrightarrow JS\,\alpha \wedge \alpha \qquad (Red_{CK,\alpha})$$

$$K_i\neg\alpha \leftrightarrow S_i\,\alpha \wedge \neg\alpha \qquad (Red_{K,\neg\alpha})$$

$$CK\neg\alpha \leftrightarrow JS\,\alpha \wedge \neg\alpha \qquad (Red_{CK,\neg\alpha})$$

$$K_i(\varphi \wedge \varphi') \leftrightarrow K_i\varphi \wedge K_i\varphi' \qquad (Red_{K,\wedge})$$

$$CK(\varphi \wedge \varphi') \leftrightarrow CK\varphi \wedge CK\varphi' \qquad (Red_{CK,\wedge})$$

$$K_i\Big(\bigvee_{\alpha\in A^+}\alpha \vee \bigvee_{\alpha\in A^-}\neg\alpha\Big) \leftrightarrow \begin{cases} \Big(\bigvee_{\alpha\in A^+} K_i\alpha\Big) \vee \Big(\bigvee_{\alpha\in A^-} K_i\neg\alpha\Big) & \textit{if } A^+\cap A^- = \emptyset \\ \top & \textit{otherwise} \end{cases} \qquad (Red_{K,\vee})$$

$$CK\Big(\bigvee_{\alpha\in A^+}\alpha \vee \bigvee_{\alpha\in A^-}\neg\alpha\Big) \leftrightarrow \begin{cases} \Big(\bigvee_{\alpha\in A^+} CK\alpha\Big) \vee \Big(\bigvee_{\alpha\in A^-} CK\neg\alpha\Big) & \textit{if } A^+\cap A^- = \emptyset \\ \top & \textit{otherwise} \end{cases} \qquad (Red_{CK,\vee})$$

Proof. For each validity, consider an arbitrary valuation $V \in 2^{ATM}$. Some proofs are skipped due to their similarity with those of standard epistemic logic [Fagin et al., 1995].

- **($Red_{K,\alpha}$).** From left to right, suppose $V \not\models \alpha \wedge S_i\,\alpha$. If $V \not\models \alpha$ then $V \not\models K_i\alpha$ because $\sim_i$ is reflexive. If $V \not\models S_i\,\alpha$ then by definition of $\sim_i$ there exists V' such that $V \sim_i V'$ and $\alpha \notin V'$, thus $V \not\models K_i\alpha$.

  From right to left, suppose $V \models S_i\,\alpha$ and $V \models \alpha$. Then by definition of $\sim_i$, every valuation V' such that $V \sim_i V'$ contains α, thus $V \models K_i\alpha$.

- Proofs for ($Red_{CK,\alpha}$), ($Red_{K,\neg\alpha}$) and ($Red_{CK,\neg\alpha}$) are analogous to the proof for ($Red_{K,\alpha}$).

- Proofs for ($Red_{K,\wedge}$) and ($Red_{CK,\wedge}$) are standard.

- **($Red_{K,\vee}$).** The case where $A^+ \cap A^- \neq \emptyset$ is obvious: K_i faces a tautological disjunction and the formula is therefore equivalent to $\top$.

  For the other case, the proof for the right-to-left direction is standard; here we examine only the left-to-right direction. As we shall see, this unusual validity is due to the fact that any subset of ATM

is an eligible world. We give a proof by contraposition. Suppose:

$$V \models \Big(\bigwedge_{\alpha \in A^+} \widehat{K}_i \neg\alpha \Big) \wedge \Big(\bigwedge_{\alpha \in A^-} \widehat{K}_i \alpha \Big), \tag{2.3}$$

we prove:

$$V \models \widehat{K}_i \Big(\bigwedge_{\alpha \in A^+} \neg\alpha \wedge \bigwedge_{\alpha \in A^-} \alpha \Big),$$

i.e., there exists a valuation related to V by $\sim_i$ where every atom from A^+ is false and every atom from A^- is true.[3] Let us define the four following sets of atoms:

$$\begin{aligned} A^+_S &= \{\alpha : \alpha \in A^+ \text{ and } S_i\,\alpha \in V\} \\ A^+_{\not S} &= \{\alpha : \alpha \in A^+ \text{ and } S_i\,\alpha \notin V\} \\ A^-_S &= \{\alpha : \alpha \in A^- \text{ and } S_i\,\alpha \in V\} \\ A^-_{\not S} &= \{\alpha : \alpha \in A^- \text{ and } S_i\,\alpha \notin V\} \end{aligned}$$

(Note that the sets are mutually disjoint because A^+ and A^- are.) We can reformulate (2.3) by splitting each conjunction into two parts:

$$V \models \Big(\bigwedge_{\alpha \in A^+_S} \widehat{K}_i \neg\alpha \Big) \wedge \Big(\bigwedge_{\alpha \in A^+_{\not S}} \widehat{K}_i \neg\alpha \Big) \wedge \Big(\bigwedge_{\alpha \in A^-_S} \widehat{K}_i \alpha \Big) \wedge \Big(\bigwedge_{\alpha \in A^-_{\not S}} \widehat{K}_i \alpha \Big). \tag{2.4}$$

Let us first look at atoms that are seen by i. Take an atom $\alpha \in A^+_S$. By (2.4), we have $V \models \widehat{K}_i \neg\alpha$, and by the definition of A^+_S, we have $V \models S_i\,\alpha$. Note that $\widehat{K}_i \neg\alpha \wedge S_i\,\alpha \leftrightarrow K_i \neg\alpha$ is plainly valid (by ($Red_{K,\alpha}$) and ($Red_{K,\neg\alpha}$)). This implies that every $\alpha \in A^+_S$ is false in V and in every related valuation. Analogously, every $\alpha \in A^-_S$ is true in V and in every related valuation. Atoms from $A^+_{\not S}$ and $A^-_{\not S}$, on the other hand, are not seen by i and thus can have any value in the current and in related valuations.

Take $V' = (V \setminus A^+_{\not S}) \cup A^-_{\not S}$. No atom α that i sees in V varies in V':

- if α is outside of A^+ and A^- then, seen or not, it is not modified between V and V';
- if α is either in A^+ or A^-, $S_i\,\alpha \in V$ implies that $\alpha \notin A^+_{\not S}$ and $\alpha \notin A^-_{\not S}$. Hence α is not modified either between V and V'.

[3] Remember that $\widehat{K}_i\varphi$ reads "φ is compatible with i's knowledge."

Therefore $V \sim_i V'$.

Now let us examine the truth values of our atoms in V'. If $\alpha \in A_S^+$, then we have seen that α is false in every valuation related to V. Moreover, if $\alpha \in A_{\not S}^+$, then by construction of V', α is false in V'. Thus every atom from A^+ is false in V'. Similarly, we can prove that every atom from A^- is true in V'. Since $V \sim_i V'$, we have:

$$V \models \widehat{K}_i\Big(\bigwedge_{\alpha \in A^+} \neg\alpha \wedge \bigwedge_{\alpha \in A^-} \alpha \Big).$$

- The proof for ($Red_{CK,\vee}$), is analogous to the proof for ($Red_{K,\vee}$).

Therefore the given formulas are plainly valid. □

Distribution of knowledge over disjunction ($Red_{K,\vee}$) entails that, for example,

$$K_i(p \vee q) \to K_i p \vee K_i q$$

is valid. However, the following formula is invalid:

$$K_i(K_j p \vee K_j \neg p) \to K_i K_j p \vee K_i K_j \neg p,$$

as it would mean that if i knows that j knows the value of p (i.e., that j sees p), then she knows that j knows p or she knows that j does not know p, and therefore that she knows the value of p herself. This implies that DEL-PAO is not closed under *propositional substitution*: we cannot replace p by $K_j p$ and q by $K_j \neg p$ in the first (valid) formula and obtain a new validity. While this is a property of standard epistemic logic, dynamic epistemic logics are not closed under propositional substitution either [van Ditmarsch et al., 2007].

2.3.2 Reduction axioms for dynamic operators

The three lemmas give equivalences for sequence of affectations, that will be useful in the proof of the proposition.

Lemma 2.3. *We have the following program equivalences, where α is an atom and p is a propositional variable:*

$$+\alpha; +\alpha \equiv +\alpha \qquad (+=+)$$

$$-\alpha; -\alpha \equiv -\alpha \qquad (-=-)$$

$$-\alpha; +\alpha \equiv \begin{cases} +\alpha & \text{if } \alpha = p \text{ or } \alpha = JS\,p \\ +\alpha; -JS\,\sigma\,p & \text{if } \alpha = S_i\,\sigma\,p \text{ with } \sigma \in OBS^* \\ +\alpha; -JS\,\sigma[1{:}]\,p & \text{if } \alpha = JS\,\sigma\,p \text{ with } \sigma \in OBS^+ \end{cases} \qquad (-=+)$$

Proof. For every equivalence, take a valuation $V \in 2^{ATM}$ and suppose we execute the sequence of assignments from V. We assume that α is not introspectively valid (otherwise $-\alpha$ fails and the equivalences are trivially valid). Remember that by the truth conditions, executing $+\alpha$ from V leads to the valuation $V \cup \alpha^{\Rightarrow}$ and executing $-\alpha$ from V leads to $V \setminus \alpha^{\Leftarrow}$. We use definitions of Section 2.2.2.

- $(+{=}+)$. Obvious: $(V \cup \alpha^{\Rightarrow}) \cup \alpha^{\Rightarrow} = V \cup \alpha^{\Rightarrow}$.
- $(-{=}-)$. Obvious again: $(V \setminus \alpha^{\Leftarrow}) \setminus \alpha^{\Leftarrow} = V \setminus \alpha^{\Leftarrow}$.
- $(-{=}+)$. We examine each three cases:
  - Suppose $\alpha = p$ or $\alpha = JS\,p$. In the first case, $\alpha^{\Leftarrow} = \alpha^{\Rightarrow} = \{p\}$; in the second case, $\alpha^{\Leftarrow} = \{JS\,p\}$ and $\alpha^{\Rightarrow} = \{\sigma\,p : \sigma \in OBS^{+}\}$. In both cases, $\alpha^{\Leftarrow} \subseteq \alpha^{\Rightarrow}$; hence $(V \setminus \alpha^{\Leftarrow}) \cup \alpha^{\Rightarrow} = V \cup \alpha^{\Rightarrow}$.
  - Now suppose $\alpha = S_i\,\sigma\,p$ with $\sigma \in OBS^{*}$. Then we have $\alpha^{\Leftarrow} = \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\} \cup \{S_i\,\sigma\,p\}$ and $\alpha^{\Rightarrow} = \{S_i\,\sigma\,p\}$. Thus

    $$\begin{aligned}
    &(V \setminus \alpha^{\Leftarrow}) \cup \alpha^{\Rightarrow} \\
    &= (V \setminus (\{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\} \cup \{S_i\,\sigma\,p\})) \cup \{S_i\,\sigma\,p\} \\
    &= (V \cup \{S_i\,\sigma\,p\}) \setminus \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\} \\
    &= (V \cup \alpha^{\Rightarrow}) \setminus JS\,\sigma\,p^{\Leftarrow},
    \end{aligned}$$

    hence the result.
  - Finally, suppose $\alpha = JS\,\sigma\,p$ with $\sigma \in OBS^{+}$. Then $\alpha^{\Leftarrow} = \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\}$ and $\alpha^{\Rightarrow} = \{\sigma'\,\sigma\,p : \sigma' \in OBS^{+}\}$. We have seen with Proposition 2.3 that $\alpha^{\Leftarrow} \cap \alpha^{\Rightarrow} = \{\alpha\}$. Therefore, intuitively, removing causes then adding consequences of α is like adding consequences, then removing all causes of α except α itself. We have $\{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\} \setminus \{JS\,\sigma\,p\} = \{JS\,\sigma[k{:}]\,p : 1 \leq k \leq depth(\sigma)\}$, thus

    $$\begin{aligned}
    &(V \setminus \alpha^{\Leftarrow}) \cup \alpha^{\Rightarrow} \\
    &= (V \setminus \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\}) \cup \{\sigma'\,\sigma\,p : \sigma' \in OBS^{+}\} \\
    &= (V \cup \{\sigma'\,\sigma\,p : \sigma' \in OBS^{+}\}) \setminus \{JS\,\sigma[k{:}]\,p : 1 \leq k \leq depth(\sigma)\} \\
    &= (V \cup \alpha^{\Rightarrow}) \setminus JS\,\sigma[1{:}]\,p^{\Leftarrow},
    \end{aligned}$$

    hence the result.

Therefore all equivalences are valid. □

Lemma 2.4. *Suppose α and β are atoms such that $\alpha \Rightarrow_I \beta$ and $\alpha \neq \beta$: then $\alpha = JS\,\sigma\,p$ with p a propositional variable. Then we have the following program equivalences:*

$$+\alpha; +\beta \equiv +\beta; +\alpha \equiv +\alpha \qquad (+\Rightarrow+)$$
$$-\alpha; -\beta \equiv -\beta; -\alpha \equiv -\beta \qquad (-\Rightarrow-)$$
$$-\alpha; +\beta \equiv +\beta; -\alpha \qquad (-\Rightarrow+)$$
$$-\beta; +\alpha \equiv \begin{cases} +\alpha & \textit{if } \alpha = JS\,p \\ +\alpha; -JS\,\sigma[1{:}]\,p & \textit{otherwise} \end{cases} \qquad (-\Leftarrow+)$$

Proof. For every equivalence, take a valuation $V \in 2^{ATM}$ and suppose we execute the sequence of assignments from V. We assume that α is not introspectively valid (otherwise $-\alpha$ fails and the equivalences are trivially valid). Like for the previous proof, we use definitions of Section 2.2.2.

- $(+\Rightarrow+)$. Since $\alpha \Rightarrow_I \beta$, by Proposition 2.4, we have $\beta^{\Rightarrow} \subseteq \alpha^{\Rightarrow}$. Hence $(V \cup \alpha^{\Rightarrow}) \cup \beta^{\Rightarrow} = (V \cup \beta^{\Rightarrow}) \cup \alpha^{\Rightarrow} = V \cup \alpha^{\Rightarrow}$.

- $(-\Rightarrow-)$. Since $\alpha \Rightarrow_I \beta$, by Proposition 2.4, we have $\alpha^{\Leftarrow} \subseteq \beta^{\Leftarrow}$. Hence $(V \setminus \alpha^{\Leftarrow}) \setminus \beta^{\Leftarrow} = (V \setminus \beta^{\Leftarrow}) \setminus \alpha^{\Leftarrow} = V \setminus \beta^{\Leftarrow}$.

- $(-\Rightarrow+)$. To prove that $(V \setminus \alpha^{\Leftarrow}) \cup \beta^{\Rightarrow} = (V \cup \beta^{\Rightarrow}) \setminus \alpha^{\Leftarrow}$, we prove that, when $\alpha \Rightarrow_I \beta$ while $\alpha \neq \beta$, $\alpha^{\Leftarrow} \cap \beta^{\Rightarrow} = \emptyset$. Suppose there exists an atom γ such that $\gamma \in \alpha^{\Leftarrow}$ and $\gamma \in \beta^{\Rightarrow}$, i.e., $\gamma \Rightarrow_I \alpha$ and $\beta \Rightarrow_I \gamma$.

  Since $\alpha \Rightarrow_I \beta$ and by Proposition 2.4, the former implies that $\gamma \Rightarrow_I \beta$. With the latter and by Proposition 2.3, it implies that $\gamma = \beta$.

  On the other hand, again since $\alpha \Rightarrow_I \beta$ and by Proposition 2.4, the latter implies $\alpha \Rightarrow_I \gamma$. With the latter and by Proposition 2.3, it implies that $\alpha = \gamma$.

  Hence both hypothesis together imply that $\alpha = \beta$, which contradicts our hypothesis. Therefore $\alpha^{\Leftarrow} \cap \beta^{\Rightarrow} = \emptyset$ and $(V \setminus \alpha^{\Leftarrow}) \cup \beta^{\Rightarrow} = (V \cup \beta^{\Rightarrow}) \setminus \alpha^{\Leftarrow}$.

- $(-\Leftarrow+)$. We examine both cases:

  - Suppose $\alpha = JS\,p$. Then $\alpha^{\Rightarrow} = \{\sigma\,p : \sigma \in OBS^{+}\}$, i.e., any sequence of visibility operators of depth at least 1, followed by p. Since $\alpha \Rightarrow_I \beta$, β is of the form $\sigma\,p$ with $\sigma \in OBS^{+}$, that is, either of the form $S_i\,\sigma'\,p$ or of the form $JS\,\sigma'\,p$ with $\sigma' \in OBS^{*}$. In both cases, each cause of β is a consequence of α: $\beta^{\Leftarrow} \subseteq \alpha^{\Rightarrow}$. Hence $(V \setminus \beta^{\Leftarrow}) \cup \alpha^{\Rightarrow} = V \cup \alpha^{\Rightarrow}$.

– Now suppose $\alpha = JS\,\sigma\,p$ with $\sigma \in OBS^+$. Then $\alpha^{\Rightarrow} = \{\sigma'\,\sigma\,p : \sigma' \in OBS^+\}$. As a consequence of α, β is either of the form $S_i\,\sigma''\,\sigma\,p$ or $JS\,\sigma''\,\sigma\,p$ with $\sigma'' \in OBS^*$.

In the first case, $\beta^{\Leftarrow} = \{JS\,\sigma''[k{:}]\,\sigma\,p : 0 \leq k \leq depth(\sigma'')\} \cup \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\} \cup \{S_i\,\sigma''\,\sigma\,p\}$. The first and last parts are included in $\alpha^{\Rightarrow}$; the second part only intersects with $\alpha^{\Rightarrow}$ for $k = 0$.

In the second case, $\beta^{\Leftarrow} = \{JS\,\sigma''[k{:}]\,\sigma\,p : 0 \leq k \leq depth(\sigma'')\} \cup \{JS\,\sigma[k{:}]\,p : 0 \leq k \leq depth(\sigma)\}$. Almost like in the previous case, the first part is included in $\alpha^{\Rightarrow}$; the second part only intersects with $\alpha^{\Rightarrow}$ for $k = 0$.

Intuitively, removing causes of β then adding consequences of α is like adding consequences of α then removing all causes of β but the ones included in $\alpha^{\Rightarrow}$:

$$\begin{aligned}
&(V \setminus \beta^{\Leftarrow}) \cup \alpha^{\Rightarrow} \\
&= (V \cup \{\sigma'\,\sigma\,p : \sigma' \in OBS^+\}) \setminus \{JS\,\sigma[k{:}]\,p : 1 \leq k \leq depth(\sigma)\} \\
&= (V \cup \alpha^{\Rightarrow}) \setminus JS\,\sigma[1{:}]\,p^{\Leftarrow},
\end{aligned}$$

hence the result.

Therefore all equivalences are valid. □

Lemma 2.5. *We have the following program equivalences, where α and β are atoms such that $\alpha \not\Rightarrow_I \beta$ and $\beta \not\Rightarrow_I \alpha$:*

$$+\alpha; +\beta \equiv +\beta; +\alpha \qquad (+\not\Rightarrow+)$$
$$-\alpha; -\beta \equiv -\beta; -\alpha \qquad (-\not\Rightarrow-)$$
$$-\alpha; +\beta \equiv +\beta; -\alpha \qquad (-\not\Rightarrow+)$$

Proof. For every equivalence, take a valuation $V \in 2^{ATM}$ and suppose we execute the sequence of assignments from V. We assume that α is not introspectively valid (otherwise $-\alpha$ fails and the equivalences are trivially valid).

- $(+\not\Rightarrow+)$. Obvious: $(V \cup \alpha^{\Rightarrow}) \cup \beta^{\Rightarrow} = (V \cup \beta^{\Rightarrow}) \cup \alpha^{\Rightarrow}$ whatever $\alpha^{\Rightarrow}$ and $\beta^{\Rightarrow}$.
- $(-\not\Rightarrow-)$. Obvious again: $(V \setminus \alpha^{\Leftarrow}) \setminus \beta^{\Leftarrow} = (V \setminus \beta^{\Leftarrow}) \setminus \alpha^{\Leftarrow}$ whatever $\alpha^{\Leftarrow}$ and $\beta^{\Leftarrow}$.
- $(-\not\Rightarrow+)$. Since $\alpha \not\Rightarrow_I \beta$ and $\beta \not\Rightarrow_I \alpha$, we know by Proposition 2.5 that $\alpha^{\Leftarrow} \cap \beta^{\Rightarrow} = \emptyset$. Therefore $(V \setminus \alpha^{\Leftarrow}) \cup \beta^{\Rightarrow} = (V \cup \beta^{\Rightarrow}) \setminus \alpha^{\Leftarrow}$.

□

Proposition 2.11. *The following formulas are plainly valid, where α and β are atoms:*

$$[+\alpha]\beta \leftrightarrow \begin{cases} \top & \textit{if } \alpha \Rightarrow_I \beta \\ \beta & \textit{otherwise} \end{cases} \qquad (Red_{+\alpha})$$

$$[-\alpha]\beta \leftrightarrow \begin{cases} \top & \textit{if } \alpha \textit{ is valid in } INTR \\ \bot & \textit{if } \alpha \textit{ is not valid in } INTR \textit{ and } \beta \Rightarrow_I \alpha \\ \beta & \textit{otherwise} \end{cases} \qquad (Red_{-\alpha})$$

$$[+\alpha]\neg\varphi \leftrightarrow \neg[+\alpha]\varphi \qquad (Red_{+\alpha,\neg})$$

$$[-\alpha]\neg\varphi \leftrightarrow \begin{cases} \top & \textit{if } \alpha \textit{ is valid in } INTR \\ \neg[-\alpha]\varphi & \textit{otherwise} \end{cases} \qquad (Red_{-\alpha,\neg})$$

$$[+\alpha](\varphi \wedge \varphi') \leftrightarrow [+\alpha]\varphi \wedge [+\alpha]\varphi' \qquad (Red_{+\alpha,\wedge})$$

$$[-\alpha](\varphi \wedge \varphi') \leftrightarrow [-\alpha]\varphi \wedge [-\alpha]\varphi' \qquad (Red_{-\alpha,\wedge})$$

$$[\pi;\pi']\varphi \leftrightarrow [\pi][\pi']\varphi \qquad (Red_{;})$$

$$[\pi \sqcup \pi']\varphi \leftrightarrow [\pi]\varphi \wedge [\pi']\varphi \qquad (Red_{\sqcup})$$

$$[\pi^*]\varphi \leftrightarrow [\pi^{\leq 2^{2|ATM(\pi)|}}]\varphi \qquad (Red_{*})$$

$$[\varphi?]\varphi' \leftrightarrow \varphi \to \varphi' \qquad (Red_{?})$$

Proof. For each validity, consider an arbitrary valuation $V \in 2^{ATM}$. Some proofs are skipped due to their similarity with those of PDL [Fischer and Ladner, 1979] or DL-PA [Balbiani et al., 2013b].

- **($Red_{+\alpha}$).** This is an extension of the corresponding validity of DL-PA for propositional variables:

$$[+p]q \leftrightarrow \begin{cases} \top & \text{if } p = q \\ q & \text{otherwise} \end{cases}$$

  to our atoms. When α is put to true, all of its introspective consequences also are, hence the condition $\alpha \Rightarrow_I \beta$ replacing $p = q$.

- **($Red_{-\alpha}$).** This is again an extension of a DL-PA validity:

$$[-p]q \leftrightarrow \begin{cases} \bot & \text{if } p = q \\ q & \text{otherwise} \end{cases}$$

  to atoms. Here we need to deal again with introspective consequences, hence $\beta \Rightarrow_I \alpha$ instead of $p = q$, and also with introspectively valid atoms: trying to remove an atom valid in $INTR$ will lead to a failure of the program (and thus to the formula being true).

- (***Red***$_{+\alpha,\neg}$). Already true in DL-PA, this is due to the program $+\alpha$ always succeeding and being deterministic.

- (***Red***$_{-\alpha,\neg}$). Provided that α is not valid in $INTR$, the program $-\alpha$ always succeeds and is deterministic. If α is valid in $INTR$, trying to remove it will lead to a failure of the program (no related valuations).

- Proofs for ($Red_{+\alpha,\wedge}$), ($Red_{-\alpha,\wedge}$), ($Red_{;}$), ($Red_{\sqcup}$) and ($Red_{?}$) are standard.

- (***Red***$_{*}$). We establish that the program π^* actually leads to at most $2^{2|ATM(\pi)|}$ different valuations, and that it implies that it is sufficient to repeat π^* at most $2^{2|ATM(\pi)|}$ times.

  Like in DL-PA, each execution of a DEL-PAO program π^* consists in a sequence of assignments τ such that $ATM(\tau) \subseteq ATM(\pi^*) = ATM(\pi)$.

  Let us take such a sequence τ. Using axioms ($-=+$), ($-\Rightarrow+$), ($-\Leftarrow+$) and ($-\not\Rightarrow+$) of lemmas 2.3, 2.4 and 2.5, we can transform τ into a sequence τ' of the form $+\alpha_1;\ldots;+\alpha_p;-\beta_1;\ldots;-\beta_q$. These exchanges may create new atoms when applying ($-=+$) or ($-\Leftarrow+$); let us write them α_R for a given atom α. Formally:

$$\alpha_R = \begin{cases} JS\,\sigma\,p & \text{if } \alpha = S_i\,\sigma\,p \text{ with } \sigma \in OBS^* \\ JS\,\sigma[1{:}]\,p & \text{if } \alpha = JS\,\sigma\,p \text{ with } \sigma \in OBS^+. \end{cases}$$

  Note that $\alpha_R \Rightarrow_I \alpha$.

  Then we can transform τ' into another sequence τ'' of the form $+\alpha_1;\ldots;+\alpha_{p'};-\beta_1;\ldots;-\beta_{q'}$, where the ordering of positive assignments as well as the ordering of negative assignments does not matter and where $p', q' \leq |ATM(\pi)|$, using axioms ($+=+$), ($-=-$), ($+\Rightarrow+$), ($-\Rightarrow-$), ($+\not\Rightarrow+$) and ($-\not\Rightarrow-$) of lemmas 2.3, 2.4 and 2.5. These transformations only decrease the number of atoms, removing causes or consequences, but do not add new atoms. Since $\alpha_R \Rightarrow_I \alpha$, at most one of them will appear positively and at most one of them will appear negatively in the final transformation, hence the result $p', q' \leq |ATM(\pi)|$.

  This means that for any sequence of assignments of atoms from a set $ATM(\pi)$, there exists an equivalent sequence of assignments composed of at most $|ATM(\pi)|$ positive assignments, then at most $|ATM(\pi)|$ negative assignments, in which the ordering does not matter. There exist at most $2^{|ATM(\pi)|}$ such sequences of positive

assignments, and at most $2^{|ATM(\pi)|}$ such sequences of negative assignments, hence at most $2^{|ATM(\pi)|} \times 2^{|ATM(\pi)|} = 2^{2|ATM(\pi)|}$ different, in the sense that they are not equivalent, sequences of assignments of atoms from $ATM(\pi)$. This implies that the program π^* can lead to at most $2^{2|ATM(\pi)|}$ different valuations.

Now to prove $\pi^* \equiv \pi^{\leq 2^{2|ATM(\pi)|}}$, if suffices to prove that there is a $k \leq 2^{2|ATM(\pi)|}$ such that $\pi^{\leq k+1} \equiv \pi^{\leq k}$. We proceed by contradiction. Suppose there is no $k \leq 2^{2|ATM(\pi)|}$ such that $\pi^{\leq k+1} \equiv \pi^{\leq k}$. So there is a valuation V and a chain of strict inclusions

$$R_{\pi^{\leq 0}}(V) \subset R_{\pi^{\leq 1}}(V) \subset \ldots \subset R_{\pi^{\leq 2^{2|ATM(\pi)|}+1}}(V)$$

where $R_\pi(V) = \{V' : VR_\pi V'\}$. In other words, there is a sequence of valuations V_1, ..., $V_{2^{2|ATM(\pi)|}+1}$ such that $V_k \notin R_{\pi^{\leq k-1}}(V)$ and $V_k \in R_{\pi^{\leq k}}(V)$ for every $k \leq 2^{2|ATM(\pi)|} + 1$. Observe that these valuations are pairwise different (if $k \neq \ell$ then $V_k \neq V_\ell$) and that all of them are contained in $R_{\pi^*}(V)$. However, we have seen there can be at most $2^{2|ATM(\pi)|}$ such valuations, so we end up in a contradiction.

Therefore there is a $k \leq 2^{2|ATM(\pi)|}$ such that $\pi^{\leq k+1} \equiv \pi^{\leq k}$. If follows that $\pi^* \equiv \pi^{\leq k}$ and thus that $\pi^* \equiv \pi^{\leq 2^{2|ATM(\pi)|}}$ (by the definition of $\pi^{\leq k}$).

Therefore the given formulas are plainly valid. □

2.3.3 Soundness and completeness

The above equivalences can be applied anywhere in a formula because the inference rule of replacement of equivalents preserves validity.

Proposition 2.12. *Let φ' be obtained from φ by replacing some occurrence of ψ in φ by ψ'. If $\psi \leftrightarrow \psi'$ is plainly valid then $\varphi \leftrightarrow \varphi'$ is plainly valid.*

Proof. This is due to the fact that the following rules of inference for K_i, CK and $[\pi]$:

$$\frac{\varphi \leftrightarrow \varphi'}{K_i\varphi \leftrightarrow K_i\varphi'} \qquad \frac{\varphi \leftrightarrow \varphi'}{CK\varphi \leftrightarrow CK\varphi'} \qquad \frac{\varphi \leftrightarrow \varphi'}{[\pi]\varphi \leftrightarrow [\pi]\varphi'}$$

all preserve plain validity. □

Theorem 2.1. *For every* DEL-PAO *formula φ there exists a formula without modal operators φ' such that $\varphi \leftrightarrow \varphi'$ is plainly valid.*

Proof. We provide a procedure to remove modal operators.

Procedure 2.1. While there is a modal operator in φ:

1. if there is a sub-formula $K_i\varphi'$ such that φ' does not contain modal operators, put φ' in conjunctive normal form and eliminate K_i by applying equivalences ($Red_{K,\wedge}$), ($Red_{K,\vee}$), ($Red_{K,\alpha}$) and ($Red_{K,\neg\alpha}$) of Proposition 2.10;

2. if there is a sub-formula $CK\varphi'$ such that φ' does not contain modal operators, put φ' in conjunctive normal form and eliminate CK by applying equivalences ($Red_{CK,\wedge}$), ($Red_{CK,\vee}$), ($Red_{CK,\alpha}$) and ($Red_{CK,\neg\alpha}$) of Proposition 2.10;

3. if there is a sub-formula $[\pi]\varphi'$ such that φ' does not contain modal operators, eliminate $[\pi]$ by applying equivalences of Proposition 2.11.

So by iterating elimination of innermost modal operators we obtain a formula without modal operators. These transformations are possible thanks to the rule of replacement of equivalents that preserves plain validity. □

Proposition 2.13. *The following formulas are valid in $INTR$, where i is an agent and α is an atom:*

$$S_i\, S_i\, \alpha \qquad (Vis_{C1})$$
$$JS\ JS\ \alpha \qquad (Vis_{C2})$$
$$JS\, S_i\, S_i\, \alpha \qquad (Vis_{C3})$$
$$JS\ \alpha \rightarrow S_i\, \alpha \qquad (Vis_{C4})$$
$$JS\ \alpha \rightarrow JS\, S_i\, \alpha \qquad (Vis_{C5})$$

Proof. These formulas are the syntactical counterparts of the five constraints (C1)-(C5) of introspective valuations. □

Let us call $\mathcal{T}_{vis}$ the collection of the above formulas, i.e.:

$$\begin{aligned}\mathcal{T}_{vis} = &\{S_i\, S_i\, \alpha : i \in Agt, \alpha \in ATM\} \cup \{JS\ JS\ \alpha : \alpha \in ATM\}\\ &\cup \{JS\, S_i\, S_i\, \alpha : i \in Agt, \alpha \in ATM\} \cup \{JS\ \alpha \rightarrow S_i\, \alpha : i \in Agt, \alpha \in ATM\}\\ &\cup \{JS\ \alpha \rightarrow JS\, S_i\, \alpha : i \in Agt, \alpha \in ATM\}.\end{aligned}$$

Proposition 2.14. *For φ without modal operators, φ is valid in $INTR$ if and only if $\mathcal{T}_{vis} \models_{\mathsf{CPL}} \varphi$, where $\models_{\mathsf{CPL}}$ is logical consequence in classical propositional logic.*

Proof. The theory $\mathcal{T}_{vis}$ describes the constraints defining the set of introspective valuations $INTR$. □

Proposition 2.15. *For formulas φ without modal operators, $\vdash_{\mathsf{DEL\text{-}PAO}} \varphi$ if and only if $\mathcal{T}_{vis} \vdash_{\mathsf{CPL}} \varphi$.*

Proof. For the left-to-right direction, first, if φ is DEL-PAO-provable then φ is valid in $INTR$, since our axioms are sound and rules of inference preserve plain validity.

Moreover, if φ is valid in $INTR$ then φ is CPL-valid in the $\mathcal{T}_{vis}$ models. This is trivial because the models are isomorphic. (It is actually an equivalence.)

Then, if φ is CPL-valid in the $\mathcal{T}_{vis}$ models then φ is derivable from $\mathcal{T}_{vis}$ in CPL, by completeness of classical propositional logic. (It is again actually an equivalence.)

Thus, if φ is DEL-PAO-provable then φ is derivable from $\mathcal{T}_{vis}$ in CPL.

For the right-to-left direction, if φ is derivable from $\mathcal{T}_{vis}$ in CPL then φ is DEL-PAO-provable is obviously true because the axiomatics of DEL-PAO contains the elements of $\mathcal{T}_{vis}$ as schemas. □

The axiomatization of DEL-PAO is given by:

- the axioms of CPL (Classical Propositional Logic);
- the reduction axioms for epistemic operators given in Proposition 2.10;
- the reduction axioms for dynamic operators given in Proposition 2.11;
- the introspection axioms given in Proposition 2.13;
- the rule of Modus Ponens and the rules of inference for K_i, CK, and $[\pi]$:

$$\frac{\varphi \leftrightarrow \varphi'}{K_i\varphi \leftrightarrow K_i\varphi'} \qquad \frac{\varphi \leftrightarrow \varphi'}{CK\varphi \leftrightarrow CK\varphi'} \qquad \frac{\varphi \leftrightarrow \varphi'}{[\pi]\varphi \leftrightarrow [\pi]\varphi'}$$

Theorem 2.2. *The axiomatization of* DEL-PAO *is sound and complete w.r.t. the set of introspective valuations.*

Proof. Let φ be a DEL-PAO formula. Let φ' be its reduction to a boolean formula.

To prove soundness, suppose φ is a theorem of DEL-PAO. By Theorem 2.1, φ' is a theorem of DEL-PAO, too. By Proposition 2.15 we have $\mathcal{T}_{vis} \vdash_{\mathsf{CPL}} \varphi'$. Then $\mathcal{T}_{vis} \models_{\mathsf{CPL}} \varphi'$ by soundness of classical propositional logic. Hence φ' is valid in $INTR$ by Proposition 2.14. Finally, φ is valid in $INTR$ because all the equivalences used in Theorem 2.1 are valid in $INTR$.

The completeness proof follows the lines of the soundness proof in reverse order, resorting to completeness of classical logic instead of soundness. □

2.4 Complexity of model checking

Given an introspective valuation V and a formula φ, we are interested in deciding whether $V \models \varphi$. An issue here is that introspective valuations are always infinite. We show that a finite number of atoms are actually necessary to interpret a formula φ, even when it contains epistemic operators. These operators can be reduced to programs if enough atoms are involved in the valuation.[4]

2.4.1 From infinite to finite models

The valuations that we are interested in DEL-PAO, i.e., the introspective valuations, are always infinite. However, the model checking problem must be defined on finite models. To this end, we define a notion of "introspective enough" valuations.

A valuation V is *introspective w.r.t. a set of atoms* A if for every atom $\alpha \in A$:

1. if α is valid in $INTR$ then $\alpha \in V$;
2. if there exists $\beta \in V$ such that $\beta \Rightarrow_I \alpha$ then $\alpha \in V$;
3. if there exists $\beta \notin V$ such that $\alpha \Rightarrow_I \beta$ then $\alpha \notin V$.

Observe that if A is finite, given an introspective valuation V, it is always possible to find a finite valuation which is introspective w.r.t. A, the simplest one being $V \cap A$. Indeed, since V is introspective:

1. every introspectively valid atom belongs to V, thus every introspectively valid atom from A belongs to $V \cap A$;
2. for every $\beta \in V$, every introspective consequence of β is in V, thus every introspective consequence of β from A is in $V \cap A$;
3. for every $\beta \notin V$, no introspective cause of β is in V, thus no introspective cause of β is in $V \cap A$.

Obviously, if V is introspective w.r.t. A, then it is introspective w.r.t. B if $B \subseteq A$. A "fully" introspective valuation $V \in INTR$ is introspective w.r.t. ATM; therefore it is also introspective w.r.t. any subset of atoms.

[4] This section was expanded compared to [Herzig et al., 2015] with a more precise definition of the problem.

2.4.2 Simulating epistemic operators with programs

Let us define the following programs:

$$\mathsf{varyIfNotSeen}(i, \alpha) = \textbf{if } \neg S_i\, \alpha \textbf{ then } (+\alpha \sqcup -\alpha)$$
$$\mathsf{varyIfNotSeen}(Agt, \alpha) = \textbf{if } \neg JS\, \alpha \textbf{ then } (+\alpha \sqcup -\alpha)$$

As their names suggest, $\mathsf{varyIfNotSeen}(i, \alpha)$ checks whether i sees α, and non-deterministically varies the truth value of α otherwise; $\mathsf{varyIfNotSeen}(Agt, \alpha)$ behaves similarly for joint visibility. We extend them to a set of atoms $A = \{\alpha_1, \ldots, \alpha_m\}$:

$$\mathsf{varyIfNotSeen}(i, A) = \mathsf{varyIfNotSeen}(i, \alpha_1); \ldots; \mathsf{varyIfNotSeen}(i, \alpha_m)$$
$$\mathsf{varyIfNotSeen}(Agt, A) = \mathsf{varyIfNotSeen}(Agt, \alpha_1); \ldots; \mathsf{varyIfNotSeen}(Agt, \alpha_m)$$

where we suppose that both programs are $skip$ if A is empty.

We do not impose any ordering on sub-programs; however, one could argue that such ordering is important. For example, suppose we execute $\mathsf{varyIfNotSeen}(i, JS\, p); \mathsf{varyIfNotSeen}(i, p)$. The first program, **if** $\neg S_i\, JS\, p$ **then** $(+JS\, p \sqcup -JS\, p)$, might put $JS\, p$ to true, thus making $S_i\, p$ true (by introspective consequence) in the process. This may change the execution of the second program, **if** $\neg S_i\, p$ **then** $(+p \sqcup -p)$, if $S_i\, p$ was previously false. However, note that $S_i\, JS\, p$ is introspectively valid: on introspective valuations, $\mathsf{varyIfNotSeen}(i, JS\, p)$ will actually never change the value of $JS\, p$. This can be generalized.

Lemma 2.6. *Let V be a valuation, i an agent and α and β two different atoms. Then:*

1. *if V is introspective w.r.t. $\{S_i\, \beta\}$, then for every valuation V_1 such that $V R_{\mathsf{varyIfNotSeen}(i,\beta)} V_1$, we have $V \models S_i\, \alpha$ if and only if $V_1 \models S_i\, \alpha$;*
2. *if V is introspective w.r.t. $\{JS\, \alpha, JS\, \beta\}$, then for every valuation V_2 such that $V R_{\mathsf{varyIfNotSeen}(Agt,\beta)} V_2$, we have $V \models JS\, \alpha$ if and only if $V_2 \models JS\, \alpha$.*

Proof. We detail both items, since the proof for joint visibility is slightly different from the one for individual visibility here. Remember that by applying the programs $+\alpha$ and $-\alpha$ to V we respectively obtain $V{+}\alpha$ and $V{-}\alpha$.

1. Recall that $\mathsf{varyIfNotSeen}(i, \beta) =$ **if** $\neg S_i\, \beta$ **then** $(+\beta \sqcup -\beta)$. We distinguish two cases.
   - Suppose $S_i\, \beta \in V$. Then the program is equivalent to $skip$ and thus $V = V_1$, making the result obvious.

- Now suppose $S_i\,\beta \notin V$. Then $\mathsf{varyIfNotSeen}(i, \beta)$ is equivalent to $+\beta \sqcup -\beta$ and thus $V_1 = V{+}\beta$ or $V_1 = V{-}\beta$. Note that β cannot be valid in $INTR$, otherwise $S_i\,\beta$ would also be valid in $INTR$ and belong to V since V is introspective w.r.t. $S_i\,\beta$. Thus $-\beta$ cannot fail and $V{-}\beta$ exists.

  We have to prove that $V \models S_i\,\alpha$ if and only if $V{+}\beta \models S_i\,\alpha$ and $V \models S_i\,\alpha$ if and only if $V{-}\beta \models S_i\,\alpha$. We again split into two cases.

  - $V{+}\beta \models S_i\,\alpha$ if and only if $S_i\,\alpha \in V{+}\beta$, i.e., $S_i\,\alpha \in V$ or $\beta \Rightarrow_I S_i\,\alpha$. Recall that $S_i\,\beta$ must not be valid in $INTR$. Suppose $\beta \Rightarrow_I S_i\,\alpha$. Thus either $\beta = S_i\,\alpha$ and $S_i\,\beta = S_i\,S_i\,\alpha$ which is valid in $INTR$, or β is of the form $JS\,\beta'$ and $S_i\,\beta = S_i\,JS\,\beta'$ which is also valid in $INTR$. This implies that $\beta \not\Rightarrow_I S_i\,\alpha$. Therefore $V{+}\beta \models S_i\,\alpha$ is equivalent to $S_i\,\alpha \in V$.
  - $V{-}\beta \models S_i\,\alpha$ if and only if $S_i\,\alpha \in V{-}\beta$, i.e., $S_i\,\alpha \in V$ and $S_i\,\alpha \not\Rightarrow_I \beta$. This time suppose $S_i\,\alpha \Rightarrow_I \beta$. Then since the only introspective consequence of $S_i\,\alpha$ is itself, $S_i\,\beta = S_i\,S_i\,\alpha$, which is introspectively valid. This is impossible since $S_i\,\beta \notin V$. This implies that $S_i\,\alpha \not\Rightarrow_I \beta$. Therefore $V{-}\beta \models S_i\,\alpha$ is equivalent to $S_i\,\alpha \in V$.

  In all cases, $V \models S_i\,\alpha$ if and only if $V_1 \models S_i\,\alpha$ for every V_1 such that $V\,R_{\mathsf{varyIfNotSeen}(i,\beta)}\,V_1$.

2. For the second case, remember that $\mathsf{varyIfNotSeen}(Agt, \beta) = \textbf{if}\ \neg JS\,\beta\ \textbf{then}\ (+\beta \sqcup -\beta)$.

   - Suppose $JS\,\beta \in V$, then again the result is obvious.
   - Now suppose $JS\,\beta \notin V$. This time we prove $V \models JS\,\alpha$ if and only if $V{+}\beta \models JS\,\alpha$ and $V \models JS\,\alpha$ if and only if $V{-}\beta \models JS\,\alpha$. The latter exists because β is not valid in $INTR$ (otherwise $JS\,\beta$ would belong to V since V is introspective w.r.t. $JS\,\beta$).
     - The first case is analogous to the one for individual visibility: $\beta \Rightarrow_I JS\,\alpha$ implies that either $\beta = JS\,\alpha$ or β is of the form $JS\,\beta'$. In both cases, $JS\,\beta$ is introspectively valid, which leads to a contradiction. Therefore $V{+}\beta \models JS\,\alpha$ if and only if $V \models JS\,\alpha$.
     - The second case is trickier. We have $V{-}\beta \models JS\,\alpha$ if and only if $JS\,\alpha \in V{-}\beta$, i.e., $JS\,\alpha \in V$ and $JS\,\alpha \not\Rightarrow_I \beta$.
       Suppose $V{-}\beta \models JS\,\alpha$. Then obviously $JS\,\alpha \in V$.
       Now suppose $V{-}\beta \models \neg JS\,\alpha$. Then $JS\,\alpha \notin V$ or $JS\,\alpha \Rightarrow_I \beta$. The latter implies that β is of the form $\sigma\,\alpha$, with $\sigma \in OBS^+$. Hence $JS\,\alpha \Rightarrow_I JS\,\beta$. Since we assumed that $JS\,\beta \notin V$ and because V is introspective w.r.t. $JS\,\alpha$, we get $JS\,\alpha \notin$

V. Thus $JS\,\alpha \Rightarrow_I \beta$ implies that $JS\,\alpha \notin V$, that is, $V - \beta \models \neg JS\,\alpha$ implies that $JS\,\alpha \notin V$.
Therefore $V - \beta \models JS\,\alpha$ if and only if $V \models JS\,\alpha$.

In all cases, $V \models JS\,\alpha$ if and only if $V_2 \models JS\,\alpha$ for every V_2 such that $V R_{\mathsf{varyIfNotSeen}(i,\beta)} V_2$.

Therefore both items are true. □

Proposition 2.16. *Let V be a valuation, i an agent and α and β two different atoms. Then:*

1. *if V is introspective w.r.t. $\{S_i\,\alpha, S_i\,\beta\}$, then for every valuation V_1, $V R_{\mathsf{varyIfNotSeen}(i,\alpha);\mathsf{varyIfNotSeen}(i,\beta)} V_1$ if and only if $V R_{\mathsf{varyIfNotSeen}(i,\beta);\mathsf{varyIfNotSeen}(i,\alpha)} V_1$;*
2. *if V is introspective w.r.t. $\{JS\,\alpha, JS\,\beta\}$, then for every valuation V_2, $V R_{\mathsf{varyIfNotSeen}(Agt,\alpha);\mathsf{varyIfNotSeen}(Agt,\beta)} V_2$ if and only if $V R_{\mathsf{varyIfNotSeen}(Agt,\beta);\mathsf{varyIfNotSeen}(Agt,\alpha)} V_2$.*

Proof. We only detail the proof of the first item since the proof for the second is analogous.

Recall that $\mathsf{varyIfNotSeen}(i,\alpha) = \textbf{if } \neg S_i\,\alpha \textbf{ then } (+\alpha \sqcup -\alpha)$. We distinguish three cases.

- Suppose $S_i\,\alpha \in V$. Then $\mathsf{varyIfNotSeen}(i,\alpha)$ is reduced to $skip$ and $V R_{\mathsf{varyIfNotSeen}(i,\alpha);\mathsf{varyIfNotSeen}(i,\beta)} V_1$ if and only if $V R_{\mathsf{varyIfNotSeen}(i,\beta)} V_1$. Moreover, by Lemma 2.6, since V is introspective w.r.t. $S_i\,\beta$, the value of $S_i\,\alpha$ is not modified by the execution of $\mathsf{varyIfNotSeen}(i,\beta)$. Therefore we also have $V R_{\mathsf{varyIfNotSeen}(i,\beta);\mathsf{varyIfNotSeen}(i,\alpha)} V_1$ if and only if $V R_{\mathsf{varyIfNotSeen}(i,\beta)} V_1$, hence the result.
- Now suppose $S_i\,\beta \in V$. The reasoning is similar and both sides are reduced to $\mathsf{varyIfNotSeen}(i,\alpha)$ due to V being introspective w.r.t. $S_i\,\alpha$.
- Finally, suppose $S_i\,\alpha \notin V$ and $S_i\,\beta \notin V$. Again by Lemma 2.6, since V is introspective w.r.t. $S_i\,\alpha$ and $S_i\,\beta$, the values of $S_i\,\alpha$ and $S_i\,\beta$ are not modified by $\mathsf{varyIfNotSeen}(i,\beta)$ and $\mathsf{varyIfNotSeen}(i,\alpha)$. This implies that $V R_{\mathsf{varyIfNotSeen}(i,\alpha);\mathsf{varyIfNotSeen}(i,\beta)} V_1$ if and only if $V R_{(+\alpha\sqcup-\alpha);(+\beta\sqcup-\beta)} V_1$, and $V R_{\mathsf{varyIfNotSeen}(i,\beta);\mathsf{varyIfNotSeen}(i,\alpha)} V_1$ if and only if $V R_{(+\beta\sqcup-\beta);(+\alpha\sqcup-\alpha)} V_1$. Using distributivity properties, the first program becomes $(+\alpha;+\beta) \sqcup (+\alpha;-\beta) \sqcup (-\alpha;+\beta) \sqcup (-\alpha;-\beta)$ and the second $(+\beta;+\alpha) \sqcup (+\beta;-\alpha) \sqcup (-\beta;+\alpha) \sqcup (-\beta;-\alpha)$. We prove that these two are equivalent by examining equivalences between sub-programs. Observe that since $S_i\,\alpha \notin V$ and $S_i\,\beta \notin V$, and again because V is introspective w.r.t. $S_i\,\alpha$ and $S_i\,\beta$, α and β are not introspectively valid, thus $-\alpha$ and $-\beta$ cannot fail.

1. We have seen with lemmas 2.4 and 2.5 that $+\alpha;+\beta \equiv +\beta;+\alpha$.
2. If $\alpha \Rightarrow_I \beta$, and since α and β are different, then α is of the form $JS\,\alpha'$. Thus $S_i\,\alpha = S_i\,JS\,\alpha'$, which is introspectively valid. This leads to a contradiction with our hypothesis that $S_i\,\alpha \notin V$ since V is introspective w.r.t. $S_i\,\alpha$. Therefore $\alpha \not\Rightarrow_I \beta$. Similarly, we have $\beta \not\Rightarrow_I \alpha$. By Lemma 2.5, we know that in this case, $+\alpha;-\beta \equiv -\beta;+\alpha$.
3. Similarly, $-\alpha;+\beta \equiv +\beta;-\alpha$ by Lemma 2.5.
4. We have seen with lemmas 2.4 and 2.5 that $-\alpha;-\beta \equiv -\beta;-\alpha$.

Therefore $(+\alpha;+\beta) \sqcup (+\alpha;-\beta) \sqcup (-\alpha;+\beta) \sqcup (-\alpha;-\beta)$ is equivalent to $(+\beta;+\alpha) \sqcup (+\beta;-\alpha) \sqcup (-\beta;+\alpha) \sqcup (-\beta;-\alpha)$.

In all cases, we have that $VR_{\mathsf{varyIfNotSeen}(i,\alpha);\mathsf{varyIfNotSeen}(i,\beta)}V_1$ if and only if $VR_{\mathsf{varyIfNotSeen}(i,\beta);\mathsf{varyIfNotSeen}(i,\alpha)}V_1$. In the general case, the ordering of sub-programs in $\mathsf{varyIfNotSeen}(i, A)$ is irrelevant whenever V is introspective w.r.t. $\{S_i\,\alpha : \alpha \in A\}$. □

The programs $\mathsf{varyIfNotSeen}(i, A)$ and $\mathsf{varyIfNotSeen}(Agt, A)$ will simulate the behaviour of K_i and CK.

Proposition 2.17. *Let V be a valuation, i an agent and φ a formula without epistemic operators. Then*

1. *if V is introspective w.r.t. $\{S_i\,\alpha : \alpha \in ATM(\varphi)\} \cup ATM(\varphi)$, then $V \models K_i\varphi \leftrightarrow [\mathsf{varyIfNotSeen}(i, ATM(\varphi))]\varphi$;*
2. *if V is introspective w.r.t. $\{JS\,\alpha : \alpha \in ATM(\varphi)\} \cup ATM(\varphi)$, then $V \models CK\varphi \leftrightarrow [\mathsf{varyIfNotSeen}(Agt, ATM(\varphi))]\varphi$.*

Proof. Again we only describe the case of individual knowledge; the proof for common knowledge is analogous.

We are going to compare relations associated to the modalities K_i and $[\mathsf{varyIfNotSeen}(i, ATM(\varphi))]$, i.e., the valuations related with V by $\sim_i$ and by $R_{\mathsf{varyIfNotSeen}(i,ATM(\varphi))}$.

First of all, since φ does not contain epistemic operators, remember that by Proposition 2.9, only atoms from $ATM(\varphi)$ are important when evaluating φ; the other atoms may vary without influencing its truth value. Therefore we will only take into account these atoms when describing relations.

The definition of $\sim_i$, restricted to atoms from φ, goes as follows:

for every $\alpha \in ATM(\varphi)$,
  if $S_i\,\alpha \in V$, then
    for every V_1 such that $V \sim_i V_1, V(\alpha) = V_1(\alpha)$ and (2.5)
  if $S_i\,\alpha \notin V$, then
    there is V_1 such that $V \sim_i V_1$ and $\alpha \in V_1$ and (2.6)
    there is V_1 such that $V \sim_i V_1$ and $\alpha \notin V_1$. (2.7)

On the other hand, the relation $R_{\mathsf{varyIfNotSeen}(i,ATM(\varphi))}$ is slightly more complicated. Remember that by Proposition 2.16, the ordering of sub-programs in $\mathsf{varyIfNotSeen}(i, ATM(\varphi))$ does not matter since V is introspective w.r.t. $\{S_i\,\alpha : \alpha \in ATM(\varphi)\}$. In other words, each sub-program $\mathsf{varyIfNotSeen}(i, \alpha)$ will have the same behaviour whenever it is executed. In particular, we have seen that truth values of conditions stay the same, and therefore are identical to the ones of the initial state.

Remember that $\mathsf{varyIfNotSeen}(i, \alpha) = \textbf{if } \neg S_i\,\alpha \textbf{ then } (+\alpha \sqcup -\alpha)$. The program $+\alpha$ adds α and all its introspective consequences to the current valuation, while $-\alpha$ removes α and all its introspective causes. In other words, each $\alpha \in ATM(\varphi)$, if not seen, may vary, but its causes or consequences may also vary. Let us abbreviate $\mathsf{varyIfNotSeen}(i, ATM(\varphi))$ by vins. We can describe the behaviour of the program as follows:

for every $\alpha \in ATM(\varphi)$,
  if $S_i\,\alpha \in V$, then
    if there exists $\beta \in ATM(\varphi)$ such that $S_i\,\beta \notin V$ and $\beta \Rightarrow_I \alpha$,
      then there is V_2 such that $V R_{\mathsf{vins}} V_2$ and $\alpha \in V_2$ and (2.8)
    if there exists $\beta \in ATM(\varphi)$ such that $S_i\,\beta \notin V$ and $\alpha \Rightarrow_I \beta$,
      then there is V_2 such that $V R_{\mathsf{vins}} V_2$ and $\alpha \notin V_2$ and (2.9)
    there is V_2 such that $V R_{\mathsf{vins}} V_2$ and $V(\alpha) = V_2(\alpha)$ and (2.10)
  if $S_i\,\alpha \notin V$, then
    there is V_2 such that $V R_{\mathsf{vins}} V_2$ and $\alpha \in V_2$ and (2.11)
    there is V_2 such that $V R_{\mathsf{vins}} V_2$ and $\alpha \notin V_2$. (2.12)

In words, if α is seen by agent i, it will not be modified "directly" by the program. Indeed, if $S_i\,\alpha$ is true the sub-program $\mathsf{varyIfNotSeen}(i, \alpha)$ will be equivalent to *skip*. However, another atom β appearing in φ and not seen by i might vary α, making it true with $+\beta$ if α is a consequence of β (case (2.8)) or false with $-\beta$ if β is a consequence of α (case (2.9)). Finally, there always exists a state where α was not modified, either because there is no β or because it was put to its initial value (case (2.10)). Otherwise, when α is not seen by i, it is directly modified by the

program (cases (2.11) and (2.12)). Observe that since V is introspective w.r.t. $\{S_i\,\alpha : \alpha \in ATM(\varphi)\}$, α is not valid in $INTR$, otherwise $S_i\,\alpha$ would belong to V. Hence it can be removed safely.

So it seems that atoms not seen or not in φ may vary while they should not. However, we are going to see that this does not happen. Take an arbitrary atom α such that $\alpha \in ATM(\varphi)$ and $S_i\,\alpha \in V$, and an arbitrary β such that $\beta \in ATM(\varphi)$ and $S_i\,\beta \notin V$. By definition, β is different from α.

- First, we consider case (2.8): suppose $\beta \Rightarrow_I \alpha$. Since they are different, β must be of the form $JS\,\beta'$. Thus $S_i\,\beta = S_i\,JS\,\beta'$, which is introspectively valid. Therefore $S_i\,\beta$ belongs to V because V is introspective w.r.t. $\{S_i\,\alpha : \alpha \in ATM(\varphi)\}$: we get a contradiction.
- Now for case (2.9): suppose $\alpha \Rightarrow_I \beta$. This time, α is of the form $JS\,\alpha'$, and $\beta = \sigma\,\alpha'$, with $\sigma \in OBS^+$. Thus $S_i\,\beta$ is also an introspective consequence of α. Since $S_i\,\beta \notin V$, then $\alpha \notin V$, otherwise V would not be introspective w.r.t. $ATM(\varphi)$. Thus $\alpha \notin V_2$ is the same as $V(\alpha) = V_2(\alpha)$.

In the end, case (2.8) cannot happen and case (2.9) reduces to case (2.10). We obtain:

for every $\alpha \in ATM(\varphi)$,
  if $S_i\,\alpha \in V$, then
    for every V_2 such that $V R_{\mathsf{vins}} V_2, V(\alpha) = V_2(\alpha)$ and (2.10')
  if $S_i\,\alpha \notin V$, then
    there is V_2 such that $V R_{\mathsf{vins}} V_2$ and $\alpha \in V_2$ and (2.11)
    there is V_2 such that $V R_{\mathsf{vins}} V_2$ and $\alpha \notin V_2$. (2.12)

Hence $\sim_i = R_{\mathsf{varyIfNotSeen}(i, ATM(\varphi))}$ on valuations introspective w.r.t. $\{S_i\,\alpha : \alpha \in ATM(\varphi)\} \cup ATM(\varphi)$ for atoms that are relevant to the truth value of φ. Therefore $V \models K_i\varphi \leftrightarrow [\mathsf{varyIfNotSeen}(i, ATM(\varphi))]\varphi$ if V is introspective w.r.t. $\{S_i\,\alpha : \alpha \in ATM(\varphi)\} \cup ATM(\varphi)$. □

Proposition 2.17 can be turned into a procedure eliminating epistemic operators: it suffices to iterate the application of the equivalences, starting with the innermost operators. These transformations are possible thanks to the rule of replacement of equivalents that preserves plain validity (see Proposition 2.12).

Procedure 2.2. While there is an epistemic operator in φ:

1. if there is a sub-formula $K_i\varphi'$ such that φ' does not contain epistemic operators, replace $K_i\varphi'$ by $[\mathsf{varyIfNotSeen}(i, ATM(\varphi))]\varphi'$;
2. if there is a sub-formula $CK\varphi'$ such that φ' does not contain epistemic operators, replace $CK\varphi'$ by $[\mathsf{varyIfNotSeen}(Agt, ATM(\varphi))]\varphi$.

2.4.3 Relevant atoms

We have seen in Proposition 2.17 that to reduce epistemic operators to programs, we needed some specific atoms to be true or false in the current world. In this section, we extend the definition of $ATM(\varphi)$ to include such atoms. We note this new set of atoms $RATM(\varphi)$, the "relevant atoms" of φ. Formally:

$$RATM(K_i\varphi) = RATM(\varphi) \cup \{S_i\,\alpha : \alpha \in RATM(\varphi)\}$$
$$RATM(CK\varphi) = RATM(\varphi) \cup \{JS\,\alpha : \alpha \in RATM(\varphi)\}$$

and equal to $ATM(\varphi)$ otherwise. So $RATM(\varphi)$ includes $ATM(\varphi)$. For instance:

$$ATM(q \wedge CK\,K_i p) = \{q, p\}$$
$$RATM(q \wedge CK\,K_i p) = \{q, p, JS\,p, S_i\,p, JS\,S_i\,p\}$$

Moreover, for a formula φ without epistemic operators, $RATM(\varphi) = ATM(\varphi)$. Note that while $RATM(\varphi)$ is finite, its cardinality can be exponential in the length (the number of symbols) of φ: for example, the cardinality of $RATM(K_{i_1} \dots K_{i_m} p)$ is in 2^m.

These relevant atoms of a formula φ are exactly the atoms appearing in the reduction of φ given by Procedure 2.2: starting with the innermost, each replacement of K_i or CK by $[\mathsf{varyIfNotSeen}(i, ATM(\varphi))]$ or $[\mathsf{varyIfNotSeen}(Agt, ATM(\varphi))]$ makes $S_i\,\alpha$ or $JS\,\alpha$ appear, for every α in the formula following the operator. Moreover, by Proposition 2.17, this reduction is equivalent to the original formula φ on any valuation introspective w.r.t. $RATM(\varphi)$.

Proposition 2.18. *Let φ be a formula. Let V, $V' \in 2^{ATM}$ such that V and V' are introspective w.r.t. $RATM(\varphi)$ and $V =_{RATM(\varphi)} V'$. Then $V \models \varphi$ if and only if $V' \models \varphi$.*

Proof. Let $(\varphi)_R$ be the formula obtained by applying Procedure 2.2 to φ. We know that φ is equivalent to $(\varphi)_R$ on valuations introspective w.r.t. $RATM(\varphi)$ and that $RATM(\varphi) = ATM((\varphi)_R)$.

We have:

$$\begin{aligned}
& V \models \varphi \\
\Leftrightarrow\ & V \models (\varphi)_R && \text{since } V \text{ is introspective w.r.t. } RATM(\varphi), \\
\Leftrightarrow\ & V' \models (\varphi)_R && \text{by Proposition 2.9, since } V =_{ATM((\varphi)_R)} V', \\
\Leftrightarrow\ & V' \models \varphi && \text{since } V' \text{ is introspective w.r.t. } RATM(\varphi).
\end{aligned}$$

Therefore $V \models \varphi$ if and only if $V' \models \varphi$. □

Proposition 2.18 implies that if $V \in INTR$, then $V \models \varphi$ if and only if $V \cap RATM(\varphi) \models \varphi$ for any φ, since we have seen that $V \cap RATM(\varphi)$ is introspective w.r.t. $RATM(\varphi)$ if V is introspective.

Before finally defining the model checking problem, we give some properties of relevant atoms that will be useful in the next chapter.

Proposition 2.19. *The equivalence*

$$K_{i_1} \dots K_{i_m}\alpha \leftrightarrow \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} \beta \Big)$$

is plainly valid, for $m \geq 0$.

Proof. We prove it by induction on m.

If $m = 0$, then $RATM(\alpha) = \alpha$ and the equivalence is obvious.

For general m, suppose $K_{i_1} \dots K_{i_m}\alpha \leftrightarrow \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} \beta \Big)$ is plainly valid. Take a valuation $V \in 2^{ATM}$ and an agent ℓ. Then:

$$\begin{aligned}
& V \models K_\ell K_{i_1} \dots K_{i_m}\alpha \\
\Leftrightarrow\ & V \models K_\ell \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} \beta \Big) \\
\Leftrightarrow\ & V \models \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} K_\ell \beta \Big) \\
\Leftrightarrow\ & V \models \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} (\beta \wedge S_\ell\, \beta) \Big) \\
\Leftrightarrow\ & V \models \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} \beta \Big) \wedge \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} S_\ell\, \beta \Big) \\
\Leftrightarrow\ & V \models \Big(\bigwedge_{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha)} \beta \Big) \wedge \Big(\bigwedge_{\beta \in \{S_\ell\, \gamma : \gamma \in RATM(K_{i_1} \dots K_{i_m}\alpha)\}} \beta \Big) \\
\Leftrightarrow\ & V \models \Big(\bigwedge_{\substack{\beta \in RATM(K_{i_1} \dots K_{i_m}\alpha) \cup \\ \{S_\ell\, \gamma : \gamma \in RATM(K_{i_1} \dots K_{i_m}\alpha)\}}} \beta \Big) \\
\Leftrightarrow\ & V \models \Big(\bigwedge_{\beta \in RATM(K_\ell K_{i_1} \dots K_{i_m}\alpha)} \beta \Big),
\end{aligned}$$

hence the result. □

We note $\langle r_1, \dots, r_p \rangle \sqsubseteq \langle 1, \dots, m \rangle$ for $\langle r_1, \dots, r_p \rangle$ a subset of $\{1, \dots, m\}$ in the same order. For example, we have $\langle 1, 3, 4 \rangle \sqsubseteq \langle 1, 2, 3, 4, 5 \rangle$ or $\langle\rangle \sqsubseteq \langle 1, 2, 3, 4, 5 \rangle$ but not $\langle 1, 4, 3 \rangle \sqsubseteq \langle 1, 2, 3, 4, 5 \rangle$. The next proposition characterizes the set of relevant atoms of a formula of the type $K_{i_1} \dots K_{i_m}\alpha$.

Proposition 2.20. *We have, for* $m \geq 0$*:*

$$RATM(K_{i_1} \dots K_{i_m} \alpha) = \{S_{i_{r_1}} \dots S_{i_{r_p}} \alpha : \langle r_1, \dots, r_p \rangle \sqsubseteq \langle 1, \dots, m \rangle\}$$

Proof. We prove it by induction on m.

The case where $m = 0$ is obvious since $RATM(\alpha) = \{\alpha\}$ and the only subset of $\langle\rangle$ is $\langle\rangle$.

For general m, suppose

$$RATM(K_{i_1} \dots K_{i_m} \alpha) = \{S_{i_{r_1}} \dots S_{i_{r_p}} \alpha : \langle r_1, \dots, r_p \rangle \sqsubseteq \langle 1, \dots, m \rangle\}.$$

Then

$$\begin{aligned}
& RATM(K_\ell K_{i_1} \dots K_{i_m} \alpha) \\
= {} & RATM(K_{i_1} \dots K_{i_m} \alpha) \cup \{S_\ell \, \alpha' : \alpha' \in RATM(K_{i_1} \dots K_{i_m} \alpha)\} \\
= {} & \{S_{i_{r_1}} \dots S_{i_{r_p}} \alpha : \langle r_1, \dots, r_p \rangle \sqsubseteq \langle 1, \dots, m \rangle\} \cup \\
& \qquad \{S_\ell \, \alpha' : \alpha' \in \{S_{i_{r_1}} \dots S_{i_{r_p}} \alpha : \langle r_1, \dots, r_p \rangle \sqsubseteq \langle 1, \dots, m \rangle\}\} \\
= {} & \{S_{i_{r_1}} \dots S_{i_{r_p}} \alpha : \langle r_1, \dots, r_p \rangle \sqsubseteq \langle 1, \dots, m \rangle\} \cup \\
& \qquad \{S_\ell \, S_{i_{r_1}} \dots S_{i_{r_p}} \alpha : \langle r_1, \dots, r_p \rangle \sqsubseteq \langle 1, \dots, m \rangle\} \\
= {} & \{S_{i_{r_1}} \dots S_{i_{r_p}} \alpha : \langle r_1, \dots, r_p \rangle \sqsubseteq \langle \ell, 1, \dots, m \rangle\},
\end{aligned}$$

hence the result. □

2.4.4 The model checking problem

The model checking problem for DEL-PAO is defined as follows:

- **Input:** a couple $\langle V \cap RATM(\varphi), \varphi \rangle$ where φ is a DEL-PAO formula and V is an introspective valuation;
- **Output:** yes if $V \models \varphi$, no otherwise.

As seen with Proposition 2.18, if V is introspective, an eligible valuation to perform model checking on is $V \cap RATM(\varphi)$. While $RATM(\varphi)$ may be exponential in the size of φ, we will see in Chapter 7 that this does not impact the complexity result.

Theorem 2.3. *The* DEL-PAO *model checking problem is* PSPACE*-complete.*

Proof. The model checking of DL-PA, which is a fragment of DEL-PAO without visibility operators S_i and JS was proven to be PSPACE-hard [Balbiani et al., 2014]. This establishes the lower bound.

We will show in Chapter 7 that the problem is in PSPACE. □

2.5 Applications

In this section we propose the Byzantine Two Generals' problem as a application, and a discussion on private announcements.

2.5.1 Two Generals' problem

The Byzantine Two Generals' problem [Akkoyunlu et al., 1975; Gray, 1978]. is an experiment illustrating the problem of coordination by communicating over an unreliable link, highlighting the importance of common knowledge. It goes as follows.

> Two generals, leading two armies, need to coordinate an attack on a fortified city; they can succeed only if they attack at the same time. The armies are located on each side of the city and they can only communicate by sending messengers that can be captured. The two generals have agreed that they will attack; however, they did not agree upon a time for the attack. This is required; but more importantly, it is required that each general knows that the other has agreed, and knows that the other knows he has agreed, and so on: common knowledge of the moment of the attack is expected.

Messengers carrying acknowledgements of receipt can also be captured: an infinite number of messages is required to reach common knowledge.

Formally, let $Agt = \{1, 2\}$ be the set of agents and $Prop = \{ta\}$ be the set of variables, with ta representing the time of the attack. We define the atom $S^m_{1,2}ta$ as:

$$S^m_{1,2}ta = \begin{cases} \underbrace{S_1\, S_2\, S_1\, \ldots S_2}_{m \text{ alternations}}\, ta & \text{if } m \text{ is even} \\ \underbrace{S_2\, S_1\, S_2\, \ldots S_2}_{m \text{ alternations}}\, ta & \text{if } m \text{ is odd} \end{cases}$$

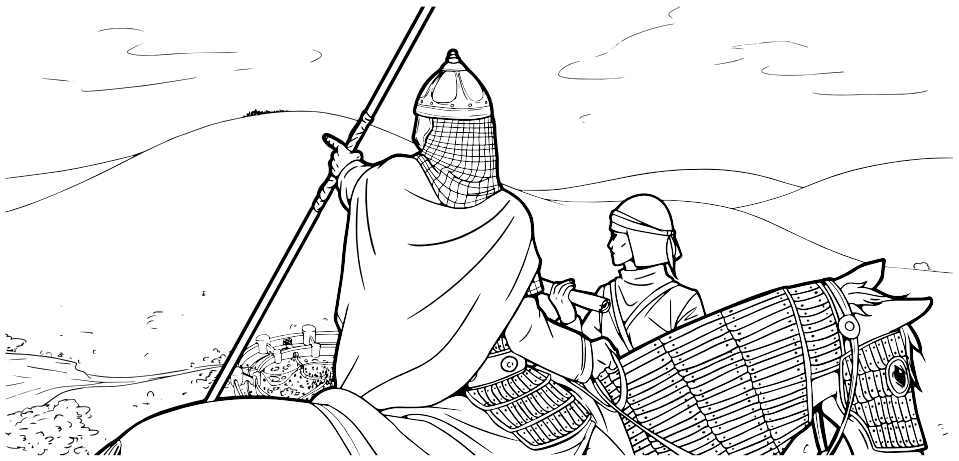
Then we define the program

$$\mathsf{sndMsg}_m = +S_1\, ta; +S_2\, ta; +S^2_{1,2}ta; \ldots; +S^m_{1,2}ta,$$

which formalizes the first general learning the time of the attack, then the second general learning it, then the first learning that the second know it, and so on until m.

Proposition 2.21. *Let $V_0 = \{\alpha : \alpha$ is valid in $INTR\} \cup \{ta\}$. We have, for every $m > 0$:*

$$V_0 \models [\mathsf{sndMsg}_m] \neg CK\, ta.$$



Proof. V_0 is introspective: it contains all introspectively valid atoms and ta. We are not interested in the truth value of the latter; making it initially true allows us to more succinctly write $CK\,ta$ instead of $CK\,ta \vee CK\neg ta$.

To have $V \models CK\,ta$, and since ta is true, we only need $JS\,ta$. Because $JS\,ta$ is not valid in $INTR$, it must be added by the program sndMsg_m. Moreover, $JS\,ta$ has no other introspective causes than itself, thus it should be added explicitly by sndMsg_m. However, $JS\,ta$ is not present in sndMsg_m, whatever the value of m. Therefore $CK\,ta$ will never be satisfied whatever $m > 0$. □

DEL-PAO allows us to simply model this kind of problem where we perform private announcements of variables and visibility of variables. We generalize this in the next section.

2.5.2 Private announcements

Public Announcement Logic PAL [Plaza, 1989] is a logic of the DEL family, extending standard epistemic logic with an operator $[\psi!]$, such that $[\psi!]\varphi$ reads "after ψ is publicly and truthfully announced, φ is true." We show in Chapter 6 how to extend DEL-PAO with public announcements, and more generally, publicly executed programs. In this section we show how to express private announcements [Baltag et al., 1998] of certain kinds of formulas within the language of DEL-PAO.

We write $i : \varphi$ for "φ is privately announced to i." We model purely private announcements here, i.e., no agent observes that φ was announced to i. Suppose we want to announce to agent i that p is true. Then we can use the following program:

$$i : p! = p?; {+}S_i\,p$$

Indeed, $[i : p!]K_i p$ is plainly valid because after the execution of $p?; +S_i\, p$, both p and $S_i\, p$ are true. This also works for the announcement of $\neg p$:

$$i : \neg p! = \neg p?; +S_i\, p$$

and more generally for the announcement of a visibility atom or the negation of a visibility atom, i.e., for literals:

$$i : \alpha! = \alpha?; +S_i\, \alpha$$
$$i : \neg\alpha! = \neg\alpha?; +S_i\, \alpha$$

Similarly, we can safely announce a conjunction of literals:

$$i : (\alpha_1 \wedge \ldots \wedge \alpha_m)! = (\alpha_1 \wedge \ldots \wedge \alpha_m)?; +S_i\, \alpha_1; \ldots; +S_i\, \alpha_m$$

Observe that the ordering of the α_j is not important since we only add atoms. This implies that we can also privately announce knowledge of literals, since they can be reduced to a conjunction of atoms. Here are some examples:

$$i : K_j p! = K_j p?; +S_i\, S_j\, p; +S_i\, p$$
$$i : K_j K_\ell \neg q! = K_j K_\ell \neg q?; +S_i\, S_j\, S_\ell\, q; +S_i\, S_\ell\, q; +S_i\, S_j\, q; +S_i\, q$$
$$i : CK S_j\, \alpha! = CK S_j\, \alpha?; +S_i\, JS\, S_j\, \alpha; +S_i\, S_j\, \alpha$$

In the general case, if φ is a sequence of epistemic operators (K_i or CK) followed by a literal, any $S_i\, \alpha$ with $\alpha \in RATM(\varphi)$ must be set to true in order to announce φ to i.

Because we only modify visibilities of atoms, we cannot properly announce disjunctions in the general case. However, we can announce a special case of disjunction: the fact of *knowing whether* a literal. For example:

$$i : (K_j p \vee K_j \neg p)! = (K_j p \vee K_j \neg p)?; +S_i\, S_j\, p$$

Observe that we add less information here than when we were announcing $K_j p$: we do not include the visibility of i on p.

We will extensively use these kinds of announcements when modelling the gossip problem in Chapter 3.

2.6 Conclusion

Using visibility atoms instead of visibility sets of propositional variables, DEL-PAO avoids the strong hypothesis of common knowledge of visibilities that other observation-based epistemic logics make. It also includes joint visibility atoms, that allows us to express common knowledge. These atoms can be modified by assignments programs, modifying

facts of the world and of knowledge of agents in a simple way. The model checking requires reducing the infinite models to finite models containing only the relevant atoms. However, the merge of knowledge and programs comes without increasing the complexity: the model checking problem is PSPACE-complete like in ECL-PC and DL-PA.

The logic DEL-PAO is the central contribution of this thesis and the base of all other logics that will be presented. We have seen that it solves the first problem of visibility-based logics that we have identified in the introduction; we are now able to reason about higher-order knowledge. This will be necessary in order to study the generalized gossip problem in Chapter 3 and other epistemic planning problems in Chapter 4. As for the second problem (distribution of knowledge operators over disjunctions), we will propose a solution to it in Chapter 6. Moreover, in Chapter 5 we will see an extension of DEL-PAO with control atoms and its application to epistemic boolean games.

3 How to share higher-order knowledge by gossiping

There are contexts where agents have to achieve higher-order knowledge, typically in order to coordinate some joint action. In the original gossip problem, all secrets are shared knowledge after $2(n-2)$ calls, but they fail to be common knowledge. Unless everybody knows the protocol and there is a global clock, such common knowledge cannot be attained. More modestly, the agents may want to achieve second-order shared knowledge: they may have the goal that everybody knows *that everybody knows all secrets. This chapter investigates how such higher-order knowledge can be achieved.*

The original version of the gossip problem [Akkoyunlu et al., 1975; Hurkens, 2000] goes as follows.

> There are six agents each of which knows some secret not known to anybody else. Two agents can make a telephone call and exchange all secrets they know. How many calls does it take to share all secrets, i.e., how many calls have to take place until everybody knows all secrets?

The problem can be generalized from six to arbitrary numbers of agents n. In the literature one can find various protocols achieving the goal in $2(n-2)$ calls. It has been proved that they are optimal: no protocol exists achieving the goal with less calls [Baker and Shostak, 1972; Tijdeman, 1971; Hajnal et al., 1972].

In this chapter, we are interested in a novel generalisation of the gossip problem including higher-order knowledge: the goal is not only that every agent knows every secret, but also that every agent knows this, and that every agent knows this... and so on until a given depth k. We are going to express this generalisation within the framework of DEL-PAO. Thanks to this formalisation, we are able to formally study

the problem, giving a protocol that achieves the goal and proving that it is optimal for any depth of knowledge k.

Contents

Résumé du chapitre

La version originale du *problème du bavardage* [Akkoyunlu et al., 1975; Hurkens, 2000] est la suivante.

> Il y a six agents, chacun connaissant un secret inconnu de tous les autres. Deux agents peuvent s'appeler au téléphone et d'échanger tous les secrets qu'ils connaissent. Combien d'appels sont nécessaires pour partager tous les secrets, c'est-à-dire, combien d'appels doivent avoir lieu pour que tout le monde connaisse tous les secrets ?

Le problème peut être généralisé de six à un nombre arbitraire d'agents n. Dans la littérature, il est possible de trouver différents protocoles qui atteignent l'objectif en $2(n-2)$ appels. Il a été prouvé qu'ils sont optimaux : il n'existe aucun protocole qui atteint l'objectif en moins d'appels [Baker and Shostak, 1972; Tijdeman, 1971; Hajnal et al., 1972].

Il existe des contextes où les agents ont besoin d'acquérir des connaissances d'ordre supérieur, généralement afin de coordonner une action conjointe. Dans le problème du bavardage original, tous les secrets sont connaissance partagée après $2(n-2)$ appels, mais à moins que tout le monde connaisse le protocole et qu'il y ait une horloge globale, une telle connaissance commune ne peut pas être atteinte. Plus modestement, nous nous intéressons à une généralisation du problème du bavardage qui inclue les connaissances d'ordre supérieur : le but est non seulement que chaque agent connaisse tous les secrets, mais aussi que chaque agent sache cela, et que chaque agent sache cela... et ainsi de suite jusqu'à une profondeur donnée k. Nous allons exprimer cette généralisation dans le cadre de DEL-PAO. Grâce à cette formalisation, nous

sommes en mesure d'étudier formellement le problème, en donnant protocole qui permet d'atteindre l'objectif, et en prouvant qu'il est optimal pour toute profondeur de connaissance k.

3.1 The generalised problem

Let $Agt = \{1, \ldots, n\}$ be the set of all agents. Let us denote the secret of agent i by s_i: $Prop = \{s_i : i \in Agt\}$. To simplify things we suppose that s_i is true.

The initial situation before the agents start gossiping is expressed by

$$\bigwedge_{i \in Agt} \Big(s_i \wedge K_i s_i \wedge \bigwedge_{j \in Agt, j \neq i} \Big(\neg K_j s_i \wedge \neg K_j \neg s_i \Big) \Big)$$

and the formula

$$\bigwedge_{i \in Agt} K_i \Big(\bigwedge_{j \in Agt} s_j \Big)$$

expresses the goal that every agent knows every secret. Let us abbreviate the conjunction $\bigwedge_{i \in J} s_i$ of secrets of agents in J by s_J; remember that $EK_J\varphi$ abbreviates the conjunction $\bigwedge_{i \in J} K_i\varphi$. We usually drop set parentheses and write write $EK_{i_1,\ldots,i_m}$ and $s_{i_1,\ldots,i_m}$ instead of $EK_{\{i_1,\ldots,i_m\}}$ and $s_{\{i_1,\ldots,i_m\}}$. Then:

- $EK_{Agt}\, s_{Agt}$ expresses that all secrets are shared knowledge: every agent knows every secret;
- $EK_{Agt} EK_{Agt}\, s_{Agt}$ expresses the goal that every agent knows that all secrets are shared knowledge;
- $\underbrace{EK_{Agt} \ldots EK_{Agt}}_{k \text{ times}}\, s_{Agt}$ expresses that all secrets are shared knowledge up to depth $k \geq 1$.

We define the iteration of EK_J inductively, for $m \geq 0$, by $EK_J^0\varphi = \varphi$ and $EK_J^{m+1}\varphi = EK_J EK_J^m \varphi$. Thus our goal is:

$$EK_{Agt}^k\, s_{Agt}$$

The result of a phone call between two agents is that their knowledge increases. We model a call between two agents i and j by a DEL-PAO action noted Call_j^i. Then

$$[\mathsf{Call}_j^i] EK_{i,j}(s_i \wedge s_j),$$

also noted

$$[\mathsf{Call}_j^i]EK_{i,j}\,s_{i,j},$$

expresses that the result of Call_j^i is that i and j know their secrets. When we say that during a call the agents communicate all they know then this not only concerns secrets, but also knowledge about secrets and more generally higher-order knowledge. Therefore calls achieve common knowledge between the calling agents:

$$[\mathsf{Call}_j^i]EK_{i,j}^m\,s_{i,j}$$

is the case for arbitrary nesting m of $EK_{i,j}$. Furthermore, the formula

$$\underbrace{[\mathsf{Call}_{j_1}^{i_1}]\ldots[\mathsf{Call}_{j_{2(n-2)}}^{i_{2(n-2)}}]}_{2(n-2)\text{ times}}\,EK_{Agt}\,s_{Agt}$$

expresses that the protocol where i_1 calls j_1 first, then i_2 calls j_2, ..., and finally $i_{2(n-2)}$ calls $j_{2(n-2)}$ achieves shared knowledge.

We note $Gossip(k,n)$ the instance of the generalized gossip problem with $n \geq 2$ agents and the goal to achieve depth $k \geq 1$ of shared knowledge. So the original problem corresponds to the instance $Gossip(1,6)$. We are going to introduce a protocol achieving shared knowledge of depth k in $(k{+}1)(n{-}2)$ calls, for $n \geq 4$. We moreover prove that our protocol is optimal: at least $(k{+}1)(n{-}2)$ calls are necessary to achieve the goal of $Gossip(k,n)$.

3.2 An algorithm achieving shared knowledge

The following algorithm generates a sequence of calls for a given instance $Gossip(k,n)$ of the generalized gossip problem, for $k \geq 1$ and $n \geq 4$. Throughout the algorithm two of the agents, which we call *left* and *right*, will have a central, fixed role: each of the other agents only communicates with either *left* or *right*. The $n{-}2$ remaining agents will be numbered 0, 1, ..., $n{-}3$.

The algorithm is made up of *turns*. During each turn, *left* and *right* collect the secrets of other agents. Together with the last agent they talked to in that turn, they thereby become what we call "semi-experts." A further call between complementary semi-experts turns them into full experts. The last agents talked to, *left* and *right*, play a crucial role. These two further semi-experts are permuted at each turn in a way that will guarantee that the goal is reached.

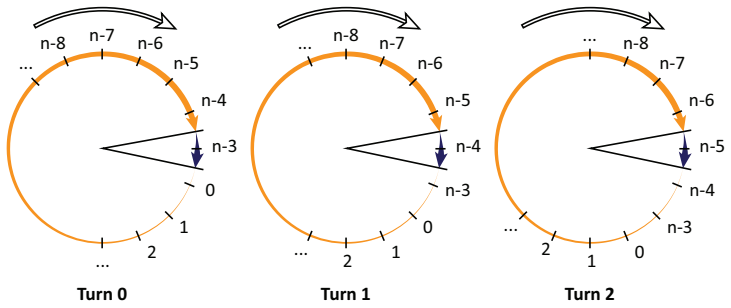


Figure 3.1: Graphical representation of the first three turns of Algorithm 3.1.

Algorithm 3.1.
procedure $gossip(k, n)$:
  for $t = 0..k$:
    agent *left* calls agent $0-t \mod n-2$;
    agent *left* calls agent $1-t \mod n-2$;
    ⋮
    agent *left* calls agent $n-4-t \mod n-2$;
    agent *right* calls agent $n-3-t \mod n-2$.

In words:

- at the first turn (turn 0), agent *left* calls agent 0, then 1, ..., then $n-4$, and finally agent *right* calls agent $n-3$;
- at the second turn (turn 1), agent *left* calls agent $n-3$, then 0, then 4, ..., then $n-5$; and finally agent *right* calls agent $n-4$;
- ... and so on.

So each turn involves $n-2$ calls, and overall the algorithm produces a sequence of $(k+1)(n-2)$ calls. In the rest of the chapter, we assume that every index of agent is taken modulo $n-2$ and we omit "$\mod n-2$."

Figure 3.1 gives a visual representation of Algorithm 3.1: agents 0, 1, ..., $n-3$ are put on a wheel which, between each turn, rotates clockwise. Agent *left* calls everyone in ascending order, except the agent at the rightmost position of the wheel, then *right* calls this agent.

Theorem 3.1. *The minimal number of calls needed to solve the instance* $Gossip(k, n)$ *of the generalized gossip problem, for* $k \geq 1$ *and* $n \geq 4$*, is* $(k+1)(n-2)$.

The main part of the chapter is devoted to the proof of the above theorem: we prove that the sequence of calls produced by the algorithm is indeed a solution in Section 3.4 and that it is optimal in Section 3.5. Our proofs will be done in the formal language of DEL-PAO.

3.3 Calls in the language of DEL-PAO

DEL-PAO provides a suitable framework to model calls between agents and to reason about the evolution of their knowledge. Before the proof of correctness of our algorithm, we show how to express calls and we give some of their properties.

In the standard version of the gossip problem, agents only communicate their factual knowledge during a call. In order to achieve higher-order knowledge they also have to tell what they know about others: for shared knowledge of level k they have to exchange all their knowledge up to depth $k-1$.

More formally, let the level k of intended shared knowledge be given. Let i and j be two agents. For a given integer m, we note $\{S_i, S_j\}^{\leq m}$ the set all non-empty sequences of visibility operators S_i and S_j of length at most m. For example:

$$\{S_i, S_j\}^{\leq 2} = \{S_i, S_j, S_i\, S_i, S_i\, S_j, S_j\, S_i, S_j\, S_j\}.$$

Then Call^i_j is the sequential composition of programs of the form

$$\textbf{if } K_i K_{i_1} \cdots K_{i_m} s \vee K_j K_{i_1} \cdots K_{i_m} s \textbf{ then } +\sigma_1 S_{i_1} \cdots S_{i_m}\, s; \ldots; +\sigma_\ell S_{i_1} \cdots S_{i_m}\, s$$

for secret s in Prop, integer $m \leq k-1$, agents $i_1, \ldots, i_m \in \mathit{Agt}$ and sequences $\{S_i, S_j\}^{\leq k-m} = \{\sigma_1, \ldots, \sigma_\ell\}$. (Remember that **if** φ **then** π abbreviates $(\varphi?;\pi) \sqcup \neg\varphi?$.) For example, for $k = 3$ the following is an element of the sequence:

$$\textbf{if } K_i K_\ell s \vee K_j K_\ell s \textbf{ then } +S_i\, S_\ell\, s; +S_j\, S_\ell\, s; +S_i\, S_i\, S_\ell\, s; +S_i\, S_j\, S_\ell\, s; +S_j\, S_i\, S_\ell\, s; +S_j\, S_j\, S_\ell\, s$$

That piece of program tests whether $K_\ell s$ is known by i or j and if so, makes $S_\ell\, s$ visible for both i and j and i's observation of $S_\ell\, s$ visible for j, and vice versa. Remember that this is procedure we used in Section 2.5.2 to model private announcements. We observe that the additions $+S_i\, S_i\, S_\ell\, s$ and $+S_j\, S_j\, S_\ell\, s$ are trivial because they are introspectively valid.

The following properties of the program Call^i_j and of its interaction with the shared knowledge operator will be useful in our proofs.

First of all, the dynamic operators $[\mathsf{Call}^i_j]$ and the shared knowledge operators EK_J are normal modal operators. So in particular $[\mathsf{Call}^i_j]\varphi \wedge [\mathsf{Call}^i_j]\psi \leftrightarrow [\mathsf{Call}^i_j](\varphi \wedge \psi)$ and $(EK_J\varphi \wedge EK_J\psi) \leftrightarrow EK_J(\varphi \wedge \psi)$ are plainly valid. Moreover, we can put coalitions together: the schema

$$(EK_{J_1}\varphi \wedge EK_{J_2}\varphi) \leftrightarrow EK_{J_1 \cup J_2}\varphi$$

is plainly valid for every $J_1, J_2 \subseteq Agt$. (To see this reduce EK according to its definition.) Finally, calls preserve positive knowledge and produce shared knowledge, which is a property that we state formally.

Proposition 3.1. *Let $s \in Prop$ and $m \geq 0$. Let φ be of the form either $K_{i_1} \dots K_{i_m} s$ or $EK_{J_1} \dots EK_{J_m} s$. Then the formulas:*

$$\varphi \to [\mathsf{Call}^i_j]\varphi \tag{$Prsv$}$$

$$K_i\varphi \to [\mathsf{Call}^i_j]EK^{k-m}_{i,j}\varphi \tag{$Incr$}$$

are plainly valid.

Proof. We prove each implication thanks to properties of DEL-PAO.

- (***Prsv***). This is due to φ not containing negations and calls only making atoms true.
- (***Incr***). Suppose $\varphi = K_{i_1} \dots K_{i_m} s$. The proof for $\varphi = EK_{J_1} \dots EK_{J_m} s$ is similar since EK_J is a conjunction of K_i.

  We have seen in DEL-PAO (see Proposition 2.19 on page 63) that $\varphi \leftrightarrow \Big(\bigwedge_{\beta \in RATM(\varphi)} \beta\Big)$ is plainly valid. Moreover, $RATM(EK^m_{i,j}\varphi) = \{\sigma\, \alpha : \alpha \in RATM(\varphi), \sigma \in \{S_i, S_j\}^{\leq m}\}$ by the definition of the relevant atoms. Therefore we want to prove that

$$K_i\varphi \to [\mathsf{Call}^i_j]\Big(\bigwedge_{\beta \in \{\sigma\, \alpha : \alpha \in RATM(\varphi), \sigma \in \{S_i, S_j\}^{\leq k-m}\}} \beta\Big).$$

  We have:

$$K_iK_{i_1} \dots K_{i_m} s \to K_iK_{i_{r_1}} \dots K_{i_{r_p}} s,$$

  for every $\langle r_1, \dots, r_p\rangle \sqsubseteq \langle 1, \dots, m\rangle$, by axiom T of standard epistemic logic. Remember that $\langle r_1, \dots, r_p\rangle \sqsubseteq \langle 1, \dots, m\rangle$ means that $\langle r_1, \dots, r_p\rangle$ is a subset of $\langle 1, \dots, m\rangle$, in the same order.

  Since $K_iK_{i_{r_1}} \dots K_{i_{r_p}} s$ obviously implies $K_iK_{i_{r_1}} \dots K_{i_{r_p}} s \vee K_jK_{i_{r_1}} \dots K_{i_{r_p}} s$, for every subset $\langle r_1, \dots, r_p\rangle \sqsubseteq \langle 1, \dots, m\rangle$, we have:

$$K_iK_{i_1} \dots K_{i_m} s \to [\mathsf{Call}^i_j]\Big(\bigwedge_{\beta \in \{\sigma\, S_{i_{r_1}} \dots S_{i_{r_p}}\, s : \sigma \in \{S_i, S_j\}^{\leq k-p}\}} \beta\Big),$$

  by the definition of programs composing Call^i_j. This means that

$$K_iK_{i_1} \dots K_{i_m} s \to [\mathsf{Call}^i_j]\Big(\bigwedge_{\beta \in \{\sigma\, S_{i_{r_1}} \dots S_{i_{r_p}}\, s : \sigma \in \{S_i, S_j\}^{\leq k-m}\}} \beta\Big),$$

because $p \leq m$.

Since this is true for any $\langle r_1, \ldots, r_p \rangle \sqsubseteq \langle 1, \ldots, m \rangle$, we can apply Proposition 2.20 on page 64 and we obtain

$$K_i K_{i_1} \ldots K_{i_m} s \to [\mathsf{Call}^i_j]\Big(\bigwedge_{\beta \in \{\sigma\, \alpha : \alpha \in RATM(K_{i_1} \ldots K_{i_m} s), \sigma \in \{S_i, S_j\}^{\leq k-m}\}} \beta \Big),$$

which is our result.

Therefore both implications are plainly valid. □

With the help of the call actions, we can define the turn t of Algorithm 3.1 as:

$$\mathsf{turn}_t = \mathsf{Call}^{left}_{0-t}; \ldots; \mathsf{Call}^{left}_{n-3}; \mathsf{Call}^{left}_{0}; \ldots; \mathsf{Call}^{left}_{n-4-t}; \mathsf{Call}^{right}_{n-3-t}.$$

3.4 Correctness

We now prove that the algorithm returns a solution.

Let us rename the agents: $Agt = \{left, right, 0, \ldots, n-3\}$. The initial state is modelled by the valuation

$$V_0 = \{s_i : i \in Agt\} \cup \{S_i\, s_i : i \in Agt\} \cup \{\alpha : \alpha \text{ is valid in } INTR\}.$$

So all secrets are true, each agent knows her own secret, and moreover the introspectively valid atoms are true (so that $V_0 \in INTR$). We have:

$$V_0 \models \bigwedge_{i \in Agt} \Big(K_i s_i \wedge \bigwedge_{j \in Agt, j \neq i} \neg K_j s_i \Big).$$

An agent is an *expert for depth* t if her personal goal for depth t is reached. Precisely, agent i is an expert for depth $t \geq 1$ if and only if we have:

$$K_i E K^{t-1}_{Agt} s_{Agt}.$$

The dynamic modalities of DEL-PAO nicely allow us to express that a further call would turn an agent i into an expert, i.e., that i is a semi-expert. Indeed, two agents i and j are *complementary for depth* t ("semi-experts"), noted $\mathsf{compl}_t(i, j)$, if a call between i and j would make them both experts for depth t. More formally:

$$\mathsf{compl}_t(i, j) = [\mathsf{Call}^i_j] E K_{i,j} E K^{t-1}_{Agt} s_{Agt}.$$

Furthermore, two pairs of agents (i_1, i_2) and (j_1, j_2) are complementary for depth t if and only if we have:

$$\mathsf{compl}_t(i_1, j_1) \wedge \mathsf{compl}_t(i_1, j_2) \wedge \mathsf{compl}_t(i_2, j_1) \wedge \mathsf{compl}_t(i_2, j_2).$$

We will prove that at each turn, two pairs of agents are complementary: the first pair is agent $left$ along with the last agent she called at this turn, and the second is agent $right$ along with the last (and only agent) she called at this turn.

The first turn is a special case where semi-experts of depth 1 are produced.

Lemma 3.1. *We have:*

$$V_0 \models [\mathsf{turn}_0]\big(EK_{left,n-4}s_{left,0,\ldots,n-4} \wedge EK_{right,n-3}s_{right,n-3}\big).$$

Proof. Let us write ij for the call between i and j. The first turn (turn 0) of Algorithm 3.1 produces the following sequence of calls:

$$left0, left1, \ldots, left(n{-}4), right(n{-}3).$$

By $(Incr)$ of Proposition 3.1 we have $V_0 \models [\mathsf{Call}_0^{left}]EK_{left,0}s_{left,0}$ and therefore $V_0 \models [\mathsf{Call}_0^{left}]K_{left}s_{left,0}$. We do the same for the next call:

$$\begin{aligned} V_0 &\models [\mathsf{Call}_0^{left}][\mathsf{Call}_1^{left}]EK_{left,1}s_{left,0,1} \\ \Rightarrow V_0 &\models [\mathsf{Call}_0^{left}][\mathsf{Call}_1^{left}]K_{left}s_{left,0,1}, \end{aligned}$$

and so on until

$$V_0 \models [\mathsf{Call}_0^{left}][\mathsf{Call}_1^{left}]\ldots[\mathsf{Call}_{n-4}^{left}]EK_{left,n-4}s_{left,0,1,\ldots,n-4}.$$

In the same vein we also have $V_0 \models [\mathsf{Call}_{n-3}^{right}]EK_{right,n-3}s_{right,n-3}$.

By $(Prsv)$ of Proposition 3.1 we then obtain

$$V_0 \models [\mathsf{Call}_0^{left}]\ldots[\mathsf{Call}_{n-4}^{left}][\mathsf{Call}_{n-3}^{right}]\big(EK_{left,n-4}s_{left,0,\ldots,n-4} \wedge EK_{right,n-3}s_{right,n-3}\big)$$

which is the same as

$$V_0 \models [\mathsf{turn}_0]\big(EK_{left,n-4}s_{left,0,\ldots,n-4} \wedge EK_{right,n-3}s_{right,n-3}\big),$$

hence the result. □

We now characterize the turns after turn_0.

Lemma 3.2. *For $t \geq 1$, we have:*

$$V_0 \models [\mathsf{turn}_0;\ldots;\mathsf{turn}_t]\big(EK_{left,n-4-t}EK_{left,0-t,\ldots,n-4-t}EK_{Agt}^{t-1}s_{Agt} \wedge EK_{right,n-3-t}EK_{right,n-3-t}EK_{Agt}^{t-1}s_{Agt}\big).$$

Proof. We prove it by induction on t. Both cases resemble the proof of Lemma 3.1.

Base case: $t = 1$. The turn 1 of Algorithm 3.1 produces the following sequence:

$$\mathit{left}(n{-}3), \mathit{left}0, \mathit{left}1, \ldots, \mathit{left}(n{-}5), \mathit{right}(n{-}4).$$

By Lemma 3.1 and (Incr) of Proposition 3.1 we have

$$\begin{aligned} V_0 &\models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}] EK_{\mathit{left},n-3} EK_{\mathit{left},n-3} s_{Agt} \\ \Rightarrow V_0 &\models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}] K_{\mathit{left}} EK_{\mathit{left},n-3} s_{Agt}. \end{aligned}$$

Then again by (Incr),

$$\begin{aligned} V_0 &\models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}][\mathsf{Call}^{\mathit{left}}_{0}] EK_{\mathit{left},0} EK_{\mathit{left},n-3,0} s_{Agt} \\ \Rightarrow V_0 &\models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}][\mathsf{Call}^{\mathit{left}}_{0}] K_{\mathit{left}} EK_{\mathit{left},n-3,0} s_{Agt}, \end{aligned}$$

and for the next call

$$\begin{aligned} V_0 &\models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}][\mathsf{Call}^{\mathit{left}}_{0}][\mathsf{Call}^{\mathit{left}}_{1}] EK_{\mathit{left},1} EK_{\mathit{left},n-3,0,1} s_{Agt} \\ \Rightarrow V_0 &\models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}][\mathsf{Call}^{\mathit{left}}_{0}][\mathsf{Call}^{\mathit{left}}_{1}] K_{\mathit{left}} EK_{\mathit{left},n-3,0,1} s_{Agt}, \end{aligned}$$

and so on until

$$V_0 \models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}][\mathsf{Call}^{\mathit{left}}_{0}][\mathsf{Call}^{\mathit{left}}_{1}] \ldots [\mathsf{Call}^{\mathit{left}}_{n-5}] EK_{\mathit{left},n-5} EK_{\mathit{left},n-3,0,1,\ldots,n-5} s_{Agt}.$$

Similarly we have

$$V_0 \models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{right}}_{n-4}] EK_{\mathit{right},n-4} EK_{\mathit{right},n-4} s_{Agt}.$$

Finally we obtain the result by (Prsv) of Proposition 3.1:

$$\begin{aligned} V_0 \models [\mathsf{turn}_0][\mathsf{Call}^{\mathit{left}}_{n-3}][\mathsf{Call}^{\mathit{left}}_{0}] \ldots [\mathsf{Call}^{\mathit{left}}_{n-5}][\mathsf{Call}^{\mathit{right}}_{n-4}] \big(& EK_{\mathit{left},n-5} EK_{\mathit{left},n-3,0,1,\ldots,n-5} s_{Agt} \\ & \wedge EK_{\mathit{right},n-4} EK_{\mathit{right},n-4} s_{Agt} \big) \\ \Leftrightarrow V_0 \models [\mathsf{turn}_0][\mathsf{turn}_1] \big(& EK_{\mathit{left},n-5} EK_{\mathit{left},n-3,0,1,\ldots,n-5} s_{Agt} \\ & \wedge EK_{\mathit{right},n-4} EK_{\mathit{right},n-4} s_{Agt} \big). \end{aligned}$$

Inductive case. The reasoning is similar, but generalized to turn $t{+}1$. Suppose the formula is true for turn t. The turn $t + 1$ is:

$$\mathit{left}(n{-}3{-}t), \mathit{left}(0{-}t), \ldots, \mathit{left}(n{-}5{-}t), \mathit{right}(n{-}4{-}t).$$

By our induction hypothesis and (Incr) of Proposition 3.1 we have

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{\mathit{left}}_{n-3-t}] EK_{\mathit{left},n-3-t} EK_{\mathit{left},n-3-t} EK_{Agt} EK^{t-1}_{Agt} s_{Agt},$$

that is,

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{left}_{n-3-t}] EK_{left,n-3-t} EK_{left,n-3-t} EK^t_{Agt} s_{Agt},$$

which implies

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{left}_{n-3-t}] K_{left} EK_{left,n-3-t} EK^t_{Agt} s_{Agt}.$$

Then by $(Prsv)$ of Proposition 3.1,

$$\begin{aligned} & V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{left}_{n-3-t}][\mathsf{Call}^{left}_{0-t}] EK_{left,0-t} EK_{left,n-3-t,0-t} EK^t_{Agt} s_{Agt} \\ \Rightarrow\ & V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{left}_{n-3-t}][\mathsf{Call}^{left}_{0-t}] K_{left} EK_{left,n-3-t,0-t} EK^t_{Agt} s_{Agt}, \end{aligned}$$

and so on until

$$\begin{aligned} V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{left}_{n-3-t}][\mathsf{Call}^{left}_{0-t}] \ldots [\mathsf{Call}^{left}_{n-5-t}] \\ EK_{left,n-5-t} EK_{left,n-3-t,0-t,\ldots,n-5-t} EK^t_{Agt} s_{Agt}. \end{aligned}$$

Moreover, by $(Incr)$,

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{right}_{n-4-t}] EK_{right,n-4-t} EK_{right,n-4-t} EK_{Agt} EK^{t-1}_{Agt} s_{Agt},$$

that is,

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{right}_{n-4-t}] EK_{right,n-4-t} EK_{right,n-4-t} EK^t_{Agt} s_{Agt}.$$

We end as usual with $(Prsv)$:

$$\begin{aligned} & V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{Call}^{left}_{n-3-t}] \ldots [\mathsf{Call}^{left}_{n-5-t}][\mathsf{Call}^{right}_{n-4-t}] \\ & \qquad \big(EK_{left,n-5-t} EK_{left,n-3-t,\ldots,n-5-t} EK^t_{Agt} s_{Agt} \wedge \\ & \qquad\ EK_{right,n-4-t} EK_{right,n-4-t} EK^t_{Agt} s_{Agt} \big) \\ \Leftrightarrow\ & V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_t][\mathsf{turn}_{t+1}] \\ & \qquad \big(EK_{left,n-5-t} EK_{left,n-3-t,\ldots,n-5-t} EK^t_{Agt} s_{Agt} \wedge \\ & \qquad\ EK_{right,n-4-t} EK_{right,n-4-t} EK^t_{Agt} s_{Agt} \big), \end{aligned}$$

which is our result for $t{+}1$. □

Lemma 3.3. *After turn $t{-}1$ of Algorithm 3.1, the pairs $(left, n{-}3{-}t)$ and $(right, 0{-}t)$ are complementary for depth t.*

Proof. From Lemma 3.2 we can deduce

$$\begin{aligned} V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_{t-1}] \big(& K_{left} EK_{left,1-t,\ldots,n-3-t} EK^{t-2}_{Agt} s_{Agt} \wedge \\ & K_{right} EK_{right,0-t} EK^{t-2}_{Agt} s_{Agt} \big). \end{aligned}$$

Applying ($Incr$) of Proposition 3.1 we obtain

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_{t-1}][\mathsf{Call}^{left}_{right}] EK_{left,right} EK_{Agt} EK^{t-2}_{Agt} s_{Agt},$$

that is,

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_{t-1}][\mathsf{Call}^{left}_{right}] EK_{left,right} EK^{t-1}_{Agt} s_{Agt},$$

which is equivalent to

$$V_0 \models [\mathsf{turn}_0; \ldots; \mathsf{turn}_{t-1}]\mathsf{compl}_t(left, right).$$

By the same reasoning for $left$ and $0{-}t$, $right$ and $n{-}3{-}t$, and finally $n{-}3{-}t$ and $0{-}t$, we obtain that each of them are complementary, hence the result. □

Lemma 3.4. *The goal for depth t, $EK^t_{Agt} s_{Agt}$, is reached after the turn t of Algorithm 3.1.*

Proof. Turn t of Algorithm 3.1 is:

$$left(0{-}t), left(1{-}t), \ldots, left(n{-}4{-}t), right(n{-}3{-}t).$$

By Lemma 3.3, after turn $t{-}1$ and the first call $left(0{-}t)$ of turn t, agents $left$ and $0{-}t$ become experts for depth t. (Hence we have $EK_{left,0-t} EK^{t-1}_{Agt} s_{Agt}$.) Then after the $n{-}4$ calls $left(1{-}t), \ldots, left(n{-}4{-}t)$ we get by ($Incr$) of Proposition 3.1:

$$K_{1-t} EK^{t-1}_{Agt} s_{Agt} \wedge \ldots \wedge K_{n-4-t} EK^{t-1}_{Agt} s_{Agt},$$

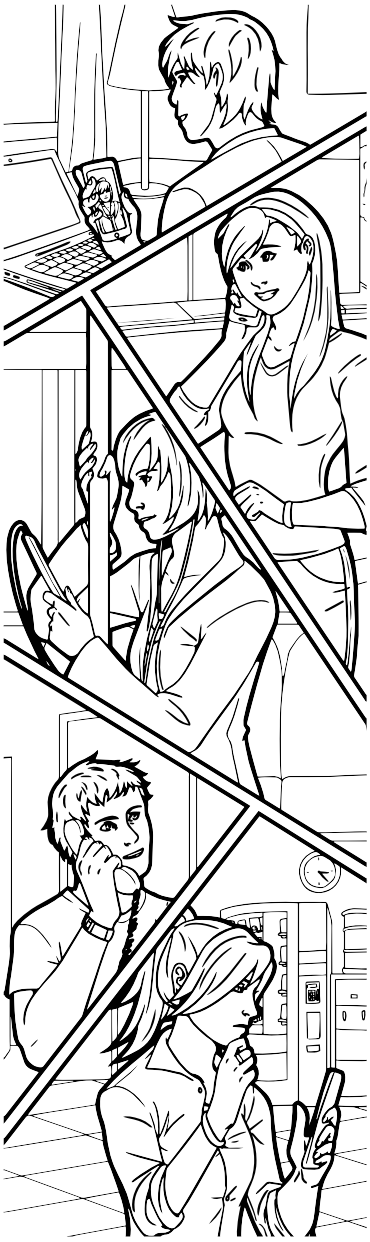
that is, $1{-}t$, …, $n{-}4{-}t$ are all experts for depth t. Finally, after the last call $right(n{-}3{-}t)$, and also by Lemma 3.3, agents $right$ and $n{-}3{-}t$ become experts for depth t. (Thus $EK_{right,n-3-t} EK^{t-1}_{Agt} s_{Agt}$.) Therefore after the $n{-}2$ calls of turn t we have $EK_{Agt} EK^{t-1}_{Agt} s_{Agt}$, which is equivalent to $EK^t_{Agt} s_{Agt}$. □

Proposition 3.2. *The sequence resulting from Algorithm 3.1 gives a solution to the generalized gossip problem for $k \geq 1$ and $n \geq 4$.*

Proof. By Lemma 3.4, the goal for depth t is reached after turn t of Algorithm 3.1. Thus the goal for depth k is reached after turn k ($k{+}1$ turns), i.e., at the end of the algorithm. □

Proposition 3.2 implies that at most $(k{+}1)(n{-}2)$ calls are required to solve the instance $Gossip(k, n)$ of the generalized gossip problem.

3.5 Optimality



In this section, we prove that the sequence of calls returned by our algorithm has an optimal length.[5]

We first show a property of the gossip problem that may seem obvious but that we prefer to clarify.

Lemma 3.5. *Suppose m agents know a fact φ not known to the remaining agents. Then it takes at least $n-m$ calls for the remaining agents to know φ.*

Proof. It suffices to prove that a call Call_j^i increases the knowledge on a fact φ of at most one agent. We distinguish four cases depending on the knowledge of i and j on φ.

- **Neither i nor j know φ.** Then $K_i\varphi \vee K_j\varphi$ is false and no agent knows φ after Call_j^i.
- **i knows φ but j does not know φ.** Then $K_i\varphi \vee K_j\varphi$ is true and both agents know φ after Call_j^i but only j learned it.
- **i does not know φ but j knows φ.** Then $K_i\varphi \vee K_j\varphi$ is true and both agents know φ after Call_j^i but only i learned it.
- **i and j know φ.** Then $K_i\varphi \vee K_j\varphi$ is true but both agents know φ after Call_j^i but no one learned it.

Therefore $n-m$ calls are necessary to spread a piece of gossip to $n-m$ agents. □

Proposition 3.3. *The minimal number of calls needed to solve the instance $Gossip(k,n)$ of the generalized gossip problem, for $k \geq 1$ and $n \geq 4$, is at least $(k+1)(n-2)$.*

[5] These results from [Cooper et al., 2016b] were not included in [Herzig and Maffre, 2016].

Proof. We prove it by induction on k.

Base case: $k = 1$. As we have seen, protocols achieving the goal in $2(n-2)$ calls have been proved to be optimal [Baker and Shostak, 1972; Tijdeman, 1971; Hajnal et al., 1972] for $k = 1$.

Inductive case. Suppose that we need at least $(k+1)(n-2)$ calls to achieve the goal for depth k. This implies that after $(k+1)(n-2) - 1$ calls, at least one agent, let us call her i, does not know a piece of information of depth $k-1$:

$$V_0 \models [\mathsf{Call}^{i_1}_{j_1}; \ldots; \mathsf{Call}^{i_{(k+1)(n-2)-1}}_{j_{(k+1)(n-2)-1}}] \neg K_i K_{\ell_1} \cdots K_{\ell_{k-1}} s.$$

Observe that i could also not know facts of a lower depth; in this case she would also not know facts of depth $k-1$ by the truth axiom T. For example, suppose that i does not know the secret of 1; then she cannot know that 2 knows the secret of 1, and that 1 knows that 2 knows the secret of 1, and so on.

Then the $(k+1)(n-2)$-th call involves i (otherwise her knowledge does not evolve) and another agent, say j. This call establishes not only $K_i K_{\ell_1} \cdots K_{\ell_{k-1}} s$ and thus the goal for depth k:

$$V_0 \models \langle \mathsf{Call}^{i_1}_{j_1}; \ldots; \mathsf{Call}^{i_{(k+1)(n-2)-1}}_{j_{(k+1)(n-2)-1}} \rangle \langle \mathsf{Call}^{i}_{j} \rangle E K^k_{Agt} s_{Agt},$$

but also the fact that i and j both know this:

$$V_0 \models \langle \mathsf{Call}^{i_1}_{j_1}; \ldots; \mathsf{Call}^{i_{(k+1)(n-2)-1}}_{j_{(k+1)(n-2)-1}} \rangle \langle \mathsf{Call}^{i}_{j} \rangle E K_{i,j} E K^k_{Agt} s_{Agt},$$

while no other agent does:

$$V_0 \models [\mathsf{Call}^{i_1}_{j_1}; \ldots; \mathsf{Call}^{i_{(k+1)(n-2)-1}}_{j_{(k+1)(n-2)-1}}] [\mathsf{Call}^{i}_{j}] \Big(\bigwedge_{\ell \in Agt \setminus \{i,j\}} \neg K_\ell E K^k_{Agt} s_{Agt} \Big).$$

To establish the goal for depth $k+1$, $E K_{Agt} E K^k_{Agt} s_{Agt}$, it is necessary to distribute $E K^k_{Agt} s_{Agt}$ from i and j to all other agents. By Lemma 3.5, we know that this takes at least $n-2$ calls. Therefore, we need $(k+1)(n-2) + n - 2 = (k+2)(n-2)$ calls to achieve the goal for depth $k+1$. □

Propositions 3.2 and 3.3 together ensure Theorem 3.1, i.e., that the minimal number of calls needed to solve the instance (k, n) of the generalized gossip problem, for $k \geq 1$ and $n \geq 4$, is exactly $(k+1)(n-2)$.

3.6 Two and three agents

Our algorithm works when four or more agents are involved; it trivially does not for two and three agents.[6]

[6] The case of two and three agents was not discussed in [Herzig and Maffre, 2016].

The former case is easy: only one call is necessary for two agents to reach knowledge on their secrets of level k, whatever k is. (This is ensured by $(Incr)$ of Proposition 3.1.) Obviously, less calls are not sufficient.

For three agents, we number them 0, 1 and 2 and we give an algorithm that takes $k{+}2$ turns of one call each.

Algorithm 3.2.
procedure $gossip(k, 3)$:
  for $t = 0..k{+}1$:
    agent 0 calls agent $(t \mod 2) + 1$.

Hence the algorithm produces sequences of the form $\mathsf{Call}_1^0; \mathsf{Call}_2^0; \mathsf{Call}_1^0; \mathsf{Call}_2^0; \ldots$, $k{+}2$ times; each turn consists in one call: $\mathsf{turn}_0 = \mathsf{Call}_1^0$, $\mathsf{turn}_1 = \mathsf{Call}_2^0$, $\mathsf{turn}_2 = \mathsf{Call}_1^0$, and so on. We prove that it is correct and optimal.

Theorem 3.2. *The minimal number of calls needed to solve the instance* $Gossip(k, 3)$ *of the generalized gossip problem, for* $k \geq 1$*, is* $k{+}2$*.*

Proposition 3.4. *The sequence resulting from Algorithm 3.2 gives a solution to the generalized gossip problem for* $k \geq 1$ *and* $n = 3$*.*

Proof. We prove it by induction on k.

Base case: $k = 1$. It is easy to check that the sequence $\mathsf{Call}_1^0; \mathsf{Call}_2^0; \mathsf{Call}_1^0$ establishes the goal for depth 1.

Inductive case. Suppose $k{+}2$ turns of the algorithm achieve the goal for depth k:

$$V_0 \models [\mathsf{turn}_0; \mathsf{turn}_1; \ldots; \mathsf{turn}_{k+2}] EK^k_{Agt} s_{Agt}.$$

Because $\mathsf{turn}_{k+2} = \mathsf{Call}^0_{(k+2 \bmod 2)+1}$, we also have

$$V_0 \models [\mathsf{turn}_0; \mathsf{turn}_1; \ldots; \mathsf{turn}_{k+2}] EK_{0,(k+2 \bmod 2)+1} EK^k_{Agt} s_{Agt},$$

which implies

$$V_0 \models [\mathsf{turn}_0; \mathsf{turn}_1; \ldots; \mathsf{turn}_{k+2}] K_0 EK^k_{Agt} s_{Agt},$$

and hence by $(Incr)$ of Proposition 3.1,

$$V_0 \models [\mathsf{turn}_0; \mathsf{turn}_1; \ldots; \mathsf{turn}_{k+2}][\mathsf{turn}_{k+3}] EK_{0,(k+3 \bmod 2)+1} EK^k_{Agt} s_{Agt}.$$

since $\mathsf{turn}_{k+3} = \mathsf{Call}^0_{(k+3 \bmod 2)+1}$. Therefore, because 0, $(k{+}2 \bmod 2) + 1$ and $k{+}3 \bmod 2) + 1$ are all different, we obtain by $(Prsv)$ of Proposition 3.1:

$$V_0 \models [\mathsf{turn}_0; \mathsf{turn}_1; \ldots; \mathsf{turn}_{k+2}][\mathsf{turn}_{k+3}] EK_{Agt} EK^k_{Agt} s_{Agt},$$

that is,

$$V_0 \models [\mathsf{turn}_0; \mathsf{turn}_1; \ldots; \mathsf{turn}_{k+2}][\mathsf{turn}_{k+3}] EK^{k+1}_{Agt}\, s_{Agt},$$

which is our goal for depth $k+1$. □

Proposition 3.5. *The minimal number of calls needed to solve the instance $Gossip(k, 3)$ of the generalized gossip problem, for $k \geq 1$, is at least $k+2$.*

Proof. This proof is similar to the proof of optimality for $n \geq 4$ (see Proposition 3.3).

Base case: $k = 1$. It was proven—and it is easy to check—that 3 calls are necessary, in the original problem, when three agents are involved [Baker and Shostak, 1972; Tijdeman, 1971; Hajnal et al., 1972].

Inductive case. Suppose that at least $k+2$ calls are necessary to achieve the goal for depth k. Then after $k+1$ calls, at least one agent i does not know a piece of information of depth $k-1$. The $k+2$-th call, between i and j, makes i know this piece of information, and i and j be aware of this. At least one call is necessary to inform the third agent that the goal for depth k was reached, establishing the goal for depth $k+1$ in $k+3$ calls. □

3.7 Gossiping with ignorance goals

In the gossip problem, we aim for full knowledge of every agent on secrets. We could also consider scenarios where we want that some agents do not learn some secrets. Assuming agents cannot omit to tell secrets they know (say, for example, because they do not know the goal), the ordering of calls might be influenced by these "ignorance goals." This section discusses some aspects of this variant which, to the best of our knowledge, was not investigated before. Unlike the version with full knowledge goal though, we do not provide a result on the number of calls or a generic algorithm, but rather protocols in specific cases and general remarks.[7]

Let us start with the original gossip problem, i.e., with $k = 1$. Suppose we do not want agent 1 to know the secret of 2, and full knowledge otherwise. Our goal is

$$\big(EK_{Agt\setminus\{1\}}\, s_{Agt}\big) \wedge \big(K_1\, s_{Agt\setminus\{2\}} \wedge \neg K_1\, s_2\big).$$

While this is obviously unsolvable for 2 agents, for at least 3 agents, a protocol would be for 1 to call every other agent but 2 before they have

[7] This discussion, inspired from [Cooper et al., 2016a], was not included in [Herzig and Maffre, 2016].

called 2. Then 2 and the remaining agents could solve the problem not involving 1. Slightly more generally, if 1 must not know the secret of 2, ..., m, she must call $m{+}1$, ..., n before they call any of 2, ..., m, then 2, ..., m, $m{+}1$, ..., n can freely acquire full knowledge.

It gets quickly more complicated, even for $k = 1$, when several agents should be ignorant. For example, suppose $n = 4$ and 1 should not know the secret of 3, while 2 should not know the secret of 4. Then 1 should call 2 first, before she calls 4 and before 2 calls 3. Then 1 can call 4, 2 can call 3 and 3 can call 4. Now suppose that 1 should not know the secret of 3, while 3 should not know the secret of 1. Then no sequence of calls lead to a solution, since every agent that 1 calls cannot be called by 3 and conversely.

Now we skip to higher-order order goals. Take $k = 2$ and suppose we want 1 not to know whether 2 knows the secret of 3 (but we do want 1 and 2 to know the secret of 3):

$$\big(EK_{Agt\setminus\{1\}}EK_{Agt}s_{Agt}\big) \wedge \big((K_1EK_{Agt\setminus\{2\}}s_{Agt} \wedge K_1K_2s_{Agt\setminus\{3\}}) \wedge \neg K_1K_2s_3\big).$$

Then the following protocol gives a solution:

1. 2 calls every agent but 3;
2. all agents but 2 call themselves until full knowledge of depth 2 is acquired;
3. 2 calls every agent but 1.

After the second step, every agent i different from 2 has almost reached her goal, except that she does not know whether 2 knows the secret of 3 (because it is not the case yet). At the third step, 2 calls everyone but 1, to learn the secret of 3, acquire the required depth of knowledge and inform every other agent.

Observe that if we increase the depth k, the goal that *only* 1 does not to know whether 2 knows the secret of 3 becomes unsolvable. By the truth axiom T of epistemic logic, we have, for example:

$$K_1K_4K_2s_3 \to K_1K_2s_3,$$

and hence the latter cannot be false without the former being false. Therefore for $k > 2$, the correct specification will be that every goal of the form

$$K_1K_{i_1} \dots K_{i_m}K_2K_{j_1} \dots K_{j_p}s_3$$

is false, for $m{+}p{+}2 \leq k$.

3.8 Conclusion

We have provided a logical analysis of the gossip problem, focusing on how higher-order shared knowledge can be obtained. DEL-PAO integration of knowledge modalities and dynamic modalities provides a handy language in order to reason about concepts such as an agent being a semi-expert, which is pivotal in our algorithm. With DEL-PAO, we were able to prove both the correctness and optimality of our algorithm, generalizing the results from [Baker and Shostak, 1972; Tijdeman, 1971; Hajnal et al., 1972] on the necessary number of calls.

The gossip problem recently attracted quite some attention in the dynamic epistemic logic community [Attamah et al., 2014a,b; van Ditmarsch et al., 2015]. We believe that our generalization as well as these variations provide interesting, canonical multi-agent problems in the field of epistemic logic and automated planning, that can be compared to the blocksworld in classical planning. First steps towards the latter, as well as an encoding of the generalized gossip problem with and without ignorance goals, are reported in Chapter 4.

4 A simple account of multi-agent epistemic planning

The generalized 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.

In this chapter, we apply DEL-PAO *to epistemic planning, as it allows us to model actions and epistemic planning tasks such as the gossip problem.*

While DEL-PAO visibility atoms and epistemic formulas allow us to model epistemic planning problems and to reduce conditions and goals to boolean formulas, we are able to formalize the existence of a plan with its dynamic operators, giving the complexity result. We also study an encoding of actions into PDDL, the standard Planning Domain Definition Language [McDermott et al., 1998]. This allows us to find a plan efficiently with a PDDL planner, which we do with the generalized gossip problem and with the "exam problem" where truth values of facts can also evolve.

Contents

Résumé

Le problème du bavardage généralisé peut être considéré comme le problème de planification multi-agent le plus simple : seuls les connaissances des agents évoluent, alors que l'état du monde reste inchangé. Dans ce chapitre, nous appliquons DEL-PAO à la planification épistémique : elle nous permet de modéliser des actions et des tâches de planification épistémiques comme le problème du bavardage.

Alors que les atomes de visibilité et les formules épistémiques de DEL-PAO nous permettent de modéliser des problèmes de planification épistémique et de réduire les conditions et les buts à des formules booléennes, nous sommes en mesure de formaliser l'existence d'un plan avec ses opérateurs dynamiques, donnant le résultat de complexité. Nous décrivons également un encodage des actions en PDDL [McDermott et al., 1998]. Cela nous permet de trouver un plan efficace à l'aide un planificateur PDDL, ce que nous faisons avec le problème du bavardage généralisé et avec le "problème de l'examen" où les valeurs de vérité des propositions peuvent aussi évoluer.

4.1 DEL-PAO^S: DEL-PAO without joint visibility

In this chapter, we consider the fragment of DEL-PAO without the JS operator, and thus without common knowledge. This avoids dealing with introspective consequences; we will come back to this in Section 4.4.4.

We call this logic DEL-PAO^S.

4.1.1 Language of DEL-PAO^S

Recall that $Prop$ is a countable non-empty set of propositional variables and Agt is a finite non-empty set of agents.

The set of visibility operators is now:

$$OBS_S = \{S_i : i \in Agt\}.$$

Like in DEL-PAO, the set of all sequences of visibility operators is noted ${OBS_S}^*$.

The set of atomic formulas becomes:

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

The language of programs and formulas of DEL-PAO^S is defined by the following grammar:

$$\pi ::= +\alpha \mid -\alpha \mid (\pi;\pi) \mid (\pi \sqcup \pi) \mid \pi^* \mid \varphi?$$
$$\varphi ::= \alpha \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi \mid [\pi]\varphi$$

where α ranges over ATM and i over Agt.

Programs and all operators are to be read like in DEL-PAO (see Section 2.1). The other boolean operators and dual modal operators abbreviate like in DEL-PAO (i.e., in the standard way). The set of atoms appearing in the formula φ and in the program π, noted $ATM(\varphi)$ and $ATM(\pi)$, are also defined like in DEL-PAO. Hence the only difference is that JS operators and the common knowledge modal operator CK are not included. We note Fml_{bool} the set of boolean formulas, i.e., formulas that do not contain K_i or $[\pi]$.

4.1.2 Semantics of DEL-PAO^S

In this chapter, we denote valuations by s or t instead of V, as automated planning usually deals with "states" instead of worlds or valuations.

Since we do not include joint visibility operators, a state $s \in 2^{ATM}$ is introspective if and only if it contains every atom of the form $\sigma\, S_i\, S_i\, \alpha$ with $\sigma \in {OBS_S}^*$. Therefore an atom is valid in $INTR$ if and only if it is of the form $\sigma\, S_i\, S_i\, \alpha$ with $\sigma \in {OBS_S}^*$.

Accessibility relations for K_i are defined exactly like in DEL-PAO:

$$s \sim_i s' \text{ iff } S_i\,\alpha \in s \text{ implies } s(\alpha) = s'(\alpha)$$

where $s(\alpha) = s'(\alpha)$ when either $\alpha \in s$ and $\alpha \in s'$, or $\alpha \notin s$ and $\alpha \notin s'$.

We verify that the introspective properties of DEL-PAO still hold.

Proposition 4.1. *All relations $\sim_i$ are equivalence relations on $INTR$.*

Proof. The proof of Proposition 2.6 on page 35 is still valid. □

Proposition 4.2. *Let* $s \in INTR$ *and* $s' \in 2^{ATM}$. *If* $s \sim_i s'$ *then* $s' \in INTR$.

Proof. Suppose that $s \sim_i s'$ for an arbitrary agent i and that $s \in INTR$.

Take an arbitrary atom α of the form $\sigma\, S_j\, S_j\, \alpha$ with $\sigma \in OBS^*$. Since $s \in INTR$, α belongs to s, and moreover $S_i\, \alpha$ also belongs to s. Thus α belongs to s' for every $s \sim_i s'$.

Therefore $s' \in INTR$. □

For the dynamic part, we do not have to deal with introspective consequences as we do not include the JS operator. Valuations updates become simpler than in DEL-PAO:

$$s{+}\alpha = s \cup \{\alpha\}$$
$$s{-}\alpha = s \setminus \{\alpha\}$$

Obviously, if s is introspective, the only way to lead to a non-introspective valuation is to remove an introspectively valid atom. Therefore when s is introspective then both $s{+}\alpha$ and $s{-}\alpha$ are as well, unless α is valid in $INTR$.

Truth conditions are as follows:

$s \models \alpha$	iff $\alpha \in s$
$s \models \neg\varphi$	iff not $(s \models \varphi)$
$s \models \varphi \wedge \varphi'$	iff $s \models \varphi$ and $s \models \varphi'$
$s \models K_i\varphi$	iff $s' \models \varphi$ for every s' such that $s \sim_i s'$
$s \models [\pi]\varphi$	iff $s' \models \varphi$ for every s' such that $sR_\pi s'$

where R_π is a binary relation on valuations defined by:

$sR_{+\alpha}s'$	iff $s' = s{+}\alpha$
$sR_{-\alpha}s'$	iff $s' = s{-}\alpha$ and α not valid in $INTR$
$sR_{\pi_1;\pi_2}s'$	iff $s(R_\pi \circ R_{\pi'})s'$
$sR_{\pi_1 \sqcup \pi_2}s'$	iff $s(R_\pi \cup R_{\pi'})s'$
$sR_{\pi^*}s'$	iff $s(\bigcup_{k\in\mathbb{N}_0} (R_\pi)^k)s'$
$sR_{\varphi?}s'$	iff $s = s'$ and $s \models \varphi$

Truth conditions are identical to DEL-PAO (see Section 2.2.5) for epistemic operators; for dynamic operators, as we have mentioned, the relation for assignments is simplified.

4.2 Epistemic planning with conditional effects

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

4.2.1 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, $\mathit{ceff}^{+}(ce_1) \cap \mathit{ceff}^{-}(ce_2) = \emptyset$.

We extend the set of atoms to an action a as expected:

$$ATM(\mathsf{a}) = ATM(pre(\mathsf{a})) \cup \Big(\bigcup_{ce \in \mathit{eff}(\mathsf{a})} ATM(cnd(ce)) \cup \mathit{ceff}^{+}(ce) \cup \mathit{ceff}^{-}(ce)\Big)$$

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:

$$\begin{aligned} pre(\mathsf{toggle}_p) &= \top \\ \mathit{eff}(\mathsf{toggle}_p) &= \{\langle p, \emptyset, \{p\}\rangle, \langle \neg p, \{p\}, \emptyset\rangle\} \end{aligned}$$

The conditions p and $\neg p$ are inconsistent, thus not leading to conflict.

Example 4.1. Suppose we want to model the original gossip problem, i.e., the gossip problem presented in Chapter 3 with a depth $k = 1$. Let $Agt = \{1, \ldots, n\}$ and $Prop = \{s_i : i \in Agt\}$. As in Chapter 3, each propositional variable s_i represents the secret of agent i. (We suppose each s_i is true.)

During the action Call^i_j, we have seen that 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:

$$\begin{aligned} \mathit{eff}(\mathsf{Call}^i_j) = \{&\langle S_i\, s_1 \vee S_j\, s_1, \{S_i\, s_1, S_j\, s_1\}, \emptyset\rangle, \\ &\ldots, \\ &\langle S_i\, s_n \vee S_j\, s_n, \{S_i\, s_n, S_j\, s_n\}, \emptyset\rangle\}. \end{aligned}$$

Unlike the specification of Chapter 3, here formulas must not contain epistemic operators. Since secrets will be initially true and actions do not make them false, the condition $K_i\, s_\ell \vee K_j\, s_\ell$ is equivalent to $S_i\, s_\ell \vee S_j\, s_\ell$, hence the current specification.

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:

$$
\begin{aligned}
s R_{\mathsf{a}} s' \text{ iff } \quad & s \models \mathit{pre}(\mathsf{a}), \text{ and} \\
& \text{for every } \mathit{ce} \in \mathit{eff}(\mathsf{a}) \text{ such that} \\
& \qquad (\mathit{ceff}^+(\mathit{ce}) \cup \mathit{ceff}^-(\mathit{ce})) \cap \{\alpha : \alpha \text{ is valid in } \mathit{INTR}\} \neq \emptyset, \\
& \qquad s \not\models \mathit{cnd}(\mathit{ce}), \text{ and} \\
& s' = \Big(s \setminus \bigcup_{\substack{\mathit{ce} \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models \mathit{cnd}(\mathit{ce})}} \mathit{ceff}^-(\mathit{ce})\Big) \cup \bigcup_{\substack{\mathit{ce} \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models \mathit{cnd}(\mathit{ce})}} \mathit{ceff}^+(\mathit{ce}).
\end{aligned}
$$

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 will see why this constraint on introspective atoms is required in Section 4.3.

4.2.2 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 exists a sequence of actions $\mathsf{a}_1, \ldots, \mathsf{a}_m \in \mathit{Act}$ and a sequence of states $t_0, \ldots, t_m \in 2^{\mathit{ATM}}$ with $m \geq 0$ such that $s_0 = t_0$, $s = t_m$ and $t_{k-1} R_{\mathsf{a}_k} t_k$ for every k such that $1 \leq k \leq m$.

A simple epistemic *planning task* is a triple $\mathcal{P} = \langle \mathit{Act}, s_0, \mathit{Goal} \rangle$ where Act is a finite set of actions, $s_0 \in 2^{\mathit{ATM}}$ is a finite state (the initial state) and $\mathit{Goal} \in \mathit{Fml}_{\mathit{bool}}$ is a boolean formula. It is *solvable* if at least one state s such that $s \models \mathit{Goal}$ is reachable from s_0 via Act; otherwise it is unsolvable.

We define the set of atoms of a planning task $\mathcal{P} = \langle \mathit{Act}, s_0, \mathit{Goal} \rangle$ as expected:

$$
\mathit{ATM}(\mathcal{P}) = \Big(\bigcup_{\mathsf{a} \in \mathit{Act}} \mathit{ATM}(\mathsf{a}) \Big) \cup s_0 \cup \mathit{ATM}(\mathit{Goal})
$$

Example 4.2. The planning task corresponding to the original gossip problem is $G_1 = \langle \mathit{Act}^{G_1}, s_0^{G_1}, \mathit{Goal}^{G_1} \rangle$ with:

- $\mathit{Act}^{G_1} = \{\mathsf{Call}^i_j : i, j \in \mathit{Agt} \text{ and } i \neq j\}$;
- $s_0^{G_1} = \{S_i\, s_i : i \in \mathit{Agt}\} \cup \{s_i : i \in \mathit{Agt}\}$;

- $Goal^{G_1} = \bigwedge_{i,j \in Agt} S_i\, s_j$,

as specified in Chapter 3 (such that, like conditions in Example 4.1, the goal formula must not contain epistemic operators).

4.3 Normal forms

The depth of atoms being unbounded, introspective states are infinite since they contain every atom of the form $\sigma\, S_i\, S_i\, \alpha$ with $\sigma \in OBS_S{}^*$. However, we want to deal with finite states, especially when encoding our problem into PDDL (see Section 4.5). We thereby impose that planning tasks are in *normal form*: they must not include an introspectively valid atom. In this form, we show that they are solvable starting from an introspective, but infinite initial state if and only if they are from the same state without atoms valid in $INTR$.

4.3.1 Boolean formulas

We say that a boolean formula φ is in *normal form* if and only if for every $\alpha \in ATM(\varphi)$, α is not valid in $INTR$.

Given a boolean formula φ, the following procedure produces a formula in normal form.

Procedure 4.1. For every $\alpha \in ATM(\varphi)$ such that α is valid in $INTR$, replace every occurrence of α in φ by $\top$.

We note the resulting formula $NF(\varphi)$.

Proposition 4.3. *Let φ be a boolean formula. Then*

$$\varphi \leftrightarrow NF(\varphi)$$

is valid in $INTR$.

Proof. Procedure 4.1 replaces every introspectively valid atom α by $\top$. For such α, $\alpha \leftrightarrow \top$ is valid in $INTR$. Therefore $NF(\varphi)$ is equivalent to φ on introspective states because the rule of replacement of equivalents preserves plain validity. □

Proposition 4.4. *For every state $s \in 2^{ATM}$, every boolean formula φ in normal form, and every α valid in $INTR$, we have*

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

Proof. Since φ is in normal form, any α which is valid in $INTR$ does not belong to $ATM(\varphi)$. Hence since φ is boolean, by Proposition 2.9 on page 38, only atoms from $ATM(\varphi)$ influence the truth value of φ, and thus, no introspectively valid α do. In other words, $s \backslash \{\alpha\} \models \varphi$ if and only if $s \cup \{\alpha\} \models \varphi$. □

4.3.2 Actions

We say that the action a is in *normal form* if and only if:

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

Given an action a, the following procedure produces an action in normal form.

Procedure 4.2.

1. For every conditional effect $ce \in \mathit{eff}(\mathsf{a})$ such that $(\mathit{ceff}^{+}(ce) \cup \mathit{ceff}^{-}(ce)) \cap \{\alpha : \alpha \text{ is valid in } \mathit{INTR}\} \neq \emptyset$, replace the precondition $pre(\mathsf{a})$ by $pre(\mathsf{a}) \wedge \neg cnd(ce)$ and remove ce from $\mathit{eff}(\mathsf{a})$;
2. Put the resulting $pre(\mathsf{a})$ and $cnd(ce)$, for every $ce \in \mathit{eff}(\mathsf{a})$, in normal form using Procedure 4.1.

We note the resulting action $NF(\mathsf{a})$.

Proposition 4.5. *Let* a *be an action. Then*

$$sR_{\mathsf{a}}t \text{ if and only if } sR_{NF(\mathsf{a})}t$$

for every $s, t \in \mathit{INTR}$.

Proof. First we can remark that by the definition of R_{a}, no introspectively valid atom can be modified by a. Therefore if $s \in \mathit{INTR}$, then $t \in \mathit{INTR}$. (In other words, we do not "miss" any valuation by assuming that $t \in \mathit{INTR}$.)

We split the set of conditional effects of a into two parts depending on whether they contain an introspectively valid atom or not:

$$\begin{aligned} CE_{\mathit{INTR}} = \{&ce \in \mathit{eff}(\mathsf{a}) : \\ &(\mathit{ceff}^{+}(ce) \cup \mathit{ceff}^{-}(ce)) \cap \{\alpha : \alpha \text{ is valid in } \mathit{INTR}\} \neq \emptyset\} \\ CE_{\not{\mathit{INTR}}} = \{&ce \in \mathit{eff}(\mathsf{a}) : \\ &(\mathit{ceff}^{+}(ce) \cup \mathit{ceff}^{-}(ce)) \cap \{\alpha : \alpha \text{ is valid in } \mathit{INTR}\} = \emptyset\} \end{aligned}$$

Therefore we can characterise $NF(\mathsf{a})$, returned by Procedure 4.2, by:

- $pre(NF(\mathsf{a})) = NF(pre(\mathsf{a}) \wedge \bigwedge_{ce \in CE_{\mathit{INTR}}} \neg cnd(ce))$;
- $\mathit{eff}(NF(\mathsf{a})) = \{\langle NF(cnd(ce)), \mathit{ceff}^{+}(ce), \mathit{ceff}^{-}(ce)\rangle : ce \in CE_{\not{\mathit{INTR}}}\}$.

Intuitively, every conditional effect from CE_{INTR} may modify an introspective atom, and thus is removed from the set of effects, while the negation of its condition is added to $pre(\mathsf{a})$. Moreover, every formula is put to normal form.

We know that:

$$sR_{\mathsf{a}}t \text{ iff } \quad s \models pre(\mathsf{a}), \text{ and} \tag{4.1}$$

$$\text{for every } ce \in CE_{INTR}, s \not\models cnd(ce), \text{ and} \tag{4.2}$$

$$t = \Big(s \setminus \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} \mathit{eff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} \mathit{eff}^{+}(ce). \tag{4.3}$$

On the other hand, the specification of $sR_{NF(\mathsf{a})}t$ is simpler because the problematic effects have been removed:

$$sR_{NF(\mathsf{a})}t \text{ iff } \quad s \models NF(pre(\mathsf{a}) \wedge \bigwedge_{ce \in CE_{INTR}} \neg cnd(ce)), \text{ and} \tag{4.4}$$

$$t = \Big(s \setminus \bigcup_{\substack{ce \in CE_{\cancel{INTR}} \\ \text{and } s \models NF(cnd(ce))}} \mathit{eff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce \in CE_{\cancel{INTR}} \\ \text{and } s \models NF(cnd(ce))}} \mathit{eff}^{+}(ce). \tag{4.5}$$

To prove that $sR_{\mathsf{a}}t$ and $sR_{NF(\mathsf{a})}t$ are equivalent, we are going to prove that (4.1) together with (4.2) are equivalent to (4.4) and that (4.2) together with (4.3) are equivalent to (4.5).

- (4.1) and (4.2) together are equivalent to:

  $$s \models pre(\mathsf{a}) \wedge \bigwedge_{ce \in CE_{INTR}} \neg cnd(ce),$$

  which, by Proposition 4.3, is equivalent to:

  $$s \models NF(pre(\mathsf{a}) \wedge \bigwedge_{ce \in CE_{INTR}} \neg cnd(ce))$$

  since s in introspective.

- Since by (4.2), for every $ce \in CE_{INTR}$, $s \not\models cnd(ce)$, (4.3) is equivalent to:

  $$t = \Big(s \setminus \bigcup_{\substack{ce \in CE_{\cancel{INTR}} \\ \text{and } s \models cnd(ce)}} \mathit{eff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce \in CE_{\cancel{INTR}} \\ \text{and } s \models cnd(ce)}} \mathit{eff}^{+}(ce)$$

  which is equivalent, by Proposition 4.3, to:

  $$t = \Big(s \setminus \bigcup_{\substack{ce \in CE_{\cancel{INTR}} \\ \text{and } s \models NF(cnd(ce))}} \mathit{eff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce \in CE_{\cancel{INTR}} \\ \text{and } s \models NF(cnd(ce))}} \mathit{eff}^{+}(ce)$$

  since s in introspective.

Therefore $sR_{\mathsf{a}}t$ if and only if $sR_{NF(\mathsf{a})}t$. □

Proposition 4.6. *For every* $s, t \in 2^{ATM}$*, every action* a *in normal form, and every* α *valid in* $INTR$*,*

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

Proof. Since a is in normal form, we have:

$$\begin{aligned} s\backslash\{\alpha\}\, R_{\mathsf{a}}\, t\backslash\{\alpha\} \;\text{ iff }\quad & s\backslash\{\alpha\} \models \mathit{pre}(\mathsf{a}), \text{ and} \\ & t\backslash\{\alpha\} = \Big((s\backslash\{\alpha\}) \setminus \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\backslash\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\backslash\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{+}(ce) \end{aligned}$$

and:

$$\begin{aligned} s\cup\{\alpha\}\, R_{\mathsf{a}}\, t\cup\{\alpha\} \;\text{ iff }\quad & s\cup\{\alpha\} \models \mathit{pre}(\mathsf{a}), \text{ and} \\ & t\cup\{\alpha\} = \Big((s\cup\{\alpha\}) \setminus \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\cup\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\cup\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{+}(ce). \end{aligned}$$

We examine each part that differs between the two definitions.

- Since a is in normal form, $\mathit{pre}(\mathsf{a})$ is in normal form. Hence by Proposition 4.4, $s\backslash\{\alpha\} \models \mathit{pre}(\mathsf{a})$ is equivalent to $s\cup\{\alpha\} \models \mathit{pre}(\mathsf{a})$ because α valid in $INTR$.
- Since a is in normal form, $\mathit{cnd}(ce)$, for every $ce \in \mathit{eff}(\mathsf{a})$, is also in normal form. Therefore, again by Proposition 4.4, $s\backslash\{\alpha\} \models \mathit{cnd}(ce)$ is equivalent to $s\cup\{\alpha\} \models \mathit{cnd}(ce)$.
- Finally, and again since a is in normal form, α does not belong to $\mathit{ceff}^{+}(ce)$ or $\mathit{ceff}^{-}(ce)$, for every $ce \in \mathit{eff}(\mathsf{a})$. Thus

  $$t\backslash\{\alpha\} = \Big((s\backslash\{\alpha\}) \setminus \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\backslash\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\backslash\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{+}(ce)$$

  if and only if

  $$t\cup\{\alpha\} = \Big((s\cup\{\alpha\}) \setminus \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\cup\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{-}(ce)\Big) \cup \bigcup_{\substack{ce\in \mathit{eff}(\mathsf{a}) \\ \text{and } s\cup\{\alpha\}\models \mathit{cnd}(ce)}} \mathit{ceff}^{+}(ce).$$

Therefore $s\backslash\{\alpha\}\, R_{\mathsf{a}}\, t\backslash\{\alpha\}$ is equivalent to $s\cup\{\alpha\}\, R_{\mathsf{a}}\, t\cup\{\alpha\}$. □

4.3.3 Planning tasks

The planning task $\mathcal{P} = \langle Act, s_0, Goal \rangle$ is in *normal form* if and only if:

- every action $\mathsf{a} \in Act$ is in normal form;
- the formula $Goal$ is in normal form.

Propositions 4.4 and 4.6 imply that $\mathcal{P} = \langle Act, s_0, Goal \rangle$ is solvable if and only if $\mathcal{P}' = \langle Act, s_0', Goal \rangle$ with $s_0' = s_0 \setminus \{\alpha : \alpha \text{ is valid in } INTR\}$ is solvable.

Observe that Example 4.2 is trivially in normal form since no higher-order knowledge is involved. We will however need to be careful when specifying the generalized gossip problem (see Section 4.6.2).

One could wonder why we did not apply the technique we used in DEL-PAO for defining the model checking problem, that consists in restricting the valuation to (relevant) atoms appearing in the actions and in the goal. This would indeed be an alternative. However, the transformation of epistemic formulas into boolean formulas is required for encoding the planning task into PDDL; we believe that this method is more intuitive and less technical than the one we previously used.

4.4 Complexity results

Consider the planning task $\mathcal{P} = \langle Act, s_0, Goal \rangle$. In this section, we show how actions from Act can be encoded into DEL-PAO^S programs. Then we prove that the solvability of $\mathcal{P}$ is in PSPACE by showing that it can be polynomially reduced to a DEL-PAO^S model checking problem.

4.4.1 Storing variables

The conditional effects of the actions that we have defined in Section 4.2 are produced in parallel. We have to simulate this in DEL-PAO^S by sequential composition. We therefore have to take care that the truth value of no condition is modified by an effect. To achieve this, we store the values of our conditions before executing the 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:

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

Lemma 4.1. *If c does not occur in φ, then the equivalence*

$$\varphi \leftrightarrow [\mathsf{str}(\varphi, c)]\, c$$

is plainly valid.

Proof. Take an arbitrary valuation $s \in 2^{ATM}$. Using axioms from Proposition 2.11 on page 50 that are plainly valid, we have:

$$\begin{aligned}
& s \models [\mathsf{str}(\varphi, c)]\, c \\
\Leftrightarrow\ & s \models [\textbf{if } \varphi \textbf{ then } +c \textbf{ else } -c]\, c \\
\Leftrightarrow\ & s \models [(\varphi?; +c) \sqcup (\neg\varphi?; -c)]\, c \\
\Leftrightarrow\ & s \models [\varphi?; +c]\, c \wedge [\neg\varphi?; -c]\, c \\
\Leftrightarrow\ & s \models (\varphi \to [+c]\, c) \wedge (\neg\varphi \to [-c]\, c) \\
\Leftrightarrow\ & s \models (\varphi \to \top) \wedge (\neg\varphi \to \bot) && \text{(since } c \text{ is not valid in } \mathit{INTR}\text{)} \\
\Leftrightarrow\ & s \models \varphi \wedge \varphi \\
\Leftrightarrow\ & s \models \varphi,
\end{aligned}$$

hence the result. □

After the execution of our program, we want all storage variables to be put to false so that we do not have to worry about them later. The program resetting the value of a given set of storage variables is simply defined as

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

4.4.2 Encoding of actions

Intuitively, an action is a DEL-PAO^S program, only executed if the precondition is fulfilled, applying each conditional effect whose condition is satisfied. For example, the action toggle_p (flipping the value of the variable p) corresponds to the program:

$$\mathsf{str}(p, c_1); \mathsf{str}(\neg p, c_2); \textbf{if } c_1 \textbf{ then } -p; \textbf{if } c_2 \textbf{ then } +p.$$

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^S program exeAct(a):

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

with $\mathit{eff}(\mathsf{a}) = \{ce_1, \ldots, ce_m\}$.

Proposition 4.7. *For every $s, t \in 2^{ATM}$ such that s does not contain any storage variable, and every action* a *in normal form:*

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

Proof. By the definition of $R_{\mathsf{exeAct(a)}}$, we have:

$$\begin{aligned}s \; R_{\mathsf{exeAct(a)}} \; t \;\text{ iff }\; & \text{there exist } s_1, s_2, s_3 \in 2^{ATM} \text{ such that} \\ & s R_{pre(\mathsf{a})?} \, s_1, \text{ and} \\ & s_1 R_{\mathsf{str}(cnd(ce_1),c_1);\ldots;\mathsf{str}(cnd(ce_m),c_m)} \, s_2, \text{ and} \\ & s_2 R_{\mathsf{exeCE}(ce_1,c_1);\ldots;\mathsf{exeCE}(ce_m,c_m)} \, s_3, \text{ and} \\ & s_3 R_{\mathsf{rst}(\{c_1,\ldots,c_m\})} t,\end{aligned}$$

that is:

$$\begin{aligned}s \; R_{\mathsf{exeAct(a)}} \; t \;\text{ iff }\; & \text{there exist } s_2, s_3 \in 2^{ATM} \text{ such that} \\ & s \models pre(\mathsf{a}), \text{ and} \\ & s R_{\mathsf{str}(cnd(ce_1),c_1);\ldots;\mathsf{str}(cnd(ce_m),c_m)} \, s_2, \text{ and} \\ & s_2 R_{\mathsf{exeCE}(ce_1,c_1);\ldots;\mathsf{exeCE}(ce_m,c_m)} \, s_3, \text{ and} \\ & s_3 R_{\mathsf{rst}(\{c_1,\ldots,c_m\})} t.\end{aligned}$$

Observe that:

- First, the storage variable c_i does not appear in the boolean formula $cnd(ce_j)$. Since $\mathsf{str}(cnd(ce_i), c_i)$ only modifies c_i, it does not change the truth value of $cnd(ce_j)$, for any j. Therefore between s and s_2, during the execution of $\mathsf{str}(cnd(ce_1), c_1); \ldots; \mathsf{str}(cnd(ce_m), c_m)$, every $cnd(ce_j)$ keeps the same value, i.e., the value it had in s.
- Second, the storage variable c_i does not appear in effects $\mathit{ceff}^+(ce_j)$ and $\mathit{ceff}^-(ce_j)$. Hence $\mathsf{exeCE}(ce_j, c_j)$ does not modify c_i, for any j. Therefore between s_2 and s_3, during the execution of $\mathsf{exeCE}(ce_1, c_1); \ldots; \mathsf{exeCE}(ce_m, c_m)$, every c_i keeps the same value, i.e., the value it had in s_2. By Lemma 4.1, we know that the value

of c_i in s_2 is the value of the corresponding condition $cnd(ce_i)$ before the execution of $\mathsf{str}(cnd(ce_i), c_i)$, i.e., in s (since we have seen that the execution of $\mathsf{str}(cnd(ce_j), c_j)$ does not change the truth value of $cnd(ce_i)$).

- Third, the program $\mathsf{rst}(\{c_1, \ldots, c_m\})$ makes every storage variable c_i false.

With this in mind, and in order to simplify the writing, let us define:

$$\begin{aligned} C_{\not\models} &= \{c_i : ce_i \in \mathit{eff}(\mathsf{a}) \text{ and } s \not\models cnd(ce_i)\} \\ C_{\models} &= \{c_i : ce_i \in \mathit{eff}(\mathsf{a}) \text{ and } s \models cnd(ce_i)\} \\ C &= \{c_i : ce_i \in \mathit{eff}(\mathsf{a})\} \\ CEFF^- &= \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} \mathit{ceff}^-(ce) \\ CEFF^+ &= \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} \mathit{ceff}^+(ce) \end{aligned}$$

Obviously $C_{\not\models} \cup C_{\models} = C$ and $C_{\not\models}$ and $C_{\models}$ are disjoint with $CEFF^-$ and $CEFF^+$.

We obtain:

$$\begin{aligned} s\, R_{\mathsf{exeAct(a)}}\, t \text{ iff there exist } s_2, s_3 \in 2^{ATM} \text{ such that} \\ s \models \mathit{pre}(\mathsf{a}), \text{ and} \\ s_2 = (s \setminus C_{\not\models}) \cup C_{\models}, \text{ and} \\ s_3 = (s_2 \setminus CEFF^-) \cup CEFF^+, \text{ and} \\ t = s_3 \setminus C, \end{aligned}$$

i.e.:

$$\begin{aligned} s\, R_{\mathsf{exeAct(a)}}\, t \text{ iff } & s \models \mathit{pre}(\mathsf{a}), \text{ and} \\ & t = ((((s \setminus C_{\not\models}) \cup C_{\models}) \setminus CEFF^-) \cup CEFF^+) \setminus C. \end{aligned}$$

Since $C_{\not\models}$ and $C_{\models}$ are disjoint with $CEFF^-$ and $CEFF^+$, we can reorder operations to obtain:

$$\begin{aligned} s\, R_{\mathsf{exeAct(a)}}\, t \text{ iff } & s \models \mathit{pre}(\mathsf{a}), \text{ and} \\ & t = ((((s \setminus CEFF^-) \cup CEFF^+) \setminus C_{\not\models}) \cup C_{\models}) \setminus C, \end{aligned}$$

which, since $C_{\not\models} \cup C_{\models} = C$, gives:

$$s\, R_{\mathsf{exeAct(a)}}\, t \text{ iff } s \models \mathit{pre}(\mathsf{a}) \text{ and } t = (s \setminus CEFF^-) \cup CEFF^+,$$

or in other words:

$$s\, R_{\mathsf{exeAct(a)}}\, t \text{ iff } s \models pre(\mathsf{a}) \text{ and } t = \Big(s \setminus \bigcup_{\substack{ce \in eff(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} ceff^{-}(ce)\Big) \cup \bigcup_{\substack{ce \in eff(\mathsf{a}) \\ \text{and } s \models cnd(ce)}} ceff^{+}(ce).$$

Therefore $s\, R_{\mathsf{exeAct(a)}}\, t$ if and only if $s\, R_{\mathsf{a}}\, t$. □

Example 4.3 (Example 4.1, ctd.)**.** The action Call^i_j, for any $i, j \in Agt$, is associated to the program:

$$\begin{aligned}\mathsf{exeAct}(\mathsf{Call}^i_j) = {} & \top?; \\ & \mathsf{str}(S_i\, s_1 \vee S_j\, s_1, c_1); \ldots; \mathsf{str}(S_i\, s_n \vee S_j\, s_n, c_n); \\ & \textbf{if } c_1 \textbf{ then } {+}S_i\, s_1; {+}S_j\, s_1; \\ & \ldots; \\ & \textbf{if } c_n \textbf{ then } {+}S_i\, s_n; {+}S_j\, s_n; \\ & \mathsf{rst}(\{c_1, \ldots, c_n\})\end{aligned}$$

Note that in this case, $pre(\mathsf{Call}^i_j)?$ can clearly be dropped.

4.4.3 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^S.

Proposition 4.8. *A planning task $\langle Act, s_0, Goal\rangle$ in normal form such that $ATM(\langle Act, s_0, Goal\rangle)$ does not contain any storage variable is solvable if and only if*

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

Proof. By the truth conditions, we have:

$$\begin{aligned}s_0 \models \big\langle \big(\bigsqcup_{\mathsf{a} \in Act} \mathsf{exeAct(a)}\big)^* \big\rangle Goal \text{ iff } & \\ & \text{there exist } t_0, \ldots, t_m \in 2^{ATM} \text{ with } m \geq 0 \text{ such that} \\ & s_0 = t_0, \text{ and} \\ & \text{for every } 1 \leq k \leq m, t_{k-1}\, R_{\bigsqcup_{\mathsf{a} \in Act} \mathsf{exeAct(a)}}\, t_k, \text{ and} \\ & t_m \models Goal,\end{aligned}$$

that is, if we detail $\bigsqcup_{\mathsf{a}\in Act} \mathsf{exeAct}(\mathsf{a})$:

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

there exist $t_0, \ldots, t_m \in 2^{ATM}$ with $m \geq 0$ such that

$s_0 = t_0$, and

for every $1 \leq k \leq m$, there exists $\mathsf{a} \in Act$ such that $t_{k-1}\, R_{\mathsf{exeAct}(\mathsf{a})}\, t_k$, and

$t_m \models Goal$.

By Proposition 4.7, we thus have:

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

there exist $t_0, \ldots, t_m \in 2^{ATM}$ with $m \geq 0$ such that

$s_0 = t_0$, and

for every $1 \leq k \leq m$, there exists $\mathsf{a} \in Act$ such that $t_{k-1}\, R_{\mathsf{a}}\, t_k$, and

$t_m \models Goal$.

For the equivalence between $R_{\mathsf{exeAct}(\mathsf{a})}$ and R_{a} to be valid, every t_{k-1} must not contain any storage variable. This is the case because:

1. First, we assumed that s_0 does not contain any storage variable;
2. Second, assuming that t_{k-1} does not contain any storage variable, ending $\mathsf{exeAct}(\mathsf{a})$ with $\mathsf{rst}(\{c_1, \ldots, c_m\})$ ensures that all used storage variables are set to false, or in other words, that t_k does not contain any storage variable.

Therefore:

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

there exist $t_0, \ldots, t_m \in 2^{ATM}$ and $\mathsf{a}_1, \ldots, \mathsf{a}_m \in Act$ with $m \geq 0$ such that

$s_0 = t_0$, and

for every $1 \leq k \leq m$, $t_{k-1}\, R_{\mathsf{a}_k}\, t_k$, and

$t_m \models Goal$,

that is:

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

there exists $t_m \in 2^{ATM}$ such that t_m is reachable from s_0 via Act, and

$t_m \models Goal$,

i.e., $s_0 \models \langle (\bigsqcup_{\mathsf{a}\in Act} \mathsf{exeAct}(\mathsf{a}))^* \rangle Goal$ if and only if $\langle Act, s_0, Goal \rangle$ is solvable. □

It may seem that we assume many hypothesis on planning tasks, such as boolean formulas and actions in normal form. (This allows a straightforward translation into PDDL.) However, the previous procedures and propositions show how to transform a general planning task into a "well-behaved" one.

Suppose you have a finite set Act of pairs a of the form $\langle pre(\mathsf{a}), \mathit{eff}(\mathsf{a})\rangle$, where $pre(\mathsf{a})$ is a formula (that may contain epistemic operators), and $\mathit{eff}(\mathsf{a})$ is a set of tuples $\langle cnd(ce), \mathit{ceff}^{+}(ce), \mathit{ceff}^{-}(ce)\rangle$, where $cnd(ce)$ is a formula (that may contain epistemic operators) and $\mathit{ceff}^{+}(ce)$ and $\mathit{ceff}^{-}(ce)$ are set of atoms such that for every $ce_1, ce_2 \in \mathit{eff}(\mathsf{a})$, we have $\mathit{ceff}^{+}(ce_1) \cap \mathit{ceff}^{-}(ce_2) = \emptyset$. We cannot call them "actions" yet since the formulas appearing in these structures are not necessarily boolean formulas. Suppose you also have an initial state $s_0 \in \mathit{INTR}$, i.e., infinite, and a goal formula Goal that may also contain epistemic operators. Then:

1. Using Procedure 2.1 on page 52 of DEL-PAO, that is based on equivalences of Proposition 2.10 on page 44 which are still plainly valid, and on the rule of replacement of equivalents that still preserves plain validity, remove epistemic operators from Goal, $pre(\mathsf{a})$ for every $\mathsf{a} \in \mathit{Act}$, and $cnd(ce)$ for every $ce \in \mathit{eff}(\mathsf{a})$. After this step, all formulas are boolean formulas.

2. Following procedures 4.1 and 4.2, put Goal and every action $\mathsf{a} \in \mathit{Act}$ in normal form. By propositions 4.3 and 4.5, we know that the resulting formulas and actions are equivalent on introspective states.

3. Remove every atom α valid in INTR from s_0. By propositions 4.4 and 4.6, we know that formulas and actions in normal form behave in the same way in the original and new s_0.

4. Remove every atom α not appearing in the planning task from s_0, i.e., replace s_0 by

$$s_0 \cap \Big(\bigcup_{\mathsf{a} \in \mathit{Act}} \mathit{ATM}(\mathsf{a}) \cup \mathit{ATM}(\mathit{Goal}) \Big).$$

   Since there is no epistemic operator, by Proposition 2.9 on page 38, we know that formulas and actions will behave in the same way in the original and new s_0.

In the end, Act is a set of actions in normal form, s_0 is finite and and Goal is in normal form. Therefore we can apply Proposition 4.8 to solve the planning task $\langle \mathit{Act}, s_0, \mathit{Goal}\rangle$.

Theorem 4.1. *Deciding the solvability of a planning task with* DEL-PAO^{S} *is* PSPACE*-complete.*

Proof. For the lower bound, we know that deciding this problem is already PSPACE-hard in classical planning [Bylander, 1994].

For the upper bound, we reduced the problem to a problem of model checking of DEL-PAO^S by Proposition 4.8. We will see in Chapter 7 that DEL-PAO^S can be translated into DL-PA-PMP whose model checking is in PSPACE.

This establishes that deciding the solvability of a planning task in DEL-PAO^S is in PSPACE. □

This result compares favourably to DEL-based epistemic planning, which is undecidable even for simple fragments [Aucher and Bolander, 2013; Charrier et al., 2016a]. The difference is due to the simplicity of our underlying epistemic logic as well as to the limited expressivity of our actions: we can basically model private announcements, while DEL has more general event models.

4.4.4 Planning actions and joint visibility

We did not include the operator of joint visibility to our language. The reason behind this choice should appear more clearly now that we have given our encoding into DEL-PAO.[8]

In planning, when conditional effects are permitted, every effect of an action is assumed to be executed in parallel. This is not possible with our DEL-PAO programs: we must execute them in sequence. This leads to the specific measures we have taken: we have to include and deal with storage variables to ensure that the execution is running correctly. However, once this is taken into account, the execution of effects are not a problem any more: assignment programs can be performed in any order—we arbitrary chose to execute all positive effects, then all negative ones.

Now suppose JS operators are included; this also brings about introspective causes and consequences. We have seen in DEL-PAO (see Lemma 2.4 on page 47) that the ordering in sequences of assignments becomes important. For example, $+JS\,p; -JS\,S_i\,p$ is neither equivalent to $-JS\,S_i\,p; +JS\,p$, nor to $+JS\,p$, nor to $-JS\,S_i\,p$. Therefore even if effects are non-conflicting, arbitrarily choosing an ordering is not an option. An alternative would be to generate all possible orderings (we will see how to do this in Chapter 5 with the star) then to quantify over them with our dynamic modalities $[\pi]$ and $\langle\pi\rangle$. In this case, actions become non-deterministic. This deflects from our main objective to design a simple approach to epistemic planning.

It is interesting to mention that we have investigated ways to include the parallel operator " $\|$ " to DEL-PAO, without success. One of the

[8] This discussion was not present in [Cooper et al., 2016a].

reasons corresponds to the problem given above: deciding what should be the outcome of $+JS\,p \parallel -JS\,S_i\,p$ is not trivial (especially without non-determinism).

4.5 Encoding into PDDL

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

4.5.1 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:

$$tr_{PDDL}(S_{i_1} \dots S_{i_m}\, p) = \begin{cases} \texttt{(p)} & \text{if } m = 0 \\ \texttt{(S-m i1 ... im p)} & \text{otherwise} \end{cases}$$

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

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

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

In words, a visibility atom $\alpha = S_{i_1} \dots S_{i_m}\, p$ 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 $S_{i_1} \dots S_{i_m}\, p$). Other boolean operators are encoded as expected.

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

4.5.2 Encoding of actions

As we consider actions with conditional effects, 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:

$$\begin{array}{l}\texttt{(when } tr_{PDDL}(cnd(ce)) \\ \quad \texttt{(and } tr_{PDDL}(\alpha_1) \ \ldots \ tr_{PDDL}(\alpha_m) \\ \qquad \texttt{(not } tr_{PDDL}(\beta_1)) \ \ldots \ \texttt{(not } tr_{PDDL}(\beta_\ell))))\end{array}$$

Note that, again, the ordering is not important.

Example 4.4 (Example 4.1, 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. Observe that we could generalize it to any i and j by:

```
(:action call
  :parameters (?i ?j)
  :effect (and (forall (?s)
    (and
      (when (or (S-1 ?i ?s) (S-1 ?j ?s))
        (and (S-1 ?i ?s) (S-1 ?j ?s)))))))
```

Almost all planners from last International Planning Competition (IPC 2014)[9] handle conditional effects and negative preconditions, and most of them handle disjunctive preconditions. For experiments, we used the planner FDSS-2014 [Röger et al., 2014] that satisfied all these properties.

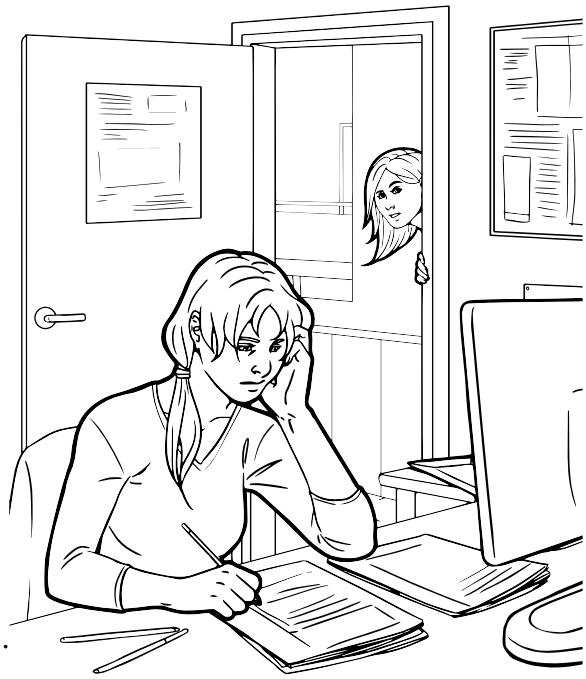
4.6 Applications

In this section, we first study the "exam problem" (a simple illustrative example concerning privacy of information), then the generalized gossip problem.[10]

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

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

4.6.1 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 us write the corresponding planning task $Exam = \langle Act^{Exam}, s_0^{Exam}, Goal^{Exam} \rangle$.

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 topic without being caught by the teacher. The goal is $S_s\, exam \wedge \neg K_t S_s\, exam \wedge \neg in_s$. In terms of visibility atoms, this gives:

$$\begin{aligned} Goal^{Exam} = {} & S_s\, exam \wedge \\ & \neg S_t\, S_s\, exam \wedge \\ & \neg in_s. \end{aligned}$$

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

$$\begin{aligned}
\mathsf{openAndGoIn}_t &= \langle \neg in_t, \{\langle \top, \{open, in_t, S_t\, S_s\, exam\}, \emptyset\rangle\}\rangle \\
\mathsf{goOutAndClose}_t &= \langle in_t \wedge \neg S_s\, exam, \langle \top, \emptyset, \{S_t\, S_s\, exam, in_t, open\}\rangle\}\rangle \\
\mathsf{goIn}_s &= \langle open \wedge \neg in_s, \{\langle \top, \{in_s\}, \emptyset\rangle\}\rangle \\
\mathsf{goOut}_s &= \langle open \wedge in_s, \{\langle \top, \emptyset, \{in_s\}\rangle\}\rangle \\
\mathsf{readExam}_s &= \langle in_s, \{\langle \top, \{S_s\, exam\}, \emptyset\rangle\}\rangle
\end{aligned}$$

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 $\neg S_s\, exam$ to ensure that the teacher cannot leave if she has witnessed the student see the exam, so that she cannot forget this fact.[11] 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 read the exam topic, acquiring the knowledge on its value.

In this case, no plan reaching the goal exist. 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: we have $S_s\, exam \rightarrow K_t S_s\, exam$. 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:

$$\begin{aligned}
\mathsf{open}_t &= \langle \neg open, \{\langle \top, \{open\}, \emptyset\rangle\}\rangle \\
\mathsf{goIn}_t &= \langle open \wedge \neg in_t, \{\langle \top, \{in_t, S_t\, S_s\, exam\}, \emptyset\rangle\}\rangle,
\end{aligned}$$

[11] It is interesting to note that this occurs because knowledge only comes from observation. It constitutes one of the limits of the approach: not seeing any more means forgetting. If we want the teacher to exit the room to perform another action only if she has seen the student read the exam (for example, inform one of her colleagues), we need to use auxiliary variables representing the fact that she knows that the student has cheated (while not currently seeing it).

- we replace $\mathsf{goOutAndClose}_t$ by:

$$\mathsf{goOut}_t = \langle open \wedge in_t \wedge \neg S_s\ exam, \{\langle \top, \emptyset, \{S_t\ S_s\ exam, in_t\}\rangle\}\rangle$$
$$\mathsf{close}_t = \langle open, \{\langle \top, \emptyset, \{open\}\rangle\}\rangle.$$

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, FDSS-2014 finds the shortest plan: $\mathsf{open}_t; \mathsf{goIn}_s; \mathsf{readExam}_s; \mathsf{goOut}_s$.

Note that 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.

4.6.2 The generalized gossip problem

We formalize the problem exactly like in Chapter 3. Let $Agt = \{1, \ldots, n\}$ and $Prop = \{s_i : i \in Agt\}$. Let us write the planning task for the generalized gossip problem of depth k $G_k = \langle Act^{G_k}, s_0^{G_k}, Goal^{G_k}\rangle$.

Our goal is

$$\varphi_{G_k} = EK^k_{Agt} s_{Agt}.$$

Remember that s_{Agt} abbreviates $\bigwedge_{i \in Agt} s_i$, that $EK_{Agt}\varphi$ abbreviates $\bigwedge_{i \in Agt} K_i\varphi$ and that EK^k_{Agt} is the iteration of EK_{Agt}, k times. As a boolean formula, this becomes:

$$Goal^{G_k} = \bigwedge_{\alpha \in RATM^-(\varphi_{G_k})} \alpha,$$

where

$$RATM^-(\varphi) = RATM(\varphi) \setminus \{\alpha : \alpha \text{ is valid in } INTR\}.$$

The set $RATM^-(\varphi)$ contains the relevant atoms of φ as defined in DEL-PAO (see Section 2.4.3), without all introspectively valid atoms, as we require the goal to be in normal form.

The initial state and the set of actions are still the same:

$$s_0^{G_k} = \{S_i\ s_i : i \in Agt\} \cup \{s_i : i \in Agt\}$$
$$Act^{G_k} = \{\mathsf{Call}^i_j : i, j \in Agt, i \neq j\},$$

where $pre(\mathsf{Call}^i_j) = \top$ and every conditional effect $ce \in \mathit{eff}(\mathsf{Call}^i_j)$ is of the form:

$$cnd(ce) = \Big(\bigwedge_{\alpha \in RATM^-(K_i K_{i_1} \ldots K_{i_m} s_\ell)} \alpha \Big) \vee \Big(\bigwedge_{\alpha \in RATM^-(K_j K_{i_1} \ldots K_{i_m} s_\ell)} \alpha \Big)$$
$$\mathit{ceff}^+(ce) = \{\sigma\, S_{i_1} \ldots S_{i_m}\, s_\ell : \sigma \in \{S_i, S_j\}^{\leq k-m}\}$$
$$\mathit{ceff}^-(ce) = \emptyset$$

for every $0 \leq m < k$ and $i_1, \ldots, i_m, \ell \in Agt$ such that for every $1 \leq p < m$, $i_p \neq i_{p+1}$, and $i \neq i_1$ and $j \neq i_1$. The description of effects ensures that there is not valid atom modified, so that Call^i_j is in normal form. Remember that $\{S_i, S_j\}^{\leq k-m}$ denotes the set all non-empty sequences of visibility operators S_i and S_j of length at most $k - m$.

It is also possible to easily model the gossip with ignorance goals as introduced in Section 3.7. Given a set of atoms A such that $A \cap \{\alpha : \alpha \text{ is valid in } \mathit{INTR}\} = \emptyset$, let us write the planning task for the generalized gossip problem of depth k with the atoms of A as the only negative goals $G\text{-}neg_{k,A} = \langle Act^{G\text{-}neg_{k,A}}, s_0^{G\text{-}neg_{k,A}}, Goal^{G\text{-}neg_{k,A}} \rangle$.

The initial state and the actions remains the same:

$$s_0^{G\text{-}neg_{k,A}} = \{S_i\, s_i : i \in Agt\} \cup \{s_i : i \in Agt\}$$
$$Act^{G\text{-}neg_{k,A}} = \{\mathsf{Call}^i_j : i, j \in Agt, i \neq j\},$$

but the goal changes:

$$Goal^{G\text{-}neg_{k,A}} = \Big(\bigwedge_{\alpha \in RATM^-(\varphi_{G_k}) \setminus A} \alpha \Big) \wedge \Big(\bigwedge_{\alpha \in A} \neg\alpha \Big).$$

Here are some examples:

- $G\text{-}neg_{1,\{S_1\, s_2\}}$ corresponds to the case where we want everyone to know all secrets, except 1 that should not know the secret of 2;
- $G\text{-}neg_{1,\{S_1\, s_3, S_2\, s_4\}}$, to the case where only 1 does not know the secret of 3 and 2, the secret of 4;
- $G\text{-}neg_{2,\{S_1\, S_2\, s_3\}}$, to the case of epistemic depth 2 where only 1 does not know whether 2 knows the secret of 3, while 1 and 2 know the secret of 3; $G\text{-}neg_{3,\{S_1\, S_2\, s_3\}}$ the same goal but of depth 3.

(We had discussed these instances in Section 3.7.)

The resources available at `http://www.irit.fr/%7EAndreas.Herzig/P/Ecai16.html` include an open-source PDDL generator that was developed in Python (the direct link is `https://github.com/FaustineMaffre/GossipProblem-PDDL-generator`). It allows us to create the domain and

problem files for the generalized gossip problem and its variant with ignorance goals. In its basic settings, the program needs the depth k and the number of agents n. It is also possible to specify sets of negatives goals, either directly by giving the index of agents (such as $\{S_1\, s_3, S_2\, s_4\}$), or with constraints (such as $\{S_i\, s_j : i \neq j \text{ and } j < 3\}$ if we want every agent not to know secrets of all agents except 1 and 2). Feeding the resulting files to a PDDL planner returns a sequence of calls which is a solution to the corresponding gossip problem. As examples, here are the shortest plans that were produced by FDSS-2014 for the examples above, with 4 agents:

- $G\text{-}neg_{1,\{S_1\, s_2\}}$: $\mathsf{Call}_3^1; \mathsf{Call}_4^1; \mathsf{Call}_4^2; \mathsf{Call}_4^3$;
- $G\text{-}neg_{1,\{S_1\, s_3, S_2\, s_4\}}$: $\mathsf{Call}_2^1; \mathsf{Call}_4^1; \mathsf{Call}_3^2; \mathsf{Call}_4^3$;
- $G\text{-}neg_{2,\{S_1\, S_2\, s_3\}}$: $\mathsf{Call}_2^1; \mathsf{Call}_4^2; \mathsf{Call}_4^3; \mathsf{Call}_3^1; \mathsf{Call}_3^2; \mathsf{Call}_4^2$ and no solution for $G\text{-}neg_{3,\{S_1\, S_2\, s_3\}}$.

4.7 Conclusion

In this chapter, 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 [Aucher and Bolander, 2013; Charrier et al., 2016a]. Of course, the low complexity of DEL-PAO^S compared to DEL comes at the price of expressivity. We have seen that our epistemic logic DEL-PAO^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.

5 Epistemic boolean games based on visibility and control

In this chapter, we extend DEL-PAO *by adding further special atoms:* control atoms. *This allows us to interpret, in addition to epistemic operators, the operator of "ceteris paribus strategic ability" and to capture epistemic boolean games.*

The atom of control $C_i\, p$ reads "agent i controls the value of p"; $C_i\, C_j\, p$ reads "i controls whether agent j controls p"; etc. We allow nesting with visibility: $C_i\, S_j\, p$ reads "i controls j's visibility of the truth value of p"; $S_i\, C_j\, p$ reads "i sees whether j controls p"; and so on. We are now able to interpret the strategic operator $\diamondsuit_J$, which intuitively means that agents in the set J can make a statement true by modifying variables they control, if the other agents do nothing [van der Hoek and Wooldridge, 2005; Herzig, 2014]. With the help of this operator, we show how to capture epistemic boolean games and to reason about equilibria in such games.

We call this logic DEL-PAOC: Dynamic Epistemic Logic of Propositional Assignment, Observation and Control.

Contents

Résumé du chapitre

Dans ce chapitre, nous étendons DEL-PAO en y ajoutant de nouveaux atomes spéciaux : des atomes des contrôle. Cela nous permet d'interpréter, en plus des opérateurs épistémiques, l'opérateur de capacité stratégique *ceteris paribus* et de capturer les jeux booléens épistémiques.

L'atome de contrôle $C_i\, p$ se lit "l'agent i contrôle la valeur de p" ; $C_i\, C_j\, p$ se lit "i contrôle si l'agent j contrôle p" ; etc. Nous permettons l'imbrication avec la visibilité : $C_i\, S_j\, p$ est lu "i contrôle la visibilité de j sur la valeur de vérité de p" ; $S_i\, C_j\, p$ est lu "i voit si j contrôle p" ; etc. Nous sommes alors en mesure d'interpréter l'opérateur stratégique $\Diamond_J$, qui signifie intuitivement que les agents de l'ensemble J peuvent rendre une propriété vraie en modifiant les variables qu'elles contrôlent, en supposant que les autres agents n'agissent pas [van der Hoek and Wooldridge, 2005; Herzig, 2014]. Avec l'aide de cet opérateur, nous montrons comment capturer les jeux booléens épistémiques et comment raisonner sur les équilibres de tels jeux.

Nous appelons cette nouvelle logique DEL-PAOC : une logique épistémique dynamique des affectations propositionnelles, de l'observation et du contrôle.

5.1 Language of DEL-PAOC

Remember that $Prop$ is a countable non-empty set of propositional variables and Agt is a finite non-empty set of agents.

Like in Chapter 4, in this chapter we consider the fragment of DEL-PAO without JS operators and thus without common knowledge. As we will discuss in Section 5.6, including the joint visibility operator implies dealing with introspective consequences and leads to problems with exclusive control.

The set of visibility operators is

$$OBS_S = \{S_i : i \in Agt\}.$$

The set of *control operators* is

$$CTRL = \{C_i : i \in Agt\},$$

where C_i stands for control of agent i. The set of all sequences of visibility and control operators is noted $(OBS_S \cup CTRL)^*$ and the set of all non-empty sequences is noted $(OBS_S \cup CTRL)^+$. As in DEL-PAO, we use $\sigma, \sigma', \ldots$ for elements of $(OBS_S \cup CTRL)^*$.

Atomic formulas are now sequences of visibility, but also control operators followed by propositional variables:

$$ATM = \{\sigma\, p : \sigma \in (OBS_S \cup CTRL)^*, p \in Prop\}.$$

Therefore our new atoms include C_i operators. Here are some examples:

- $C_1\, p$ reads "1 controls the value of p." This means that 1 has the ability to change the truth value of the variable p.
- $C_1\, C_2\, p$ reads "1 controls whether 2 controls the value of p."
- $S_1\, C_2\, q$ reads "1 sees whether 2 controls the value of q." Recall that this does not imply that 2 controls the value of q.
- $C_1\, S_2\, p$ reads "1 controls whether 2 sees the value of p."
- ... and so on.

The language of DEL-PAOC is then defined by the following grammar:

$$\pi ::= +\alpha \mid -\alpha \mid (\pi;\pi) \mid (\pi \sqcup \pi) \mid \pi^* \mid \varphi?$$
$$\varphi ::= \alpha \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi \mid [\pi]\varphi \mid \Diamond_J\varphi$$

where α ranges over ATM, i over Agt and J over 2^{Agt}.

Programs and all operators except $\Diamond_J$ should be read like in DEL-PAO (see Section 2.1). $\Diamond_J\varphi$ is a strategic operator that reads "agents in J can achieve φ by modifying variables they control if other agents do not act." This refers to the *ceteris paribus* strategic ability in the sense of [van der Hoek and Wooldridge, 2005; Herzig, 2014].

The other boolean operators and dual modal operators abbreviate like in DEL-PAO (i.e., in the standard way). Moreover, $\Box_J\varphi$ ("agents in J cannot avoid φ by changing values of variables they control while other agents do not act") abbreviates $\neg\Diamond_J\neg\varphi$.

The set of atoms appearing in the formula φ and in the program π, noted $ATM(\varphi)$ and $ATM(\pi)$, are also defined like in DEL-PAO, with the addition of:

$$ATM(\Diamond_J\varphi) = ATM(\varphi).$$

Following [van der Hoek and Wooldridge, 2005], we can express the "coalition logic-like" strategic formula $\langle\!\langle J\rangle\!\rangle\varphi$ that reads "agents in J are able to achieve φ by modifying variables they control, whatever the other agents do" from $\Diamond_J$ by

$$\langle\!\langle J\rangle\!\rangle\varphi = \Diamond_J\Box_{Agt\setminus J}\varphi,$$

along with its dual $[\![J]\!]\varphi = \neg\langle\!\langle J\rangle\!\rangle\neg\varphi$.

5.2 Semantics of DEL-PAOC

Like in Chapter 4, a valuation $V \in 2^{ATM}$ is introspective if and only if it contains every atom of the form $\sigma\, S_i\, S_i\, \alpha$ with $\sigma \in OBS_S{}^*$, and not $(OBS_S \cup CTRL)^*$, and $\alpha \in \{\sigma\, p : \sigma \in (OBS_S \cup CTRL)^*, p \in Prop\}$. This means that, for example, $C_j\, S_i\, S_i\, p$ is not valid in $INTR$ (but $S_i\, S_i\, C_j\, p$ is).

Accessibility relations for K_i are defined exactly like in DEL-PAO (see Section 2.2.3):

$$V \sim_i V' \text{ iff } S_i\, \alpha \in V \text{ implies } V(\alpha) = V'(\alpha)$$

where, again, $V(\alpha) = V'(\alpha)$ when either $\alpha \in V$ and $\alpha \in V'$, or $\alpha \notin V$ and $\alpha \notin V'$. As we have seen in Chapter 4, the properties of DEL-PAO still hold: all $\sim_i$ are equivalence relations on $INTR$ and if $V \in INTR$, if $V \sim_i V'$ then $V' \in INTR$.

For the dynamic part, we again do not have to deal with introspective consequences as we do not include the JS operator. Like in Chapter 4, valuations updates are simpler than in DEL-PAO:

$$V{+}\alpha = V \cup \{\alpha\}$$
$$V{-}\alpha = V \setminus \{\alpha\}$$

With our new set of atoms, we have seen that when V is introspective then both $V{+}\alpha$ and $V{-}\alpha$ are so, unless α is valid in $INTR$.

Truth conditions are as follows:

$V \models \alpha$ iff $\alpha \in V$

$V \models \neg\varphi$ iff not $(V \models \varphi)$

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

$V \models K_i\varphi$ iff $V' \models \varphi$ for every V' such that $V \sim_i V'$

$V \models [\pi]\varphi$ iff $V' \models \varphi$ for every V' such that $V R_\pi V'$

$V \models \Diamond_J\varphi$ iff there exist $\alpha_1, \ldots, \alpha_m \in ATM$ with $m \geq 0$ such that for every $1 \leq k \leq m$, there is $i \in J$ such that $C_i\, \alpha_k \in V$ and $V \models \langle(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)\rangle\varphi$

where R_π is a binary relation on valuations defined by:

$V R_{+\alpha} V'$ iff $V' = V{+}\alpha$

$V R_{-\alpha} V'$ iff $V' = V{-}\alpha$ and α not valid in $INTR$

$V R_{\pi_1;\pi_2} V'$ iff $V(R_\pi \circ R_{\pi'})V'$

$V R_{\pi_1 \sqcup \pi_2} V'$ iff $V(R_\pi \cup R_{\pi'})V'$

$V R_{\pi^*} V'$ iff $V(\bigcup_{k \in \mathbb{N}_0} (R_\pi)^k)V'$

$V R_{\varphi?} V'$ iff $V = V'$ and $V \models \varphi$

Truth conditions are identical to DEL-PAO (see Section 2.2.5) for epistemic operators. For dynamic operators, the relation for assignments is simplified like in DEL-PAO^S. The new operator $\Diamond_J$ is interpreted like in [Herzig et al., 2011]: agents in J can make φ true while others do not act if there exists a sequence of assignments of variables they control that leads to a state where φ is satisfied. Remember that $\langle\pi\rangle\varphi$ reads "there exists an execution of π after which φ is true."[12] We suppose that the program becomes $skip$ if $m = 0$, that is, agents in J can simply leave the world in its current state.

Introspective validity and plain validity are defined like in DEL-PAO (see Section 2.2.5).

5.3 Axiomatization

Our axiomatization will consist, like for DEL-PAO, in reduction axioms.

We axiomatize the richer language where the JS operator is included (along with common knowledge), since they are needed to characterize introspective valuations. In other words, we axiomatize the language of DEL-PAO with control atoms. In this language, valuation updates $V{+}\alpha$ and $V{-}\alpha$ are defined like in DEL-PAO (see Section 2.2.4). The semantics of the new operator $\Diamond_J$ is unchanged, but assignments may now concern atoms with JS operators.

Since their truth conditions were not modified, reduction axioms for knowledge operators (see Proposition 2.10 on page 44) and dynamic operators (see Proposition 2.11 on page 50) are still valid. We show how to reduce the strategic operator to a program.

5.3.1 Reduction of the strategic operator

In the vein of $\mathsf{varyIfNotSeen}(i, A)$ for epistemic operators defined in Section 2.4.2, we are going to define $\mathsf{varyIfCtrl}(J, A)$ which, as its name suggest, will vary the variables of A if the coalition J controls them.

Our truth condition for $\Diamond_J$ indicates that agents must control each variable initially, before the modification of any other variable. Since we can consider control over control of variables, this means that control may evolve and that we have to store its initial value, like we did for conditions of conditional effects in Chapter 4. We reuse the notion of

[12] The semantics of $\Diamond_J\varphi$ were modified between [Herzig et al., 2016] and here. In [Herzig et al., 2016], truth conditions were defined as: $V \models \Diamond_J\varphi$ if and only if $V \models \varphi$ for some V' such that VR_JV', where VR_JV' if only atoms controlled by at least one agent in J were modified between V and V'. This led to technical issues with introspective valuations and axiomatization, such as the possibility to remove introspectively valid atoms. Some of these issues are discussed in Section 5.6.

storage variables, noted c, and the programs

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

which stores the value of the formula φ into the storage variable c, and

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

which resets the value of a given set of storage variables.

Then, for a coalition of agents J and a set of atoms $A = \{\alpha_1, \ldots, \alpha_m\}$, we define the program

$$\begin{aligned}\mathsf{varyIfCtrl}(J, A) = \mathsf{str}(\bigvee_{i\in J} C_i\,\alpha_1, c_1); \ldots; \mathsf{str}(\bigvee_{i\in J} C_i\,\alpha_m, c_m);\\ \Big(\bigsqcup_{\alpha_k\in A} \mathsf{varyIf}(c_k, \alpha_k)\Big)^*;\\ \mathsf{rst}(A),\end{aligned}$$

where $\mathsf{varyIf}(c, \alpha) = c?; (+\alpha \sqcup -\alpha)$.

The programs first stores, for every variable α_k in A, whether at least one agent in J controls α_k. Then the program may vary each α_k that was originally controlled, and this an undetermined number of times, generating all possible sequences of assignments of variables from A. We then reset the value of control variables so that, like in Chapter 4, they do not interfere with following executions.

Proposition 5.1. *Let $V \in 2^{ATM}$ be a valuation that does not contain any storage variable, J a set of agents and φ a formula without modal operators. Then we have*

$$V \models \Diamond_J \varphi \leftrightarrow \langle \mathsf{varyIfCtrl}(J, ATM(\varphi))\rangle\varphi.$$

Proof. We are going to compare the truth conditions associated to the modalities $\Diamond_J\varphi$ and $\langle \mathsf{varyIfNotSeen}(i, ATM(\varphi))\rangle\varphi$. Take an arbitrary valuation $V \in 2^{ATM}$:

$V \models \Diamond_J\varphi$ iff

there exist $\alpha_1, \ldots, \alpha_m \in ATM$ such that

for every $1 \leq k \leq m$, there exists $i \in J$ such that $C_i\,\alpha_k \in V$, and

$V \models \langle(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)\rangle\varphi$,

that is:

$V \models \Diamond_J\varphi$ iff

there exist $\alpha_1, \ldots, \alpha_m \in ATM$ such that

for every $1 \leq k \leq m, V \models \bigvee_{i\in J} C_i\,\alpha_k$, and

$V \models \langle(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)\rangle\varphi$,

i.e.:

$V \models \Diamond_J \varphi$ iff
there exist $\alpha_1, \ldots, \alpha_m \in ATM$ and $U_0, \ldots, U_m, V' \in 2^{ATM}$ such that
$V = U_0, V' = U_m,$ and
for every $1 \leq k \leq m, V \models \bigvee_{i \in J} C_i\, \alpha_k$ and $U_{k-1} R_{+\alpha_k \sqcup -\alpha_k} U_k,$ and
$V' \models \varphi$.

By Proposition 2.9 on page 38, we know that we can restrict atoms to the ones appearing in φ. Thus we have:

$V \models \Diamond_J \varphi$ iff
there exist $\alpha_1, \ldots, \alpha_m \in ATM(\varphi)$ and $U_0, \ldots, U_m, V' \in 2^{ATM}$ such that
$V = U_0, V' = U_m,$ and
for every $1 \leq k \leq m, V \models \bigvee_{i \in J} C_i\, \alpha_k$ and $U_{k-1} R_{+\alpha_k \sqcup -\alpha_k} U_k,$ and
$V' \models \varphi$.

On the other hand:

$V \models \langle \mathsf{varIfCtrl}(J, ATM(\varphi)) \rangle \varphi$ iff
there exist $V_1, V_2, V' \in 2^{ATM}$ such that
$V R_{\mathsf{str}(\bigvee_{i \in J} C_i\, \alpha_1, c_1); \ldots; \mathsf{str}(\bigvee_{i \in J} C_i\, \alpha_m, c_m)} V_1,$ and
$V_1 R_{\left(\bigsqcup_{\alpha_\ell \in ATM(\varphi)} \mathsf{varIf}(c_\ell, \alpha_\ell)\right)^*} V_2,$ and
$V_2 R_{\mathsf{rst}(ATM(\varphi))} V',$ and
$V' \models \varphi$.

Similarly to the proof of Proposition 4.7 on page 99, we define

$$C_{\not\models} = \{c_\ell : \alpha_\ell \in ATM(\varphi) \text{ and } V \not\models \bigvee_{i \in J} C_i\, \alpha_\ell\}$$
$$C_{\models} = \{c_\ell : \alpha_\ell \in ATM(\varphi) \text{ and } V \models \bigvee_{i \in J} C_i\, \alpha_\ell\}$$
$$C = \{c_\ell : \alpha_\ell \in ATM(\varphi)\}$$

Obviously $C_{\not\models} \cup C_{\models} = C$.

We obtain:

$$
\begin{array}{l}
V \models \langle \mathsf{varIfCtrl}(J, ATM(\varphi))\rangle\varphi \text{ iff} \\
\quad \text{there exist } V_1, V_2, V' \in 2^{ATM} \text{ such that} \\
\quad\quad V_1 = (V \setminus C_{\not\models}) \cup C_{\models}, \text{ and} \\
\quad\quad V_1 R_{\left(\bigsqcup_{\alpha_\ell \in ATM(\varphi)} \mathsf{varIf}(c_\ell, \alpha_\ell)\right)^*} V_2, \text{ and} \\
\quad\quad V' = V_2 \setminus C, \text{ and} \\
\quad\quad V' \models \varphi,
\end{array}
$$

that is:

$$
\begin{array}{l}
V \models \langle \mathsf{varIfCtrl}(J, ATM(\varphi))\rangle\varphi \text{ iff} \\
\quad \text{there exist } V_1, V_2, V' \in 2^{ATM} \text{ such that} \\
\quad\quad V_1 = (V \setminus C_{\not\models}) \cup C_{\models}, \text{ and} \\
\quad\quad \text{there exist } U_0, \ldots, U_m \in 2^{ATM} \text{ such that } V_1 = U_0, V_2 = U_m, \text{ and} \\
\quad\quad\quad \text{for every } 1 \leq k \leq m, U_{k-1} R_{\bigsqcup_{\alpha_\ell \in ATM(\varphi)} \mathsf{varIf}(c_\ell, \alpha_\ell)} U_k, \text{ and} \\
\quad\quad V' = V_2 \setminus C, \text{ and} \\
\quad\quad V' \models \varphi,
\end{array}
$$

i.e.:

$$
\begin{array}{l}
V \models \langle \mathsf{varIfCtrl}(J, ATM(\varphi))\rangle\varphi \text{ iff} \\
\quad \text{there exist } V_1, V_2, V' \in 2^{ATM} \text{ such that} \\
\quad\quad V_1 = (V \setminus C_{\not\models}) \cup C_{\models}, \text{ and} \\
\quad\quad \text{there exist } \alpha_1, \ldots, \alpha_m \in ATM(\varphi) \text{ and } U_0, \ldots, U_m \in 2^{ATM} \text{ such that} \\
\quad\quad\quad V_1 = U_0, V_2 = U_m, \text{ and} \\
\quad\quad\quad \text{for every } 1 \leq k \leq m, U_{k-1} R_{\mathsf{varIf}(c_k, \alpha_k)} U_k, \text{ and} \\
\quad\quad V' = V_2 \setminus C, \text{ and} \\
\quad\quad V' \models \varphi.
\end{array}
$$

which becomes, when decomposing $\mathsf{varIf}(c_k, \alpha_k)$:

$$
\begin{array}{l}
V \models \langle \mathsf{varIfCtrl}(J, ATM(\varphi))\rangle\varphi \text{ iff} \\
\quad \text{there exist } V_1, V_2, V' \in 2^{ATM} \text{ such that} \\
\quad\quad V_1 = (V \setminus C_{\not\models}) \cup C_{\models}, \text{ and} \\
\quad\quad \text{there exist } \alpha_1, \ldots, \alpha_m \in ATM(\varphi) \text{ and } U_0, \ldots, U_m \in 2^{ATM} \text{ such that} \\
\quad\quad\quad V_1 = U_0, V_2 = U_m, \text{ and} \\
\quad\quad\quad \text{for every } 1 \leq k \leq m, U_{k-1} \models c_k \text{ and } U_{k-1} R_{+\alpha_k \sqcup -\alpha_k} U_k, \text{ and} \\
\quad\quad V' = V_2 \setminus C, \text{ and} \\
\quad\quad V' \models \varphi.
\end{array}
$$

We are getting close to the truth condition of $\Diamond_J\varphi$, with additional modifications of storage variables.

As we have seen in the proof of Proposition 4.7 on page 99, only storage variables are modified between V and V_1, where each c_k takes the value of $\bigvee_{i\in J} C_i\,\alpha_k$. Afterwards, during the execution of $\mathsf{varIf}(c_\ell, \alpha_\ell)$, the value of storage variables is not modified. Therefore saying $U_{k-1} \models c_k$ is like $V \models \bigvee_{i\in J} C_i\,\alpha_k$. Finally, all storage variables, which by definition, do not appear in φ, are removed.

Therefore $V \models \Diamond_J\varphi \leftrightarrow \langle \mathsf{varIfCtrl}(J, ATM(\varphi))\rangle\varphi$. □

5.3.2 Soundness and completeness

The rule of replacement of equivalents still preserves plain validity.

Proposition 5.2. *Let φ' be obtained from φ by replacing some occurrence of ψ in φ by ψ'. If $\psi \leftrightarrow \psi'$ is plainly valid then $\varphi \leftrightarrow \varphi'$ is plainly valid.*

Proof. This is due to the fact that, along with the ones for K_i and $[\pi]$ (see Proposition 2.12 on page 52) the following rule of inference for $\Diamond_J$:

$$\frac{\varphi \leftrightarrow \varphi'}{\Diamond_J\varphi \leftrightarrow \Diamond_J\varphi'}$$

preserves plain validity. □

Theorem 5.1. *For every* DEL-PAOC *formula φ there exists a formula without modal operators φ' such that $\varphi \leftrightarrow \varphi'$ is plainly valid.*

Proof. We extend Procedure 2.1 on page 52 to strategic operators.

Procedure 5.1. While there is a modal operator in φ:

1. if there is a sub-formula $K_i\varphi'$ or $[\pi]\varphi'$ such that φ' does not contain modal operators, apply Procedure 2.1;
2. if there is a sub-formula $\Diamond_J\varphi'$ such that φ' does not contain modal operators, replace $\Diamond_J\varphi'$ by $\langle \mathsf{varIfCtrl}(J, ATM(\varphi'))\rangle\varphi'$, according to Proposition 5.1.

So by iterating elimination of innermost modal operators we obtain a formula without modal operators. These transformations are possible thanks to the rule of replacement of equivalents that preserves plain validity. □

The axiomatization of DEL-PAOC is given by:

- the axioms of DEL-PAO given in Section 2.3;
- the reduction axiom for strategic operators given in Proposition 5.1.

Theorem 5.2. *The axiomatization of* DEL-PAOC *is sound and complete w.r.t. the set of introspective valuations.*

Proof. The proof follows the lines of its counterpart for DEL-PAO (see Theorem 2.2 on page 54). □

5.4 Complexity of model checking

For model checking, we will reduce the epistemic operator K_i to the program $\mathsf{varyIfNotSeen}(i, A)$ as we did in DEL-PAO (see Section 2.4.2). We will do the same with the strategic operator $\Diamond_J$, reducing it to the program $\mathsf{varyIfCtrl}(J, A)$ thanks to Proposition 5.1. Like in DEL-PAO, this transformation may add new atoms to the formula. In order to include these new atoms, we extend the definition of relevant atoms $RATM(\varphi)$, defined in Section 2.4.3, to strategic operators as follows:

$$RATM(\Diamond_J \varphi) = RATM(\varphi) \cup \{C_i\, \alpha : i \in J, \alpha \in RATM(\varphi)\}$$

Observe that the relevant atoms of φ are exactly the atoms of its reduction obtained by Procedure 5.1, minus storage variables that do not influence the final value of φ.

We also extend Proposition 2.18 on page 62 to DEL-PAOC.

Proposition 5.3. *Let φ be a formula. Let V, $V' \in 2^{ATM}$ such that V and V' are introspective w.r.t. $RATM(\varphi)$ and $V =_{RATM(\varphi)} V'$. Then $V \models \varphi$ if and only if $V' \models \varphi$.*

Proof. The proof follows the line of Proposition 2.18. The new set of relevant atoms ensures that all atoms appearing in the reduction of φ (except storage variables) are equal in V and V' (including atoms of control). □

Then the model checking problem for DEL-PAOC is defined as follows:

- **Input:** a couple $\langle V \cap RATM(\varphi), \varphi \rangle$ where φ is a DEL-PAOC formula and V is an introspective valuation;
- **Output:** yes if $V \models \varphi$, no otherwise.

Theorem 5.3. *The* DEL-PAOC *model checking problem is* PSPACE*-complete.*

Proof. The model checking of DL-PA, which is a fragment of DEL-PAOC without visibility operators S_i and control operators C_i was proven to be PSPACE-hard [Balbiani et al., 2014]. This establishes the lower bound.

We will show in Chapter 7 that the problem is in PSPACE. □

5.5 Epistemic boolean games

In this section, we show how to express that a given strategy is a Nash equilibrium and whether a Nash equilibrium exists in an epistemic boolean game within DEL-PAOC.

5.5.1 Epistemic boolean games in DEL-PAOC

We define ATM_{OBS_S} to be the subset of ATM which only visibility atoms:

$$ATM_{OBS_S} = \{\sigma\, p : \sigma \in OBS_S{}^*, p \in Prop\}.$$

Informally, ATM_{OBS_S} is exactly the set of atoms considered in Chapter 4, i.e., DEL-PAO atoms without joint visibility and without control.

An *epistemic boolean game* is a tuple

$$\mathcal{B} = (Agt, ATM^{\mathsf{f}}_{OBS_S}, (\Psi_i)_{i\in Agt}, (\gamma_i)_{i\in Agt})$$

where $ATM^{\mathsf{f}}_{OBS_S}$ is a finite subset of ATM_{OBS_S}, the sets $(\Psi_i)_{i\in Agt}$ (the sets of control) partition $ATM^{\mathsf{f}}_{OBS_S}$, and the formulas γ_i (the goals) are such that $RATM(\gamma_i) \subseteq ATM^{\mathsf{f}}_{OBS_S}$.

Remember that the relevant atoms of φ, introduced in Section 2.4.3 and noted $RATM(\varphi)$, contains the atoms of φ plus visibility atoms related to epistemic operators appearing in φ (e.g., $RATM(K_i p) = \{p, S_i\, p\}$). This means that if $\gamma_j = K_i p$, then p but also $S_i\, p$ must belong to $ATM^{\mathsf{f}}_{OBS_S}$ because both influence the truth value of γ_j.

5.5.2 Strategies and introspection

Strategies of individuals and coalitions are defined like in the context of standard boolean games: $s_i \in 2^{\Psi_i}$ and $s_J = \bigcup_{i\in J} s_i$. We however impose a constraint on strategy profiles s_{Agt}: they must be introspective w.r.t. $ATM^{\mathsf{f}}_{OBS_S}$.

We have seen in Section 2.4 that valuations introspective w.r.t. a specific set of atoms can be finite while being "introspective enough" for the model checking to work correctly. Without introspective consequences, the definition given in Section 2.4.1 reduces to: a valuation V is introspective w.r.t. to a set of atoms A if every introspectively valid atom from A belongs to V. Strategy profiles are sets of atoms and therefore are of the same nature as valuations. When requiring them to be introspective w.r.t. $ATM^{\mathsf{f}}_{OBS_S}$, we implicitly impose that every agent controlling an introspectively valid atom chooses to make it true; it appears like a reasonable condition.

Besides strategies, utilities over strategy profiles, best responses and Nash equilibria are defined exactly as in the context of standard boolean games (see Section 1.3.2). Remember that the utility of an agent is 1 if she reaches her goal, and 0 otherwise.

5.5.3 Nash equilibrium

The Nash equilibria of a given epistemic boolean game can be characterised in the language of DEL-PAOC. Given the sets of control $(\Psi_i)_{i\in Agt}$, we define $Ctrl = \{C_i\,\alpha : i \in Agt, \alpha \in \Psi_i\}$ which describes the control abilities of agents in the boolean game in terms of atoms.

Proposition 5.4. *Let $\mathcal{B} = (Agt, ATM^{\mathsf{f}}_{OBS_S}, (\Psi_i)_{i\in Agt}, (\gamma_i)_{i\in Agt})$ be an epistemic boolean game and let*

$$Nash = \bigwedge_{i\in Agt} (\Diamond_{\{i\}}\gamma_i \to \gamma_i).$$

Then s_{Agt} is a Nash equilibrium for $\mathcal{B}$ if and only if

$$s_{Agt} \cup Ctrl \models Nash.$$

The contraposition of the formula is easier to understand: $\neg\gamma_i \to \neg\Diamond_{\{i\}}\gamma_i$ intuitively reads "if i has not reached her goal, then she cannot reach it by modifying the variables under her control." In other words, her strategy is a best response to the strategies of the other agents. If this is the case for every agent, i.e., if $\bigwedge_{i\in Agt}(\Diamond_{\{i\}}\gamma_i \to \gamma_i)$ is true, then the current situation is a Nash equilibrium.

Proof. Take an arbitrary agent i. We prove that s_i is a best response to $s_{Agt\setminus\{i\}}$ (with $\langle s_i, s_{Agt\setminus\{i\}}\rangle = s_{Agt}$) if and only if $s_{Agt} \cup Ctrl \models \Diamond_{\{i\}}\gamma_i \to \gamma_i$. We distinguish two cases.

- First suppose $s_{Agt} \models \gamma_i$. Then $U_i(s_{Agt}) = 1$ and therefore s_i is trivially a best response to $s_{Agt\setminus\{i\}}$.

  Now if $s_{Agt} \models \gamma_i$ then $s_{Agt} \cup Ctrl \models \gamma_i$ by Proposition 2.9 on page 38 since γ_i does not contain any atom from $Ctrl$. Thus $s_{Agt} \cup Ctrl \models \Diamond_{\{i\}}\gamma_i \to \gamma_i$.

- Now suppose $s_{Agt} \not\models \gamma_i$. Then $U_i(s_{Agt}) = 0$ and s_i is a best response to $s_{Agt\setminus\{i\}}$ if and only if for every strategy $s'_i \in 2^{\Psi_i}$, $U_i(\langle s'_i, s_{Agt\setminus\{i\}}\rangle) = 0$, i.e., if and only if for every strategy $s'_i \in 2^{\Psi_i}$, $s'_i \cup s_{Agt\setminus\{i\}} \not\models \gamma_i$, that is, if and only if for every s'_{Agt} such that $s_{Agt} =_{ATM^{\mathsf{f}}_{OBS_S}\setminus\Psi_i} s'_{Agt}$, $s'_{Agt} \not\models \gamma_i$.

  On the other hand, again by Proposition 2.9, $s_{Agt} \not\models \gamma_i$ if and only if $s_{Agt} \cup Ctrl \not\models \gamma_i$, i.e., $s_{Agt} \cup Ctrl \models \Box_{\{i\}}\neg\gamma_i$ since $s_{Agt} \cup Ctrl \models \Diamond_{\{i\}}\gamma_i \to \gamma_i$. In other words, for every $\alpha_1, \ldots, \alpha_m \in ATM$ such that for every $1 \le k \le m$, $C_i\,\alpha_k \in s_{Agt} \cup Ctrl$, we have $s_{Agt} \cup Ctrl \models [(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)]\neg\gamma_i$. Since the control of i is defined by Ψ_i, this is equivalent to: for every $\alpha_1, \ldots, \alpha_m \in \Psi_i$, we have $s_{Agt} \cup Ctrl \models [(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)]\neg\gamma_i$. Our formula does

not contain any control atom or strategic operator any more, hence by Proposition 2.9, this is equivalent to: for every $\alpha_1, \ldots, \alpha_m \in \Psi_i$, we have $s_{Agt} \models [(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)]\neg\gamma_i$, i.e., for every s'_{Agt} such that $s_{Agt} =_{ATM^{\mathsf{f}}_{OBS_S} \setminus \Psi_i} s'_{Agt}$, $s'_{Agt} \not\models \gamma_i$.

In both cases, we have that s_i is a best response to $s_{Agt\setminus\{i\}}$ if and only if $s_{Agt} \cup Ctrl \models \Diamond_{\{i\}}\gamma_i \to \gamma_i$. The formula $Nash$ generalizes this to the whole set of agents. □

The following proposition provides a characterization in the logic DEL-PAOC of the existence of a Nash equilibrium in a certain epistemic boolean game.

Proposition 5.5. *Let $\mathcal{B} = (Agt, ATM^{\mathsf{f}}_{OBS_S}, (\Psi_i)_{i\in Agt}, (\gamma_i)_{i\in Agt})$ be an epistemic boolean game. Then $\mathcal{B}$ has at least one Nash equilibrium if and only if*

$$Ctrl \cup \{\alpha \in ATM^{\mathsf{f}}_{OBS_S} : \alpha \textit{ is valid in } INTR\} \models \Diamond_{Agt} Nash.$$

Proof. The set $\{\alpha \in ATM^{\mathsf{f}}_{OBS_S} : \alpha \text{ is valid in } INTR\}$ ensures that $Ctrl \cup \{\alpha \in ATM^{\mathsf{f}}_{OBS_S} : \alpha \text{ is valid in } INTR\}$ is introspective w.r.t. $ATM^{\mathsf{f}}_{OBS_S}$. Let us write this set $ATM^{\mathsf{f}}_{OBS_S|INTR}$.

We have: $Ctrl \cup ATM^{\mathsf{f}}_{OBS_S|INTR} \models \Diamond_{Agt} Nash$ if and only if there exists $\alpha_1, \ldots, \alpha_m \in ATM$ such that for every $1 \leq k \leq m$, there exists $i \in Agt$ such that $C_i\, \alpha_k \in Ctrl \cup ATM^{\mathsf{f}}_{OBS_S|INTR}$, and $Ctrl \cup ATM^{\mathsf{f}}_{OBS_S|INTR} \models \langle(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)\rangle Nash$. Since the sets $(\Psi_i)_{i\in Agt}$ partition $ATM^{\mathsf{f}}_{OBS_S}$, agents together control exactly the set $ATM^{\mathsf{f}}_{OBS_S}$ and this is equivalent to: there exists $\alpha_1, \ldots, \alpha_m \in ATM^{\mathsf{f}}_{OBS_S}$ such that $Ctrl \cup ATM^{\mathsf{f}}_{OBS_S|INTR} \models \langle(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)\rangle Nash$. Therefore the program $(+\alpha_1 \sqcup -\alpha_1); \ldots; (+\alpha_m \sqcup -\alpha_m)$ will either add or remove atoms from $ATM^{\mathsf{f}}_{OBS_S}$. Observe that the only atoms from $ATM^{\mathsf{f}}_{OBS_S}$ that are already in $Ctrl \cup ATM^{\mathsf{f}}_{OBS_S|INTR}$ are introspectively valid and therefore cannot be removed. Thus the program can only add new atoms among $\alpha_1, \ldots, \alpha_m$ (the others will stay false). Hence the previous statement is equivalent to: there exists $\alpha_1, \ldots, \alpha_k \in ATM^{\mathsf{f}}_{OBS_S}$ (with $k \leq m$) such that $Ctrl \cup ATM^{\mathsf{f}}_{OBS_S|INTR} \cup \{\alpha_1, \ldots, \alpha_k\} \models Nash$. $ATM^{\mathsf{f}}_{OBS_S|INTR} \cup \{\alpha_1, \ldots, \alpha_k\}$ can be viewed as a strategy for all agents which is, as required, introspective w.r.t. $ATM^{\mathsf{f}}_{OBS_S}$. In other words, there exists s_{Agt} such that $Ctrl \cup s_{Agt} \models Nash$, i.e., there exists a strategy s_{Agt} such that s_{Agt} is a Nash equilibrium by Proposition 5.4, that is, the game has at least one Nash equilibrium. □

The preceding two propositions together with Theorem 5.3 about complexity of model checking for DEL-PAOC provide a complexity result

both for the membership problem, described in Proposition 5.4, and for the existence problem, described in Proposition 5.5, of Nash equilibria in epistemic boolean games.

Theorem 5.4. *The membership problem and the existence problem of Nash equilibria in epistemic boolean games are both in* PSPACE.

Proof. From propositions 5.4 and 5.5, both problems polynomially reduce to the DEL-PAOC model checking problem. □

The PSPACE-completeness remains an open problem.

Let us illustrate epistemic boolean games with an example of a coordination game.

Example 5.1. Suppose we have two agents 1 and 2 each of which knows a bit of information, respectively noted b_1 and b_2, that the other agent does not know. Both 1 and 2 have the same goal: 1 wants 2 to know her secret only if 1 knows the secret of 2, and similarly for 2. Each agent can change her secret, and can either talk or keep quiet; in other words, each agent has control on her bit and on the other agent's visibility of her bit.

We define $\mathcal{B}_b = (Agt, ATM^{\mathsf{f}}_{OBS_S}, (\Psi_i)_{i\in Agt}, (\gamma_i)_{i\in Agt})$ where:

- $Agt = \{1, 2\}$;
- $ATM^{\mathsf{f}}_{OBS_S} = \{b_1, b_2, S_2\, b_1, S_1\, b_2\}$;
- $\Psi_1 = \{b_1, S_2\, b_1\}$ and $\Psi_2 = \{b_2, S_1\, b_2\}$;
- $\gamma_1 = \gamma_2 = (K_1 b_2 \vee K_1 \neg b_2) \leftrightarrow (K_2 b_1 \vee K_2 \neg b_1)$.

Two interesting Nash equilibria are the following. Either they both share their secrets: $s_{Agt} = \{b_1, b_2, S_2\, b_1, S_1\, b_2\}$; or they both remain silent: $s_{Agt} = \{b_1, b_2\}$.

Indeed, with $Ctrl = \{C_1\, b_1, C_2\, b_2, C_1\, S_2\, b_1, C_2\, S_1\, b_2\}$, we have:

$$\{b_1, b_2, S_2\, b_1, S_1\, b_2\} \cup Ctrl \models (\Diamond_{\{1\}}\gamma_1 \to \gamma_1) \wedge (\Diamond_{\{2\}}\gamma_2 \to \gamma_2)$$
$$\{b_1, b_2\} \cup Ctrl \models (\Diamond_{\{1\}}\gamma_1 \to \gamma_1) \wedge (\Diamond_{\{2\}}\gamma_2 \to \gamma_2)$$

Intuitively, each agent can only modify the other agent's visibility of her secret; if only one of them changes her strategy then it will break the equivalence of the goal.

Observe that we omit other Nash equilibria such as $\{b_1, S_2\, b_1, S_1\, b_2\}$ or $\{b_2\}$, where the values of b_i is modified. It actually suffices that $S_2\, b_1$ and $S_1\, b_2$ are either both true or both false to obtain a Nash equilibrium.

5.6 Relationship between exclusive control and visibility

We have seen that we did not include the joint visibility operator JS in the language of DEL-PAOC. This would indeed lead to problems with exclusive control, that is required in boolean games, because of introspective causes and consequences.[13]

For example, suppose agent 1 controls $JS\,p$, and that agent 2 controls $S_i\,p$. It seems that exclusive control is ensured; however if 1 makes $JS\,p$ true, then an infinite number of other atoms, including $S_i\,p$, will become true. Therefore the control on $S_i\,p$ is actually not exclusive. The other way round, if 2 makes $S_i\,p$ false, then $JS\,p$ will also become false. It is interesting to note that 1 does not have a "full control" on $S_i\,p$: she can only make it true. Similarly, 2 can only make $JS\,p$ false.

A way to solve this issue would be to define constraints on atoms of control, like introspective constraints defined in Section 2.2.1:

$$
\begin{aligned}
&S_i\,S_i\,\alpha\\
&JS\,JS\,\alpha\\
&JS\,S_i\,S_i\,\alpha\\
&JS\,\alpha \to S_i\,\alpha\\
&JS\,\alpha \to JS\,S_i\,\alpha
\end{aligned}
$$

New constraints would define a new set of introspective valuations $INTR$ where, for example, no agent can control an introspectively valid atom such as $\sigma\,S_i\,S_i\,\alpha$ with $\sigma \in OBS^*$ or $\sigma\,JS\,\alpha$ with $\sigma \in OBS^+$. This would require at least constraints of the form:

$$
\begin{aligned}
&\neg C_j\,S_i\,S_i\,\alpha\\
&\neg C_j\,JS\,JS\,\alpha\\
&\neg C_j\,JS\,S_i\,S_i\,\alpha
\end{aligned}
$$

Yet these are not sufficient to ensure $\neg C_j\,\sigma\,S_i\,S_i\,\alpha$ for every $\sigma \in OBS^*$ and $\neg C_j\,\sigma\,JS\,\alpha$ for every $\sigma \in OBS^+$. We also need to "generate" every sequence σ using the JS operator. However, the constraints:

$$
\begin{aligned}
&\neg C_j\,JS\,\alpha \to \neg C_j\,S_i\,\alpha\\
&\neg C_j\,JS\,\alpha \to \neg C_j\,JS\,S_i\,\alpha
\end{aligned}
$$

echoing the last two introspective constraints, are intuitively too strong: an agent should be able to make individual visibility true without controlling joint attention.

[13] This discussion was not included in [Herzig et al., 2016].

As observed in the previous paragraph, we might consider a more subtle notion of control, with positive and negative control over variables: $C_i^+ \alpha$ means that i can only make α true; $C_i^- \alpha$ means that i can only make α false. Then $C_i \alpha$ is a shortcut for $C_i^+ \alpha \wedge C_i^- \alpha$. Our constraints become:

$$\neg C_j^- \, S_i \, S_i \, \alpha$$
$$\neg C_j^- \, JS \, JS \, \alpha$$
$$\neg C_j^- \, JS \, S_i \, S_i \, \alpha$$
$$\neg C_j^- \, JS \, \alpha \rightarrow \neg C_j^- \, S_i \, \alpha$$
$$\neg C_j^- \, JS \, \alpha \rightarrow \neg C_j^- \, JS \, S_i \, \alpha$$

or if we rewrite the last two:

$$C_j^- \, S_i \, \alpha \rightarrow C_j^- \, JS \, \alpha$$
$$C_j^- \, JS \, S_i \, \alpha \rightarrow C_j^- \, JS \, \alpha$$

This seems to be more intuitive: if one can make individual visibility false, then she can make joint visibility false. From this we can derive:

$$\text{if } C_j^- \, \alpha \in V \text{ and } \beta \Rightarrow_I \alpha, \text{ then } C_j^- \, \beta \in V,$$

that is, if agent j negatively controls α, then she also negatively controls all its introspective causes. Then the constraints that come to mind for positive control are:

$$C_j^+ \, JS \, \alpha \rightarrow C_j^+ \, S_i \, \alpha$$
$$C_j^+ \, JS \, \alpha \rightarrow C_j^+ \, JS \, S_i \, \alpha$$

implying that:

$$\text{if } C_j^+ \, \alpha \in V \text{ and } \alpha \Rightarrow_I \beta, \text{ then } C_j^+ \, \beta \in V.$$

This set of constraints seems acceptable, however it remains to ensure that if V follows all of them, then every V' such that $V \sim_i V'$ or $V \sim_{Agt} V'$ also follows them, which is not the case right now. Another work to be done is to tune the relation R_π of programs so that it also does not exit the new set of introspective valuations. These were the hardest part when defining the introspective constraints in DEL-PAO; it is plausible that with an extended set of constraints, this task becomes much more challenging.

This might be rewarding however, as with such constraints we may also impose further relations between control and visibility, such as the requirement "control implies visibility":

$$C_i \, \alpha \rightarrow S_i \, \alpha$$

or knowledge of control:

$$S_i\, C_i\, \alpha.$$

5.7 Conclusion

We have studied an extension of DEL-PAO with control in which one can deduce strategic abilities of coalitions of agents. It accounts for concepts from boolean games such as the existence of a Nash equilibrium that can be extended to epistemic boolean games in a straightforward way.

As the model checking of the DEL-PAOC is PSPACE-complete, it follows that the membership problem and existence problem of a Nash equilibria in epistemic boolean games are both in PSPACE. Agents can now modify the visibility conditions of propositional variables, including higher-order visibility. In the previous work of Ågotnes et al., agents can only affect the truth value of propositional variables and, indirectly, the knowledge of those agents' who can see the truth value of these variables; but agents cannot modify the visibility conditions of propositional variables. In other words, visibility conditions remain static, whereas in our framework, they can change. It remains to establish the lower bound.

6 Adding public announcements and programs

Another main drawback of visibility-based epistemic logics, including DEL-PAO*, is that the epistemic operators may distribute over disjunctions:* $K_i(p \vee q) \rightarrow (K_i p \vee K_i q)$ *is valid. This is annoying because it does not allow us to model things such as the muddy children puzzle (where each child knows that one of the children is muddy without knowing which). In the present chapter, we propose a solution to this problem.*

Following [Castelfranchi, 1994; van Linder et al., 1997], we observe that an agent's knowledge may originate from three processes: observation, communication, and inference. We do not consider knowledge obtained via inference and assume that agents are omniscient. We therefore do not model formation of knowledge via (time-consuming) application of inference rules and leave it to future work to integrate existing logics of time-bounded reasoning [Alechina et al., 2004; Grant et al., 2000; Balbiani et al., 2016]. While current observability-based approaches only account for the former, we here take into account the most basic form of communication modifying the agents' knowledge: public announcements. We do so by adding a *public information state* to the model, which is a set of valuations, as proposed in [Lomuscio et al., 2000; Su et al., 2007] and recently used in [Charrier and Schwarzentruber, 2015; van Benthem et al., 2015]. Public announcements make the public information state shrink just as in public announcement logic. We can then model that although no p-worlds nor q-worlds are accessible for i because the information state contains no such world, there is an accessible $p \wedge q$-world. This may be due to the announcement of $p \wedge q$, or to the announcement (in some order) of p and of q, or to the announcement (in some order) of p and of $p \rightarrow q$, etc.

In this chapter, we view DEL-PAO programs as being executed publicly, thereby updating the public information state. Public announcements become a particular case of such programs: they are publicly

executed tests.

We call this logic DEL-PAO-PP: Dynamic Epistemic Logic of Propositional Assignment and Observation with Public Programs.

Contents

Résumé du chapitre

Un autre inconvénient majeur des logiques épistémiques basées sur la visibilité, y compris DEL-PAO, est que les opérateurs épistémiques peuvent distribuer sur les disjonctions : $K_i(p \vee q) \rightarrow (K_i p \vee K_i q)$ est valide. Ceci est gênant, car cela nous empêche de modéliser les problèmes comme celui des enfants sales (où chaque enfant sait que l'un des enfants est boueux sans savoir qui). Dans ce chapitre, nous proposons une solution à ce problème.

En suivant [Castelfranchi, 1994; van Linder et al., 1997], nous observons que la connaissance d'un agent peut provenir de trois processus : l'observation, la communication, et l'inférence. Nous ne considérons pas la connaissance obtenue par inférence et supposons que les agents sont omniscients. Alors que les approches actuelles basées sur l'observabilité ne prennent en compte que le premier processus, nous incluons ici la forme la plus basique de communication modifiant les connaissances des agents : les annonces publiques. Nous le faisons en ajoutant un *état d'information* au modèle, c'est-à-dire un ensemble de valuations, tel que proposé dans [Lomuscio et al., 2000; Su et al., 2007] et récemment utilisé dans [Charrier and Schwarzentruber, 2015; van Benthem et al., 2015]. Les annonces publiques font rétrécir l'état d'information, comme en logique des annonces publiques. Nous pouvons alors modéliser que même si aucun p-monde ni q-monde n'est accessible pour i car l'état d'information ne contient pas un tel monde, il existe un $p \wedge q$-monde accessible. Cela peut être dû à l'annonce de $p \wedge q$, ou bien à celle (dans un ordre quelconque) de p et de q, ou à celle (dans un ordre quelconque) de p et de $p \rightarrow q$, etc.

Dans ce chapitre, nous considérons les programmes DEL-PAO comme étant exécutés publiquement, mettant ainsi à jour l'état d'information. Les annonces publiques deviennent un cas particulier de ces programmes : elles sont des tests exécutés publiquement.

Nous appelons cette nouvelle logique DEL-PAO-PP : une logique épistémique dynamique des affectations propositionnelles, de l'observation et du contrôle avec des programmes publics.

6.1 Language of DEL-PAO-PP

Remember that $Prop$ is a countable non-empty set of propositional variables and and Agt is a finite non-empty set of agents. The set of observability operators OBS and atoms ATM is defined like in DEL-PAO:

$$OBS = \{S_i : i \in Agt\} \cup \{JS\}$$
$$ATM = \{\sigma\, p : \sigma \in OBS^*, p \in Prop\}$$

The language of programs and formulas of DEL-PAO-PP is defined by the following grammar:

$$\pi ::= +\alpha \mid -\alpha \mid (\pi;\pi) \mid (\pi \sqcup \pi) \mid \varphi?$$
$$\varphi ::= \alpha \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid K_i\varphi \mid CK\varphi \mid [\pi!]\varphi$$

where α ranges over ATM and i over Agt.

The language is therefore the one of DEL-PAO, except that programs are now executed publicly: we read $[\pi!]\varphi$ as "φ will be true after the update of the current local and information states by π." As we shall see, the publicly performed test $\varphi?$ behaves exactly as the public announcement of φ. Unlike in DEL-PAO, we do not include the Kleene star * to the language of (public) programs as it is not required in applications and complexity with it has to be decided.

$K_i\varphi$ reads "φ is known by i" and $CK\varphi$ reads "φ is common knowledge among all agents." Another difference with DEL-PAO in that, while S_i still expresses sensor information, now K_i expresses information coming from both sensors and communication.

The other boolean operators and dual epistemic operators abbreviate like in DEL-PAO (in the standard way). Moreover, $\langle\pi!\rangle\varphi$ ("there exists a public execution of π after which φ is true") abbreviates $\neg[\pi!]\neg\varphi$. The set of atoms appearing in the formula φ and in the program π, noted $ATM(\varphi)$ and $ATM(\pi)$, are also defined like in DEL-PAO, with the addition of:

$$ATM([\pi!]\varphi) = ATM(\pi) \cup ATM(\varphi).$$

Example 6.1 (Representing the muddy children puzzle)**.** Let us illustrate by means of the muddy children puzzle [Lehmann, 1984; Fagin et al., 1995] how knowledge can be represented in our language.

The version of this puzzle with two agents goes as follows. Two children come back from playing in the park. When their father sees them, he notices that they both have mud on their foreheads. He says, "at least one of you has mud on her forehead," and then asks, "Do you know if you have mud on your forehead?" The children simultaneously answer "No." Then the father repeats his question, "Do you know if you have mud on your forehead?" This time the two children simultaneously answer, "Yes, I have."

Let $Agt = \{1, \ldots, n\}$ be the set of children. Let $Prop = \{m_1, \ldots, m_n\}$ be the propositional variables, where m_i expresses that child i is muddy.

Suppose all children are muddy. This fact is described by the conjunction:

$$Muddy = \bigwedge_{i \in Agt} m_i.$$

Second, agents' observational capabilities are expressed by:

$$Obs = \Big(\bigwedge_{i \in Agt} \neg S_i\, m_i \Big) \wedge \Big(\bigwedge_{\substack{i,j \in Agt, \\ i \neq j}} S_i\, m_j \Big) \wedge \Big(\bigwedge_{i,j \in Agt} JS\, S_i\, m_j \Big).$$

The first two conjuncts express that the agents see the states of other agents but not their own states. The last conjunct expresses that the agents jointly observe their observational capabilities. For instance, $JS\, S_i\, m_i$ reads "all agents jointly see whether i sees m_i." Recall that it does not imply that i sees m_i however.

Third, the agents' (sensor and communicational) information in the initial situation is described by the formula:

$$Ign = \bigwedge_{i \in Agt} (\neg K_i m_i \wedge \neg K_i \neg m_i).$$

So the muddy children puzzle where all children are muddy is fully described by the conjunction:

$$Muddy \wedge Obs \wedge Ign.$$

Let us now look at the consequences of this description in the semantics to be defined. First, the implication

$$(Muddy \wedge Obs \wedge Ign) \rightarrow \langle (\bigvee_{i \in Agt} m_i)?! \rangle Ign$$

will be valid for $n \geq 2$: each child is still ignorant about her muddiness after the announcement that one of them is muddy. Furthermore, the implications

$$(Muddy \wedge Obs \wedge Ign) \rightarrow \langle (\bigvee_{i \in Agt} m_i) ?! \rangle \langle Ign ?! \rangle^k Ign$$

will be valid for $0 \leq k \leq n-2$ and $n \geq 2$, where $\langle \pi ! \rangle^k$ is the iteration of $\langle \pi ! \rangle$, k times: the children keep on being ignorant about their state after $n-2$ rounds of the announcement of that ignorance.

It is important to note that, unlike in DEL-PAO, in DEL-PAO-PP the intended meaning of the formulas $\neg K_i m_i \wedge \neg K_i \neg m_i$ and $\neg S_i \, m_i$ is different: the former says that according to her information state, i is ignorant about m_i, while the latter says that i does not observe m_i. The status of the latter remains unchanged when the children gain new information via the public announcement of Ign. In contrast, the status of the former changes after $n-1$ announcements:

$$(Muddy \wedge Obs \wedge Ign) \rightarrow \langle (\bigvee_{i \in Agt} m_i) ?! \rangle \langle Ign ?! \rangle^{n-1} (\bigwedge_{i \in Agt} K_i m_i)$$

and even

$$(Muddy \wedge Obs \wedge Ign) \rightarrow \langle (\bigvee_{i \in Agt} m_i) ?! \rangle \langle Ign ?! \rangle^{n-1} CK (\bigwedge_{i \in Agt} m_i)$$

will be valid: knowledge and even common knowledge of muddiness is achieved after $n-1$ rounds.

6.2 Semantics of DEL-PAO-PP

We interpret our formulas on *pointed models*. A pointed model is a couple $\langle U, w \rangle$ where:

- $U \subseteq 2^{ATM}$ is the *public information state*, or simply information state. This set of worlds will evolve when programs are publicly executed. For example, the program $\varphi ?$ will make the public information state shrink so that only worlds satisfying φ are kept.
- $w \in 2^{ATM}$ is a world, or valuation, or sometimes local state (in contrast to information state).

A valuation w of DEL-PAO-PP will behave like a valuation of DEL-PAO. In particular, we note $INTR$ the set of worlds w where the five constraints (C1)-(C5) defined in Section 2.2.1 hold. We write them w to empathize that we do not necessarily consider all possible combinations

of atoms from ATM any more, but only the ones from the public information state U. Our valuations hence become closer to standard Kripke "worlds."

Accessibility relations for K_i and CK are defined exactly like in DEL-PAO (see Section 2.2.3):

$$w \sim_i w' \text{ iff } S_i\,\alpha \in w \text{ implies } w(\alpha) = w'(\alpha)$$
$$w \sim_{Agt} w' \text{ iff } JS\,\alpha \in w \text{ implies } w(\alpha) = w'(\alpha)$$

where, again, with $w(\alpha) = w'(\alpha)$ when either $\alpha \in w$ and $\alpha \in w'$, or $\alpha \notin w$ and $\alpha \notin w'$.

The same properties hold: the relation $\sim_{Agt}$ and every $\sim_i$ are equivalence relations on $INTR$ and if $w \in INTR$, if $w \sim_i w'$ or $w \sim_{Agt} w'$ then $w' \in INTR$.

Example 6.2 (Two muddy children)**.** Consider the valuation

$$\begin{aligned} w = \{&S_1\,m_2, S_2\,m_1\} \\ &\cup \{\alpha : \alpha \text{ is valid in } INTR\} \\ &\cup \{\sigma S_i\,m_j : \sigma \in OBS^+, i, j \in \{1,2\}\}. \end{aligned}$$

The last two lines ensure that w is introspective: the second adds all introspective validities and the third adds $JS\,S_i\,m_j$ and all its introspective consequences. Then the four relevant introspective valuations are depicted in Figure 6.1, together with relations $\sim_1$ and $\sim_2$.

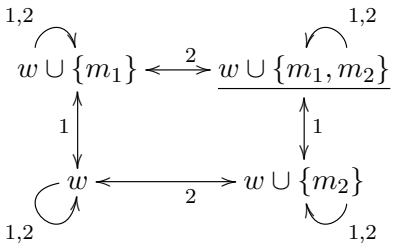


Figure 6.1: Two muddy children.

For the dynamic part, we reuse our notation for updates of valuations defined in Section 2.2.4:

$$w{+}\alpha = w \cup \alpha^{\Rightarrow}$$
$$w{-}\alpha = w \setminus \alpha^{\Leftarrow}$$

where $\alpha^{\Leftarrow}$ is the set of introspective causes of α and $\alpha^{\Rightarrow}$ the set of its introspective consequences, defined like in DEL-PAO (see Section 2.2.1).

We extend it to a public information state $U \subseteq 2^{ATM}$:

$$U{+}\alpha = \{w{+}\alpha : w \in U\}$$
$$U{-}\alpha = \{w{-}\alpha : w \in U\}$$

Again, Proposition 2.7 on page 37 ensures that when w is introspective then both $w{+}\alpha$ and $w{-}\alpha$ are so, too (unless α is valid in $INTR$). By definition, this is still obviously true for the information state: when U only contains introspective worlds then both $U{+}\alpha$ and $U{-}\alpha$ only contains introspective worlds, too (unless α is valid in $INTR$).

Truth conditions are as follows:

$U, w \models \alpha$ iff $\alpha \in w$
$U, w \models \neg\varphi$ iff not $(U, w \models \varphi)$
$U, w \models \varphi \wedge \varphi'$ iff $U, w \models \varphi$ and $U, w \models \varphi'$
$U, w \models K_i\varphi$ iff $U, w' \models \varphi$ for every $w' \in U$ such that $w \sim_i w'$
$U, w \models CK\varphi$ iff $U, w' \models \varphi$ for every $w' \in U$ such that $w \sim_{Agt} w'$
$U, w \models [\pi!]\varphi$ iff $U', w' \models \varphi$ for every $\langle U', w'\rangle$ such that $\langle U, w\rangle \mathcal{P}_\pi \langle U', w'\rangle$

where $\mathcal{P}_\pi$ is a binary relation on pointed models is defined by:

$\langle U, w\rangle \mathcal{P}_{+\alpha} \langle U', w'\rangle$ iff $U' = U{+}\alpha$ and $w' = w{+}\alpha$
$\langle U, w\rangle \mathcal{P}_{-\alpha} \langle U', w'\rangle$ iff $U' = U{-}\alpha$ and $w' = w{-}\alpha$ and α is not valid in $INTR$
$\langle U, w\rangle \mathcal{P}_{\pi;\pi'} \langle U', w'\rangle$ iff $\langle U, w\rangle (\mathcal{P}_\pi \circ \mathcal{P}_{\pi'}) \langle U', w'\rangle$
$\langle U, w\rangle \mathcal{P}_{\pi\sqcup\pi'} \langle U', w'\rangle$ iff $\langle U, w\rangle (\mathcal{P}_\pi \cup \mathcal{P}_{\pi'}) \langle U', w'\rangle$
$\langle U, w\rangle \mathcal{P}_{\chi?} \langle U', w'\rangle$ iff $U, w \models \chi,\ w' = w,$ and $U' = \{u \in U : U, u \models \chi\}$

Note that we do not require $w \in U$.

For epistemic operators, truth conditions are similar to DEL-PAO's, but require the related world w' to be in the public information state U. For the new dynamic operator, the relation $\mathcal{P}_\pi$ includes the modification of the information state. This operator is therefore distinct from the dynamic operator $[\pi]$ of DEL-PAO. Observe that tests make the information state shrink while assignments may shrink, but also transpose it.

Example 6.3 (Two muddy children, continued)**.** Remember that

$$Obs = (\neg S_1\, m_1 \wedge \neg S_2\, m_2) \wedge (S_1\, m_2 \wedge S_2\, m_1) \wedge (JS\ S_1\, m_1 \wedge JS\ S_1\, m_2 \wedge JS\ S_2\, m_1 \wedge JS\ S_2\, m_2)$$

and

$$Ign = (\neg K_1 m_1 \wedge \neg K_1 \neg m_1) \wedge (\neg K_2 m_2 \wedge \neg K_2 \neg m_2).$$

Let U be the set of all valuations in Figure 6.1. We have:

$$\begin{aligned} U, u &\models \mathit{Obs} \wedge \mathit{Ign}, \text{ for every } u \in U \\ U, w \cup \{m_1, m_2\} &\models \langle (m_1 \vee m_2)?! \rangle \mathit{Ign} \\ U, w \cup \{m_1, m_2\} &\models \langle (m_1 \vee m_2)?! \rangle \langle \mathit{Ign}?! \rangle (K_1 m_1 \wedge K_2 m_2) \end{aligned}$$

The proof of last line will be detailed in Section 6.5.

Let $\mathcal{C}$ be a class of pointed models. A formula φ is *satisfiable in* $\mathcal{C}$ if and only if there is a $\langle U, w \rangle \in \mathcal{C}$ such that $U, w \models \varphi$; it is *valid in* $\mathcal{C}$ if and only if $\neg\varphi$ is unsatisfiable. For example, the equivalence $[\chi?!]\bot \leftrightarrow \neg\chi$ is valid in the class of all pointed models.

An *introspective pointed model* is a pointed model $\langle U, w \rangle$ such that:

- $w \in U$;
- $U \subseteq \mathit{INTR}$ is a set of introspective valuations.

For example, $S_i \, \mathit{JS} \, p$ is valid in the class of epistemic models. The following conditions guarantee that when we interpret a formula in an introspective pointed model we stay within the class of introspective pointed models.

Proposition 6.1. *Let $\langle U, w \rangle$ be an introspective pointed model. Then the following hold:*

1. *If $w \sim_i w'$ and $w' \in U$ then $\langle U, w' \rangle$ is an introspective pointed model.*
2. *If $w \sim_{Agt} w'$ and $w' \in U$ then $\langle U, w' \rangle$ is an introspective pointed model.*
3. *If $\langle U, w \rangle \mathcal{P}_\pi \langle U', w' \rangle$ then $\langle U', w' \rangle$ is an introspective pointed model.*

Proof. (1) and (2) are obvious since U does not change and we impose that $w' \in U$.

The third item can be proven by induction on the form of π:

- $\boldsymbol{\pi = +\alpha}$. We have seen that if $U \subseteq \mathit{INTR}$, then $U{+}\alpha \subseteq \mathit{INTR}$. Moreover, by definition of $U{+}\alpha$, $w{+}\alpha$ is obviously contained in $U{+}\alpha$.

  Therefore $\langle U{+}\alpha, w{+}\alpha \rangle$ is an introspective pointed model.

- $\boldsymbol{\pi = -\alpha}$. If α is valid in INTR, then there is no world related to $\langle U, w \rangle$ by $\mathcal{P}_{-\alpha}$. Otherwise, the proof follows the lines of the proof for $\pi = +\alpha$.

- The proofs for $\pi = \pi_1; \pi_2$ and $\pi = \pi_1 \sqcup \pi_2$ are straightforward.

- $\pi = \chi?$. If $U, w \not\models \chi$, then there is no world related to $\langle U, w\rangle$ by $\mathcal{P}_{\chi?}$.

  Otherwise, the resulting information state $\{u \in U : U, u \models \chi\} \subseteq INTR$ since it is included in $U \subseteq INTR$. Moreover, $w \in \{u \in U : U, u \models \chi\}$ since $U, w \models \chi$.

  Therefore $\langle\{u \in U : U, u \models \chi\}, w\rangle$ is an introspective pointed model.

It follows that every $\langle U', w'\rangle$ such that $\langle U, w\rangle \mathcal{P}_\pi \langle U', w'\rangle$ is an introspective pointed model. □

Therefore, as it was the case for introspective valuations in DEL-PAO, we cannot "exit" the class of introspective pointed models when interpreting $K_i\varphi$, $CK\varphi$ or $[\pi!]\varphi$.

It is important to observe that the schemas

$$S_i\,\alpha \wedge \alpha \to K_i\alpha$$
$$S_i\,\alpha \wedge \neg\alpha \to K_i\neg\alpha$$

are still valid in the class of all pointed models (even non-epistemic ones). In contrast, the converse of the implication is now invalid. For example, imagine a public information state where p was publicly tested, i.e., publicly announced: $U = \{w \in 2^{ATM} : p \in w\}$. Then we have $\langle U, \{p\}\rangle \models K_i p$ because in every world from the public information state, p is true, while $\langle U, \{p\}\rangle \not\models S_i\,p$ since $S_i\,p$ is currently false.

6.3 Complexity of model checking

We define the relevant atoms of a DEL-PAO-PP formula φ like in DEL-PAO (see Section 2.4.3), as epistemic operators will reduce in the same way:

$$RATM(K_i\varphi) = RATM(\varphi) \cup \{S_i\,\alpha : \alpha \in RATM(\varphi)\}$$
$$RATM(CK\varphi) = RATM(\varphi) \cup \{JS\,\alpha : \alpha \in RATM(\varphi)\}$$

and equal to $ATM(\varphi)$ otherwise.

Then the model checking problem for DEL-PAO-PP is defined as follows:

- **Input:** a couple $\langle w \cap RATM(\varphi), \varphi\rangle$ where φ is a DEL-PAO-PP formula and w is an introspective valuation;
- **Output:** yes if $INTR, w \models \varphi$, no otherwise.

Note that we do not consider the more general problem of checking a triple $\langle U, w, \varphi\rangle$ where U is a set of valuations. The reason is that the

explicit representation of U may require exponential space in the size of w (that may be double-exponential in the length of φ). One might consider representing U by a boolean formula, as done in [Lomuscio et al., 2000; Su et al., 2007; van Benthem et al., 2015]; however, one cannot represent the set of all introspective valuations $INTR$ in that way.

Theorem 6.1. *The* DEL-PAO-PP *model checking problem is* PSPACE*-complete.*

Proof. The model checking of DL-PA, which is a fragment of DEL-PAO-PP without visibility operators S_i and JS and public information states was proven to be PSPACE-hard [Balbiani et al., 2014]. This establishes the lower bound.

We will show in Chapter 7 that the problem is in PSPACE. □

6.4 Properties of the public programs operator

In this section, we discuss a list of properties of the public programs operator $[\pi!]$.

6.4.1 Expressing public announcement of formulas

Consider the operator $[\chi!]$ of public announcement of a formula χ as studied in dynamic epistemic logics [van Ditmarsch et al., 2007]. In the present setting, its truth condition has to be formulated as follows:

$$U, w \models [\chi!]\varphi \qquad \text{iff } U, w \models \chi \text{ implies } \{u \in U : U, u \models \chi\}, w \models \varphi.$$

The set $\{u \in U : U, u \models \chi\}$ is called the relativisation of U to the extension of χ in U.

Let us compare this to the public performance of tests: the relativisation of U to the extension of χ in U is nothing but the result of the public update of U by $\chi?$. Indeed, $\langle U, w\rangle \mathcal{P}_{\chi?} \langle U', w'\rangle$ is the case if and only if $U, w \models \chi$ and U' is the restriction of U to the extension of χ in U. So $[\chi!]\varphi$ and $[\chi?!]\varphi$ have identical truth conditions.

6.4.2 Some reduction axioms

We could not find a complete axiomatization based on reduction axioms: as we have seen, the equivalences

$$K_i\alpha \leftrightarrow S_i\,\alpha \wedge \alpha$$
$$K_i\neg\alpha \leftrightarrow S_i\,\alpha \wedge \neg\alpha$$

are not valid any more, preventing the reduction of the epistemic operators like in DEL-PAO. We will see in the next chapter that these operators can nevertheless be reduced to another kind of program that do not change the public information state when executed. However, these programs, while being useful for deciding the complexity, cannot be fully reduced and therefore are not sufficient to provide an axiomatization either. We discuss some valid and invalid equivalences.

The following equivalences reduce all programs to either atomic programs or public tests:

$$[\pi;\pi'!]\varphi \leftrightarrow [\pi!][\pi'!]\varphi$$
$$[\pi \sqcup \pi'!]\varphi \leftrightarrow [\pi!]\varphi \wedge [\pi'!]\varphi$$

The proofs given in Proposition 2.11 on page 50 still apply.

As to public tests, they can be reduced against boolean operators and the individual knowledge operator; the following equivalences are valid on introspective pointed models:

$$[\chi?!]\beta \leftrightarrow \neg\chi \vee \beta$$
$$[\chi?!]\neg\varphi \leftrightarrow \neg\chi \vee \neg[\chi?!]\varphi$$
$$[\chi?!](\varphi \wedge \varphi') \leftrightarrow [\chi?!]\varphi \wedge [\chi?!]\varphi'$$
$$[\chi?!]K_i\varphi \leftrightarrow \neg\chi \vee K_i[\chi?!]\varphi$$
$$[\chi?!]CK\varphi \leftrightarrow \neg\chi \vee CK[\chi?!]\varphi$$

where β is an atom. Observe that these axioms are exactly the reduction axioms of public announcement logic PAL [Wang and Cao, 2013] plus an axiom for common knowledge that is due to the similarity between this operator and the individual knowledge operator. As we have seen, our public test has the same truth conditions as public announcements.

As to positive public assignments, they are deterministic and distribute over the boolean operators:

$$[+\alpha!]\beta \leftrightarrow \begin{cases} \top & \text{if } \alpha \Rightarrow_I \beta \\ \beta & \text{otherwise} \end{cases}$$
$$[+\alpha!]\neg\varphi \leftrightarrow \neg[+\alpha!]\varphi$$
$$[+\alpha!](\varphi \wedge \varphi') \leftrightarrow [+\alpha!]\varphi \wedge [+\alpha!]\varphi'$$

Similarly, for negative public assignments we have

$$[-\alpha!]\beta \leftrightarrow \begin{cases} \top & \text{if } \alpha \text{ is valid in } \mathit{INTR} \\ \bot & \text{if } \alpha \text{ is not valid in } \mathit{INTR} \text{ and } \beta \Rightarrow_I \alpha \\ \beta & \text{otherwise} \end{cases}$$

$$[-\alpha!]\neg\varphi \leftrightarrow \begin{cases} \top & \text{if } \alpha \text{ is valid in } \mathit{INTR} \\ \neg[-\alpha!]\varphi & \text{otherwise} \end{cases}$$

$$[-\alpha!](\varphi \wedge \varphi') \leftrightarrow [-\alpha!]\varphi \wedge [-\alpha!]\varphi'$$

We have already proven them within the framework of DEL-PAO (see Proposition 2.11 on page 50).

6.4.3 Replacement of equivalents

The above equivalences can be applied anywhere in a formula because the inference rule of replacement of equivalents preserves validity.

Proposition 6.2. *Let φ' be obtained from φ by replacing some occurrence of χ in φ by χ'. Let U be a set of valuations. If $U, w \models \chi \leftrightarrow \chi'$ for every $w \in U$ then $U, w \models \varphi \leftrightarrow \varphi'$ for every $w \in U$.*

Proof. This is due to the fact that, along with the ones for K_i and CK (see Proposition 2.12 on page 52) the following rule of inference for $[\pi!]$:

$$\frac{\varphi \leftrightarrow \psi}{[\pi!]\varphi \leftrightarrow [\pi!]\psi}$$

preserves validity. □

6.5 Muddy children, proved

In this section we formally prove the statement we made in Example 6.3 about the muddy children puzzle for the case of two children. Remember that we have assumed that each child is muddy:

$$\mathit{Muddy} = m_1 \wedge m_2.$$

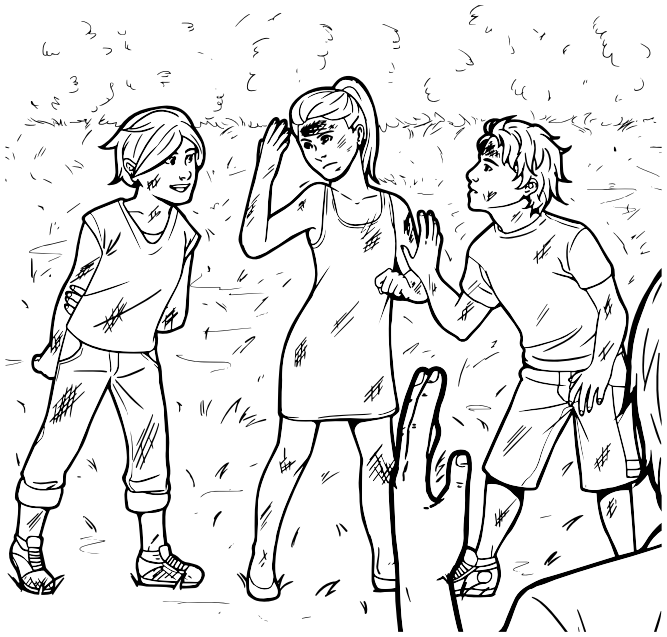
As usual, each child sees the other but cannot see herself:

$$\begin{aligned} \mathit{Obs} = {} & (\neg S_1\, m_1 \wedge \neg S_2\, m_2) \wedge (S_1\, m_2 \wedge S_2\, m_1) \wedge \\ & (\mathit{JS}\, S_1\, m_1 \wedge \mathit{JS}\, S_1\, m_2 \wedge \mathit{JS}\, S_2\, m_1 \wedge \mathit{JS}\, S_2\, m_2), \end{aligned}$$

and initially each child does not know whether she is muddy or not:

$$\mathit{Ign} = (\neg K_1 m_1 \wedge \neg K_1 \neg m_1) \wedge (\neg K_2 m_2 \wedge \neg K_2 \neg m_2).$$

We use the following validity of PAL.



Proposition 6.3. *Let φ and φ' be boolean formulas. Then*

$$(K_i\varphi \wedge \neg K_i\varphi' \wedge \neg K_i\neg\varphi') \to [(\varphi \vee \varphi')?!](\neg K_i\varphi' \wedge \neg K_i\neg\varphi')$$

is valid in introspective pointed models.

Intuitively, the above proposition says that if agent i knows a fact φ but does not know the fact φ', then publicly announcing that φ or φ' is true does not increase her knowledge about φ'.

Proof. Suppose $U, w \models K_i\varphi$. Then for every $\langle U', w'\rangle$ such that $\langle U, w\rangle \sim_i \langle U', w'\rangle$, $U', w' \models \varphi$. Moreover, suppose $U, w \models \neg K_i\varphi' \wedge \neg K_i\neg\varphi'$. Then there exists a $\langle U_1, w_1\rangle$ such that $U_1, w_1 \models \varphi'$ and there exists a $\langle U_2, w_2\rangle$ such that $U_2, w_2 \not\models \varphi'$.

Now suppose we announce that $\varphi \vee \varphi'$. Every pointed model which was previously related to $\langle U, w\rangle$ by $\sim_i$ will still be since φ, and hence $\varphi \vee \varphi'$, is true in all of them. Therefore there still exists a related pointed model where φ' is true and a related pointed model where φ' is false. Thus after the announcement of $\varphi \vee \varphi'$, we still have $\neg K_i\varphi' \wedge \neg K_i\neg\varphi'$. □

The formulas

$$(K_1m_2 \wedge \neg K_1m_1 \wedge \neg K_1\neg m_1) \to [m_2 \vee m_1?!](\neg K_1m_1 \wedge \neg K_1\neg m_1)$$

and

$$(K_2m_1 \wedge \neg K_2m_2 \wedge \neg K_2\neg m_2) \to [m_1 \vee m_2?!](\neg K_2m_2 \wedge \neg K_2\neg m_2)$$

are instances of the above PAL validity. Observe that $Muddy \wedge Obs$ implies $K_1 m_2$ and $K_2 m_1$ while Ign implies $\neg K_1 m_1 \wedge \neg K_1 \neg m_1$ and $\neg K_2 m_2 \wedge \neg K_2 \neg m_2$. Therefore:

$$(Muddy \wedge Obs \wedge Ign) \rightarrow [m_1 \vee m_2 ?!] Ign.$$

Moreover:

$$Muddy \rightarrow \langle m_1 \vee m_2 ?! \rangle \top.$$

Putting the last two implications together we obtain:

$$(Muddy \wedge Obs \wedge Ign) \rightarrow \langle m_1 \vee m_2 ?! \rangle Ign.$$

Similarly, we can establish that

$$(Muddy \wedge Obs \wedge Ign) \rightarrow \langle m_1 \vee m_2 ?! \rangle \langle Ign ?! \rangle (K_1 m_1 \wedge K_2 m_2)$$

is valid in introspective pointed models.

6.6 Conclusion

We have extended DEL-PAO in such a way that knowledge of agents is deduced from what they see and from what is publicly announced to them. We thereby solve the second issue of previous observability-based approaches: knowledge operators do not distribute over disjunction any more. This latter feature allows us to formalize the muddy children puzzle in a natural way.

Beyond public announcements, we can reason about publicly executed programs: public announcements are special cases of publicly executed tests. This allows us to formalize variants of the muddy children puzzle where the children e.g. clean their forehead [van Ditmarsch et al., 2005]. The addition of public programs again comes without increasing the complexity as the model checking problem of DEL-PAO-PP is still PSPACE-complete. It however remains to find an axiomatization.

7 Complexity of model checking: upper bound

*In this chapter, we prove that the model checking problem for all logics presented in this thesis—*DEL-PAO *in Chapter 2,* DEL-PAO^S *in Chapter 4,* DEL-PAOC *in Chapter 5 and* DEL-PAO-PP *in Chapter 6—is in* PSPACE*. We do so by providing an algorithm for a language into which all formulas from every previous logic can be translated. This language includes publicly executed programs, used to encode* DEL-PAO-PP *programs, as well as "mental programs" that do not modify the public information state, allowing us to encode* DEL-PAO *programs.*

To cover all languages, we provide semantics based on a public information state and a valuation (like in Chapter 6). However, public programs of DEL-PAO-PP modify the public information state and thus differ from programs presented in DEL-PAO, DEL-PAO^S and DEL-PAOC. We have also seen in Chapter 2 and in Chapter 5 how to reduce epistemic and strategic operators to programs. Therefore it will be useful to include in our language both public programs and a new operator K_π, where π will be a "mental program," i.e., will not modify the public information state. Such operators were introduced e.g. in [van Benthem et al., 2006; Charrier and Schwarzentruber, 2015; Herzig et al., 2015]. Intuitively, $[\pi!]\varphi$ is read "φ will be true after the public update of the current local and information states by π," while $K_\pi\varphi$ is read "φ will be true after the update of the current local state by π (keeping the current information state constant)." So at a given local state w and information state U, the public program operator $[\pi!]$ updates both U and w; in contrast, the mental program operator K_π keeps U constant and only updates w. The latter can be viewed as traversing the space of current epistemic possibilities.[14]

[14] This chapter is inspired by a section of [Charrier et al., 2016a], but has been revised

We call this logic DL-PA-PMP: Dynamic Logic of Propositional Assignments with Public and Mental Programs.

Contents

Résumé du chapitre

Dans ce chapitre, nous prouvons que le problème de vérification de modèle pour toutes les logiques présentées dans la thèse—DEL-PAO dans le chapitre 2, DEL-PAO^S dans le chapitre 4, DEL-PAOC dans le chapitre 5 et DEL-PAO-PP dans le chapitre 6—est en PSPACE. Nous le faisons en fournissant un algorithme pour un langage dans lequel toutes les formules de toutes les précédentes logiques peuvent être traduites. Ce langage comprend des programmes exécutés publiquement, utilisés pour encoder les programmes de DEL-PAO-PP, ainsi que des "programmes mentaux" qui ne modifient pas l'état d'information, ce qui nous permet d'encoder des programmes de DEL-PAO.

Pour couvrir tous les langages, nous fournissons une sémantique s'appuyant sur un état d'information et une valuation (comme dans le chapitre 6). Toutefois, les programmes publics de DEL-PAO-PP modifient l'état d'information et diffèrent donc des programmes présentés dans DEL-PAO, DEL-PAO^S et DEL-PAOC. Nous avons également vu dans le chapitre 2 et dans le chapitre 5 comment réduire les opérateurs épistémiques et stratégiques à ces programmes. Par conséquent, il sera utile d'inclure dans notre langage à la fois les programmes publics et un nouvel opérateur K_π, où π sera un "programme mental", c'est-à-dire, qui ne modifiera pas l'état d'information. Ces opérateurs ont été introduits par

in order to include properly all presented logics: the Kleene star has been added and translations from every language, along with proofs of correctness, have been added. The model checking algorithms remain unchanged (apart from the star that was included, following [Charrier et al., 2016b]).

exemple dans [van Benthem et al., 2006; Charrier and Schwarzentruber, 2015; Herzig et al., 2015]. Intuitivement, $[\pi!]\varphi$ est lu "φ sera vrai après la mise à jour des états local et d'information actuels par π", tandis que $K_\pi\varphi$ est lu "φ sera vrai après la mise à jour de l'état local actuel par π (en gardant l'état d'information courant constant)". Donc, étant donnés un état local w et un état d'information U, l'opérateur de programme public $[\pi!]$ met à jour à la fois U et w, alors que l'opérateur de programme mental K_π garde U constant et met seulement à jour w. Ce dernier peut être vu comme traversant l'espace des possibilités épistémiques actuelles.

Nous appelons cette logique DL-PA-PMP : une logique dynamique des affectations propositionnelles avec des programmes publics et mentaux.

7.1 DL-PA-PMP: DL-PA with public and mental programs

To show that the problem is in PSPACE, we adapt the alternating algorithm in [Charrier and Schwarzentruber, 2015], originally designed for a variant of a dynamic logic with propositional assignments, public announcements and arbitrary public announcements. Here we consider another, novel variant without arbitrary public announcements but with public programs: DL-PA-PMP.

7.1.1 Language of DL-PA-PMP

Remember that $Prop$ is a countable non-empty set of propositional variables and and Agt is a finite non-empty set of agents.

The set of visibility operators OBS is defined like in DEL-PAO:

$$OBS = \{S_i : i \in Agt\} \cup \{JS\}.$$

We also include the set of control operators $CTRL$ from DEL-PAOC:

$$CTRL = \{C_i : i \in Agt\},$$

so that atoms are about both visibility and control:

$$ATM = \{\sigma\, p : \sigma \in (OBS \cup CTRL)^*, p \in Prop\}.$$

Then the syntax of DL-PA-PMP is defined by the following grammar:

$$\begin{array}{rl}
\pi_{\mathtt{p}} ::= & \alpha{\leftarrow}\top \mid \alpha{\leftarrow}\bot \mid (\pi_{\mathtt{p}}; \pi_{\mathtt{p}}) \mid (\pi_{\mathtt{p}} \sqcup \pi_{\mathtt{p}}) \mid \varphi? \\
\pi_{\mathtt{m}} ::= & \alpha{\leftarrow}\top \mid \alpha{\leftarrow}\bot \mid (\pi_{\mathtt{m}}; \pi_{\mathtt{m}}) \mid (\pi_{\mathtt{m}} \sqcup \pi_{\mathtt{m}}) \mid {\pi_{\mathtt{m}}}^* \mid \varphi? \\
\varphi ::= & \alpha \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid [\pi_{\mathtt{p}}!]\varphi \mid K_{\pi_{\mathtt{m}}}\varphi
\end{array}$$

where α ranges over ATM.

Unlike in previous logics, here assignments are noted $\alpha{\leftarrow}\top$ and $\alpha{\leftarrow}\bot$ to highlight that they are different from our previous $+\alpha$ and $-\alpha$. Indeed, even if we include joint visibility operators JS, we will see that their semantics do not take introspective consequences and introspectively valid atoms into account.

The other boolean operators abbreviate as usual, in the standard way. The set of atoms appearing in the formula φ and in the program π, noted $ATM(\varphi)$ and $ATM(\pi)$, are also defined like in DEL-PAO for boolean operators. Moreover:

$$ATM([\pi_{\mathtt{p}}!]\varphi) = ATM(\pi_{\mathtt{p}}) \cup ATM(\varphi)$$
$$ATM(K_{\pi_{\mathtt{m}}}\varphi) = ATM(\pi_{\mathtt{m}}) \cup ATM(\varphi)$$

The language of DL-PA-PMP is slightly different from the one presented in [Charrier and Schwarzentruber, 2015]; here, we:

- consider general programs $[\pi_{\mathtt{p}}!]\varphi$ instead of only public announcements;
- drop arbitrary public announcements;
- include the Kleene star in mental programs.

This implies that the model checking procedure will have to consider general publicly executed programs and the Kleene star. The latter is however included in an extended version of [Charrier and Schwarzentruber, 2015] that can be found in [Charrier et al., 2016b].

7.1.2 Semantics of DL-PA-PMP

Formulas are interpreted on pointed models $\langle U, w\rangle$ (see Section 6.2): public programs will modify the public information state while mental programs will not. The latter will allow us to simulate DEL-PAO programs within the framework of DL-PA-PMP.

As we have mentioned, assignments do not take into account introspective consequences even with JS operators, so that a finite number of atoms is modified; it is the translation of other logics formulas into DL-PA-PMP that will deal with consequences. We define:

$$w{+}\alpha = w \cup \{\alpha\}$$
$$w{-}\alpha = w \setminus \{\alpha\}$$

and

$$U{+}\alpha = \{w{+}\alpha : w \in U\}$$
$$U{-}\alpha = \{w{-}\alpha : w \in U\}$$

Then the truth conditions of DL-PA-PMP are as follows:

$$
\begin{array}{ll}
U, w \models \alpha & \text{iff } \alpha \in w \\
U, w \models \neg\varphi & \text{iff not } (U, w \models \varphi) \\
U, w \models \varphi \wedge \varphi' & \text{iff } U, w \models \varphi \text{ and } U, w \models \varphi' \\
U, w \models [\pi!]\varphi & \text{iff } U', w' \models \varphi \text{ for every } \langle U', w'\rangle \text{ such that } \langle U, w\rangle \mathcal{P}_\pi \langle U', w'\rangle \\
U, w \models K_\pi\varphi & \text{iff } U, w' \models \varphi \text{ for every } w' \in U \text{ such that } \langle U, w\rangle \mathcal{M}_\pi \langle U, w'\rangle
\end{array}
$$

where $\mathcal{P}_\pi$ is the public relation on pointed models is defined by:

$$
\begin{array}{ll}
\langle U, w\rangle \mathcal{P}_{\alpha\leftarrow\top} \langle U', w'\rangle & \text{iff } U' = U{+}\alpha \text{ and } w' = w{+}\alpha \\
\langle U, w\rangle \mathcal{P}_{\alpha\leftarrow\bot} \langle U', w'\rangle & \text{iff } U' = U{-}\alpha \text{ and } w' = w{-}\alpha \\
\langle U, w\rangle \mathcal{P}_{\pi;\pi'} \langle U', w'\rangle & \text{iff } \langle U, w\rangle (\mathcal{P}_\pi \circ \mathcal{P}_{\pi'}) \langle U', w'\rangle \\
\langle U, w\rangle \mathcal{P}_{\pi\sqcup\pi'} \langle U', w'\rangle & \text{iff } \langle U, w\rangle (\mathcal{P}_\pi \cup \mathcal{P}_{\pi'}) \langle U', w'\rangle \\
\langle U, w\rangle \mathcal{P}_{\chi?} \langle U', w'\rangle & \text{iff } U, w \models \chi,\ w' = w, \text{ and } U' = \{u \in U : U, u \models \chi\}
\end{array}
$$

and $\mathcal{M}_\pi$ is the mental relation on pointed models that is defined by:

$$
\begin{array}{ll}
\langle U, w\rangle \mathcal{M}_{\alpha\leftarrow\top} \langle U', w'\rangle & \text{iff } U' = U \text{ and } w' = w{+}\alpha \\
\langle U, w\rangle \mathcal{M}_{\alpha\leftarrow\bot} \langle U', w'\rangle & \text{iff } U' = U \text{ and } w' = w{-}\alpha \\
\langle U, w\rangle \mathcal{M}_{\pi;\pi'} \langle U', w'\rangle & \text{iff } \langle U, w\rangle (\mathcal{M}_\pi \circ \mathcal{M}_{\pi'}) \langle U', w'\rangle \\
\langle U, w\rangle \mathcal{M}_{\pi\sqcup\pi'} \langle U', w'\rangle & \text{iff } \langle U, w\rangle (\mathcal{M}_\pi \cup \mathcal{M}_{\pi'}) \langle U', w'\rangle \\
\langle U, w\rangle \mathcal{M}_{\pi^*} \langle U', w'\rangle & \text{iff } \langle U, w\rangle (\bigcup_{k\in\mathbb{N}_0} (\mathcal{M}_\pi)^k) \langle U', w'\rangle \\
\langle U, w\rangle \mathcal{M}_{\chi?} \langle U', w'\rangle & \text{iff } U' = U,\ w' = w \text{ and } U, w \models \chi
\end{array}
$$

Observe that mental programs indeed do not change the public information state: when $\langle U, w\rangle \mathcal{M}_\pi \langle U', w'\rangle$ then $U' = U$. However, the execution of mental programs may exit the information state. To see this, take as an example $U = \{w \in 2^{ATM} : p \notin w\}$, i.e., the set of valuations where p is false. $\mathcal{M}_{p\leftarrow\top}$ relates the pointed model $\langle U, \emptyset\rangle$ to the model $\langle U, \{p\}\rangle$, but $\{p\} \notin U$.

As we will translate DEL-PAO-PP's epistemic operators, that depend on the information state, into mental programs, we impose that truth condition for K_π requires $w' \in U$: valuations outside the U will not be taken into account.

Note also that unlike in all other semantics presented in this thesis, introspectively valid atoms can be removed by the public or mental execution of $\alpha\leftarrow\bot$. Like introspective consequences, the translation into DL-PA-PMP will have to deal with them.

7.2 Translation into DL-PA-PMP

In this section, we show how to translate formulas of the previously presented logics into the language of DL-PA-PMP. We prove that the translation is correct on models we are interested in: the formula is satisfied in a given (introspective) model of its original framework if and only if it is satisfied in the corresponding (finite) model of DL-PA-PMP that we will use for model checking.

We define the restriction of a set of valuations to a set of atoms A as:

$$U|_A = \{u \cap A : u \in U\}.$$

Note that $U|_A$ is a finite set of finite valuations whenever A is finite.

To avoid confusion, we indicate to which semantics we refer to by subscripting the satisfaction relation $\models$ by the name of the logic. When we mention translation functions, we always refer to the ones defined at the beginning of the current subsection.

7.2.1 From DEL-PAO

Take a DEL-PAO formula φ. We define the following procedure of translation of φ into DL-PA-PMP.

Procedure 7.1.

1. eliminate all epistemic operators K_i and CK from φ using Procedure 2.2 on page 61; call the resulting formula φ';
2. translate φ' into the DL-PA-PMP formula $tr^{\mathtt{fml}}_{ATM(\varphi')}(\varphi')$ according to the following definition:

$$\begin{aligned} tr^{\mathtt{fml}}_A(\alpha) &= \alpha \\ tr^{\mathtt{fml}}_A(\neg\varphi) &= \neg tr^{\mathtt{fml}}_A(\varphi) \\ tr^{\mathtt{fml}}_A(\varphi_1 \wedge \varphi_2) &= tr^{\mathtt{fml}}_A(\varphi_1) \wedge tr^{\mathtt{fml}}_A(\varphi_2) \\ tr^{\mathtt{fml}}_A([\pi]\varphi) &= K_{tr^{\mathtt{prg}}_A(\pi)} tr^{\mathtt{fml}}_A(\varphi) \end{aligned}$$

where $tr^{\mathtt{prg}}_A(\pi)$ is defined as:

$$\begin{aligned} tr^{\mathtt{prg}}_A(+\alpha) &= \beta_1{\leftarrow}\top; \ldots; \beta_m{\leftarrow}\top \\ tr^{\mathtt{prg}}_A(-\alpha) &= \begin{cases} \mathit{fail} & \text{if } \alpha \text{ valid in } \mathit{INTR} \\ \beta'_1{\leftarrow}\bot; \ldots; \beta'_p{\leftarrow}\bot & \text{otherwise} \end{cases} \\ tr^{\mathtt{prg}}_A(\pi_1;\pi_2) &= tr^{\mathtt{prg}}_A(\pi_1); tr^{\mathtt{prg}}_A(\pi_2) \\ tr^{\mathtt{prg}}_A(\pi_1 \sqcup \pi_2) &= tr^{\mathtt{prg}}_A(\pi_1) \sqcup tr^{\mathtt{prg}}_A(\pi_2) \\ tr^{\mathtt{prg}}_A(\pi^*) &= tr^{\mathtt{prg}}_A(\pi)^* \\ tr^{\mathtt{prg}}_A(\chi?) &= tr^{\mathtt{fml}}_A(\chi)? \end{aligned}$$

with

$$\{\beta_1, \ldots, \beta_m\} = \alpha^{\Rightarrow} \cap A$$
$$\{\beta'_1, \ldots, \beta'_p\} = \alpha^{\Leftarrow} \cap A$$

Remember that $\alpha^{\Leftarrow}$ and $\alpha^{\Rightarrow}$ are the introspective causes and consequences of α. We note $tr_{\text{DEL-PAO}}(\varphi)$ the formula obtained after applying Procedure 7.1 to φ.

Lemma 7.1. *Let π be a* DEL-PAO *program and $A \subseteq ATM$ a set of atoms. Let w be a* DL-PA-PMP *valuation such that $w \in INTR$. Then for every w' such that $\langle INTR|_A, w \cap A\rangle \mathcal{M}_{tr_A^{\texttt{prg}}(\pi)} \langle INTR|_A, w'\rangle$, we have $w' \in INTR|_A$.*

Proof. Observe that $w \in INTR$ implies that $w \cap A \in INTR|_A$. We prove the property by induction on the form of π.

- $\boldsymbol{\pi = +\alpha}$**.** In this case, we have $tr_A^{\texttt{prg}}(\pi) = \beta_1{\leftarrow}\top; \ldots; \beta_m{\leftarrow}\top$ with $\{\beta_1, \ldots, \beta_m\} = \alpha^{\Rightarrow} \cap A$. Hence by the truth conditions of DL-PA-PMP, $w' = (w \cap A) \cup (\alpha^{\Rightarrow} \cap A) = (w \cup \alpha^{\Rightarrow}) \cap A$. Since $w \in INTR$, we have seen in DEL-PAO that $w \cup \alpha^{\Rightarrow} \in INTR$, hence $(w \cup \alpha^{\Rightarrow}) \cap A \in INTR|_A$.
- $\boldsymbol{\pi = -\alpha}$**.** If α is valid in $INTR$, $tr_A^{\texttt{prg}}(-\alpha) = \mathit{fail}$ and the property is trivially valid. Otherwise, the proof is similar to the case $\pi = +\alpha$.
- The proofs for $\pi = \pi_1; \pi_2$ and $\pi = \pi_1 \sqcup \pi_2$ are straightforward since their translation is homomorphic and the associated relation $\mathcal{M}_{tr_A^{\texttt{prg}}(\pi)}$ is a composition or union of $\mathcal{M}_{tr_A^{\texttt{prg}}(\pi_1)}$ and $\mathcal{M}_{tr_A^{\texttt{prg}}(\pi_2)}$.
- $\boldsymbol{\pi = \chi?}$**.** Then $tr_A^{\texttt{prg}}(\pi) = tr_A^{\texttt{fml}}(\chi)?$. Like when assigning to false an introspectively valid atom, the proof is obvious if the program fails. Otherwise, the truth condition indicates that $w' = w \cap A$, hence w' trivially belongs to $INTR|_A$.

In all cases we have $w' \in INTR|_A$. □

Lemma 7.1 implies that on specific states, translations of DEL-PAO programs cannot exit the public information state. This helps us prove the next equivalences.

Proposition 7.1. *Let $A \subseteq ATM$ be a set of atoms and $w \in INTR$ an introspective valuation. Then we have:*

$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr_A^{\texttt{prg}}(\pi_1;\pi_2)}\varphi \leftrightarrow K_{tr_A^{\texttt{prg}}(\pi_1)} K_{tr_A^{\texttt{prg}}(\pi_2)}\varphi$$
$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr_A^{\texttt{prg}}(\pi_1 \sqcup \pi_2)}\varphi \leftrightarrow K_{tr_A^{\texttt{prg}}(\pi_1)}\varphi \wedge K_{tr_A^{\texttt{prg}}(\pi_2)}\varphi$$
$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr_A^{\texttt{prg}}(\pi^*)}\varphi \leftrightarrow K_{tr_A^{\texttt{prg}}(\pi^{2^{2|ATM(\pi)|}})}\varphi$$
$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr_A^{\texttt{prg}}(\chi?)}\varphi \leftrightarrow tr_A^{\texttt{fml}}(\chi) \rightarrow \varphi$$

Proof. All these equivalences are plainly valid in DEL-PAO (see Proposition 2.11 on page 50) and most are equivalences of dynamic logics. They still apply in DL-PA-PMP on certain models because, first, the translation of these program operators is homomorphic, and second, because of Lemma 7.1 that ensures that on this kind of model, translations of programs for the set A do not exit the public information state. This is especially important for the sequence: $K_{tr_A^{\mathtt{prg}}(\pi_1;\pi_2)}\varphi \leftrightarrow K_{tr_A^{\mathtt{prg}}(\pi_1)}K_{tr_A^{\mathtt{prg}}(\pi_2)}\varphi$ is not valid in the general case. □

Proposition 7.2. *Let $V \in INTR$ be an introspective valuation and φ a* DEL-PAO *formula. Then we have:*

$$V \models_{\text{DEL-PAO}} \varphi \text{ if and only if } INTR|_{RATM(\varphi)}, V \cap RATM(\varphi) \models_{\text{DL-PA-PMP}} tr_{\text{DEL-PAO}}(\varphi)$$

Proof. We follow the translation procedure and use properties of DEL-PAO and DL-PA-PMP programs to prove the equivalence.

- First of all, remove epistemic operators from φ, and write the resulting formula φ'; by Proposition 2.17 on page 59, it is equivalent to φ since V is introspective. We have seen that $RATM(\varphi) = ATM(\varphi')$. Call this set A. Translate the resulting formula into DL-PA-PMP; we obtain $tr_A^{\mathtt{fml}}(\varphi')$.

- In DEL-PAO, apply validities $(Red_;)$, $(Red_\sqcup)$, (Red_*) and $(Red_?)$ of Proposition 2.11 on page 50 to φ'. We obtain a formula φ'' equivalent to φ' and with the same atoms, but with only boolean operators and assignment programs.

  In DL-PA-PMP, apply validities of Proposition 7.1 to $tr_A^{\mathtt{fml}}(\varphi')$, without "breaking" sequences of assignments that originate from the same DEL-PAO assignment. (For example, suppose we translate $[+JS\,p]S_i\,p$, we will obtain $K_{JS\,p\leftarrow\top;S_i\,p\leftarrow\top}S_i\,p$, with $JS\,p\leftarrow\top; S_i\,p\leftarrow\top$ both originating from $+JS\,p$.) We obtain a new formula that is equivalent to $tr_A^{\mathtt{fml}}(\varphi')$, and actually identical to the translation of φ'' (noted $tr_A^{\mathtt{fml}}(\varphi'')$) since, first, translations of program operators ;, $\sqcup$, * and $?$ are homomorphic and second, because the equivalences that we apply are identical to the ones of Proposition 2.11.

Therefore we only need to prove that

$$V \models_{\text{DEL-PAO}} \varphi'' \text{ if and only if } INTR|_A, V \cap A \models_{\text{DL-PA-PMP}} tr_A^{\mathtt{fml}}(\varphi'').$$

We do it by induction on the form of φ'' (remember that it only contains boolean operators and assignments programs):

- **φ'' is boolean.** In this case, $tr_A^{\mathtt{fml}}(\varphi'') = \varphi''$. Then $V \models_{\mathsf{DEL\text{-}PAO}} \varphi''$ is equivalent to $V \cap A \models_{\mathsf{DEL\text{-}PAO}} \varphi''$ by Proposition 2.9 on page 38, which is equivalent to $U, V \cap A \models_{\mathsf{DL\text{-}PA\text{-}PMP}} \varphi''$ for any public information state U since the truth conditions for boolean formulas are identical in DEL-PAO and in DL-PA-PMP and only depend on the local state. In particular, $INTR|_A, V \cap A \models_{\mathsf{DL\text{-}PA\text{-}PMP}} \varphi''$.

- **$\varphi'' = [+\alpha]\psi$.** Then we have $tr_A^{\mathtt{fml}}(\varphi'') = K_{\beta_1 \leftarrow \top; \ldots; \beta_m \leftarrow \top} tr_A^{\mathtt{fml}}(\psi)$ with $\{\beta_1, \ldots, \beta_m\} = \alpha^{\Rightarrow} \cap A$. Hence:

$$
\begin{aligned}
& INTR|_A, V \cap A \models_{\mathsf{DL\text{-}PA\text{-}PMP}} K_{\beta_1 \leftarrow \top; \ldots; \beta_m \leftarrow \top} tr_A^{\mathtt{fml}}(\psi) \\
\Leftrightarrow\ & INTR|_A, (V \cap A) \cup (\alpha^{\Rightarrow} \cap A) \models_{\mathsf{DL\text{-}PA\text{-}PMP}} tr_A^{\mathtt{fml}}(\psi) \\
\Leftrightarrow\ & INTR|_A, (V \cup \alpha^{\Rightarrow}) \cap A \models_{\mathsf{DL\text{-}PA\text{-}PMP}} tr_A^{\mathtt{fml}}(\psi) \\
\Leftrightarrow\ & V \cup \alpha^{\Rightarrow} \models_{\mathsf{DEL\text{-}PAO}} \psi \qquad \text{(by induction hypothesis)} \\
\Leftrightarrow\ & V \models_{\mathsf{DEL\text{-}PAO}} [+\alpha]\psi.
\end{aligned}
$$

- **$\varphi'' = [-\alpha]\psi$.** The proof is obvious if α is introspectively valid (as it translates to *fail*) and similar to the case $\varphi'' = [+\alpha]\psi$ otherwise.

Therefore the translation is correct. □

This settles the case of DEL-PAO. The properties of other logics are slightly different due to their respective operators and semantics, but we will see that the method is the same.

7.2.2 From DEL-PAO^S

Observe that DEL-PAO^S is a fragment of DEL-PAOC: neither of them consider the operator of joint visibility JS or common knowledge, and their semantics are strictly identical for boolean, epistemic and dynamic operators; DEL-PAOC simply further includes the strategic operator $\Diamond_J$. Therefore we do not detail any procedure for DEL-PAO^S, as it is completely identical to the one for DEL-PAOC, without the reduction of strategic operators at the beginning.

7.2.3 From DEL-PAOC

Take a DEL-PAOC formula φ. We define the following procedure of translation of φ into DL-PA-PMP.

Procedure 7.2.

1. eliminate all epistemic operators K_i and strategic operators $\Diamond_J$ from φ using Procedure 5.1 on page 121; call the resulting formula φ';

2. translate φ' into the DL-PA-PMP formula $tr^{\mathtt{fml}}(\varphi')$ according to the following definition:

$$\begin{aligned} tr^{\mathtt{fml}}(\alpha) &= \alpha \\ tr^{\mathtt{fml}}(\neg\varphi) &= \neg tr^{\mathtt{fml}}(\varphi) \\ tr^{\mathtt{fml}}(\varphi_1 \wedge \varphi_2) &= tr^{\mathtt{fml}}(\varphi_1) \wedge tr^{\mathtt{fml}}(\varphi_2) \\ tr^{\mathtt{fml}}([\pi]\varphi) &= K_{tr^{\mathtt{prg}}(\pi)} tr^{\mathtt{fml}}(\varphi) \end{aligned}$$

where $tr^{\mathtt{prg}}(\pi)$ is defined as:

$$\begin{aligned} tr^{\mathtt{prg}}(+\alpha) &= \alpha{\leftarrow}\top \\ tr^{\mathtt{prg}}(-\alpha) &= \begin{cases} \mathit{fail} & \text{if } \alpha \text{ valid in } \mathit{INTR} \\ \alpha{\leftarrow}\bot & \text{otherwise} \end{cases} \\ tr^{\mathtt{prg}}(\pi_1;\pi_2) &= tr^{\mathtt{prg}}(\pi_1); tr^{\mathtt{prg}}(\pi_2) \\ tr^{\mathtt{prg}}(\pi_1 \sqcup \pi_2) &= tr^{\mathtt{prg}}(\pi_1) \sqcup tr^{\mathtt{prg}}(\pi_2) \\ tr^{\mathtt{prg}}(\pi^*) &= tr^{\mathtt{prg}}(\pi)^* \\ tr^{\mathtt{prg}}(\chi?) &= tr^{\mathtt{fml}}(\chi)? \end{cases} \end{aligned}$$

Observe that we do not need to keep the set of atoms of φ' in this case, as we do not have to deal with introspective causes and consequences in assignments. We note $tr_{\mathsf{DEL\text{-}PAOC}}(\varphi)$ the formula obtained after applying Procedure 7.2 to φ.

Lemma 7.2. *Let π be a* DEL-PAOC *program and $A \subseteq ATM$ a set of atoms such that $ATM(\pi) \subseteq ATM$. Let w be a* DL-PA-PMP *valuation such that $w \in INTR$. Then for every w' such that $\langle INTR|_A, w \cap A\rangle \mathcal{M}_{tr^{\mathtt{prg}}(\pi)} \langle INTR|_A, w'\rangle$, we have $w' \in INTR|_A$.*

Proof. We prove it by induction on the form of π.

- $\boldsymbol{\pi = +\alpha}$**.** In this case, $tr^{\mathtt{prg}}(\pi) = \alpha{\leftarrow}\top$. Hence by the truth conditions of DL-PA-PMP, $w' = (w \cap A) \cup \{\alpha\} = (w \cup \{\alpha\}) \cap A$ since $\alpha \in A$ (because $ATM(\pi) \subseteq ATM$). Since $w \in INTR$, we have seen in DEL-PAOC that $w \cup \{\alpha\} \in INTR$, hence $(w \cup \{\alpha\}) \cap A \in INTR|_A$.

- $\boldsymbol{\pi = -\alpha}$**.** If α is valid in $INTR$, $tr^{\mathtt{prg}}(-\alpha) = \mathit{fail}$ and the property is trivially valid. Otherwise, the proof is similar to the case $\pi = +\alpha$.

- The proofs for $\pi = \pi_1;\pi_2$, $\pi = \pi_1 \sqcup \pi_2$ and $\pi = \chi?$ are similar to their cases in the proof of Lemma 7.1 as their truth conditions are identical to DEL-PAO's.

In all cases we have $w' \in INTR|_A$. □

This counterpart of Lemma 7.1 indicates that, again, the execution of mental programs cannot exit the public information state in models that interest us.

Proposition 7.3. *Let $A \subseteq ATM$ be a set of atoms and $w \in INTR$ an introspective valuation. Then we have:*

$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr^{\text{prg}}(\pi_1;\pi_2)}\varphi \leftrightarrow K_{tr^{\text{prg}}(\pi_1)}K_{tr^{\text{prg}}(\pi_2)}\varphi$$
$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr^{\text{prg}}(\pi_1\sqcup\pi_2)}\varphi \leftrightarrow K_{tr^{\text{prg}}(\pi_1)}\varphi \wedge K_{tr^{\text{prg}}(\pi_2)}\varphi$$
$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr^{\text{prg}}(\pi^*)}\varphi \leftrightarrow K_{tr^{\text{prg}}(\pi^{2^{2|ATM(\pi)|}})}\varphi$$
$$INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} K_{tr^{\text{prg}}(\chi?)}\varphi \leftrightarrow tr^{\texttt{fml}}(\chi) \to \varphi$$

Proof. The proof is similar to Proposition 7.1 for the translation of DEL-PAO programs. It likewise relies on Lemma 7.2. □

Proposition 7.4. *Let $V \in INTR$ be a valuation and φ a* DEL-PAOC *formula. Then we have:*

$V \models_{\text{DEL-PAOC}} \varphi$ *if and only if*
$$INTR|_{RATM(\varphi)}, V \cap RATM(\varphi) \models_{\text{DL-PA-PMP}} tr_{\text{DEL-PAOC}}(\varphi)$$

Remember that relevant atoms of DEL-PAOC also include control atoms that are "hidden" in strategic operators.

Proof. The proof follows the lines of Proposition 7.2 from the previous section: we first remove epistemic and strategic operators from φ and obtain a formula φ', equivalent to φ by Proposition 5.1 on page 118 and such that $RATM(\varphi) = ATM(\varphi')$. Then we apply validities ($Red_;$), ($Red_\sqcup$), (Red_*) and ($Red_?$) of Proposition 2.11 on page 50 to φ', getting φ'', and equivalences of Proposition 7.3 to its translation $tr^{\texttt{fml}}(\varphi')$, obtaining $tr^{\texttt{fml}}(\varphi'')$ like in the previous setting. We again need to prove that

$$V \models_{\text{DEL-PAOC}} \varphi'' \text{ if and only if } INTR|_A, V \cap A \models_{\text{DL-PA-PMP}} tr^{\texttt{fml}}(\varphi''),$$

with $A = ATM(\varphi')$. We again do it by induction on the form of φ''.

- The case when φ'' is boolean is the same as for Proposition 7.2 (because the translation is still homomorphic).
- $\boldsymbol{\varphi'' = [+\alpha]\psi}$**.** This time we have $tr^{\texttt{fml}}(\varphi'') = K_{\alpha\leftarrow\top}tr^{\texttt{fml}}(\psi)$. Observe that $\alpha \in A$. Hence:

$$\begin{aligned}
&INTR|_A, V \cap A \models_{\text{DL-PA-PMP}} K_{\alpha\leftarrow\top}tr^{\texttt{fml}}(\psi)\\
\Leftrightarrow\ &INTR|_A, (V \cap A) \cup \{\alpha\} \models_{\text{DL-PA-PMP}} tr^{\texttt{fml}}(\psi)\\
\Leftrightarrow\ &INTR|_A, (V \cup \{\alpha\}) \cap A \models_{\text{DL-PA-PMP}} tr^{\texttt{fml}}(\psi)\\
\Leftrightarrow\ &V \cup \{\alpha\} \models_{\text{DEL-PAOC}} \psi \qquad \text{(by induction hypothesis)}\\
\Leftrightarrow\ &V \models_{\text{DEL-PAOC}} [+\alpha]\psi.
\end{aligned}$$

- $\varphi'' = [-\alpha]\psi$. The proof is obvious if α is introspectively valid (as it translates to $fail$) and similar to the case $\varphi'' = [+\alpha]\psi$ otherwise.

Therefore the translation is correct. □

7.2.4 From DEL-PAO-PP

In DEL-PAO-PP, we consider public programs and epistemic operators. The formers will be translated homomorphically (except, as before, for assignments) to DL-PA-PMP public programs. We have already seen in DEL-PAO how to reduce epistemic operators to programs; we reuse these in DEL-PAO-PP. As epistemic operators do not modify the public information state and as we only keep indistinguishable worlds that are in this information state, we translate them to mentally executed programs, whose semantics fit perfectly.

Take a DEL-PAO-PP formula φ. We define the following procedure of translation of φ into DL-PA-PMP.

Procedure 7.3.

1. translate φ into the DL-PA-PMP formula $tr^{\mathtt{fml}}_{RATM(\varphi)}(\varphi)$ according to the following definition:

$$\begin{aligned}
tr^{\mathtt{fml}}_A(\alpha) &= \alpha \\
tr^{\mathtt{fml}}_A(\neg\varphi) &= \neg tr^{\mathtt{fml}}_A(\varphi) \\
tr^{\mathtt{fml}}_A(\varphi_1 \wedge \varphi_2) &= tr^{\mathtt{fml}}_A(\varphi_1) \wedge tr^{\mathtt{fml}}_A(\varphi_2) \\
tr^{\mathtt{fml}}_A(K_i\varphi) &= K_{tr^{\mathtt{prg}}_A(\mathsf{varyIfNotSeen}(i,RATM(\varphi)))} tr^{\mathtt{fml}}_A(\varphi) \\
tr^{\mathtt{fml}}_A(CK\varphi) &= K_{tr^{\mathtt{prg}}_A(\mathsf{varyIfNotSeen}(Agt,RATM(\varphi)))} tr^{\mathtt{fml}}_A(\varphi) \\
tr^{\mathtt{fml}}_A([\pi!]\varphi) &= [tr^{\mathtt{prg}}_A(\pi)!] tr^{\mathtt{fml}}_A(\varphi)
\end{aligned}$$

where $tr^{\mathtt{prg}}_A(\pi)$ is defined as:

$$\begin{aligned}
tr^{\mathtt{prg}}_A(+\alpha) &= \beta_1{\leftarrow}\top; \ldots; \beta_m{\leftarrow}\top \\
tr^{\mathtt{prg}}_A(-\alpha) &= \begin{cases} fail & \text{if } \alpha \text{ valid in } INTR \\ \beta'_1{\leftarrow}\bot; \ldots; \beta'_p{\leftarrow}\bot & \text{otherwise} \end{cases} \\
tr^{\mathtt{prg}}_A(\pi_1; \pi_2) &= tr^{\mathtt{prg}}_A(\pi_1); tr^{\mathtt{prg}}_A(\pi_2) \\
tr^{\mathtt{prg}}_A(\pi_1 \sqcup \pi_2) &= tr^{\mathtt{prg}}_A(\pi_1) \sqcup tr^{\mathtt{prg}}_A(\pi_2) \\
tr^{\mathtt{prg}}_A(\chi?) &= tr^{\mathtt{fml}}_A(\chi)?
\end{aligned}$$

with

$$\begin{aligned}
\{\beta_1, \ldots, \beta_m\} &= \alpha^{\Rightarrow} \cap A \\
\{\beta'_1, \ldots, \beta'_p\} &= \alpha^{\Leftarrow} \cap A
\end{aligned}$$

Observe that unlike in Procedure 7.1 for DEL-PAO and in Procedure 7.2 for DEL-PAOC, we use $RATM(\varphi)$ instead of $ATM(\varphi')$ (with φ' the formula resulting from eliminating epistemic operators from φ) as epistemic operators have not be eliminated yet. This is however equivalent, as we have seen that $RATM(\varphi) = ATM(\varphi')$ in previous cases. We simulate epistemic operators with our programs $\mathsf{varyIfNotSeen}(.,.)$ on the relevant atoms of the nested formula as it may contain other epistemic operators; then we translate these programs so that assignments behave correctly. This translation is identical for public programs (and is the same as previous translations of DEL-PAO and DEL-PAOC programs). We do not include the star since it does not appear in $\mathsf{varyIfNotSeen}(i, A)$ and $\mathsf{varyIfNotSeen}(Agt, A)$. We note $tr_{\mathsf{DEL\text{-}PAO\text{-}PP}}(\varphi)$ the formula obtained after applying Procedure 7.3 to φ.

We verify that epistemic operators are indeed equivalent to their mentally executed programs counterparts.

Proposition 7.5. *Let U be a public information state such that $U \subseteq INTR$, $w \in U$ a valuation, i an agent and φ a formula without epistemic operators. Then:*

1. *for every $A \subseteq ATM$ such that $RATM(K_i\varphi) \subseteq A$, $U, w \models_{\mathsf{DEL\text{-}PAO\text{-}PP}} K_i\varphi$ if and only if $U|_A, w \cap A \models_{\mathsf{DL\text{-}PA\text{-}PMP}} K_{tr_A^{\mathtt{prg}}(\mathsf{varyIfNotSeen}(i, ATM(\varphi)))} tr_A^{\mathtt{fml}}(\varphi)$;*
2. *for every $A \subseteq ATM$ such that $RATM(CK\varphi) \subseteq A$, $U, w \models_{\mathsf{DEL\text{-}PAO\text{-}PP}} CK\varphi$ if and only if $U|_A, w \cap A \models_{\mathsf{DL\text{-}PA\text{-}PMP}} K_{tr_A^{\mathtt{prg}}(\mathsf{varyIfNotSeen}(Agt, RATM(\varphi)))} tr_A^{\mathtt{fml}}(\varphi)$.*

Proof. We only examine the first case as the case of common knowledge is similar.

Observe that since φ is boolean, $RATM(\varphi) = ATM(\varphi)$. We have seen in the proof of Proposition 2.17 on page 59 that $\mathsf{varyIfNotSeen}(i, ATM(\varphi))$ correctly simulates the epistemic operator on valuations that are introspective enough (which is the case of $w \cap A$ since $RATM(K_i\varphi) \subseteq A$). However, we are now also dealing with public information states.

According to the semantics of mental programs of DL-PA-PMP, we only take into account worlds related by $\mathcal{M}_{tr_A^{\mathtt{prg}}(\mathsf{varyIfNotSeen}(i, ATM(\varphi)))}$ that are in the public information state $U|_A$. This is important when simulating epistemic operators as their truth conditions in DEL-PAO-PP specify that we only keep worlds related by $\sim_i$ that are in U (which is the main purpose of the public information state).

Take an atom $\alpha \in ATM(\varphi)$.

- If $S_i\,\alpha \in w$, then $w(\alpha) = u(\alpha)$ for every u such that $w \sim_i u$, hence for every $u \in U$ such that $w \sim_i u$. This moreover implies that $S_i\,\alpha \in w \cap A$ as $\alpha \in A$ (since $RATM(K_i\varphi) \subseteq A$). As

$\mathsf{varyIfNotSeen}(i, ATM(\varphi))$ behaves like K_i, we have that $w(\alpha) = u'(\alpha)$ for every u' such that $w \cap A \mathcal{M}_{tr_A^{\mathrm{prg}}(\mathsf{varyIfNotSeen}(i,ATM(\varphi)))} u'$, and thus for every $u' \in U|_A$ such that $w \cap A \mathcal{M}_{tr_A^{\mathrm{prg}}(\mathsf{varyIfNotSeen}(i,ATM(\varphi)))} u'$.

- Now suppose $S_i\,\alpha \notin w$. We distinguish three cases depending on the public information state.

  - Suppose α is true in the information state, i.e., $\alpha \in v$ for every $v \in U$. Then of course, $\alpha \in u$ for every $u \in U$ such that $w \sim_i u$. Our hypothesis also implies that $\alpha \in v'$ for every $v' \in U|_A$, as $\alpha \in A$. Therefore $\alpha \in u'$ for every $u' \in U|_A$ such that $w \cap A \mathcal{M}_{tr_A^{\mathrm{prg}}(\mathsf{varyIfNotSeen}(i,ATM(\varphi)))} u'$.
  - Now suppose $\alpha \notin v$ for every $v \in U$. This case is similar to the previous one: this implies that $\alpha \notin v'$ for every $v' \in U|_A$, and thus α is always false in all valuations related by $\sim_i$ or by $\mathcal{M}_{tr_A^{\mathrm{prg}}(\mathsf{varyIfNotSeen}(i,ATM(\varphi)))}$.
  - Finally, suppose $\alpha \in v_1$ for some $v_1 \in U$ and $\alpha \notin v_2$ for some $v_2 \in U$. This implies that $\alpha \in v_1'$ for some $v_1' \in U|_A$ and $\alpha \notin v_2'$ for some $v_2' \in U|_A$. Since they behave in the same way, and because w and $w \cap A$ agree on atoms from A and thus on atoms from $ATM(\varphi)$, we know that if $w \sim_i v_1$ for some $v_1 \in U$ such that $\alpha \in v_1$, then $w \cap A \mathcal{M}_{tr_A^{\mathrm{prg}}(\mathsf{varyIfNotSeen}(i,ATM(\varphi)))} v_1'$ for some $v_1' \in U|_A$ such that $\alpha \in v_1'$, and that if $w \sim_i v_2$ for some $v_2 \in U$ such that $\alpha \notin v_2$, then $w \cap A \mathcal{M}_{tr_A^{\mathrm{prg}}(\mathsf{varyIfNotSeen}(i,ATM(\varphi)))} v_2'$ for some $v_2' \in U|_A$ such that $\alpha \notin v_2'$; and conversely.

In all cases, if one relation leads to a world within the public information state where α is true, then the other can lead to a world within the information state where α is true; if one relation leads to a world within the information state where α is false, then the other can lead to a world within the information state where α is false.

Therefore $K_i\varphi$ and $K_{tr_A^{\mathrm{prg}}(\mathsf{varyIfNotSeen}(i,ATM(\varphi)))}\varphi$ are still equivalent on the given models. □

Like in previous sections for mental programs, here we study the properties of public programs.

Proposition 7.6. *Let $A \subseteq ATM$ be a set of atoms. Let U be a public information state such that $U \subseteq INTR|_A$ and $w \in U$ a valuation. Then*

we have:

$$U, w \models_{\text{DL-PA-PMP}} [tr_A^{\mathtt{prg}}(\pi_1;\pi_2)!]\varphi \leftrightarrow [tr_A^{\mathtt{prg}}(\pi_1)!][tr_A^{\mathtt{prg}}(\pi_2)!]\varphi$$
$$U, w \models_{\text{DL-PA-PMP}} [tr_A^{\mathtt{prg}}(\pi_1 \sqcup \pi_2)!]\varphi \leftrightarrow [tr_A^{\mathtt{prg}}(\pi_1)!]\varphi \wedge [tr_A^{\mathtt{prg}}(\pi_2)!]\varphi$$
$$U, w \models_{\text{DL-PA-PMP}} [tr_A^{\mathtt{prg}}(\chi?)!]\beta \leftrightarrow \neg tr_A^{\mathtt{fml}}(\chi) \vee \beta$$
$$U, w \models_{\text{DL-PA-PMP}} [tr_A^{\mathtt{prg}}(\chi?)!]\neg\varphi \leftrightarrow \neg tr_A^{\mathtt{fml}}(\chi) \vee \neg[tr_A^{\mathtt{prg}}(\chi?)!]\varphi$$
$$U, w \models_{\text{DL-PA-PMP}} [tr_A^{\mathtt{prg}}(\chi?)!](\varphi \wedge \varphi') \leftrightarrow [tr_A^{\mathtt{prg}}(\chi?)!]\varphi \wedge [tr_A^{\mathtt{prg}}(\chi?)!]\varphi'$$
$$U, w \models_{\text{DL-PA-PMP}} [tr_A^{\mathtt{prg}}(\chi?)!]K_{tr_A^{\mathtt{prg}}(\mathsf{varyIfNotSeen}(RATM(\varphi),))}\varphi \leftrightarrow \neg tr_A^{\mathtt{fml}}(\chi) \vee K_{tr_A^{\mathtt{prg}}(\mathsf{varyIfNotSeen}(i,RATM(\varphi))))}[\chi?!]\varphi$$
$$U, w \models_{\text{DL-PA-PMP}} [tr_A^{\mathtt{prg}}(\chi?)!]K_{tr_A^{\mathtt{prg}}(\mathsf{varyIfNotSeen}(Agt,RATM(\varphi)))}\varphi \leftrightarrow \neg tr_A^{\mathtt{fml}}(\chi) \vee K_{tr_A^{\mathtt{prg}}(\mathsf{varyIfNotSeen}(Agt,RATM(\varphi)))}[\chi?!]\varphi$$

where β is an atom.

Proof. This is again due to translation being homomorphic for the given program operators, and because these equivalences are already valid in DEL-PAO-PP (see Section 6.4). The test is a special case that as we have seen, behaves like the public announcement of PAL. The last two equivalences are the counterparts of $[\chi?!]K_i\varphi \leftrightarrow \neg\chi \vee K_i[\chi?!]\varphi$ and $[\chi?!]CK\varphi \leftrightarrow \neg\chi \vee CK[\chi?!]\varphi$ of DEL-PAO-PP. □

Lemma 7.3. *Let π be a* DEL-PAO-PP *program and $A \subseteq ATM$ a set of atoms. Let U be a public information state such that $U \subseteq INTR|_A$ and w a valuation such that $w \in U$. Then for every $\langle U', w'\rangle$ such that $\langle U, w\rangle \mathcal{P}_{tr_A^{\mathtt{prg}}(\pi)} \langle U', w'\rangle$, we have $U' \subseteq INTR|_A$ and $w' \in U'$.*

Proof. First we can remark that since $INTR|_A = \{u \cap A : u \in INTR\}$, we have $U \subseteq INTR|_A$ if and only if for every $u \in U$, there exists $v \in INTR$ such that $v \cap A = u$.

We prove the property by induction on the form of π.

- $\boldsymbol{\pi = +\alpha}$**.** Then $tr_A^{\mathtt{prg}}(\pi) = \beta_1{\leftarrow}\top; \ldots; \beta_m{\leftarrow}\top$ with $\{\beta_1, \ldots, \beta_m\} = \alpha^{\Rightarrow} \cap A$. Hence since $\langle U, w\rangle \mathcal{P}_{tr_A^{\mathtt{prg}}(\pi)} \langle U', w'\rangle$, we have: for every $u' \in U'$, there is $u \in U$ such that $u' = u \cup (\alpha^{\Rightarrow} \cap A)$, by the semantics of $\mathcal{P}_{tr_A^{\mathtt{prg}}(\pi)}$. We have seen that this is equivalent to, for every $u' \in U'$, there is $v \in INTR$ such that $v \cap A = u$ and $u' = u \cup (\alpha^{\Rightarrow} \cap A)$, i.e., for every $u' \in U'$, there is $v \in INTR$ such that $u' = (v \cap A) \cup (\alpha^{\Rightarrow} \cap A) = (v \cup \alpha^{\Rightarrow}) \cap A$. Since if $v \in INTR$, then $v \cup \alpha^{\Rightarrow} \in INTR$, and we obtain: for every $u' \in U'$, there is $v' \in INTR$ such that $u' = v' \cap A$, which is equivalent to $U' \subseteq INTR|_A$. Moreover w' is the trivially in new public information state U' since $w \in U$.

- $\boldsymbol{\pi = -\alpha}$**.** If α is valid in $INTR$, $tr_A^{\mathtt{prg}}(-\alpha) = \mathit{fail}$ and the property is trivially valid. Otherwise, the proof is similar to the case $\pi = +\alpha$.

- The proofs for $\pi = \pi_1;\pi_2$ and $\pi = \pi_1 \sqcup \pi_2$ are straightforward since their translation is homomorphic and the associated relation $\mathcal{P}_{tr_A^{\mathtt{prg}}(\pi)}$ is a composition or union of $\mathcal{P}_{tr_A^{\mathtt{prg}}(\pi_1)}$ and $\mathcal{P}_{tr_A^{\mathtt{prg}}(\pi_2)}$.

- $\boldsymbol{\pi = \chi?}$**.** Like when assigning to false an introspectively valid atom, the proof is obvious if the program fails. Otherwise, we have seen that a public test can only shrink the public information state. Moreover $w' = w$ if and only if it satisfies χ, hence it is in the resulting state.

In all cases we have $U' \subseteq INTR|_A$ and $w' \in U'$. □

Lemma 7.3 implies that as we will start performing model checking in $INTR|_A$, we will always stay in subset of this public information state, where all our properties apply.

Proposition 7.7. *Let $w \in INTR$ be an introspective valuation and φ a* DEL-PAO-PP *formula. Then we have:*

$$INTR, w \models_{\text{DEL-PAO-PP}} \varphi \textit{ if and only if}$$
$$INTR|_{RATM(\varphi)}, w \cap RATM(\varphi) \models_{\text{DL-PA-PMP}} tr_{\text{DEL-PAO-PP}}(\varphi)$$

Proof. We follow a procedure similar to the ones of propositions 7.2 and 7.4. First translate φ into DL-PA-PMP; we obtain $tr_A^{\mathtt{fml}}(\varphi)$.

We have seen in Proposition 7.5 that epistemic operators are equivalent to their translation into mental programs on every public information state $U \subseteq INTR|_A$, which are the only ones that are reachable by Lemma 7.3. Hence we focus on public programs.

We have shown with Proposition 7.6 that the properties for operators ;, $\sqcup$ and $?$ of DL-PA-PMP are identical to the ones of DEL-PAO-PP (see Section 6.4.2). Since their translation is homomorphic, we make the same reasoning than in previous sections: we remove them from φ, getting φ', and $tr_A^{\mathtt{fml}}(\varphi)$, getting $tr_A^{\mathtt{fml}}(\varphi')$ and prove that the resulting formulas are equivalent:

$$INTR, w \models_{\text{DEL-PAO-PP}} \varphi' \text{ if and only if } INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} tr_A^{\mathtt{fml}}(\varphi'),$$

for φ' with boolean operators and public assignments and $A = RATM(\varphi)$. We do it by induction on the form of φ'.

- The case when φ' is boolean is the same as for Proposition 7.2 (because the translation is still homomorphic).

- $\boldsymbol{\varphi' = [+\alpha!]\psi}$**.** Then we have $tr_A^{\mathtt{fml}}(\varphi') = [\beta_1{\leftarrow}\top; \ldots; \beta_m{\leftarrow}\top!]tr_A^{\mathtt{fml}}(\psi)$

with $\{\beta_1, \ldots, \beta_m\} = \alpha^{\Rightarrow} \cap A$. Hence:

$$
\begin{aligned}
& INTR|_A, w \cap A \models_{\text{DL-PA-PMP}} [\beta_1 \leftarrow \top; \ldots; \beta_m \leftarrow \top!] tr_A^{\mathtt{fml}}(\psi) \\
\Leftrightarrow\ & \{u \cup (\alpha^{\Rightarrow} \cap A) : u \in INTR|_A\}, \\
& \qquad (w \cap A) \cup (\alpha^{\Rightarrow} \cap A) \models_{\text{DL-PA-PMP}} tr_A^{\mathtt{fml}}(\psi) \\
\Leftrightarrow\ & \{(u \cap A) \cup (\alpha^{\Rightarrow} \cap A) : u \in INTR\}, \\
& \qquad (w \cap A) \cup (\alpha^{\Rightarrow} \cap A) \models_{\text{DL-PA-PMP}} tr_A^{\mathtt{fml}}(\psi) \\
\Leftrightarrow\ & \{(u \cup \alpha^{\Rightarrow}) \cap A : u \in INTR\}, \\
& \qquad (w \cup \alpha^{\Rightarrow}) \cap A \models_{\text{DL-PA-PMP}} tr_A^{\mathtt{fml}}(\psi) \\
\Leftrightarrow\ & INTR{+}\alpha, w \cup \alpha^{\Rightarrow} \models_{\text{DEL-PAO-PP}} \psi \qquad \text{(by induction hypothesis)} \\
\Leftrightarrow\ & INTR, w \models_{\text{DEL-PAO-PP}} [+\alpha]\psi.
\end{aligned}
$$

- $\boldsymbol{\varphi' = [-\alpha!]\psi}$**.** The proof is obvious if α is introspectively valid (as it translates to *fail*) and similar to the case $\varphi' = [+\alpha!]\psi$ otherwise.

Therefore the translation is correct. □

7.3 The model checking problem

The model checking problem for DL-PA-PMP is defined as follows:

- **Input:** a couple $\langle w \cap ATM(\varphi), \varphi \rangle$ where φ is a DL-PA-PMP formula and w is an introspective valuation;
- **Output:** yes if $INTR, w \models \varphi$, no otherwise.

Remember that DL-PA-PMP does not include epistemic or strategic operators; hence $ATM(\varphi) = RATM(\varphi)$.

Theorem 7.1. *The* DL-PA-PMP *model checking problem is in* PSPACE.

The next section provides an algorithm that establishes the upper bound.

7.4 The PSPACE algorithm

In this section, we give the procedures performing the model checking algorithm on formulas of DL-PA-PMP. These formulas have been obtained from every language presented in the thesis by the translation presented in the previous section.

Let us define the size of a pair $\langle A, \varphi \rangle$, with $A \subseteq ATM$ a set of atoms and φ a formula, as the cardinality of A plus the length of φ (i.e., the number of symbols used to write it down). Note that the size of the translation φ' of φ may be exponential in the size of φ because of the

reduction of epistemic and strategic operators. However, the size of $\langle w \cap ATM(\varphi'), \varphi' \rangle$ is polynomial in the size of $\langle w \cap RATM(\varphi), \varphi \rangle$. Therefore we have a polynomial reduction from the model checking of our logics to the model checking of DL-PA-PMP.

Algorithms 7.1, 7.2, 7.3 and 7.4 are adapted from [Charrier and Schwarzentruber, 2015], and Algorithm 7.5 is from [Charrier et al., 2016b].

The set of valuations $INTR|_{ATM(\varphi)}$ is fixed in the beginning: we write it $INTR_0$ in the following procedures. Remember that $INTR_0$ is a finite set of finite valuations, as $ATM(\varphi)$ is finite.

The procedures are alternating, meaning that existential and universal choices are performed. For instance, the instruction "(∀) choose $w' \in U$" succeeds if every choice of w' leads to the accepting state. We also quantify over choices of sub-procedures: for instance, "(∃) $algo_1$ **or** $algo_2$" means that at least one of the calls $algo_1$ or $algo_2$ must succeed. More precisely, (∀)$algo_1$ **and** $algo_2$ is an abbreviation for:

(∀) choose $i \in \{1, 2\}$
if $i = 1$:
    $algo_1$
else:
    $algo_2$

and similarly for (∃)$algo_1$ **or** $algo_2$. Observe that in each case, either $algo_1$ or $algo_2$ is executed, but not both.

- Algorithm 7.1 describes $mc(w, \varphi)$, the main model checking procedure for DL-PA-PMP.
  - **Parameters:** a finite valuation w and a DL-PA-PMP formula φ.
  - **Behaviour:** calls the procedure mc_{yes} with an empty list and accepts the input if mc_{yes} does not reject it.
- Algorithm 7.2 describes $mc_{yes}(L, w, \varphi)$, the model checking subprocedure.
  - **Parameters:** a list of announcements and assignments L, a finite valuation w and a DL-PA-PMP formula φ.
  - **Behaviour:** verifies the formula φ depending on its form. The surviving set of valuations is represented by L, a list of announcements and assignments. (For this reason the set $INTR_0$ is represented by the empty list $[]$.) Making an assignment or a public test recursively calls mc_{yes} with the new

assignment or test (if it passes) appended to L. Then evaluating $K_{\pi_m}\psi$ means finding a valuation w' related to w by π_m that is in the public information state, described by L, and where ψ is verified.

- Algorithm 7.3 describes $survives_{yes}(L, w)$, the procedure checking whether a valuation survives announcements and tests made.
  - **Parameters:** a list of announcements and assignments L and a finite valuation w.
  - **Behaviour:** checks whether w satisfies every public assignment and test made until now. The procedure unstacks the list L and verifies that tested formulas are satisfied and atoms assigned to true or false are indeed true or false. Whenever the list is not empty, the procedure calls itself with every valuation possible before the effect of the assignment or the test.
- Algorithm 7.4 describes $ispath_{yes}(L, w, w', \pi_m)$, the path searching procedure for mental programs.
  - **Parameters:** a list of announcements and assignments L, two finite valuations w and w' and a DL-PA-PMP mental program π_m.
  - **Behaviour:** checks whether there is a π_m-path from w to w', depending on the form of π_m. The list L is specifically required in the case of the test, where the procedure calls mc_{yes}. When $\pi_m = {\pi_m}'^*$ we have seen with Axiom (Red_*) of Proposition 2.11 on page 50 that it actually suffices to repeat the program at most $2^{2|ATM(\pi_m')|}$ times.
- Algorithm 7.5 describes $iter_{yes}(L, w, w', \pi_m, i)$, the program iterating procedure.
  - **Parameters:** a list of announcements and assignments L, two finite valuations w and w', a DL-PA-PMP mental program π_m and an integer $i \geq 0$.
  - **Behaviour:** checks whether w' is reachable from w by repeating π_m i times. If i is even, we recursively call $iter_{yes}$, dividing i in two; if i is odd, we execute the program once to reduce to a case where i is even.

We implicitly define the dual of mc_{yes}, called mc_{no}, by replacing **and** by **or**, $(\forall)$ by $(\exists)$, mc_{yes} by mc_{no}, etc., in the pseudo-code of mc_{yes}. While mc_{yes} reads "reject if L, w does not satisfy φ," its dual mc_{no} reads "reject

if L, w satisfies φ." The procedure $survives_{no}$, $ispath_{no}$ and $iter_{no}$ are defined similarly from the code of $survives_{yes}$, $ispath_{yes}$ and $iter_{yes}$.

Algorithm 7.1.
procedure $mc(w, \varphi)$:
    $mc_{yes}([], w, \varphi)$
    accept

Algorithm 7.2.
procedure $mc_{yes}(L, w, \varphi)$:
    match φ **with**
        case $\varphi = \alpha$:
            if $\alpha \notin w$:
                reject
        case $\varphi = \neg\psi$:
            $mc_{no}(L, w, \psi)$
        case $\varphi = (\psi_1 \wedge \psi_2)$:
            (∀) choose $i \in \{1, 2\}$
            $mc_{yes}(L, w, \psi_i)$
        case $\varphi = K_{\pi_{\mathtt{m}}}\psi$:
            (∀) choose $w' \in INTR_0$
            (∃) $ispath_{no}(L, w, w', \pi_{\mathtt{m}})$
                or $survives_{no}(L, w')$
                or $mc_{yes}(L, w', \psi)$
        case $\varphi = [\alpha{\leftarrow}\top!]\psi$:
            $mc_{yes}(L :: (\alpha{\leftarrow}\top), w[\alpha{\leftarrow}\top], \psi)$
        case $\varphi = [\alpha{\leftarrow}\bot!]\psi$:
            $mc_{yes}(L :: (\alpha{\leftarrow}\bot), w[\alpha{\leftarrow}\bot], \psi)$
        case $\varphi = [(\pi_{\mathtt{p}_1}; \pi_{\mathtt{p}_2})!]\psi$:
            $mc_{yes}(L, w, [\pi_{\mathtt{p}_1}!][\pi_{\mathtt{p}_2}!]\psi)$
        case $\varphi = [(\pi_{\mathtt{p}_1} \sqcup \pi_{\mathtt{p}_2})!]\psi$:
            $mc_{yes}(L, w, [\pi_{\mathtt{p}_1}!]\psi \wedge [\pi_{\mathtt{p}_2}!]\psi)$
        case $\varphi = [\chi?!]\psi$:
            (∀) $mc_{no}(L, w, \chi)$
                or $mc_{yes}(L :: (\chi!), w, \psi)$

Algorithm 7.3.
procedure $survives_{yes}(L, w)$:
    match L **with**
        case $L = []$:
            do nothing
        case $L = L' :: (\alpha \leftarrow \top)$:
            (∀) $\alpha \in w$
                and
                (∃) $survives_{yes}(L', w[\alpha \leftarrow \top])$
                    or $survives_{yes}(L', w[\alpha \leftarrow \bot])$
        case $L = L' :: (\alpha \leftarrow \bot)$:
            (∀) $\alpha \notin w$
                and
                (∃) $survives_{yes}(L', w[\alpha \leftarrow \top])$
                    or $survives_{yes}(L', w[\alpha \leftarrow \bot])$
        case $L = L' :: (\varphi!)$:
            (∀) $mc_{yes}(L', w, \varphi)$
                and $survives_{yes}(L', w)$

Algorithm 7.4.
procedure $ispath_{yes}(L, w, w', \pi_{\mathtt{m}})$:
    match $\pi_{\mathtt{m}}$ **with**
        case $\pi_{\mathtt{m}} = \alpha \leftarrow \top$:
            if $w' \neq w \cup \{\alpha\}$:
                reject
        case $\pi_{\mathtt{m}} = \alpha \leftarrow \bot$:
            if $w' \neq w \setminus \{\alpha\}$:
                reject
        case $\pi_{\mathtt{m}} = (\pi_{\mathtt{m}1}; \pi_{\mathtt{m}2})$:
            (∃) choose a valuation $v \in INTR_0$
            (∀) $ispath_{yes}(L, w, v, \pi_{\mathtt{m}1})$
                and $ispath_{yes}(L, v, w', \pi_{\mathtt{m}2})$
        case $\pi_{\mathtt{m}} = (\pi_{\mathtt{m}1} \sqcup \pi_{\mathtt{m}2})$:
            (∃) choose $k \in \{1, 2\}$
            $ispath_{yes}(L, w, w', \pi_{\mathtt{m}k})$
        case $\pi_{\mathtt{m}} = \chi?$:
            (∀) $w = w'$
                and $mc_{yes}(L, w, \chi)$
        case $\pi_{\mathtt{m}} = {\pi_{\mathtt{m}}}'^{*}$:
            (∃) choose $i \in \{0, \ldots, 2^{2|ATM({\pi_{\mathtt{m}}}')|}\}$
            $iter_{yes}(L, w, w', {\pi_{\mathtt{m}}}', i)$

Algorithm 7.5.
procedure $iter_{yes}(L, w, w', \pi_{\mathtt{m}}, i)$:
    match i **with**
        case $i = 0$:
            if $w' \neq w$:
                reject
        case i even:
            (∃) choose a valuation $v \in INTR_0$
            (∀) $iter_{yes}(L, w, v, \pi_{\mathtt{m}}, i/2)$
                and $iter_{yes}(L, v, w', \pi_{\mathtt{m}}, i/2)$
        case i odd:
            (∃) choose a valuation $v \in INTR_0$
            (∀) $ispath_{yes}(L, w, v, \pi_{\mathtt{m}})$
                and $iter_{yes}(L, v, w', \pi_{\mathtt{m}}, i-1)$

Proposition 7.8. *The procedure* mc *is implementable by an alternating Turing machine running in polynomial time.*

Proof. The critical case of procedure mc_{yes} is when $\varphi = K_{\pi_{\mathtt{m}}}\psi$. It was already proven that the procedure $ispath_{yes}$ without the case of the Kleene star * could be implemented by an alternating Turing machine in polynomial time [Charrier and Schwarzentruber, 2015]. For the case of the star, remember that when running (∀)$algo_1$ **and** $algo_2$ or (∃)$algo_1$ **or** $algo_2$, either $algo_1$ or $algo_2$ is executed, but not both. Hence in $ispath_{yes}$, the procedure $iter_{yes}$ is called for only one value of i. For the same reason, the procedure $survives_{yes}$ runs in polynomial time. □

The model checking of DL-PA-PMP is then in AP. Since AP = PSPACE [Chandra and Stockmeyer, 1976], we have shown that our model checking is in PSPACE.

7.5 Conclusion

This technical chapter introduces an auxiliary logic into which every other logic presented in the thesis can be translated. We are not interested in concepts it may express or in a potential axiomatization (which might be difficult to find for the same reasons as for DEL-PAO-PP) but in the PSPACE complexity of its model checking as it proves the upper bound of the model checking problems of DEL-PAO, DEL-PAO^S, DEL-PAOC and DEL-PAO-PP. We established this complexity by providing an alternating algorithm that runs in polynomial time.

8 Conclusion and perspectives

This thesis follows recent works in epistemic logic which study how knowledge can be constructed using information on visibilities of agents over variables. Observability-based logics offer a nice alternative to standard semantics, but come with the two main drawbacks that we identified. We proposed solutions to both issues by encoding visibility information with special "visibility atoms" of unbounded depth instead of sets of propositional variables and by including communication in terms of publicly executed programs. We moreover introduced a "joint visibility" operator to build common knowledge and control atoms that allow us to discuss strategic abilities of agents.

It is notable that the addition of higher-order observability and of joint observability, and also of strategic operators and publicly executed programs, comes without supplementary cost for model checking, which stays PSPACE-complete. We have seen that its definition is however not direct as it requires to transform our infinite models into finite, but "introspective enough" ones. It remains to show the complexity of the satisfiability problem.

We have illustrated our framework with toy examples of epistemic logic, such as the muddy children problem, but also with more general applications. We believe that the generalisation of the gossip problem that we introduced may be the simplest and most tunable epistemic planning problem, much like the blocksworld problem in classical planning, and we have seen that our framework is well suited for such a task. In the long term, we aim to generalize our approach to temporal planning, where actions are durative and may overlap, to flexible planning, where actions may happen between intervals of time, and to contingent planning, with uncertainty on the initial state or on the effects of actions.

We close this thesis with a discussion on one central topic of it: knowledge and ignorance. The latter is an important part of the mo-

tivation behind this work. As we have seen in the introduction, logics of visibility were first investigated because modelling ignorance with possible worlds semantics was counter-intuitive and led to models impractical for system verification. Moreover, our first goal when we designed DEL-PAO was to relax the assumption that visibilities are common knowledge, so that higher-order knowledge, but more importantly higher-order ignorance, becomes non-trivial.

How to ensure introspection of knowledge with our visibility atoms appeared clear: we simply imposed that an agent always sees whether she sees something. However, ensuring that introspection is preserved in all related worlds was not so straightforward and led to the definition of the five constraints (C1)-(C5). It is interesting to note that we have made several attempts at generalizing our framework to beliefs without success. A way we investigated was to replace the visibility operator S_i by two operators O_i and C_i, respectively reading "i has an opinion on" and "i is correct on." However, we were not able to find counterparts of the five introspective constraints that ensure KD45 properties in the current and all related valuations. This is only important if we consider building an axiomatization; it remains an interesting line of work even only conceptually. Beliefs would provide a more general framework in which, with the help of our current dynamic operators, one could intuitively formalize a theory of mind.

Another interesting future work is to define the indistinguishability relation for common knowledge as usual as the reflexive and transitive closure of the union of individual indistinguishability relations. This would allow us to consider common knowledge for an arbitrary number of agents instead of only for the whole set of agents. This however should be done with care, as the axiomatization of the common knowledge operator would be different and as this might also influence the complexity result.

Finally, one may wonder whether, after including visibility and public announcements, there still exists a validity that is not a validity of standard epistemic logic. Here is one:

$$K_1(p \vee q) \wedge \neg K_1 p \wedge \neg K_1 q \rightarrow K_1 K_2 (p \vee q).$$

Intuitively: if I know that $p \vee q$ while not knowing that p and not knowing that q, then I must have learned it via a public announcement. Therefore I know that the others also know $p \vee q$. A way to relax this validity is by integrating private announcements. We have shown that we can model at least some forms of private announcements thanks to our visibility atoms; it would be interesting to investigate to which extent one can capture action models of dynamic epistemic logics. One might then imagine to find combinations of gossip problems and muddy children puzzles...

Conclusion et perspectives

Cette thèse fait suite à des travaux récents dans la logique épistémique qui étudient comment la connaissance peut être construite en utilisant des informations sur les visibilités des agents sur les variables. Les logiques basées sur l'observabilité offrent une bonne alternative à la sémantique standard, mais sont accompagnées avec les deux principaux inconvénients que nous avons identifiés. Nous avons proposé des solutions aux deux problèmes en encodant les informations de visibilité avec des "atomes de visibilité" spéciaux de profondeur illimitée au lieu d'ensembles de variables propositionnelles et en incluant la communication sous forme de programmes exécutés publiquement. Nous avons en outre introduit un opérateur de "visibilité jointe" pour construire la connaissance commune et des atomes de contrôle qui nous permettent de considérer les capacités stratégiques des agents.

Il est à noter que l'ajout de l'observabilité d'ordre supérieur et de l'observabilité jointe, ainsi que des opérateurs stratégiques et des programmes exécutés publiquement, n'est pas accompagné d'un coût supplémentaire pour le model checking, qui reste PSPACE-complet. Nous avons vu que sa définition n'est toutefois pas directe car elle nécessite de transformer nos modèles infinis en modèles finis, mais "suffisamment introspectifs". Il reste à démontrer la complexité du problème de satisfiabilité.

Nous avons illustré notre logique sur des exemples classiques de la logique épistémique, tels que le problème des enfants sales, mais aussi sur des applications plus générales. Nous estimons que la généralisation du problème du bavardage que nous avons introduit est peut-être le problème de planification épistémique le plus simple et paramétrable, comme l'est le problème "blocksworld" en planification classique, et nous avons vu que notre logique est adaptée à ce type de tâche. À long terme, nous visons à généraliser notre approche à la planification temporelle, où les actions ont une durée et peuvent se chevaucher, à la planification flexible, où les actions peuvent se produire entre des intervalles de temps, et à la planification contingente, avec incertitude sur l'état initial ou sur les effets des actions.

Nous terminons cette thèse avec une discussion sur un sujet central de celle-ci : la connaissance et l'ignorance. Cette dernière est une partie importante de la motivation derrière ce travail. Comme nous l'avons vu dans l'introduction, les logiques de visibilité ont d'abord été étudiées car la modélisation de ignorance avec la sémantique des mondes possibles est contre-intuitive et conduit à la construction de modèles peu adaptés à la vérification de système. De plus, notre premier objectif lorsque nous avons conçu DEL-PAO était de relâcher le fait que les visibilités soient

connaissance commune, de sorte que les connaissances d'ordre supérieur, ou plus important encore l'ignorance d'ordre supérieur, soient non triviales.

Comment assurer l'introspection des connaissances avec nos atomes de visibilité parut clair : nous avons simplement imposé qu'un agent voit toujours si elle voit quelque chose. Cependant, veiller à ce que cette propriété soit conservée dans tous les mondes liés n'a pas été aussi simple et a conduit à la définition des cinq contraintes (C1)-(C5). Il est intéressant de noter que nous avons fait plusieurs tentatives de généralisation de notre logique aux croyances sans succès. Une façon que nous avons étudiée était de remplacer l'opérateur de visibilité S_i par deux opérateurs O_i et C_i, respectivement lus "i a une opinion sur" et "i est correcte sur". Cependant, nous n'avons pas réussi à trouver les homologues des cinq contraintes introspectives pour assurer les propriétés KD45 dans la valuation courante et dans toutes les valuations reliées. Ceci est seulement important si l'on considère la construction d'une axiomatique ; cela reste intéressant à étudier, même seulement sur le plan conceptuel. Les croyances fournirait un cadre plus général dans lequel, avec l'aide de nos opérateurs dynamiques actuels, il serait possible de formaliser intuitivement une théorie de l'esprit.

Un autre travail intéressant pour le futur est de définir la relation d'indistinguabilité pour la connaissance commune comme d'habitude, par la fermeture réflexive et transitive de l'union des relations indistinguabilité individuelles. Cela nous permettrait d'envisager la connaissance commune pour un nombre arbitraire d'agents au lieu de seulement pour l'ensemble des agents. Cela doit toutefois être examiné avec soin : l'axiomatique de l'opérateur de connaissance commune serait différente et cela pourrait également influer sur le résultat de complexité.

Enfin, on peut se demander si, après y inclus la visibilité et les annonces publiques, il existe encore des validités qui ne sont pas valides en logique épistémique standard. En voici une :

$$K_1(p \vee q) \wedge \neg K_1 p \wedge \neg K_1 q \rightarrow K_1 K_2(p \vee q).$$

Intuitivement : si je sais que $p \vee q$ tout en ne sachant pas que p et en ne sachant pas que q, alors j'ai du l'apprendre par une annonce publique. Par conséquent, je sais que les autres savent aussi $p \vee q$. Une façon de se relâcher cette validité est d'intégrer les annonces privées. Nous avons montré que nous pouvons modéliser au moins certaines formes d'annonces privées grâce à nos atomes de visibilité ; il serait intéressant d'étudier dans quelle mesure nous pouvons capturer des modèles d'action des logiques épistémiques dynamiques. On pourrait alors imaginer des combinaisons des problèmes du bavardage et des enfants sales...

Bibliography

Ågotnes, T., Harrenstein, P., van der Hoek, W., and Wooldridge, M. (2013). Boolean games with epistemic goals. In Grossi, D., Roy, O., and Huang, H., editors, *Proceedings of the 4th International Workshop on Logic, Rationality, and Interaction (LORI 2013)*, pages 1–14. Springer.

Akkoyunlu, E. A., Ekanadham, K., and Hubert, R. V. (1975). Some constraints and tradeoffs in the design of network communications. In *Proceedings of the 5th ACM Symposium on Operating Systems Principles (SOSP 1975)*, pages 67–74. ACM Press.

Alechina, N., Logan, B., and Whitsey, M. (2004). A complete and decidable logic for resource-bounded agents. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 606–613. IEEE Computer Society.

Alur, R., Henzinger, T. A., and Kupferman, O. (2002). Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713.

Attamah, M., van Ditmarsch, H., Grossi, D., and van der Hoek, W. (2014a). A framework for epistemic gossip protocols. In Bulling, N., editor, *Proceedings of the 12th European Conference on Multi-Agent Systems (EUMAS 2014)*, pages 193–209. Springer.

Attamah, M., van Ditmarsch, H., Grossi, D., and van der Hoek, W. (2014b). Knowledge and gossip. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, pages 21–26.

Aucher, G. and Bolander, T. (2013). Undecidability in epistemic planning. In Rossi, F., editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 27–33. AAAI Press.

Baker, B. and Shostak, R. (1972). Gossips and telephones. *Discrete Mathematics*, 2(3):191–193.

Balbiani, P., Fernandez-Duque, D., and Lorini, E. (2016). A logical theory of belief dynamics for resource-bounded agents. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*.

Balbiani, P., Gasquet, O., and Schwarzentruber, F. (2013a). Agents that look at one another. *Logic Journal of the IGPL*, 21(3):438–467.

Balbiani, P., Herzig, A., Schwarzentruber, F., and Troquard, N. (2014). DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE. *CoRR*, abs/1411.7.

Balbiani, P., Herzig, A., and Troquard, N. (2013b). Dynamic logic of propositional assignments: a well-behaved variant of PDL. In Kupferman, O., editor, *Proceedings of the 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS 2013)*, pages 143–152.

Baltag, A., Moss, L. S., and Solecki, S. (1998). The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 1998)*, pages 43–56. Morgan Kaufmann.

Bolander, T. and Andersen, M. B. (2011). Epistemic planning for single and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34.

Bonzon, E., Lagasquie-Schiex, M.-C., Lang, J., and Zanuttini, B. (2006). Boolean games revisited. In Brewka, G., Coradeschi, S., Perini, A., and Traverso, P., editors, *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, pages 265–269.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69:165–204.

Castelfranchi, C. (1994). Guarantees for autonomy in cognitive agent architecture. In Wooldridge, M. J. and Jennings, N. R., editors, *Proceedings of the 1994 ECAI Workshop on Agent Theories, Architectures and Languages*, volume 890, pages 56–70. Springer-Verlag.

Chandra, A. K. and Stockmeyer, L. J. (1976). Alternation. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS 1976)*, pages 98–108.

Charrier, T., Lorini, E., Herzig, A., Maffre, F., and Schwarzentruber, F. (2016a). Building epistemic logic from observations and public announcements. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016)*.

Charrier, T., Pinchinat, S., and Schwarzentruber, F. (2016b). Public announcements with mental programs: relevance and computational complexity. Unpublished manuscript.

Charrier, T. and Schwarzentruber, F. (2015). Arbitrary public announcement logic with mental programs. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, pages 1471–1479. International Foundation for Autonomous Agents and Multiagent Systems.

Cooper, M. C., Herzig, A., Maffre, F., Maris, F., and Régnier, P. (2016a). A simple account of multi-agent epistemic planning. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*, pages 193–201.

Cooper, M. C., Herzig, A., Maffre, F., Maris, F., and Régnier, P. (2016b). Simple epistemic planning: generalised gossiping. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016)*, pages 1563–1564.

Cooper, M. C., Herzig, A., Maffre, F., Maris, F., and Régnier, P. (2016c). Simple epistemic planning: generalised gossiping. *ArXiv e-prints*, abs/1606.0.

Dunne, P. E., van der Hoek, W., Kraus, S., and Wooldridge, M. (2008). Cooperative boolean games. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pages 1015–1022. International Foundation for Autonomous Agents and Multiagent Systems.

Engeler, E. (1967). Algorithmic properties of structures. *Mathematical Systems Theory*, 1(3):183–195.

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. (1995). *Reasoning about Knowledge*. MIT Press.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208.

Fischer, M. J. and Ladner, R. E. (1979). Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211.

Fitting, M. (1983). *Proof methods for modal and intuitionistic logics*. D. Reidel ; Sold and distributed in the U.S.A. and Canada by Kluwer Boston.

Gasquet, O., Goranko, V., and Schwarzentruber, F. (2014). Big brother logic: logical modeling and reasoning about agents equipped with surveillance cameras in the plane. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pages 325–332.

Ghallab, M., Nau, D. S., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Elsevier.

Grant, J., Kraus, S., and Perlis, D. (2000). A logic for characterizing multiple bounded agents. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(4):351–387.

Gray, J. (1978). Notes on data base operating systems. In *Operating Systems, An Advanced Course*, pages 393–481. Springer-Verlag.

Grossi, D., Lorini, E., and Schwarzentruber, F. (2015). The ceteris paribus structure of logics of game forms. *Journal of Artificial Intelligence Research*, 53:91–126.

Hajnal, A., Milner, E. C. B., and Szemerédi, E. (1972). A cure for the telephone disease. *Canadian Mathematical Bulletin*, 15(3):447–450.

Harrenstein, P., van der Hoek, W., Meyer, J.-J., and Witteveen, C. (2001). Boolean games. In *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2001)*, pages 287–298. Morgan Kaufmann Publishers Inc.

Herzig, A. (2014). Logics of knowledge and action: critical analysis and challenges. *Journal of Autonomous Agents and Multi-Agent Systems*, 29(5):1–35.

Herzig, A., Lorini, E., and Maffre, F. (2015). A poor man's epistemic logic based on propositional assignment and higher-order observation. In van der Hoek, W., Holliday, W. H., and Wang, W.-f., editors, *Proceedings of the 5th International Conference on Logic, Rationality and Interaction (LORI 2015)*, pages 156–168. Springer Verlag.

Herzig, A., Lorini, E., Maffre, F., and Schwarzentruber, F. (2016). Epistemic boolean games based on a logic of visibility and control. In Kambhampati, S., editor, *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*. AAAI Press.

Herzig, A., Lorini, E., Maffre, F., and Walther, D. (2014). Alternating-time Temporal Logic with Explicit Programs. In *Proceedings of the 7th Workshop on Logical Aspects of Multi-Agent Systems (LAMAS 2014)*.

Herzig, A., Lorini, E., Troquard, N., and Moisan, F. (2011). A dynamic logic of normative systems. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 228–233.

Herzig, A., Lorini, E., and Walther, D. (2013). Reasoning about actions meets strategic logics. In Grossi, D., Roy, O., and Huang, H., editors, *Proceedings of the 4th International Workshop on Logic, Rationality, and Interaction (LORI 2013)*, pages 162–175. Springer.

Herzig, A. and Maffre, F. (2016). How to share knowledge by gossiping. In *Proceedings of the 3rd International Conference on Agreement Technologies (AT 2015)*. Springer-Verlag.

Hintikka, J. (1962). *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press.

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580.

Hurkens, C. A. J. (2000). Spreading gossip efficiently. *Nieuw Archief voor Wiskunde*, 5/1(2):208–210.

Kominis, F. and Geffner, H. (2015). Beliefs in multiagent planning: from one agent to many. In Brafman, R. I., Domshlak, C., Haslum, P., and Zilberstein, S., editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS 2015)*, pages 147–155. AAAI Press.

Lehmann, D. J. (1984). Knowledge, common Knowledge and related puzzles (extended summary). In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing (PODC 1984)*, pages 62–67.

Lomuscio, A. and Raimondi, F. (2006). Model checking knowledge, strategies, and games in multi-agent systems. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 161–168.

Lomuscio, A., van der Meyden, R., and Ryan, M. (2000). Knowledge in multiagent systems: initial configurations and broadcast. *ACM Transactions on Computational Logic*, 1(2):247–284.

Lorini, E. and Herzig, A. (2014). Direct and indirect common belief. In Konzelmann Ziv, A. and Schmid, H. B., editors, *Institutions, Emotions, and Group Agents: Contributions to Social Ontology*, pages 355–372. Springer.

Löwe, B., Pacuit, E., and Witzel, A. (2011). DEL planning and some tractable cases. In *Proceedings of the 3rd International International Workshop on Logic, Rationality and Interaction (LORI 2011)*, pages 179–192. Springer Berlin Heidelberg.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL – The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control.

Muise, C., Belle, V., Felli, P., McIlraith, S. A., Miller, T., Pearce, A. R., and Sonenberg, L. (2015). Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 3327–3334. AAAI Press.

Pauly, M. (2001). *Logic for Social Software*. PhD thesis, University of Amsterdam.

Pauly, M. (2002). A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166.

Petrick, R. P. A. and Bacchus, F. (2004). Extending the Knowledge-Based Approach to Planning with Incomplete Information and Sensing. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 2–11.

Plaza, J. (1989). Logics of public communications. In Emrich, M. L., Pfeifer, M. S., Hadzikadic, M., and Ras, Z., editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems (ISMIS 1989)*, pages 201–216. Oak Ridge National Laboratory, ORNL/DSRD- 24.

Pratt, V. R. (1976). Semantical considerations on Floyd-Hoare logic. In *Proceedings of the 17th Annual Symposium on Foundations of Computer Science (FOCS 1976)*, pages 109–121.

Röger, G., Pommerening, F., and Seipp, J. (2014). Fast downward stone soup 2014. In *The 2014 International Planning Competition*.

Salwicki, A. (1970). Formalized algorithmic languages. *Bulletin de l'Academie Polonaise des Sciences, Serie des sciences mathematiques, astronomiques et physiques*, 18:227–232.

Su, K., Sattar, A., and Luo, X. (2007). Model checking temporal logics of knowledge via OBDDs. *The Computer Journal*, 50(4):403–420.

Tijdeman, R. (1971). On a telephone problem. *Nieuw Archief voor Wiskunde*, 19(3):188–192.

van Benthem, J., van Eijck, J., Gattinger, M., and Su, K. (2015). Symbolic model checking for Dynamic Epistemic Logic. In van der Hoek, W., Holliday, W. H., and Wang, W.-f., editors, *Proceedings of the 5th International Conference on Logic, Rationality and Interaction (LORI 2015)*, pages 366–378. Springer-Verlag.

van Benthem, J., van Eijck, J., and Kooi, B. (2006). Logics of communication and change. *Information and Computation*, 204(11):1620–1662.

van der Hoek, W., Iliev, P., and Wooldridge, M. (2012). A logic of revelation and concealment. In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pages 1115–1122. IFAAMAS.

van der Hoek, W., Troquard, N., and Wooldridge, M. (2011). Knowledge and control. In Sonenberg, L., Stone, P., Tumer, K., and Yolum, P., editors, *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 719–726. IFAAMAS.

van der Hoek, W. and Wooldridge, M. (2005). On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119.

van Ditmarsch, H., van der Hoek, W., and Kooi, B. (2005). Dynamic epistemic logic with assignment. In *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pages 141–148.

van Ditmarsch, H., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic*. Springer Publishing Company, Incorporated, 1st edition.

van Ditmarsch, H., van Eijck, J., Pardo, P., Ramezanian, R., and Schwarzentruber, F. (2015). Dynamic gossip. *ArXiv e-prints*, abs/1511.0.

van Linder, B., van der Hoek, W., and Meyer, J.-J. (1997). Seeing is believing: and so are hearing and jumping. *Journal of Logic, Language and Information*, 6(1):33–61.

Wang, Y. and Cao, Q. (2013). On axiomatizations of public announcement logic. *Synthese*, 190(Supplement-1):2013.

Yanov, J. (1959). On equivalence of operator schemes. *Problems of Cybernetic*, 1:1–100.

Yu, Q., Li, Y., and Wang, Y. (2015). A dynamic epistemic framework for conformant planning. In *Proceedings of the 15th conference on Theoretical Aspects of Rationality and Knowledge (TARK 2015)*, pages 249–259.

Dans les logiques épistémiques, la connaissance est généralement modélisée par un graphe de mondes possibles, qui correspondent aux alternatives à l'état actuel du monde. Ainsi, les arêtes entre les mondes représentent l'*indistinguabilité*. Connaître une proposition signifie que cette proposition est vraie dans toutes les alternatives possibles. Les informaticiens théoriques ont cependant remarqué que cela a conduit à plusieurs problèmes, à la fois intuitifs et techniques : plus un agent est ignorant, plus elle a d'alternatives à examiner ; les modèles peuvent alors devenir trop grands pour la vérification de système. Ils ont récemment étudié comment la connaissance pourrait être réduite à la notion de *visibilité*. Intuitivement, l'idée de base est que quand un agent voit quelque chose, alors elle sait sa valeur de vérité. A l'inverse, toute combinaison de valeurs de vérité des variables non observables est possible pour l'agent. Ces informations d'observabilité permettent de reconstituer la sémantique standard de la connaissance : deux mondes sont indistinguables pour un agent si et seulement si chaque variable observée par cet agent a la même valeur dans les deux mondes. Notre objectif est de démontrer que les logiques épistémiques fondées sur la visibilité constituent un outil approprié pour plusieurs applications importantes dans le domaine de l'intelligence artificielle.

Dans le cadre actuel de ces logiques de visibilité, chaque agent a un ensemble de variables propositionnelles qu'elle peut observer ; ces visibilités sont constantes à travers le modèle. Cela accompagne une hypothèse forte : les visibilités sont connues de tous, et sont même *connaissance commune*. De plus, la construction de la connaissance à partir de la visibilité entraîne des validités contre-intuitives, la plus importante étant que l'opérateur de la connaissance distribue sur les disjonctions de littéraux : si un agent sait que p ou q est vrai, alors elle sait que p est vrai ou que q est vrai, parce qu'elle peut les voir.

Dans cette thèse, nous proposons des solutions à ces deux problèmes et les illustrons sur diverses applications telles que la planification épistémique ou les jeux booléens épistémiques, et sur des exemples plus spécifiques tels que le problème des enfants sales ou le problème du bavardage. Nous étudions en outre des propriétés formelles des logiques que nous concevons, fournissant axiomatisations et résultats de complexité.