Reasoning about concurrent actions and its applications to epistemic and temporal planning

Julien Vianey

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Julien VIANEY

Le 15 décembre 2020

Raisonnement avec des action concurrentes et ses applications à la planification épistémique et temporelle

---

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Andreas HERZIG Frédéric MARIS

Jury

M. Bruno ZANUTTINI, Rapporteur
M. Tiago DE LIMA, Rapporteur
Mme Anne-Gwenn BOSSER, Examinatrice
M. Andreas HERZIG, Directeur de thèse
M. Frederic MARIS, Co-directeur de thèse
M. Martin COOPER, Président

Reasoning about concurrent actions and its applications to epistemic and temporal planning

Raisonnement avec des actions concurrentes et ses applications à la planification épistémique et temporelle

Julien Vianey

December 15, 2020

Résumé

La planification est un problème d'intelligence artificielle pouvant être appliquée à de nombreux domaines. Dans cette thèse nous nous intéressons à étendre les possibilités de la planification pour représenter des problèmes réalistes. Nous opposons les problèmes jouets comme les enfants sales (« muddy children ») aux problèmes que l'on peut rencontrer dans le monde réel. Les problèmes du monde réel vont présenter différentes caractéristiques qu'il faudra pouvoir prendre en compte dans leur résolution. Ils sont bien souvent multi-agent et demandent de pouvoir raisonner sur la connaissance des agents, ce que l'on appelle raisonnement épistémique. Les actions des agents peuvent nécessiter une certaine durée pour se réaliser et les agents peuvent les réaliser en parallèle. Enfin, les actions peuvent avoir des conséquences imprévisibles ou des événements indépendants peuvent se produire.

Différents domaines de planification ont été étudiés pour ajouter ces différents aspects à la planification classique. Bien que l'aspect multi-agent ait été étudié en combinaison avec les trois autres, les autres combinaisons ne l'ont pas ou peu été. Le but de cette thèse est d'apporter des éléments pour permettre de résoudre des tâches de planification multi-agent, temporelles et épistémiques. Ces trois aspects (multi-agent, temporel et épistémique, abrégé en MaTEp) nous semblent les plus importants à associer. L'incertain représente un ajout bien plus conséquent puisqu'il peut être présent à de multiples niveaux dans les problèmes et il peut être géré de très nombreuses manières.

Nous commençons par présenter une famille de problèmes de planification MaTEp, les problèmes de bavardage temporels et épistémiques. Le problème du bavardage épistémique est un problème où plusieurs agents ont chacun une information connue d'eux seuls. Ils peuvent s'appeler pour partager l'intégralité des connaissances qu'ils ont, sur les informations de chacun mais aussi sur la connaissance des agents sur ces informations. Le but est alors d'avoir une connaissance partagée par tous les agents jusqu'à une certaine profondeur. Avec une profondeur de 1 on voudra que tous les agents connaissent tous les secrets. Avec une profondeur de 2 on voudra également que tous sachent que tous connaissent tous les secrets. Nous généralisons ici ce problème en ajoutant des contraintes temporelles sur les communications. Les agents ne peuvent s'appeler ou sont forcés de s'appeler à certains moments. Nous montrons que cette famille de problèmes est NP-complète, et ce même si on ajoute des buts négatifs comme avoir l'agent i qui ignore le secret de l'agent j.

Nous présentons ensuite une logique dynamique que l'on appelle logique dynamique d'affectations propositionnelles parallèles (DL-PPA) avec des programmes parallèles. Nous montrons que les problèmes de satisfiabilité et de model-checking sont tous les deux PSPACE-complets. Cette preuve est faite via une réduction polynomiale vers la logique dynamique d'affectations propositionnelles (DL-PA) sur laquelle DL-PPA est basée. Cette logique est ensuite appliquée à la planification classique. Le problème de recherche de l'existence d'une solution à un problème de planification peut être traduit en une formule DL-PPA. Nous présentons des traductions pour des solutions permettant l'exécution d'actions en parallèle ainsi que pour des solutions n'autorisant qu'une exécution séquentielle. Ces même programmes peuvent être modifiés pour modéliser la recherche de solution à horizon borné, c'est à dire des plans de longueur restreinte.

La dernière partie présente une traduction d'un problème de planification épistémique vers un problème de planification classique. Le problème de planification épistémique est décrit dans une logique statique, EL-O, avec une description des transitions d'états via une fonction partielle. Nous montrons que ce problème peut être traduit polynomialement de la planification épistémique avec EL-O vers de la planification classique. Cette traduction implique que la planification épistémique présentée est dans PSPACE. Dans le même esprit, en nous basant sur notre modèle de planification épistémique nous avons créé des domaines et problèmes PDDL. Ces problèmes ont été testés avec des planificateurs de la compétition internationale de planification (IPC) de 2018. Nous avons comparé les temps d'exécution nécessaires aux planificateurs pour donner une solution. Ces résultats sont similaires pour les différents planificateurs testés et montrent que ces types de problèmes ne sont pas évidents pour des planificateurs au top de l'état de l'art.

Mots clés : logique épistémique, multi-agent, raisonnement temporel, planification

Abstract

Planning is a problem of the domain of artificial intelligence that can be applied to many areas. In this thesis we are interested in extending the possibilities of planning to model realistic problems. We oppose toy problems such as muddy children with problems that can be encountered in the real world. Real world problems have features that need to be taken into account in their resolution. They are often multi-agent and require the ability to reason on the knowledge of the agents, we call this epistemic reasoning. The actions of the agents may require a certain amount of time to be fully executed and the agents may perform actions in parallel. Finally, actions may have unpredictable consequences or independent events may occur.

Different areas of planning have been studied to add these aspects to classical planning. Although the multi-agent aspect has been studied in combination with the other three, the other combinations have been studied little or not at all. The aim of this thesis is to provide elements to allow planning of multi-agent, temporal and epistemic problems. These three aspects (multi-agent, temporal and epistemic, abbreviated to MaTEp) seem to us the most important to associate. Uncertainty is a much more significant addition since it can be present at multiple levels in problems and can be managed in many different ways.

We begin by presenting a family of MaTEp planning problems, the temporal and epistemic gossip problems. The epistemic gossip problem is a problem where there are several agents, each with information known only to them. They can call each other to share the entirety of the knowledge they have, about each other's information but also about the agents' knowledge about this information. The goal is then to have a knowledge shared by all the agents up to a certain depth. With a depth of 1, we want all agents to know all the secrets. With a depth of 2 we also want everyone to know that everyone knows all the secrets. Here we generalise this problem by adding time constraints on communications. Agents can only call each, other or are forced to call each other, at some given instants. We show that this family of problems is NP-complete, even if we add negative goals such as having agent i ignore the secret of agent j.

We then present a dynamic logic called dynamic logic of parallel propositional assignments (DL-PPA) with parallel programs. We show that the problems of satisfiability and model-checking are both PSPACE-complete.

This proof is made through a polynomial reduction to the dynamic logic of propositional assignments (DL-PA) logic on which DL-PPA is based. This logic is then applied to classical planning. The problem of finding the existence of a solution to a planning problem can be translated into a DL-PPA formula. We present translations for solutions allowing parallel execution of actions as well as for solutions allowing only sequential execution. These same programs can be modified to model the search for solutions with a limited horizon, i.e. plans of limited length.

The last part presents a translation from an epistemic planning problem to a classical planning problem. The epistemic planning problem is described in a static logic, EL-O, with a description of state transitions via a partial function. We show that this problem can be polynomially translated from epistemic planning with EL-O to classical planning. This translation implies that the epistemic planning problem presented is in PSPACE. In the same vein, based on our epistemic planning model we created PDDL domains and problems. These problems have been tested with planners from the 2018 International Planification Competition (IPC). We compared the execution times needed by the planners to give a solution. These results are similar for the different planners tested and show that these types of problems are not obvious to planners at the top of the state of the art.

Keywords: epistemic logic, multi-agent, temporal reasoning, planning

Remerciements

Je tiens tout d'abord à remercier mes directeurs de thèse, Andreas Herzig et Frédéric Maris. Votre encadrement m'a permis de bien développer mes connaissances en informatique théorique et mon autonomie. Merci également pour m'avoir amené à voyager et rencontrer d'autres chercheurs avec les groupes de travail MAFTEC et avec l'université de Freiburg, entre autres.

Merci également à Bruno Zanuttini et Tiago de Lima d'avoir accepté d'être mes rapporteurs. À vous et aux deux autres membres du jury, Anne-Gwenn Bosser et Martin Cooper, merci d'avoir été présents pour ma soutenance. Vos remarques et questions ont rendu ma soutenance bien plus agréable que ce que je l'avais pensé. Je vous remercie aussi pour vos suggestions d'amélioration pour cette thèse.

Merci à Karen Godary-Dejean et Alain Redlinger pour vos lettres de recommandation qui, je n'en doute pas, m'ont aidé à pouvoir réaliser cette thèse.

Merci à tous les gens que j'ai rencontrés durant ses trois ans, dans les groupes de travail, en conférence et bien sûr à l'IRIT. Les bons moments et super discussions que l'on a pu avoir contribuent à mon intérêt pour la recherche universitaire. Tous vous citer ici serait fastidieux et je suis certain que j'oublierais des noms, aussi je me limiterai à quelques noms en étant sûr que les autres se reconnaitront. Florence, Elise et Victor, merci de m'avoir aidé pour la rédaction et l'administratif lié à la soutenance.

Merci aussi, bien sûr, à tous ceux en dehors du milieu universitaire, ma famille et mes amis Robin, Guillaume, Antony, Laurie, Martin et Michel pour votre soutien.

Contents

1 Introduction

This first chapter describes the definitions used in the thesis. The first section is about some generic definitions that are not fully spelled out. The following sections describe some instantiations. These sections are about some variants of automated planning problems. The first one is classical planning, the simplest form of planning. The second adds some reasoning over the agents knowledge. The third is about planning with temporally extended actions. The last one combines all of this in epistemic temporal planning.

The last section of this chapter explains the contribution of this thesis.

Content of the chapter

Résumé du chapitre en français

Ce chapitre présente les définitions utilisées dans le reste de la thèse. On suppose donnés un ensemble d'états, un ensemble d'actions et un ensemble de formules construites sur un ensemble de fluents. Une tâche de planification est composée d'un état initial, une fonction modélisant les actions et une formule de but. On distingue deux types de plans, les plans séquentiels et les plans parallèles. Dans un plan séquentiel, une action est appliquée à un état pour créer l'état suivant tandis que dans un plan parallèle il s'agit d'un ensemble d'actions qui sont appliquées. Un plan séquentiel est donc un ensemble ordonné d'actions et un plan parallèle est un ensemble ordonné d'ensembles d'actions.

La forme la plus simple de planification est appelée planification classique. En planification classique les fluents sont des variables propositionnelles. Un état est un ensemble de fluents, ceux ci sont les fluents vrais dans cet état, les autres étant faux. Le but est une formule propositionnelle. Les actions sont composées d'une précondition qui doit être vraie pour que l'action puisse être exécutée et d'un ensemble d'effets. Dans leur forme la plus simple les effets sont séparés en effets positifs (ensemble de fluents ajoutés à l'état) et effets négatifs (ensemble de fluents supprimés de l'état). Cette définition étant limitée, différentes extensions ont été proposées, notamment les effets conditionnels. Un effet conditionnel est composé d'une condition et d'ensembles d'effets positifs et négatifs. Si une action est exécutée, tous les effets dont la condition est vraie verront leurs effets appliqués, les autres effets ne seront pas appliqués. En planification classique, un plan est dit plan solution si son execution depuis l'état initial amène à un état où le but est vrai.

La planification épistémique est la forme de planification incluant un raisonnement sur la connaissance des agents. Selon le langage utilisé, la connaissance peut être ajoutée aux fluents ou aux formules en utilisant des opérateurs de connaissance. La logique dynamique épistémique la plus courante (DEL) utilise la deuxième solution. DEL est intéressante par sa grande expressivité mais cela implique une complexité élevée. La planification DEL a été prouvée indécidable. Pour pouvoir étudier des problèmes de planification épistémiques solvables en un temps raisonnable, différentes approches ont été étudiées. Ces approches ajoutent des restrictions soit sur l'aspect épistémique soit sur l'aspect dynamique. Pour le premier il s'agit généralement de contraindre l'utilisation de l'opérateur de connaissance sur les fluents. Les contraintes dynamiques modifient la connaissance des agents sur les actions, celles ci peuvent

par exemple être toutes publiques. Dans la suite de cette thèse nous utilisons EL-O pour la partie épistémique. Il s'agit d'une logique épistémique allégée basée sur des opérateurs de visibilité. Le fonctionnement de ces opérateurs – un agent voit si plutôt que de savoir que – permet de contourner certaines contraintes ajoutées par le choix d'avoir un opérateur appliqué seulement aux fluents tout en conservant les gains en complexité liés à ce choix.

L'objectif de cette thèse était de créer des outils d'aide à la planification pour les problèmes de la vie réelle. J'oppose ici les problèmes de la vie réelle aux problèmes jouets qui ont un intérêt philosophique mais qui ne se trouveront jamais dans la vie réelle. Je me suis concentré sur la réunion de trois aspects différents que l'on trouve couramment dans les problèmes de la vie réelle : multi-agents, les actions étendues dans le temps et le raisonnement sur la connaissance des agents. Les problèmes de planification temporelle et épistémique multi-agents peuvent être utiles de nombreuses façons. Nous pouvons les utiliser pour aider les gens à décider de la manière d'accomplir une tâche, par exemple pour les lignes de production d'une usine. Ils peuvent être utilisés dans les jeux vidéo pour l'IA de personnages non jouables ou pour générer la dynamique d'une histoire. Ou encore pour des robots plus intelligents, afin de déterminer les actions qu'ils doivent effectuer et quand ils doivent les faire. Comme nous voulons aider à planifier des problèmes de la vie réelle, nous voulons disposer d'outils qui peuvent trouver une solution dans un délai raisonnable. Comme la planification temporelle temporelle et épistémique est au moins aussi difficile que la planification classique, nous avons choisi leur complexité PSPACE-complète comme limite inférieure. Dans cette classe de complexité les problèmes avec un faible nombre d'actions et d'étapes peuvent être résolus mais les problèmes plus importants peuvent nécessiter une quantité énorme de temps qui pourrait ne pas être pratique. Nous voulions donc que les problèmes de planification soient dans PSPACE. Comme cela représentait déjà beaucoup de travail, nous avons choisi de ne considérer que les problèmes déterministes. Cela implique que l'aspect multi-agents est présent dans la description et l'exécution du problème mais pas dans la recherche de la solution. Nous voyons la résolution comme étant effectuée de manière centralisée. Cela implique que cette localisation peut communiquer la partie pertinente du plan à chaque agent avant son exécution. Cela fonctionne bien pour les problèmes de coopération entre les agents. Pour les problèmes avec compétition entre les agents, le vainqueur de la compétition est choisi lorsque le problème est modélisé.

Dans le chapitre 2, je présente les problèmes de bavardage épistémique temporel (TEGP). Ils sont une généralisation du problème du bavardage épistémique avec des contraintes temporelles sur les appels. Nous avons ajouté une durée pour toutes les actions d'appel et des contraintes sur les moments auxquels un appel entre deux agents est autorisé. Ce chapitre montre que les TEGP sont des problèmes NP-complets même avec des buts négatifs. Un exemple de but négatif est que l'on ne veut pas que l'agent i connaisse le secret de l'agent j. Ces problèmes montrent l'intérêt d'ajouter un aspect temporel aux problèmes épistémiques pour les rendre plus réalistes. Lorsque nous voulons communiquer avec d'autres personnes, il se peut que nous ne soyons pas en mesure de les appeler quand nous le voulons et que le plan le plus court ne soit pas le meilleur lorsque ces contraintes sont prises en compte. Nous avons alors voulu ajouter plus de flexibilité aux contraintes temporelles. Cela a conduit aux deux chapitres suivants. Le chapitre 3 présente une extension de DL-PA pour ajouter du parallélisme dans la logique. Ce chapitre montre que DL-PPA conserve la complexité PSPACE-complète de DL-PA. Ce résultat est ensuite utilisé pour une tâche de planification classique à horizon borné. Deux programmes qui sont équivalents au problème de l'existence d'un plan de k étapes ou moins sont décrits. Une autre approche de la planification épistémique parallèle est décrite au chapitre 4. Cette fois ci, nous utilisons des actions EL-O et nous avons donc des actions épistémiques au lieu des actions classiques du chapitre précédent. Nous montrons que les actions EL-O peuvent être traduites en planification classique parallèle. Avec cette traduction, nous pouvons montrer que la planification parallèle EL-O est PSPACE-complète. Nous utilisons également la traduction vers la planification classique pour traduire certains problèmes épistémiques en PDDL. Cette dernière traduction nous a permis de tester certains de ces problèmes avec des planificateurs IPC et de comparer le temps d'obtention d'un plan.

1.1 Generic definitions

We suppose given a set S of *states* and a set Act of *action* names, and a set of formulas Fml formed on a set $\mathbb{F}$ of fluents. I will use both action or action name with the same meaning. The set of boolean formulas Fml is defined by the following grammar:

$$\varphi ::= f \mid \bot \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2$$

where f ranges over the set of fluents $\mathbb{F}$. We will use the abbreviations $\top = \neg\bot$ and $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$.

The precise nature of these sets will depend on the kind of planning under concern and will be detailed in the sequel. A *planning task* is a triple $(s_0, \tau, Goal)$ with

- $s_0 \in S$ is the *initial state*
- $\tau : Act \times S \to S$ is a partial function modeling the actions in Act[1]
- $Goal \in Fml$ is the *goal*

We say that an action a can be *executed* at a given state s if $\tau(\mathsf{a}, s)$ is defined. In the following we use τ_{a} as an abbreviation of $\tau(\mathsf{a}, \cdot)$. Then the result of the execution is a state $s' = \tau_{\mathsf{a}}(s)$.

The solution of a planning task is an ordered set of actions called a *plan*. We call plans made of single actions *sequential plans* and plans made of sets of actions (*steps*) *parallel plans*. The *execution* of a plan consists of the ordered execution of the actions starting from a given state.

In order to define the execution of parallel plans several notions of interference have been proposed (Knoblock 1994; Dimopoulos, Nebel, and Koehler 1997). We choose the $\forall$-Step plan semantic of (Rintanen, Heljanko, and Niemelä 2006) to build the interference rules. In such plans, actions of a step can be executed in any order with the same outcome. This rather restrictive definition is used in most approaches to parallel classical planning. For this, we extend the function τ to set of actions, with some restrictions. The execution of an empty set of actions $\tau_\emptyset$ is always defined and does not

[1] τ can be generalized to $\tau : Act \times 2^S \to 2^S$ to have incomplete knowledge about the initial state and non deterministic actions.

change the state: $\tau_\emptyset(s) = s$. We require τ_A to be defined at a state s if and only if, for every action $\mathsf{a} \in A$ $\tau_{\{\mathsf{a}\}}$ is defined and in that case we require that $\tau_A(s) = \tau_\mathsf{a}(\tau_{A\setminus\mathsf{a}}(s)) = \tau_{A\setminus\mathsf{a}}(\tau_\mathsf{a}(s))$. This way, if $\tau_A(s)$ is defined for a set $A = \{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$, then $\tau_A(s) = \tau_{\mathsf{a}_1}(\ldots(\tau_{\{\mathsf{a}_n\}}(s))\ldots)$ for any ordering of the set.

Formally, the execution of a sequential plan $\langle \mathsf{a}_0, \ldots, \mathsf{a}_n \rangle$ at a state s_0 is the sequence of states $(s_0, \ldots, s_{n+1})$ where:

$$s_{i+1} = \tau_{\mathsf{a}_i}(s_i)$$

and the execution of a parallel plan $\langle A_1, \ldots, A_n \rangle$ at a state s_0 is defined as the sequence of states $(s_0, \ldots, s_{n+1})$ where:

$$s_{i+1} = \tau_{A_i}(s_i)$$

To illustrate the different kinds of planning described here, I will use different variations of the gossip problem. The simplest one (Akkoyunlu, Ekanandham, and Huber 1975; Hurkens 2000) has six agents. Each agent has a secret, something known by them and only them at the beginning, and the goal is for all the agents to know all the secrets. In order to achieve this goal, the agents can call each other to share all their knowledge.

1.2 Classical planning

The simplest form of automated planning (Ghallab, Nau, and Traverso 2004) is called *classical planning*. Such planning tasks are modeled with propositional variables as fluents.

In the generalized gossip problem (Maffre 2016) there are n agents (noted i, j, etc.) who each have a secret, something known by them ($ksec_{i,i}$) but no one else ($\neg ksec_{j,i}$ for $j \neq i$). The goal of the planning task is for all the agents to know all the secrets. In order to achieve this goal the agents can call each other.

During a one-way call (such as a letter or email) information only passes in one direction, whereas during a two-way call (such as a telephone conversation) information passes in both directions. In the case of parallel communication, several calls between distinct pairs of agents may take place simultaneously, but an agent can only participate in one call (either by or to another agent)

at the same instant. Parallel communications can model usual multi agent communication problems. The sequential version, in which only one call can take place at the same time, is of interest when the aim is to minimise the total number of calls or when the communication channel is shared by the agents.

1.2.1 State description

The states of a classical planning task are sets of fluents. All the fluents in the set are true in this state and the other fluents are false. In the generalized gossip problem, the initial state is $s_0 = \{ksec_{i,i} \ : \ i \in \{1..n\}\}$.

1.2.2 Action description

In classical planning, an action a is described with a precondition $pre(\mathsf{a})$ which must be true for the action to be executed, and a set of effects $\mathit{eff}(\mathsf{a})$, describing the consequences of the action. The effects are fluents that are added or deleted from the current state. These effects may have conditions of their own, in that case they are called conditional effects. We require that the actions descriptions are *consistent*, that is when executed in a given state s, for any fluent f, an action cannot delete and add f.

1.2.2.1 Without conditional effects

The simplest form of action description is that of STRIPS (Fikes and Nilsson 1971) (Stanford Research Institute Problem Solver). In this language, preconditions are limited to a conjunction of fluents that must be true and fluents that must be false in order for the action to be executable. The same limitation applies to the goal formula.

The effects are divided in two sets: $\mathit{eff}^+(\mathsf{a})$ is the set of fluents that are added to the state after the execution of a and $\mathit{eff}^-(\mathsf{a})$ is the set of fluents that are deleted from the state. The partial function τ_{a} is as follows:

$$\tau_{\mathsf{a}}(s) = \begin{cases} (s \setminus \mathit{eff}^-(\mathsf{a})) \cup \mathit{eff}^+(\mathsf{a}) & \text{if } s \models pre(\mathsf{a}) \\ \mathsf{undef} & \text{otherwise} \end{cases}$$

As a's description is consistent it does not matter in which order we apply negative and positive effects.

For the gossip problem it is:

$$\tau_{\mathsf{Call}_{i,j}}(s) = s \cup \{ksec_{i,k} \ : \ k \in \{1..n\}, ksec_{j,k} \in s\} \\ \cup \{ksec_{j,k} \ : \ k \in \{1..n\}, ksec_{i,k} \in s\}$$

In the case of parallel plans, we require that the set of actions A executed on a given step has no *contradictory effects*. This means that for any pair of actions $\mathsf{a}_1, \mathsf{a}_2 \in A$, $\mathit{eff}^+(\mathsf{a}_1) \cap \mathit{eff}^-(\mathsf{a}_2) = \emptyset$. For example, opening and closing a door have contradictory effects. We also want to avoid *cross-interaction*. Here, this means that for two distinct actions $\mathsf{a}_1, \mathsf{a}_2 \in A$, a_1 cannot add a fluent that must be false for $\mathit{pre}(\mathsf{a}_2)$ nor delete a fluent that must be true for $\mathit{pre}(\mathsf{a}_2)$. For example, closing a door will interact with crossing it. We say that a set of actions without contradictory effects and cross-interaction in a state s is *consistent in* s. Those constraints are more restrictive than accepting all ∀-Step plans but are simpler to test within our logics and give only ∀-Step plans. Under these restrictions testing whether a ∀-step plan is valid is polytime while it is CoNP-complete for any generic ∀-step plan (Rintanen, Heljanko, and Niemelä 2006).

1.2.2.2 With conditional effects

STRIPS being limited, some other languages were proposed to extend the description of actions. The most widespread one, and the only one I will write about, is PDDL (McDermott et al. 1998) (Planning Domain Definition Language).

In PDDL, the preconditions of actions and the goal can be any boolean formula and the actions can have *conditional effects*. Here we will only consider non recursive conditional effects, which is usually called conditional STRIPS. In this case, a conditional effect ce has a condition $\mathit{cnd}(ce)$ and positive $\mathit{eff}^+(ce)$ and negative $\mathit{eff}^-(ce)$ effects. As we can express unconditional effects as conditional effects that have $\top$ as a condition, we can reduce the description of an action to a tuple $\mathsf{a} = (\mathit{pre}(\mathsf{a}), \mathit{eff}(\mathsf{a}))$. When the action a is executed in a state s, all the conditional effects $ce \in \mathit{eff}(\mathsf{a})$ of a are executed. When a conditional effect is executed, if its condition is true in the state s, all the fluents in $\mathit{eff}^+(ce)$ are added to the state and all the fluents in $\mathit{eff}^-(ce)$ are deleted.

Formally:

$$\tau_{\mathsf{a}}(s) = \begin{cases} (s \setminus \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ s \models \mathit{cnd}(ce)}} \mathit{eff}^{-}(ce)) \cup \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ s \models \mathit{cnd}(ce)}} \mathit{eff}^{+}(ce) & \text{if } s \models \mathit{pre}(\mathsf{a}) \\ \mathsf{undef} & \text{otherwise} \end{cases}$$

As in STRIPS planning, for parallel plans we require consistent actions sets. A set of actions A is said to be free of contradictory effects in a state s if and only if for any pair of actions $\mathsf{a}_1, \mathsf{a}_2 \in A$ and for every $ce_1 \in \mathit{eff}(\mathsf{a}_1)$ and $ce_2 \in \mathit{eff}(\mathsf{a}_2)$, if $s \models ce_1 \wedge ce_2$ then $\mathit{eff}^{+}(ce_1) \cap \mathit{eff}^{-}(ce_2) = \emptyset$.

For the cross-interaction, we generalise the definition used for STRIPS planning. We say that two states s and s' *agree* on a formula φ if and only if either $s \models \varphi$ and $s' \models \varphi$ or $s \not\models \varphi$ and $s' \not\models \varphi$. Then we say that two different actions a_1 and a_2 *have no cross-interaction at* s if the following hold:

1. s and $\tau_{\mathsf{a}_1}(s)$ agree on $\mathit{pre}(\mathsf{a}_2)$ and on the condition $\mathit{cnd}(ce_2)$ of every conditional effect $ce_2 \in \mathit{eff}(\mathsf{a}_2)$;

2. s and $\tau_{\mathsf{a}_2}(s)$ agree on $\mathit{pre}(\mathsf{a}_1)$ and on the condition $\mathit{cnd}(ce_1)$ of every conditional effect $ce_1 \in \mathit{eff}(\mathsf{a}_1)$.

1.2.3 Goal and preconditions description

In classical planning the goal and the preconditions are both boolean formulas over the set of fluents. However, depending on the language used, there can be some restrictions on these formulas. They are usually the same for both the goal and the preconditions.

In STRIPS the formulas can only be conjunctions of fluents and negations of fluents. In PDDL the formulas can use the full set of boolean formulas described above.

For the generalised gossip problem, the goal is $\mathit{Goal} = \bigwedge_{i,j \in \{1..n\}} \mathit{ksec}_{i,j}$.

1.2.4 Solution description

For a plan to be a solution of a classical planning task, the goal formula must be true in the state reached after the execution of the plan on the initial state. When describing a planning task with the definitions above this cannot be

verified in classical propositional logic. In the use cases of STRIPS or PDDL the verification is done in some meta-language. Therefore, the dynamic aspect of planning is implicit when describing a planning task in these languages. However, we may want to have an explicit definition of both the planning task and the validity of a solution for this task in a formal logic. To do this, one can use two dynamic logics, one for sequential planning only and one for both sequential and parallel plans.

1.2.4.1 Sequential

For sequential plans, we will use DL-PA (Herzig, Lorini, Troquard, et al. 2011) (Dynamic Logic of Propositional Assignments). With DL-PA, we can describe a planning task with any boolean formulas for the goal and preconditions and conditions of conditional effects. We can also describe τ_A but only for single actions (i.e. A is a singleton).

DL-PA is interesting for classical planning (Herzig, Menezes, et al. 2014), in particular because of the complexity of satisfiability checking is in PSPACE. This matches with the complexity of determining plan existence (Bylander 1994) which is PSPACE-complete.

1.2.4.2 Parallel

Many authors have investigated how planning problems and their solutions can be represented in Propositional Dynamic Logic PDL, including work on conformant and contingent planning and planning with epistemic actions (Spalazzi and Traverso 2000; Andersen, Bolander, and Jensen 2012; Li, Yu, and Wang 2017; Bolander, Engesser, et al. 2018; Cong, Pinchinat, and Schwarzentruber 2018). However and to the best of our knowledge, it has not been investigated yet how planning with concurrent actions can be captured in dynamic logic. Probably the reason for that is that there is no consensus on the semantics of parallel actions in the PDL community: there are proposals interpreting parallel composition as interleaving (Mayer and Stockmeyer 1996; Benevides and Schechter 2014), as intersection (Balbiani and Vakarelov 2003), and as relations between states and sets of states (Peleg 1987; Goldblatt 1992). There are also extensions with modalities from resource separation logics (Balbiani and Boudou 2018; Benevides, Freitas, and Viana 2011; P. A. S. Veloso, S. R. M. Veloso, and Benevides 2014).

1.2.5 Tools to find a solution to a planning task

Theorem provers for logics such as DL-PA allow us to decide whether there exists a plan and to verify a given plan. However using these may not be the most efficient way to find a solution plan for a given planning task. To find a plan for a specific planning task we can use planners. There are many different kinds of planners that gives different kinds of solutions. Some will return an optimal plan, a plan with the minimum number of steps to achieve the goal. Some will return a suboptimal plan but will focus on the time and memory needed to find this plan. The plans can also be constrained with a different cost for each action. Such a plan will be optimal if it minimises the total cost.

The easiest way to find a state-of-the-art planner is to look at the International Planning Competition (IPC) results. All of the planners here use PDDL as input language and represent top state-of-the-art performances. We can also use SATPLAN (Mali and Kambhampati 1999; Vidal 2001) which is based on the translation from a planning problem to a SAT problem. Many SAT solvers give a solution when there exists one. This solution can then be translated to a solution plan for the planning problem. As SAT solvers are more widely used than planners, they may have better performances. However, SAT is a NP-complete problem when classical planning is a PSPACE-complete problem. To reduce a planning problem to NP, we search for horizon bounded plans. To find an optimal plan we have to search for a plan for different lengths then take the shorter one. The search can be done in different ways. We can search incrementally and stop when the SAT problem has a solution. We can also search by dichotomy. When the SAT problem has a solution, the problem is either longer than the optimal solution or it is an optimal solution. And when the SAT problem has no solution, it is shorter than the optimal solution.

1.3 Epistemic planning

Epistemic planning is the form of planning that involves reasoning about the agents' knowledge. Depending on the logic used, knowledge will be modeled either via special fluents, using knowledge based variables, or via the formulas by adding some knowledge operators.

For instance the gossip problem was generalised in (Cooper, Herzig, Maffre,

Maris, and Régnier 2016b; Cooper, Herzig, Maffre, Maris, and Régnier 2016c) to the epistemic gossip problem. In the epistemic gossip problem, we are not only interested in the knowledge of the secrets for each agent but also the knowledge they have of the other agents' knowledge. The goal of an epistemic gossip problem planning task can be shared knowledge of order 3: that all the agents know that all of them know that all of them know every secret. It can also be that a specific agent knows that the other agents know all the secrets.

We use two different formats for the propositional variables in the epistemic gossip problem. We can keep the same variables as in classical planning ($ksec_{i,j}$) that means that agent i knows the secret of agent j. Alternatively, we consider than the secrets are booleans and we keep the knowledge out of the propositions. To do this, we use sec_j instead, which means that the secret of agent j is true. In such a case, the secret may be, for instance, about the truth of a gossip that everyone heard about agent j.

1.3.1 DEL

This subsection is based on (Bolander 2017) and (Cooper, Herzig, Maffre, Maris, Perrotin, et al. 2019).

Epistemic planning is the general setting of planning in Dynamic Epistemic Logic DEL (Bolander and Andersen 2011; Löwe, Pacuit, and Witzel 2011). DEL combines standard epistemic logic (the static component) with event models describing actions and their perception by the agents (the dynamic component). As DEL-based planning is undecidable (Bolander and Andersen 2011), restrictions of either the static or the dynamic component were explored. Most approaches focused on the latter. It turned out that undecidability is already the case under severe restrictions: basically, DEL planning tasks are only decidable when all actions are public (Aucher and Bolander 2013; Bolander, Jensen, and Schwarzentruber 2015; Cong, Pinchinat, and Schwarzentruber 2018), which is not the case in many real world multiagent applications.

Given a set of propositions $p \in \mathbb{P}$, the set of DEL formulas Fml_{DEL} is defined by the following grammar:

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{K}_i\varphi \mid CKi\varphi$$

1.3.1.1 State description

To represent the states in a DEL planning task we need a Kripke structure. A Kripke structure is a triple $\mathcal{M}(W, R, h)$ where:

- W is a set of possible words
- R is a set of symmetric binary relations R_i, the *indistinguishably relation* for agent i, on W
- $h : \mathbb{P} \to 2^W$ is a function that, for each propositional variable, gives the set of worlds where that variable is true

This way, we can have a model of all the worlds that are possible for our agents. If an agent i does not have enough knowledge to distinguish a world w_1 from a world w_2, then there is a relation in R_i between these worlds.

Then, the epistemic state is the tuple $(w_0, \mathcal{M})$ where $w_0 \in W$.

1.3.1.2 Action description

The actions of DEL are described with *action models*. An action model is a tuple $(E, R, pre, \mathit{eff})$ where:

- E is a set of events
- R is the set of indistinguishably relations
- $pre : E \to Fml_{\mathsf{DEL}}$ assigns a precondition to each event
- $\mathit{eff} : E \to Fml_{\mathsf{DEL}}$ assigns an effect to each event, for all $e \in E$, $\mathit{eff}(e)$ is a conjunction of literals over $\mathbb{P}$

For instance, the action model corresponding to the truthful public announcement of p has a single event e, total relations, p as precondition and an empty effect: $(e, Agt, p, skip)$. Another example may be a truthful semiprivate announcement whether p to agent i. Agent i learns whether p and other agents only learn that i learns whether p without learning whether p. This is modeled by an action model with two events, e^+ and e^-. The preconditions are $pre(e^+) = p$ and $pre(e^-) = \neg p$ and the effects are undefined. The set of indistinguishably relations is $R_j = \{e^+, e^-\} \times \{e^+, e^-\}$ for every $j \neq i$ and $R_i = \{(e^+, e^+), (e^-, e^-)\}$. The designated event is e^+ if we want i to learn that p or e^- if we want i to learn that $\neg p$.

1.3.2 Other decidable approaches

We overview existing work on decidable epistemic planning in the DEL paradigm. We only present sequential plans: as far as we know parallel plans have not been investigated yet.

The approach of (Muise et al. 2015) is based on a lightweight epistemic logic where the scope of the epistemic operator $\mathbf{K}_i$ is restricted to literals, or literals that are preceded by a sequence of epistemic operators (Demolombe and Pozos Parra 2000; Steedman and Petrick 2007; Lakemeyer and Lespérance 2012), and negations (Muise et al. 2015); in other words, no conjunctions or disjunctions can occur in the scope of $\mathbf{K}_i$. Such restrictions however exclude formulas such as $\mathbf{K}_i(\mathbf{K}_j p \vee \mathbf{K}_j \neg p)$ expressing that agent i knows that agent j knows whether the propositional variable p is true. This is a major drawback because such formulas are fundamental in communication and more generally in any forms of interaction: a situation where agent i does not know whether or not p is the case ($\neg\mathbf{K}_i p \wedge \neg\mathbf{K}_i \neg p$) but knows that j knows ($\mathbf{K}_i(\mathbf{K}_j p \vee \mathbf{K}_j \neg p)$) may lead agent i to ask j about p. Such 'knowing-whether' information also conveniently describes the initial situation of the gossip problem:

$$\textstyle\bigwedge_i \left((\mathbf{K}_i sec_i \vee \mathbf{K}_i \neg sec_i) \wedge \bigwedge_{j \neq i} (\neg\mathbf{K}_i sec_j \wedge \neg\mathbf{K}_i \neg sec_j)\right)$$

where sec_i is the secret of agent i, as well as the goal $\bigwedge_{i,j} (\mathbf{K}_i sec_j \vee \mathbf{K}_i \neg sec_j)$. In our EL-O-based approach all these formulas can be expressed.

(Kominis and Geffner 2015) keep the language of standard epistemic logic (so their language is not restricted to epistemic literals) and restrict the dynamic component. It requires that the initial state is common knowledge and that all action occurrences are either public or semi-public. This makes it impossible to account for many natural everyday situations such as gossiping.

A series of papers by Liu et col. investigates epistemic planning with common knowledge based on the situation calculus paradigm (Q. Liu and Y. Liu 2018; Huang et al. 2017). They represent KD45 knowledge in a particular normal form that generalises Moss's characteristic formulas. Their actions have very general effects, such as a disjunction becoming common knowledge, which requires the integration of belief update and revision operations.

Le et al. study DEL-based planning with common knowledge under compact representations of the initial epistemic state and of event models (Le et al. 2018). For the former they use what they call S5-theories (although their epistemic logic is K, not S5); for the latter they use the action language $m\mathcal{A}$, which has statements of the kind "agent i observes action a". This differs from

our modelling where agents observe propositional variables, which makes it difficult to compare the two approaches. We can however also model their 'coin in the box' example which they claim no approach can deal with. A limitation of $m\mathcal{A}$ is that only literals can be announced. Agents therefore cannot communicate higher-order knowledge, as required in the generalised gossip problem. We note that they mention the issue of interfering actions but do not develop this further because it is not clear how to compute effects under $m\mathcal{A}$.

We here simplify the static component: we replace standard epistemic logic by a lightweight version, Epistemic Logic of Observation (EL-O), which is based on the notion of observability of a propositional variable by an agent. In EL-O it is supposed that agent i knows that p is true when p is true and i observes p. Symmetrically, i knows that p is false when p is false and i observes p. Thus when i observes p then i knows either that p is true or that p is false. The other way round, when i does not observe p then i does not know whether p: both p and $\neg p$ are possible for i. This extends to higher-order observability: i may observe whether j observes p, and so on. (Cooper, Herzig, Maffre, Maris, and Régnier 2016a) showed that EL-O is suitable for sequential epistemic planning.

1.3.2.1 State description

We recall Epistemic Logic of Observation, abbreviated EL-O. Its language is a fragment of that of the dynamic epistemic logic DEL-PAO (Herzig, Lorini, and Maffre 2015) where models were always infinite: we now simulate them by finite models, via a modification of the interpretation. This makes the semantics immediately suitable for model checking, which had required some more work in the original presentation.

The epistemic aspect is captured by *observability operators*. The set of observability operators is:

$$OBS = \{\mathbf{S}_i \ : \ i \in Agt\} \cup \{\mathbf{JS}\},$$

where $\mathbf{S}_i$ stands for individual visibility of agent i and **JS** stands for joint visibility of all agents. The set of all sequences of visibility operators of length at most k is noted $OBS^{\leq k}$; and the set of all sequences of visibility operators of length k is noted $OBS^{=k}$. Then the set of all finite sequences of

visibility operators is $OBS^* = \bigcup_{k\geq 0} OBS^{=k}$ and the set of all finite non-empty sequences is $OBS^+ = \bigcup_{k\geq 1} OBS^{=k}$. Elements of OBS^* are noted σ, σ', etc. The *depth* of a sequence of operators σ, noted $depth(\sigma)$, is the number of operators composing it.

Visibility atoms, or atoms for short, are finite sequences of visibility operators followed by a propositional variable. The set of all atoms is:

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

For example, $\mathbf{S}_1\, p$ reads "1 sees the value of p"; it means that 1 knows whether p is true or false. $\mathbf{JS}\, \mathbf{S}_2\, q$ reads "all agents jointly see whether agent 2 sees the value of q": there is joint attention in the group of all agents concerning 2's observation of q; agent 2 may or may not see the value of q, and in both cases this is jointly observed. $\mathbf{S}_1\, \mathbf{S}_2\, \mathbf{S}_3\, p$ reads "1 sees whether 2 sees whether 3 sees p". Atoms with an empty sequence of observability operators are nothing but propositional variables.

Principles of introspection play an important role in epistemic logic: when agent i knows that p then i also *knows* that she knows that p; and when agent i does not know that p then i also *knows* that she does not know that p. In our visibility-based epistemic logic, introspection can be expressed as $\mathbf{S}_i\, \mathbf{S}_i\, \alpha$. Likewise, joint introspection is expressed as $\mathbf{JS}\, \mathbf{JS}\, \alpha$. The latter implies $\sigma \mathbf{JS}\, \alpha$ for every non-empty σ because joint visibility implies any nesting of individual visibility. We therefore call an atom *introspective* if it contains two consecutive $\mathbf{S}_i$, or a $\mathbf{JS}$ that is preceded by a non-empty sequence of observability operators. In other words, an atom is introspective if it is of the form $\sigma\, \mathbf{S}_i\, \mathbf{S}_i\, \alpha$ for some $\sigma \in OBS^*$, or of the form $\sigma\, \mathbf{JS}\, \alpha$ for some $\sigma \in OBS^+$. The set of all introspective atoms is

$$\begin{aligned} \textit{I-Atm} = \{\sigma\, \mathbf{S}_i\, \mathbf{S}_i\, \alpha \ : \ \sigma \in OBS^* \text{ and } \alpha \in ATM\} \cup \\ \{\sigma\, \mathbf{JS}\, \alpha \ : \ \sigma \in OBS^+ \text{ and } \alpha \in ATM\}. \end{aligned}$$

The complement of *I-Atm* is the set of relevant atoms: $\textit{R-Atm} = ATM \setminus \textit{I-Atm}$.

1.3.2.2 Action description

The actions descriptions of an EL-O planning task is made in the same way as for classical planning. The fluents are EL-O atoms, without $\mathbf{JS}$ operators,

instead of propositions. One of the contributions of chapter 4 is the ability to use **JS** in EL-O planning tasks.

We will usually use actions with conditional effects. The precondition $pre(\mathsf{a})$ of the action a is an EL-O formula, as are the conditions $cnd(ce)$ of a conditional effect $ce \in \mathit{eff}(\mathsf{a})$. The effects of ce, $\mathit{eff}^+(ce)$ and $\mathit{eff}^-(ce)$ are sets of EL-O atoms.

The second example of section 1.3.1.2 is modeled in EL-O with an action with p as a precondition and an unconditional effect with $\mathbf{S}_i\, p$ and $\mathbf{S}_j\, \mathbf{S}_i\, p$ for every $j \neq i$.

1.3.2.3 Goal and precondition description

The language of EL-O is defined by the same grammar as defined above with visiblity atoms (noted $\alpha, \alpha', \beta, \ldots$) as fluents:

$$\varphi ::= \alpha \mid \bot \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2)$$

The set of EL-O formulas is noted $\mathit{Fml}_{\mathsf{EL\text{-}O}}$. The set of *relevant formulas* is $\mathit{R\text{-}Fml}_{\mathsf{EL\text{-}O}} = \{\varphi \in \mathit{Fml}_{\mathsf{EL\text{-}O}} \ : \ \mathit{ATM}(\varphi) \subseteq \mathit{R\text{-}Atm}\}$. The set $\mathit{ATM}(\varphi)$ is the set of atoms occurring in φ. It is defined inductively by:

$$\begin{aligned} \mathit{ATM}(\alpha) &= \{\alpha\} \\ \mathit{ATM}(\neg\varphi) &= \mathit{ATM}(\varphi) \\ \mathit{ATM}(\varphi \wedge \varphi') &= \mathit{ATM}(\varphi) \cup \mathit{ATM}(\varphi') \end{aligned}$$

For example, $\mathit{ATM}(\mathbf{JS}\, q \wedge \mathbf{S}_2\, p) = \{\mathbf{JS}\, q, \mathbf{S}_2\, p\}$ and $\mathit{ATM}(\mathbf{S}_1\, \mathbf{JS}\, p) = \{\mathbf{S}_1\, \mathbf{JS}\, p\}$. Note that neither p nor $\mathbf{JS}\, p$ are atoms of $\mathbf{S}_1\, \mathbf{JS}\, p$.

As for the formulas, the whole definition of planning tasks and actions are like in classical planning. We only need to use atoms as fluents instead of propositional variables.

1.3.2.4 The epistemic gossip problem

In the case of the gossip problem, the set of propositional variables $\mathbb{P} = \{\mathit{ksec}_{i,j} \mid i, j \in \mathit{Agt}\}$ where $\mathit{ksec}_{i,j}$ reads "i knows j's secret". So $\mathbf{S}_k\, \mathit{ksec}_{i,j}$ means that agent k sees if i knows j's secret. The goal is to achieve some knowledge over those secrets. For any planning task, the goal is a formula

over the atoms of the problem ($Goal$), we say that $\max(\{depth(\alpha) \ : \ \alpha \in ATM(Goal)\})$ is the depth d of the problem.

The actions in the gossip problem are calls between two agents leading to an update of the two agents' knowledge. In one-way calls, after a call from agent i to agent j, agent j sees everything that agent i knew before the call and they jointly see that. As there is no need for atoms of depth over d[2], we can simplify the jointly seeing by saying that they both see that they see that they see these atoms, and so up to the depth d;

Formally, for the epistemic gossip problem of depth 2, the execution of an action $\mathsf{Call}_{i,j}$ is modeled by the function:

$$\tau_{\mathsf{Call}_{i,j}}(s) = s \cup \{\mathbf{S}_j\,\alpha, \mathbf{S}_j\,\mathbf{S}_i\,\alpha, \mathbf{S}_i\,\mathbf{S}_j\,\alpha, \ldots \ : \ \alpha \in ATM \text{ and } \mathbf{S}_i\,\alpha \in s\}$$

1.3.2.5 Solution description

Like in classical planning to describe both the planning task and the validity of a solution for this task in a formal logic, we need a dynamic logic. For this we use DEL-PAO which has the same dynamic aspect as DL-PA and the same atoms as EL-O. This implies the same restrictions as both logics, mainly that it cannot represent parallel plans. However, DEL-PAO satisfiability checking complexity is the same as DL-PA. This way, we have epistemic planning problems with the same complexity as classical planning problems.

1.3.3 Tools to find a solution

Unlike classical planning, there is no standardised way of writing an epistemic planning task for a solver. That's why there are different approaches for epistemic planners. (Le et al. 2018) describes an epistemic forward search planner called EFP to search for solutions for problems such as their 'coin in the box' example. They are also approaches to use STRIPS or classical planning, with external functions to handle the planning task and the epistemic reasoning separately (Hu, Miller, and Lipovetzky 2019), or simply by translating the problem into classical planning (Cooper, Herzig, Maffre, Maris, and Régnier 2016b).

[2]The calls are actions without preconditions, so the goal is the only formula that has to be true in these problems.

As in classical planning, there are SAT-based approaches for epistemic planning. (Lagniez et al. 2018) presents some experimental results comparing state of the art solvers for modal logics with the MoSaiC solver using a RECAR approach.

1.4 Temporal planning

In classical planning actions have no explicit duration. We can usually consider that the actions have no duration and are instantaneous. This assumption may not work for all problems. For instance, in the gossip problem all the calls are instantaneous but we may have to give them a duration. Calls with a duration allows problems with agents that takes longer to exchange the informations. Or we could add restrictions on when the agents can call each other.

Temporal planning is an extension of classical planning with actions with durations. It allows one to model such problems.

1.4.1 State description

If there is a duration for each action, there has to be a link between the states and time. There are two different approaches to reason about the time. Either with a logic of instants (McDermott 1982) or with a logic of intervals (Allen 1984). Then, the states are bounded either with the instants or with the end-points of the intervals.

1.4.2 Action description

An extension of PDDL, PDDL 2.1 (Fox and Long 2003) proposes actions with a duration. With this extension, actions can have effects either at the beginning or at the end of the action and preconditions are split in three parts. The preconditions can be needed at the beginning, the end or over the entire duration of the action. This is shown in Figure 1.1.

This means that during the execution of an action, multiple state transitions may occur.

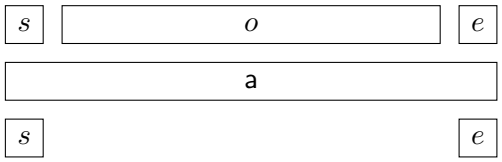

Figure 1.1: PDDL 2.1 action
The top line represents the preconditions, the middle one the action a and the bottom one the effects. The horizontal axis represents the time. The letters are *s* for "at-start", *e* for "at-end" and *o* for "over-all" (over the entire duration).

1.4.3 Solution description

As in classical planning, a temporal planning task may have a sequential solution. However, there are some temporal planning tasks which have only parallel solutions. For instance to hammer a nail, one has to hold the nail on the wall while hammering it. If both actions are not executed concurrently, the nail will fall on the ground. Those planning tasks are called *temporally expressive* (Cushing et al. 2007).

While asking for a sequential or a parallel solution does not change the complexity in classical planning, the distinction between temporally expressive and temporally simple planning tasks involves a difference of complexity. A temporally simple planning task can be linearly reduced to a classical planning task. However the problem of the existence of a solution for a temporally expressive planning is EXPSPACE-complete (Rintanen 2007).

1.4.4 Tools to find a solution

As for classical planners that use the standard language PDDL, temporal planners have been developed for the language PDDL 2.1 involving temporal aspects. Most of these temporal planners are usually working separately for planning and scheduling. While the selection of actions in a solution plan can be done by a planning module, scheduling such actions can be done in parallel via a TMM (Time Map Manager) to handle temporal constraints (Dean and McDermott 1987). These temporal constraints can be modeled for example as a STP (Simple Temporal Problem) (Dechter, Meiri, and Pearl 1991) or as a DTP (Disjunctive Temporal problem) (Stergiou and Koubarakis

2000). We do not detail such techniques here as it would add a lot of details not relevant for this thesis.

In the same way as SAT-based approaches for classical or epistemic planning, SAT modulo theory (SMT) can be used to find a solution for temporal planning. This approach was studied in several works (Maris and Régnier 2008; Rintanen 2015; Shin and Davis 2005).

1.5 Aim and contribution of the thesis

The aim of this thesis was to make tools to help planning for real life problems. I here oppose real life problems to toy problems which have a philosophical interest but may never be found in real life. I focussed on bringing together three different aspects commonly found in real life problems: multi-agent, temporally extended actions and reasoning about the knowledge of the agents.

Multi-agent temporal and epistemic planning problems can be useful in many ways. We can use them for helping people decide how to accomplish a task, for instance to optimise the production lines in a factory. They can also be used in video games for the AI of non playable characters or to dynamically generate a story. Or for smarter robots, to find which actions they have to do and when they have to do them.

As we want to help planning for real life problems, we want to have tools that can find a solution in a reasonable amount of time. As both temporal and epistemic planning is at least as difficult as classical planning we choose their PSPACE-complete complexity as a lower bound. Within this complexity problems with a low number of actions and steps can be solved but bigger problems may need a huge amount of time that might not be practical. Therefore we wanted to have all planning problems in PSPACE.

As this was already a lot of work, we choose to consider only deterministic problems. This implies that the multi-agents aspect is present in the problem description and execution but not in the search of the solution. We think of the solving being done centrally. This implies that this location can communicate the relevant part of the plan to each agent before its execution. This works well for problems with cooperation between the agents. For problems with competition between the agents, the winner of the competition is chosen when the problem is modeled.

1.5.1 Outline of the thesis

In Chapter 2, I present the temporal epistemic gossip problems (TEGP for short). They are a generalisation of the epistemic gossip problem with temporal constraints on the calls. We added a duration for all call actions and constraints on the times allowed for a call between two agents. This chapter shows that TEGP are NP-complete problems even with negative goals. An example of negative goal is that we do not want agent i to know the secret of agent j. These problems show the interest of adding a temporal aspect to epistemic problems to make them more realistic. When we want to communicate with other people, we may not be able to call them whenever we want and the shortest plan may not be the fastest when those constraints are taken into account.

We then wanted to add more flexibility to the temporal constraints. This led to the two following chapters. Chapter 3 shows an extension to DL-PA to add parallelism in the logic. This chapter shows that DL-PPA maintains the PSPACE-complete complexity of DL-PA. This result is then used for bounded horizon classical planning task. Two programs that are equivalent to the problem of the existence of a plan of k steps or less are described.

Another approach to parallel epistemic planning is described in chapter 4. This time we use EL-O actions so we have epistemic actions instead of the classical actions of the previous chapter. We show that EL-O actions can be translated to parallel classical planning. With this translation we can show that EL-O parallel planning is PSPACE-complete. We also use the translation to classical planning to translate some epistemic problems to PDDL. With this last translation we tested some of these problems with some IPC planners and compared the time to get a plan.

2 Temporal Epistemic Gossip Problem

Here, we enrich the epistemic gossip problem by allowing to limit any communication to a set of instants (such as an interval during which the two agents involved in the call are both available). For an example, one can think of satellites on different orbits which can only communicate when they 'see' each other. In a more down-to-earth example, the interval during which a mobile communication is available is often limited by the charge capacity of the battery. Another variant we study is when certain calls must occur at given instants (for example for maintenance or security reasons).

We show that the temporal epistemic gossip problem is in NP even for a complex goal given in the form of a CNF and in the presence of constraints on the instants when calls can or must take place. This positive result, when compared to classical planning follows from the reasonable assumption that knowledge is never destroyed. Moreover, we show that in the absence of temporal constraints and negative goals, temporal epistemic gossiping is in P. We then show maximality of this tractable subproblem in the sense that the problem becomes NP-complete in the following cases: in the presence of temporal constraints (even as weak as a simple upper bound on the execution time of a plan) and in the presence of negative goals (such as agent i should not learn the secret of agent j).

The content of this chapter corresponds to an article published at the European Conference on Multi-Agent Systems (EUMAS 2018) (Cooper, Herzig, Maris, and Vianey 2018).

Content of the chapter

Résumé du chapitre en français

Ici, nous enrichissons le problème du bavardage épistémique en autorisant de limiter toute communication à un ensemble d'instants (comme un intervalle pendant lequel les deux agents impliqués dans la communication sont tous deux disponibles). Par exemple, on peut penser à des satellites sur des orbites différentes qui ne peuvent communiquer que lorsqu'ils se "voient" mutuellement. Dans un exemple plus terre à terre l'intervalle de temps pendant lequel une communication mobile est disponible est souvent limité par la capacité de charge de la batterie. Une autre variante que nous étudions est celle où certains appels doivent avoir lieu à des instants donnés (par exemple, pour des raisons de maintenance ou de sécurité).

Nous montrons que les *problèmes de bavardage épistémique temporel* (TEGP) est en NP même pour un but complexe donné sous la forme d'une CNF et avec des contraintes sur les instants où les communications peuvent ou doivent avoir lieu. Ce résultat positif, par rapport à la planification classique découle de l'hypothèse raisonnable que la connaissance n'est jamais détruite. De plus, nous montrons qu'en l'absence de contraintes temporelles et de buts négatifs, le bavardage épistémique temporel est dans P. Nous montrons ensuite que le problème devient NP-complet dans les cas suivants : en présence de contraintes temporelles (même aussi faibles qu'une simple borne supérieure sur le temps d'exécution d'un plan) et en présence de buts négatifs (comme l'agent i ne doit pas apprendre le secret de l'agent i).

Pour prouver l'appartenance à NP, nous cherchons la longueur d'un plan sans boucle pour un problème ayant une solution, dans le pire des cas. Dans ce cas, pour qu'un agent apprenne la valeur d'un atome il faut que cette information transite par tous les agents. Si l'on a n agents, il faut donc $n-1$ appels. Pour avoir une connaissance de profondeur d d'une proposition, il faut répéter cette opération d fois. Et si la CNF contient m clauses, pour qu'elles soient toutes vraies il faut répéter cela m fois. On a donc un plan solution de longueur polynomiale par rapport à la taille du problème dès lors qu'il existe une solution. Ceci est suffisant pour dire que les TEGP appartiennent à NP.

Le sous problème n'utilisant ni contraintes temporelles ni fluents négatifs est équivalent au problème du bavardage épistémique. Ce problème est prouvé comme étant dans P. Pour les autres sous problèmes, nous en avons choisis quelques uns pour chercher une meilleure borne inférieure à la complexité des TEGP.

En effet, le simple ajout de contraintes temporelles simples (ici on n'ajoute qu'une limite aux appels, après un certain temps tous les appels sont interdits) suffit pour avoir une borne inférieure NP-difficile. On peut facilement construire un graphe de communications traduisant un problème SAT dès lors que le nombre d'appels est limité. Pour le cas où l'on autoriserait les appels en parallèle, il suffit d'utiliser les contraintes temporelles pour forcer des appels séquentiels.

Un autre apport des TEGP par rapport au problème du bavardage épistémique est l'ajout de buts négatif (qu'un agent ne connaisse pas un secret par exemple). Même en l'absence de contraintes temporelles, cet ajout impose une borne inférieure de complexité NP-difficile. Avec ces résultats, nous prouvons que les TEGP sont NP-complets.

2.1 Definitions

The set of all atoms of depth at most d is noted ATM. Observe that if the depth of atom $\alpha \in ATM$ is strictly less than d then $\mathbf{S}_i\,\alpha$ also belongs to ATM.

An instance of the depth-d temporal epistemic gossip problem (TEGP) is given by a tuple $\Pi = (s_0, \mathit{Goal}, \mathit{Agt}, I_p, I_n)$:

$$\begin{array}{rcl} s_0 & \subseteq & \mathit{ATM} \text{ such that } s_0 \text{ contains every } \mathit{ksec}_{i,i}, \text{ for } i \in \mathit{Agt} \\ \mathit{Goal} & \in & \mathit{Fml} \text{ is a conjunction of clauses} \\ I_p & \subseteq & \mathbb{N} \times (\mathbb{N} \cup \{\infty\}) \times \mathit{Agt} \times \mathit{Agt} \\ I_n & \subseteq & \mathbb{N} \times \mathit{Agt} \times \mathit{Agt} \end{array}$$

where s_0 is the initial state; Goal is the goal we want to achieve in the form of a CNF formula (that we identify with a set of clauses); I_p is the set of intervals during which two agents can call each other and I_n is the set of instants when two agents must call each other. All the number used in this chapter are encoded in binary.

The set I_n of necessary calls may correspond to calls that have been programmed in the network for some other purpose. We suppose that I_n is included in I_p, in the sense that for every $(t, i, j) \in I_n$ there is a $(t_1, t_2, i, j) \in I_p$ such that $t_1 \leq t \leq t_2$. In this thesis we always consider the initial state $s_0 = \{\mathit{ksec}_{i,i} \mid i \in \mathit{Agt}\}$ in which all agents know their own secrets.

A set of calls A between agents induces a partial function between states, i.e. from 2^{ATM} to 2^{ATM}. For a state $s \in 2^{ATM}$:

$$\tau_A(s) = \begin{cases} \mathsf{undef} & \text{if } \exists \mathsf{a} \in A : s \not\models \mathit{pre}(\mathsf{a}), \text{ or } \exists \mathsf{a}_1, \mathsf{a}_2 \in A : \\ & \mathsf{a}_1 = \mathsf{Call}_{i_1,j_1} \text{ and } \mathsf{a}_2 = \mathsf{Call}_{i_2,j_2} \\ & \quad \text{with } \{i_1, j_1\} \cap \{i_2, j_2\} \neq \emptyset \\ s \cup \bigcup\limits_{\substack{\mathsf{a} \in A, \\ ce \in \mathit{eff}(\mathsf{a}), \\ \text{and } s \models \mathit{cnd}(ce)}} \mathit{eff}^+(ce) & \text{otherwise} \end{cases}$$

A *plan* is a relation $P \subseteq \mathbb{N} \times \mathit{Agt} \times \mathit{Agt}$. Given a plan P and a natural number t, the set of calls happening at instant t is $P(t) = \{(i, j) : (t, i, j) \in P\}$. We use $|P|$ to denote the number of distinct instants t for which $P(t) \neq \emptyset$. We use $T_P(k)$ to denote the k-th instant (in strictly increasing order of time)

at which a call happens in P: i.e. $T_P(1) < \ldots < T_P(|P|)$ and $\forall t$, $P(t) \neq \emptyset \Leftrightarrow \exists k \in \{1, \ldots, |P|\}$, $T(k) = t$. Our modelling of time by the natural numbers implicitly imposes a fixed duration of one time unit for each call.

Given a TEGP $\Pi = (s_0, \mathit{Goal}, \mathit{Agt}, I_p, I_n)$, a plan P *satisfies the temporal constraints* of Π if and only if all the necessary calls are in P and every call in P is possible; formally: $I_n \subseteq P$ and for every $(t, i, j) \in P$ there is a $(t_1, t_2, i, j) \in I_p$ such that $t_1 \leq t \leq t_2$. Moreover, P *solves* the TEGP if and only if it satisfies the temporal constraints and there is a sequence of states $(s_0, \cdots, s_{|P|})$ such that

- $s_0 = \mathit{Init}$
- $s_{|P|} \models \mathit{Goal}$
- $s_{k+1} = P(T_P(k+1))(s_k)$ for every k with $0 \leq k < |P|$

where $P(T_p(k+1))$ is the set of actions at instant $T_p(k+1)$ and $P(T_p(k+1))(s_k)$ is the result of executing these actions in state s_k. By the definition above of $A(s)$, the set of actions $P(T_p(k+1))$ at instant $T_p(k+1)$ cannot contain two calls involving the same agent. In the sequential version of the TEGP, a solution plan P must also satisfy $\forall t, \mathit{card}(P(t)) \leq 1$.

A TEGP defines in a natural way a call digraph G in which the vertices are the agents and the directed edges the possible calls. In the two-way version, G is a graph.

Example 1. *Consider a network of five servers (which we call a, b, c, d and e) where each server can only communicate with a subset of the others. Note that all calls are assumed to be one-way in this example. As part of the maintenance program, a, b, c and d send a backup of their data to e every night and these backups can be sent to any server during the day (between 8:00 and 18:00). The others servers can communicate with each other at any moment if there is a communication link between them. The communication graph is depicted in Figure 2.1.*

On the same network, another question that we can ask is whether c can know a's data without a being aware of this. In this case, the goal is $ksec_{c,a} \wedge \neg \mathbf{S}_a\, ksec_{c,a}$. *The answer is 'yes' since the following plan establishes the goal: the necessary* $\mathsf{Call}_{a,e}$ *at instant* 2, *followed by* $\mathsf{Call}_{e,c}$ *at an instant* $t \in [8, 18]$ *(together with the other necessary calls* $\mathsf{Call}_{b,e}$ *at instant* 4, $\mathsf{Call}_{c,e}$ *at instant* 20 *and* $\mathsf{Call}_{d,e}$ *at instant* 22*).*

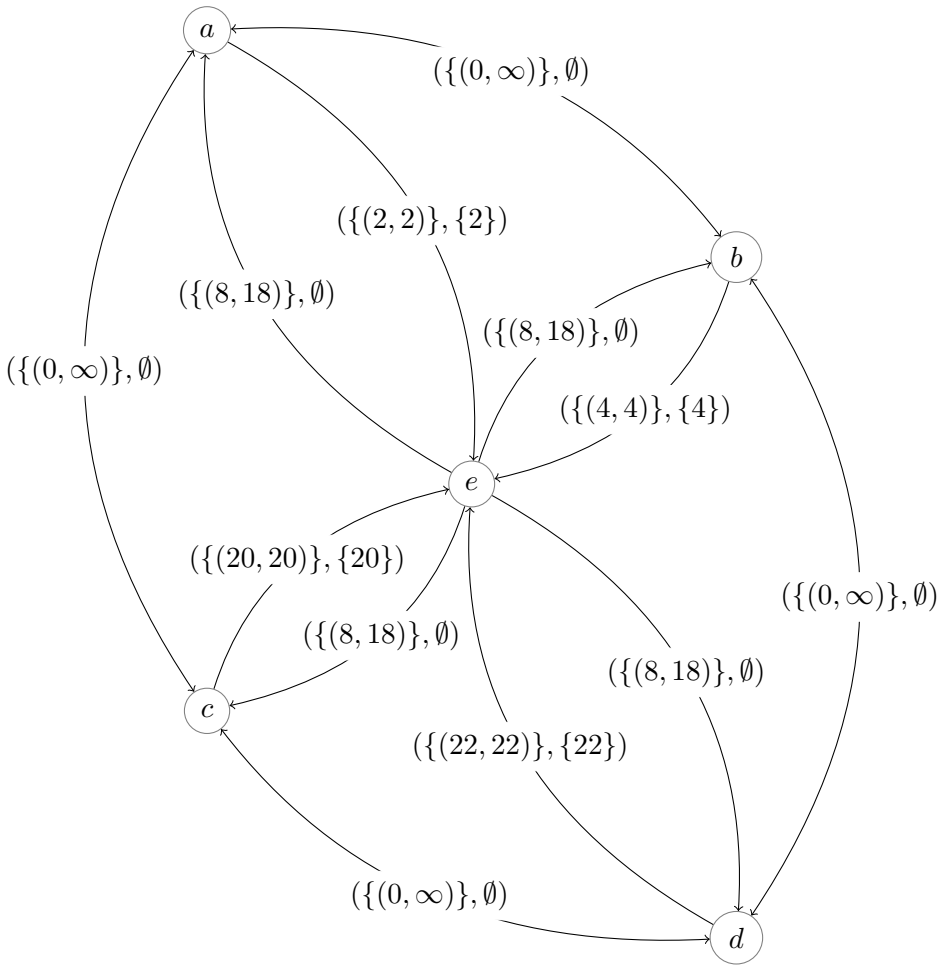

Figure 2.1: Call graph for Example 1 involving necessary calls with e. A double-ended arrow represents two directed edges (i.e., the possibility of one-way calls in both directions). The tuples on the arcs represents the subsets of I_p and I_n for the calls from agent i (at the beginning of the arc) to j (at the end of the arc). If there is no arc from i to j, then no call is possible from i to j.

2.2 Membership in NP

Proposition 1. Let m be the number of clauses in the CNF of the goal. Let d be the the depth of atoms for the problem. If a plan for an instance of a TEGP exists, then there is a plan with $md(n-1) + |I_n|$ calls or less

Proof. Let $\alpha \in ATM$ be an atom. $\mathbf{S}_i\,\alpha$ can only be true if there is a path in the graph G between i and some agent who knows α. Without loss of generality, we can assume that this path is cycle-free. The length of this path is at most $n-1$. Then, for any atom with an epistemic depth d, at most $d(n-1)$ calls are needed for this atom to be true, by the concatenation of d paths of length $n-1$.

The number of calls needed for a disjunction of formulas to be true is at most the maximum of the number of calls needed for each formula. In a CNF, there are only disjunctions over atoms, so the number of calls needed for a disjunction is $d(n-1)$.

The number of calls needed for a conjunction of formulas is the sum of the number of calls needed for every formula, which here is at most $d(n-1)$. So, with m being the number of conjunctions in the CNF of the goal, at most $md(n-1)$ calls are needed for a problem with only possible calls. Thus, if a solution plan P exists, then P contains a subset Q of at most $md(n-1)$ calls which are sufficient to establish all positive atoms in the goal.

For a problem with necessary calls, it can happen that the plan Q does not contain all the necessary calls I_n. However, as P is a solution plan, it must contain every necessary calls. Thus, there exists a subset of P, Q' that is also a solution plan with at most $md(n-1) + |I_n|$ calls. □

2.3 A subproblem of the temporal gossip problem in P

We say that a TEGP instance is *positive* if its goal is a CNF containing only positive atoms. A special case of TEGP is the class of positive non temporally constrained epistemic gossip problems $\Pi = (s_0, \mathit{Goal}, \mathit{Agt}, I_p, I_n)$ where $I_p = \{(0, \infty, i, j) : (i, j) \in E\}$, for some E, and Goal is a positive CNF. In this case, E is the set of edges in the call digraph: if a call is possible (as specified by E), it is possible at any instant. On the other hand, there is no restriction on the set of necessary calls I_n.

Proposition 2. The class of positive non temporally constrained epistemic gossip problems can be solved in polynomial time.

Proof. There is a simple polynomial-time algorithm for positive non temporally constrained epistemic gossip problems: make all possible calls in some fixed order and repeat this operation $md(n-1)+|I_n|$ times. Call this sequential plan Q. By the proof of Proposition 1, if a solution plan exists, there is a sequential solution plan P of length at most $md(n-1)+|I_n|$. The actions of P necessarily appear as a subsequence of Q. Since the goal and preconditions of actions contain only positive atoms, the extra actions of Q cannot destroy any goals or preconditions. It follows that if a solution plan exists, then Q is also a solution plan. Thus this simple algorithm solves the class of positive non temporally constrained epistemic gossip problems in polynomial time. □

Given an arbitrary instance of TEGP, we can construct a positive non temporally constrained instance by ignoring negative goals and temporal constraints (specified by I_p). This is a polynomial-time solvable relaxation of the original TEGP instance. This provides a relaxation which is inspired by the well-known delete-free relaxation of classical planning problems and is orthogonal to the relaxation of temporal planning problems based on establisher-uniqueness and monotonic fluents (Cooper, Maris, and Régnier 2014). Consider that this relaxation of an instance of TEGP has no solution, then the instance of TEGP does not have a solution. This can be used to find TEGP's instances without solution in polynomial-time.

2.4 NP-completeness when execution time is bounded

The simplest temporal constraint is just a time limit on the execution of a plan. In the case of sequential plans this simply corresponds to placing a bound on plan length (which is equal to the number of calls) whereas in the parallel case execution time corresponds to the number of steps. We show in this section that this single constraint (a time limit on plan execution) is sufficient to render the epistemic gossip problem NP-complete. It is worth noting that the PSPACE complexity of classical planning is not affected by the possibility of placing an arbitrary limit on plan length, but the special case of delete-free planning passes from P to NP-hard when a bound is placed on plan length (Bylander 1994). We show that this remains true for the specific case of gossiping problems.

We begin by studying the sequential case of TEGP.

Proposition 3. The epistemic gossip problem with no temporal constraints but with a bound on the number of calls is NP-complete, even when the goal is a conjunction of positive atoms.

Proof. We will exhibit a polynomial-time reduction from the well-known NP-complete problem SAT to the version of the epistemic gossip problem whose question is whether there is a sequential solution plan of length at most L. To do so, for a given set of clauses $\{C_1, \ldots, C_m\}$ we need the following agents:

- an agent S (the source),
- literal agents, i.e., agents for every variable and every negation of a variable (which we name, respectively, x^+ and x^-) for each variable x of the SAT instance,
- clause agents, i.e., agents for every clause (which we name C_i for the ith clause of the SAT instance).

Before performing this construction, we first add a dummy clause $(x \vee \neg x)$ for each SAT variable x. This clearly does not change the semantics of the instance but it does force us to specify the truth value of each variable in a solution of the SAT instance.

The source agent S and clause agents can only communicate with literal agents. The source agent S can communicate with every literal agent. A literal agent can only communicate with S and those clauses it is a member of. The graph G of communications is shown in Figure 2.2 for a particular SAT instance. In this example, $C_1 = (\neg x \vee y \vee z)$, $C_2 = (\neg y \vee z)$ and the clauses C_3, C_4, C_5 are the dummy clauses $(x \vee \neg x)$, $(y \vee \neg y)$, $(z \vee \neg z)$.

A variable x is considered to be true (false) if S's secret passes through x^+ (respectively, x^-) in the solution plan on its way to the agent representing the dummy clause $(x \vee \neg x)$. The bound on the number of actions will prevent the possibility of S's secret passing through both x^+ and x^-. So the choice of whether S's secret passes through x^+ or x^- determines an assignment to the variable x in the SAT problem.

The goal of this instance of TEGP is that every clause agent knows the secret of S ($Goal = \bigwedge_{C_i} ksec_{C_i,S}$). Now set the bound on plan length to be $L = 2n + m$, where n is the number of variables in the SAT instance and m

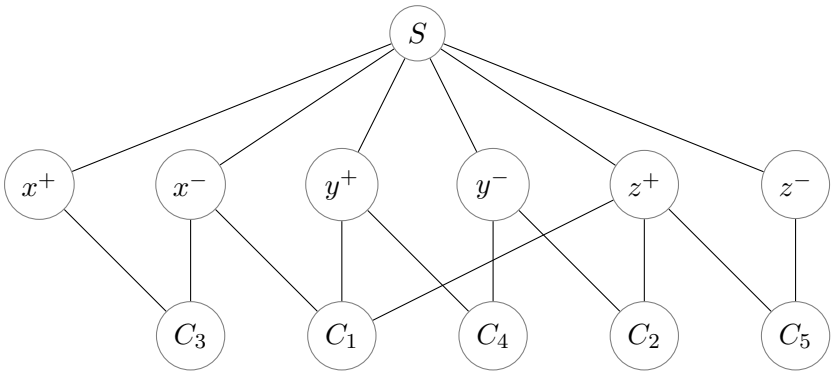

Figure 2.2: Representation of the formula $(\neg x \vee y \vee z) \wedge (\neg y \vee z)$ as a temporal epistemic gossip problem in which the question is whether there is a plan using no more than 8 calls.

the number of clauses in the original instance. With the new dummy clauses, the total number of clauses is $n + m$.

In a solution plan P we require at least n calls, one to either x^+ or x^-, for the n SAT variables x, in order for S's secret to be able to reach the agent corresponding to the dummy clause $x \vee \neg x$. P must also contain at least $n+m$ calls to the clause agents C_i (including the dummy clauses) to establish the goals $ksec_{C_i,S}$. A solution plan of length precisely $2n + m$ corresponds to a solution of the corresponding SAT instance since such a plan defines a unique assignment to all variables that satisfies all clauses. For example, the solution $x = false$, $y = false$, $z = true$ to the SAT instance of Figure 2.2 corresponds to the following solution plan of length 8: S calls x^-; S calls y^-; S calls z^+; x^- calls C_1; z^+ calls C_2; x^- calls C_3; y^- calls C_4; z^+ calls C_5. This reduction from SAT is clearly polynomial.

Proposition 1 proves the existence of a polynomial-length certificate for positive instances of the decision version of TEGP. Such certificates (solutions) can be verified in polynomial time. Thus TEGP $\in$ NP. Since the epistemic gossip problem with no temporal constraints but with a bound on the number of calls is clearly still in NP, this completes the proof of NP-completeness. □

The proof of Proposition 3 was given for the case of two-way communications. It is trivial to adapt it to the case of one-way communications (for example, by only allowing calls from S to literal agents and from literal agents to clause

agents). One can note that any two-way temporal epistemic gossip problem can be polynomially translated to a problem with one-way calls. A two-way call between agents i and j is translated to two one-way calls, one from i to j and one from j to i.

We now consider the parallel version of the TEGP. Recall that in the parallel version of the TEGP, several calls may take place at each step, provided no agent is concerned by more than one call at each step.

Proposition 4. The parallel version of the epistemic gossip problem with no temporal constraints except for a bound on the number of steps is NP-complete even when the goal is a conjunction of positive atoms.

Proof. By the same argument as in the proof of Proposition 1, the problem is in NP. We complete the proof by exhibiting a polynomial reduction from 3SAT which is well known to be NP-complete. Given an instance I_{3SAT} of 3SAT, by introducing sufficiently many new variables x' which are copies of old variables x (together with the clauses $x \vee \neg x'$, $\neg x \vee x'$ to impose equality of x and x') we can transform I_{3SAT} into an equivalent instance in which each literal does not occur in more than three clauses. This is a polynomial reduction since we need to introduce at most one copy of each variable x per clause in which it occurs in I_{3SAT}. Therefore, from now on, we suppose that each literal occurs in at most two clauses in I_{3SAT}.

We construct an instance I of the epistemic gossip problem which has a parallel solution plan of length $2p$ if and only if I_{3SAT} is satisfiable. We choose the value of p to be strictly greater than $n + 3$, where n is the number of variables in I_{3SAT}. To be concrete, we can choose $p = n+4$. We add to I_{3SAT} $p - n$ new dummy variables $x_{n+1}, \ldots, x_p$ none of which occur in the clauses of I_{3SAT}. In I there is an agent S (the source), literal agents x_i^+, x_i^- for each variable x_i $(i = 1, \ldots, p)$, and a clause agent C_j for each of the clauses C_j $(j = 1, \ldots, m)$ of I_{3SAT}. For each variable x_i $(i = 1, \ldots, p)$, we also add a dummy-clause agent D_i which we can consider as representing the dummy clause $x_i \vee \neg x_i$. Instead of linking these basic agents directly, we place paths of new agents between these basic agents. Between agent S and agent x_i^+ we add a path of length $p + 1 - i$. Similarly, we add a new path of the same length between S and agent x_i^-. For $i = 1, \ldots, p$, we add two new paths both of length p between the literal agents x_i^+ and x_i^- and the dummy-clause agent D_i. For each clause C_j of I_{3SAT}, we also add three new paths of length q from the agents corresponding to the literals of C_j to the agent C_j, where

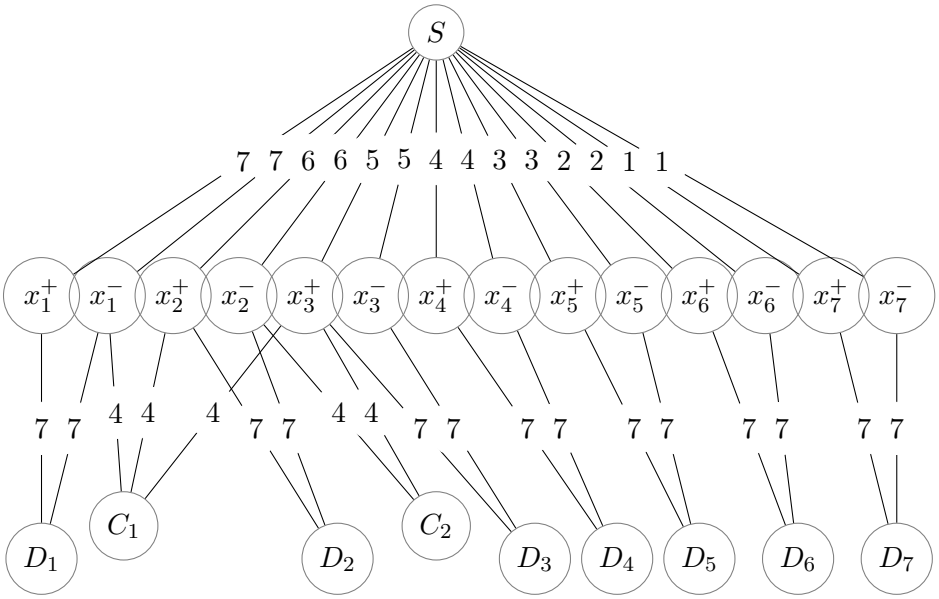

Figure 2.3: Representation of the formula $(\neg x \vee y \vee z) \wedge (\neg y \vee z)$ as a temporal epistemic gossip problem in which the question is whether there is a parallel plan using no more than 14 steps.

$q = p - 3 = n + 1$. The resulting network is shown in Figure 2.3 for an example instance. The numbers on edges in this figure represent the length of the corresponding path. For example, there are 6 intermediate agents (not shown so as not to clutter up the figure) between the agents S and x_1^+. The goal of I is

$$\left(\bigwedge_{i=1}^{p} ksec_{D_i,S}\right) \quad \wedge \quad \left(\bigwedge_{j=1}^{m} ksec_{C_j,S}\right)$$

In order to establish the goal $ksec_{D_i,S}$, the secret of S has to follow a path from S to D_i. The shortest paths from S to D_1 are of length $2p$ and pass through either x_1^+ or x_1^-. Recall that our aim is to find a plan whose execution requires at most $2p$ steps. Thus, to establish $ksec_{D_1,S}$ in $2p$ steps, during the first step, S must call the first agent on the path to x_1^+ or the first agent on the path to x_1^-. The shortest path from S to D_2 is of length $2p-1$, so during the second step, S must call the first agent either on the path to x_2^+ or the first agent on the path to x_2^-. By a simple inductive argument, we can see that during step i ($i = 1, \ldots, p$), S must call the first agent on the path to x_i^+ or x_i^-. We can consider that the choice of whether S's secret passes through x_i^+ or x_i^- determines an assignment to the variables x_i. Due to the diminishing lengths of these paths as i increases, S's secret arrives simultaneously at the literal agents, either x_i^+ or x_i^-, for $i = 1, \ldots, p$. Another p steps are then required to send in parallel this secret to the dummy-clause agents D_j, for a total number of steps of $2p$. Almost simultaneously (within two time units), S' secret arrives at the clause agents C_j, provided it has passed through one of the agents corresponding to the literals of C_j. The length of paths from literal agents (x_i^+ or x_i^-) to clause agents C_j is $q = p - 3$ which is slightly less than p to allow for the fact that a literal agent, say x_i^+, may have to send S's secret along at most four paths: first towards D_i, then towards the (at most) three clauses in which x_i occurs.

It is important to note that S is necessarily occupied during the first p steps, as described above, so if S were to try to send its secret both to x_i^+ and x_i^- for some $i \leq n$ the secret could not arrive via the second of these paths at a clause agent C_j in less than $2p + 1 - n + q = 3p - n - 2$ steps which is greater than the upper bound of $2p$ steps (since $p = n + 4$). By our construction, the goal $ksec_{C_j,S}$ is established only if the assignment to the variables x_i determined by the solution plan satisfies the clause C_j. Hence,

parallel solution plans of length $2p$ steps correspond precisely to solutions of I_{3SAT}. We have therefore demonstrated a polynomial reduction from 3SAT to the parallel version of the epistemic gossip problem with a bound on the number of steps. □

The following corollary follows from the fact that we can place an upper bound L on the number of steps in a plan by simply imposing via I_p an interval of possible instants $[1, L]$ for all calls.

Corollary 5. TEGP is NP-complete.

2.5 NP-completeness of gossiping with negative goals

We show in this section that even without temporal constraints or a bound on plan length, when we allow negative goals the problem of deciding the existence of a solution plan is NP-complete.

Proposition 6. The epistemic gossip problem with possibly negative goals is NP-complete even in the absence of any temporal constraints or bound on plan length.

Proof. The same argument as in the proof of Proposition 1 shows that the problem belongs to NP since it is a subproblem of TEGP.

To complete the proof, it suffices to give a polynomial reduction from SAT. Let I_{SAT} be an instance of SAT. We will construct a call graph G and a set of goals such that the corresponding instance I_{Gossip} of the epistemic gossip problem is equivalent to I_{SAT}. Recall that the nodes of the call graph G are the agents and the edges of G the communication links between agents.

For each propositional variable x in I_{SAT}, we add four nodes x^+, x^-, b_x, d_x to G joined by the edges shown in Figure 2.4(b). There is a source node S in G and edges (S, x^+), (S, x^-) for each variable x in I_{SAT}. For each clause C_j in I_{SAT}, we add a node C_j joined to the nodes corresponding to the literals of C_j. This is illustrated in Figure 2.4(a) for the clause $C_j = \neg x \vee y \vee z$. The solution plan to I_{Gossip} will make S's secret transit through x^+ (on its way from S to some clause node C_j) if and only if $x = true$ in the corresponding solution to I_{SAT}.

For each clause C_j in I_{SAT}, G contains a clause gadget as illustrated in Figure 2.4(a) for the clause $\neg x \vee y \vee z$. We also add $ksec_{C_j,S}$ to the set of

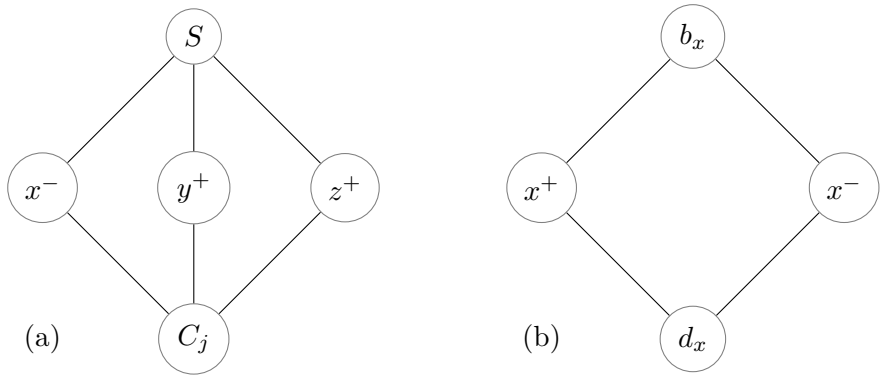

Figure 2.4: (a) gadget imposing the clause $C_j = \neg x \vee y \vee z$; (b) gadget imposing the choice between x and $\neg x$.

goals. Clearly, S's secret must transit through one of the nodes corresponding to the literals of C_j (x^-, y^+ or z^+ in the example of Figure 2.4) to achieve the goal $ksec_{C_j,S}$.

To complete the reduction, it only remains to impose the constraint that a's secret transits through at most one of the nodes x^+, x^-, for each variable x of I_{SAT}. This is achieved by the negation gadget shown in Figure 2.4(b) for each variable x. We add the goals $ksec_{d_x,b_x}$ and $\neg ksec_{d_x,S}$ for each variable x, and the goal $\neg ksec_{C_j,b_x}$ for each variable x and each clause C_j (containing the literal x or $\neg x$). The goal $ksec_{d_x,b_x}$ ensures that b_x's secret transits through x or $\neg x$. Now, recall that we assume that during a call, agents communicate all their knowledge. Suppose that b_x's secret transits through x^+: then S's secret cannot transit through x^+ before b_x's secret (because of the negative goal $\neg ksec_{d_x,S}$) and cannot transit through x^+ after b_x's secret (because of the negative goal $\neg ksec_{C_j,b_x}$). By a similar argument, if b_x's secret transits through x^-, then S's secret cannot transit through x^+. Thus, this gadget imposes that S's secret transits through exactly one of the nodes x^+, x^-.

We have shown that I_{SAT} has a solution if and only if I_{Gossip} has a solution. Since the reduction is clearly polynomial, this completes the proof. □

The NP-completeness shown in the proof of Proposition 6 holds for both the sequential and parallel versions of the gossip problem.

2.6 Discussion and conclusion

We have defined temporal epistemic gossip problems and have investigated their complexity. Our results are in line with previous results concerning epistemic planning: it is possible to add an epistemic dimension to planning, thus increasing expressibility, without increasing complexity (Cooper, Herzig, Maffre, Maris, and Régnier 2016a).

In our approach agents are not introspective: $ksec_{i,j}$ does not imply $\mathbf{S}_i\, ksec_{i,j}$, and $\mathbf{S}_k\, ksec_{i,j}$ does not imply $\mathbf{S}_k\, \mathbf{S}_k\, ksec_{i,j}$. This only concerns positive introspection: negative introspection cannot be expressed. Positive introspection can however be enforced by adding axioms $ksec_{i,j} \rightarrow \mathbf{S}_i\, ksec_{i,j}$, and $\mathbf{S}_k\, ksec_{i,j} \rightarrow \mathbf{S}_k\, \mathbf{S}_k\, ksec_{i,j}$. We however did not do so in order to simplify presentation.

We have assumed a centralized approach in which a centralized planner decides the actions of all agents. Several other researchers have recently studied distributed versions of the classical gossip problem where the agents have to decide themselves whom to call, based on the knowledge (and ignorance) they have (Apt, Grossi, and Hoek 2015; Apt, Grossi, and Hoek 2018; Apt, Kopczynski, and Wojtczak 2017; Apt and Wojtczak 2017b; Apt and Wojtczak 2017a; van Ditmarsch, Eijck, et al. 2017; van Ditmarsch, Grossi, et al. 2016). An interesting avenue of future research would be to consider the epistemic gossip problem in this framework.

Several other variants of our centralized model could also be investigated, including the precondition that i has to know the telephone number of j in order to call j and telephone numbers are communicated in the same way as secrets. In another variant, the secrets can be passwords which are no longer constants since each agent i can change their own password (Cooper, Herzig, Maffre, Maris, and Régnier 2016c).

3 Dynamic Logic of Parallel Propositional Assignments

In chapter 1 we presented some semantics of parallel actions in PDL. We here take a different route and build on a simple version of dynamic logic where atomic programs are assignments of formulas to propositional variables. Dynamic Logic of Propositional Assignments DL-PA has numerous applications in knowledge representation (Herzig 2014) in particular classical planning (Herzig, Menezes, et al. 2014), and is considerably simpler than PDL. In particular, complexity is much lower: satisfiability checking is in PSPACE. We add an operator of parallel composition to DL-PA, as well as an operator of inclusive nondeterministic composition (as opposed to PDL's exclusive nondeterministic composition).

The chapter is organised as follows. We extend DL-PA to DL-PPA in Section 3.1 and show in Section 3.2 that complexity stays in PSPACE. In Section 3.3 we show how sequential and parallel planning as well as their bounded versions can be polynomially translated to DL-PPA. Section 3.4 concludes.

The content of this chapter corresponds to an article published at the International Joint Conference on Artificial Intelligence (IJCAI 2019) (Herzig, Maris, and Vianey 2019).

Content of the chapter

Résumé du chapitre en français

Dans le chapitre 1, nous avons présenté une sémantique des actions parallèles en PDL. Nous prenons ici un chemin différent et nous nous basons sur une version simple de la logique dynamique où les programmes atomiques sont des affectations de formules à des variables propositionnelles. La *logique dynamique d'affectations propositionnelles* DL-PA a de nombreuses applications dans la représentation des connaissances (Herzig 2014) en particulier la planification classique (Herzig, Menezes et al. 2014), et est considérablement plus simple que PDL. En particulier, la complexité est beaucoup plus faible : la vérification de la satisfiabilité est dans PSPACE. Nous ajoutons un opérateur de composition parallèle à DL-PA, ainsi qu'un opérateur de composition nondéterministe inclusive (par opposition à la composition nondéterministe exclusive de PDL).

Ce chapitre est organisé comme suit. Nous étendons DL-PA en DL-PPA (*logique dynamique d'affectations propositionnelles parallèles*) dans la section 3.1 et montrons dans la section 3.2 que la complexité reste dans PSPACE. Dans la section 3.3, nous montrons comment la planification séquentielle et parallèle ainsi que leurs versions bornées peuvent être traduites de manière polynomiale en DL-PPA. La section 3.4 conclut.

Nous étendons le langage de DL-PA avec deux nouveaux opérateurs $\pi_1 \sqcap \pi_2$ et $\pi_1 \sqcup \pi_2$. Le premier permet d'indiquer que les deux programmes π_1 et π_2 sont exécutés en simultané, et le second est l'opérateur de composition nondéterministe inclusive. De la même manière que l'opérateur de composition nondéterministe exclusive, il permet l'exécution d'un des deux programmes, mais il permet également l'exécution des deux en parallèle. Le résultat de $\pi_1 \sqcup \pi_2$ sera alors soit l'exécution de π_1, soit l'exécution de π_2, soit l'exécution de $\pi_1 \sqcap \pi_2$. Nous avons ensuite créé des programmes en DL-PA permettant de traduire ces deux opérateurs en un temps polynomial, ce qui montre que DL-PPA est dans PSPACE, comme DL-PA.

L'appartenance à cette classe de complexité rend DL-PPA intéressante pour traduire des problèmes de planification classique. Ceci ayant déjà été réalisé pour la recherche de l'existence de plan séquentiel en utilisant DL-PA, nous avons complété ces résultats en ajoutant la recherche de plans parallèles. Pour cela nous avons choisi d'utiliser la sémantique ∀-Step qui est généralement utilisée en planification. Pour respecter cette sémantique, nous avons fixé deux règles. La première est que deux actions effectuées en parallèle ne doivent pas avoir d'effets contradictoires. Une action qui ajoute un fluent ne peut pas être exécutée en parallèle d'une action qui le détruit. Par exemple l'action « porter l'objet » ne peut pas s'exécuter en même temps que l'action « poser l'objet ». L'une rend faux que l'objet est au sol alors que l'autre le rend vrai. La seconde est que les conditions ne peuvent pas être altérées par une autre action, ce que l'on appelle interaction croisée. La précondition d'une action ne peut pas être rendue fausse par une autre action. Par exemple, l'action de porter un objet pour l'agent i et l'action de porter le même objet pour l'agent j ne peuvent pas se produire en parallèle. Les deux ont pour précondition la présence de l'objet au sol et comme effet l'absence de cet objet au sol. De même pour les conditions des effets conditionnels qui ne peuvent pas devenir fausses (ou vraies si elles étaient fausses) suite aux effets d'une autre action exécutée en parallèle. Nous avons également utilisé l'opérateur de k ou moins répétitions d'un programme pour modéliser la recherche d'un plan en k étapes ou moins. En effet, bien qu'il soit prouvé qu'en planification classique à horizon non borné l'existence d'un plan séquentiel implique l'existence d'un plan parallèle et vice versa ce n'est pas forcément le cas pour la recherche à horizon borné, un plan parallèle en k étapes ou moins peut exister sans qu'un plan séquentiel soit possible avec le même horizon.

3.1 DL-PA and DL-PPA

Our Dynamic Logic of Parallel Propositional Assignments DL-PPA extends Dynamic Logic of Propositional Assignments DL-PA by two program operators: an operator of parallel composition $\sqcap$ and a new operator of nondeterministic composition $\sqcup$. The distinction between $\sqcup$ and the standard operator of nondeterministic composition $\cup$ is similar to that between inclusive and exclusive disjunction: the interpretation of $\pi_1 \cup \pi_2$ is "do either π_1 or π_2", while that of $\pi_1 \sqcup \pi_2$ is "do either π_1, or π_2, *or both*". We call the former *exclusive* nondeterministic choice and the latter *inclusive* nondeterministic choice. The interpretation of parallel composition $\pi_1 \sqcap \pi_2$ is that both programs are executed on local copies of the variables, and then the resulting state is obtained by merging these local states: $\pi_1 \sqcap \pi_2$ fails if two assigned variables get assigned different truth values by π_1 and π_2; otherwise new truth values override old ones.

3.1.1 Language

The language of DL-PPA is built from a given countably infinite set of propositional variables $\mathbb{P}$. Programs π and formulas φ are defined by the grammar

$$\begin{array}{lcl} \varphi & ::= & p \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\pi\rangle\varphi \\ \pi & ::= & p{\leftarrow}\varphi \mid \varphi? \mid \pi;\pi \mid \pi \cup \pi \mid \pi \sqcup \pi \mid \pi \sqcap \pi \mid \pi^* \end{array}$$

where p ranges over $\mathbb{P}$. The formula $\langle\pi\rangle\varphi$ reads "there is a possible execution of π such that φ is true afterwards". The program $p{\leftarrow}\varphi$ assigns the truth value of φ to p and is called *atomic program*. For example, $p{\leftarrow}\neg p$ swaps the truth value of p: when p is true then it becomes false and vice versa. $\varphi?$ tests that φ is true (which fails when φ is false). $\pi_1;\pi_2$ executes π_1 and π_2 in sequence. $\pi_1 \cup \pi_2$ nondeterministically chooses between executing either π_1 or π_2; and $\pi_1 \sqcup \pi_2$ nondeterministically chooses between executing either π_1, or π_2, or both. $\pi_1 \sqcap \pi_2$ is the parallel composition of π_1 or π_2. The set of all formulas is *Fml*. In the following, we will use the standard abbreviations for $\vee$, $\rightarrow$ and $\leftrightarrow$.

The language of DL-PA is the fragment of DL-PPA without $\sqcup$ and $\sqcap$.

The set of propositional variables occurring in a formula φ is noted $\mathbb{P}_\varphi$; similarly, the set of variables occurring in a program π is noted $\mathbb{P}_\pi$. For example, $\mathbb{P}_{p{\leftarrow}q\vee r} = \{p, q, r\}$.

The *length* of DL-PPA formulas and of DL-PPA programs is the number of symbols required to write them down, excluding parentheses and considering that the length of propositional variables is 1. We note them $\ell(\varphi)$ and $\ell(\pi)$.

We use standard formula connectives such as $\top$ and $\varphi \to \psi$. We use the program connectives of n-times iteration of programs, recursively defined by

$$\pi^{\leq 0} = \top?$$
$$\pi^{\leq n+1} = \top? \cup (\pi; \pi^{\leq n})$$

Furthermore, $;_{p_i \in P} p_i \leftarrow \varphi_i$ is a shorthand for $p_1 \leftarrow \varphi_1; \cdots; p_n \leftarrow \varphi_n$, for $P = \{p_1, \ldots, p_n\}$. When $P = \emptyset$ we identify the sequence with $\top?$. We have to be careful here because sequential composition ';' is not commutative; e.g., the interpretation of $p \leftarrow q; q \leftarrow p$ will differ from that of $q \leftarrow p; p \leftarrow q$. Each time we use the abbreviation we will make sure that the order does not matter.

3.1.2 Semantics of DL-PPA

Semantics is in terms of valuations, alias states, which are subsets of $\mathbb{P}$. So the set of all valuations is $2^{\mathbb{P}}$. We use V, V', U, W, ... to denote valuations.

$$\begin{aligned}
\|p\|_{\text{DL-PA}} &= \{V \ : \ p \in V\} \\
\|\langle\pi\rangle\varphi\|_{\text{DL-PA}} &= \left\{ V \ : \ \text{there is } U \text{ such that } \begin{array}{c} (V,U) \in \|\pi\|_{\text{DL-PA}} \\ \text{and } U \in \|\varphi\|_{\text{DL-PA}} \end{array} \right\} \\
\|p \leftarrow \varphi\|_{\text{DL-PA}} &= \{(V, V \cup \{p\}) \ : \ V \in \|\varphi\|_{\text{DL-PA}}\} \\
&\quad \cup \{(V, V \setminus \{p\}) \ : \ V \notin \|\varphi\|_{\text{DL-PA}}\} \\
\|\varphi?\|_{\text{DL-PA}} &= \{(V,V) \ : \ V \in \|\varphi\|_{\text{DL-PA}}\} \\
\|\pi_1; \pi_2\|_{\text{DL-PA}} &= \|\pi_1\|_{\text{DL-PA}} \circ \|\pi_2\|_{\text{DL-PA}} \\
\|\pi_1 \cup \pi_2\|_{\text{DL-PA}} &= \|\pi_1\|_{\text{DL-PA}} \cup \|\pi_2\|_{\text{DL-PA}} \\
\|\pi^*\|_{\text{DL-PA}} &= (\|\pi\|_{\text{DL-PA}})^* \ = \ \bigcup_{k \in \mathbb{N}_0} (\|\pi\|_{\text{DL-PA}})^k
\end{aligned}$$

Table 3.1: Interpretation of DL-PA formulas and programs

Formulas and programs are interpreted by mutual recursion. In DL-PA,

the interpretation of a formula φ was a set of valuations $\|\varphi\|_{\mathsf{DL\text{-}PA}} \subseteq 2^{\mathbb{P}}$ and the interpretation of a program π is a binary relation on valuations $\|\pi\|_{\mathsf{DL\text{-}PA}} \subseteq 2^{\mathbb{P}} \times 2^{\mathbb{P}}$; the main clauses are recalled in Table 3.1. In contrast, in DL-PPA the interpretation of a program π is a ternary relation on the set of valuations $\|\pi\| \subseteq 2^{\mathbb{P}} \times 2^{\mathbb{P}} \times 2^{\mathbb{P}}$. When $(V, U, W) \in \|\pi\|$ then there is an execution of π from state V to state U assigning the variables in W. The definition is again by mutual recursion and the main clauses are given in Table 3.2; the others are standard.

Let us comment on the clauses that are new w.r.t. DL-PA.

The semantics of the assignment $p{\leftarrow}\varphi$ is: p is made true if φ is true and is made false if φ is false, and in both cases the set of assigned variables is the singleton $\{p\}$.

The semantics of parallel composition is that each subprogram π_k is executed locally; then it is checked whether the modifications (in terms of assigned variables) are compatible: this is the case when all variables that are assigned by both subprograms (namely the variables in $W_1 \cap W_2$) get assigned the same truth value. If this is not the case then the parallel composition fails; otherwise the resulting valuation U is computed by putting together (1) the unchanged part of V (i.e., $V \setminus W$), (2) the updates of π_1 (i.e., $U_1 \cap W_1$), (3) the updates of π_2 (i.e., $U_2 \cap W_2$). Moreover, the set of variables W assigned by the parallel composition is the union of those assigned by the subprograms.

The semantics of inclusive nondeterministic composition $\pi_1 \sqcup \pi_2$ is, as announced, the exclusive nondeterministic composition of the three programs π_1, π_2 and $\pi_1 \sqcap \pi_2$.

Here are some examples of interpretations of programs:

$$\begin{aligned}
\|p{\leftarrow}\bot\| &= \{(V, V \setminus \{p\}, \{p\}) \ : \ V \subseteq \mathbb{P}\} \\
\|\top?\sqcap p{\leftarrow}\bot\| &= \|p{\leftarrow}\bot\| \\
\|p{\leftarrow}\top \sqcap p{\leftarrow}\bot\| &= \emptyset \\
\|p{\leftarrow}\top \sqcap q{\leftarrow}\bot\| &= \{(V, (V \setminus \{q\}) \cup \{p\}, \{p, q\}) \ : \ V \subseteq \mathbb{P}\} \\
\|p{\leftarrow}p \sqcap p{\leftarrow}\bot\| &= \{(V, V, \{p\}) \ : \ V \subseteq \mathbb{P} \text{ and } p \notin V\}
\end{aligned}$$

Proposition 7. Let π be a DL-PPA program and let $(V, U, W) \in \|\pi\|$. Then:

- $W \subseteq \mathbb{P}_\pi$;
- $V \setminus U \subseteq W$ and $U \setminus V \subseteq W$;

$$
\begin{aligned}
\|p\| &= \{V \ : \ p \in V\} \\
\|\langle\pi\rangle\varphi\| &= \{V \ : \ \text{there are } U, W \text{ such that } (V, U, W) \in \|\pi\| \text{ and } U \in \|\varphi\|\} \\
\|p{\leftarrow}\varphi\| &= \{(V, V{\cup}\{p\}, \{p\}) \ : \ V \in \|\varphi\|\} \cup \{(V, V{\setminus}\{p\}, \{p\}) \ : \ V \notin \|\varphi\|\} \\
\|\varphi?\| &= \{(V, V, \emptyset) \ : \ V \in \|\varphi\|\} \\
\|\pi_1; \pi_2\| &= \left\{ (V, U, W) \ : \ \text{there are } U_1, W_1, W_2 \text{ s.th. } \begin{array}{l} (V, U_1, W_1) \in \|\pi_1\|, \\ (U_1, U, W_2) \in \|\pi_2\| \text{ and} \\ W = W_1 \cup W_2 \end{array} \right\} \\
\|\pi_1 \cup \pi_2\| &= \|\pi_1\| \cup \|\pi_2\| \\
\|\pi_1 \sqcup \pi_2\| &= \|\pi_1\| \cup \|\pi_2\| \cup \|\pi_1 \sqcap \pi_2\| \\
\|\pi_1 \sqcap \pi_2\| &= \left\{ (V, U, W) \ : \ \begin{array}{l} \text{there are } U_1, W_1, U_2, W_2 \text{ such that} \\ (V, U_1, W_1) \in \|\pi_1\|, (V, U_2, W_2) \in \|\pi_2\|, \\ W_1 \cap W_2 \cap U_1 = W_1 \cap W_2 \cap U_2, \\ U = (V \setminus W) \cup (U_1 \cap W_1) \cup (U_2 \cap W_2), \text{ and} \\ W = W_1 \cup W_2 \end{array} \right\} \\
\|\pi^*\| &= \bigcup_{k \in \mathbb{N}_0} \|\pi\|^k
\end{aligned}
$$

Table 3.2: Interpretation of DL-PPA programs

- If $\mathbb{P}_\pi \subseteq P$ then $V \subseteq P$ implies $U \subseteq P$.

The first item can be restricted a bit further: W is a subset of the variables occurring on the left-hand side of assignments of π.

The next result says that the variables not occurring in a formula or a program do not matter in their interpretation.

Proposition 8. Let φ be a DL-PPA formula and π a DL-PPA program. Let P be a set of variables none of which occurs in φ or π, i.e., P is such that $P \cap \mathbb{P}_\varphi = P \cap \mathbb{P}_\pi = \emptyset$. Then

- $V{\cup}P \in \|\varphi\|$ iff $V{\setminus}P \in \|\varphi\|$;
- $(V{\cup}P, U{\cup}P, W) \in \|\pi\|$ iff $(V{\setminus}P, U{\setminus}P, W) \in \|\pi\|$.

A formula φ is *satisfiable* if and only if $\|\varphi\| \neq \emptyset$. Thanks to Propositions 7 and 8, when checking validity or satisfiability of a formula φ it suffices to check triples (V, U, W) whose elements are all subsets of $\mathbb{P}_\varphi$.

When the variables of π_1 and π_2 are disjoint then they can be sequentialised:

Proposition 9. Let π_1 and π_2 be two programs such that $\mathbb{P}_{\pi_1} \cap \mathbb{P}_{\pi_2} = \emptyset$. Then $\|\pi_1 \sqcap \pi_2\| = \|\pi_1; \pi_2\| = \|\pi_2; \pi_1\|$.

Observe that Proposition 9 cannot be strengthened by only requiring that the variables that are assigned by π_1 and π_2 are disjoint. For example, the sets of valuations $\|p{\leftarrow}\top \sqcap q{\leftarrow}p\|$ and $\|p{\leftarrow}\top; q{\leftarrow}p\|$ are different.

Just as in DL-PA (Balbiani, Herzig, Schwarzentruber, et al. 2014), the Kleene star can be eliminated in DL-PPA: intuitively, when there exists a run of π^* going from x to y then it is possible to go from x to y in no more than $2^{|\mathbb{P}_\pi|}$ iterations of π. (Indeed, if there is a longer run, it necessarily goes through the same valuation twice and can be shortened.)

Proposition 10. For all DL-PPA programs π, $\|\pi^*\| = \|\pi^{\leq 2^{|\mathbb{P}_\pi|}}\|$.

The above result can be strengthened: one might replace $|\mathbb{P}_\pi|$ with the number of atomic programs appearing in π; even better, one might replace it with the number of distinct variables appearing on the left-hand side of assignments of π. For instance, $p{\leftarrow}q \vee r^*$ can then be reduced to $p{\leftarrow}q \vee r^{\leq 2}$ instead of $p{\leftarrow}q \vee r^{\leq 8}$.

3.1.3 Counters

In DL-PA one can implement k-bit counters (see e.g. (Balbiani, Herzig, and Troquard 2013, Section II.C)), and this transfers to DL-PPA. The encoding of a binary number of length k requires a vector of $\lceil \log k \rceil$ propositional variables. We represent the state of the counter by a formula ct, which stands for the conjunction of these $\lceil \log k \rceil$ variables or negations thereof. Furthermore we use the following abbreviations:

- $ct{\leftarrow}0$ is a program that resets ct: all variables of the counter are set to false;
- For every integer i, $ct{=}i$ is a formula that is true if and only if the value encoded by ct is i;
- $ct{+}{+}$ increments the value of ct.

We suppose that variables used in a counter do not occur elsewhere, so that they do not interfere with other programs.

Counters allow us to formulate in a more compact way our abbreviation $\pi^{\leq k}$ whose expansion has length exponential in k: using a $\lceil \log k \rceil$-bit counter we can identify $\pi^{\leq k}$ with the program

$$ct{\leftarrow}0; (\neg ct{=}k\,?\,; \pi; ct{+}{+})^*$$

whose length is linear in $\ell(\pi) + \log k$. Indeed, although these two programs do not have the same interpretation, they behave the same as far as the variables of π are concerned.

Proposition 11. For valuations $V, U, W \subseteq \mathbb{P}_\pi$, we have $(V, U, W) \in \|\pi^{\leq k}\|$ if and only if

$$(V, U{\cup}U', W{\cup}W') \in \|ct{\leftarrow}0; (\neg ct{=}k\,?\,; \pi; ct{+}{+})^*\|$$

for some subsets U' and W' of the set of counter variables.

3.2 Reduction from DL-PPA to DL-PA

We are now going to give a polynomial-time reduction of formulas and programs of DL-PPA to DL-PA. Our translation eliminates $\sqcap$ and $\sqcup$ thanks to the introduction of (several kinds of) copies of propositional variables.

3.2.1 Auxiliary variables

First, the variable δ_p stores that p has been assigned; it will allow us to simulate the third component W of the interpretation of programs that keeps trace of assigned variables.

Second, when translating parallel composition we sequentialise the parallel execution of two programs π_1 and π_2, and in order to safely do so we let each of them work on local copies of variables p, respectively noted $(p)^1$ and $(p)^2$.

The translation of inclusive nondeterministic composition $\pi_1 \sqcup \pi_2$ also requires some care: the naive $f(\pi_1 \sqcup \pi_2) = \pi_1 \cup \pi_2 \cup (\pi_1 \sqcap \pi_2)$ would come with an exponential growth. Our translation re-uses the above copying technique.

We extend copying from variables to sets of variables and to programs. We associate to each set of variables $P \subseteq \mathbb{P}$ the copies $(P)^1 = \{(p)^1 \ : \ p \in P\}$ and $(P)^2 = \{(p)^2 \ : \ p \in P\}$; and we map programs π to programs $(\pi)^1$ by replacing all variables p of π by $(p)^1$; similarly, we map π to $(\pi)^2$. For example, $(p \leftarrow q \wedge r)^2 = (p)^2 \leftarrow (q)^2 \wedge (r)^2$.

Proposition 12. Let $V, U, W \subseteq \mathbb{P}_\pi$ and let k be either 1 or 2. Then:

- $(V, U, W) \in \|\pi\|$ iff $((V)^k, (U)^k, (W)^k) \in \|(\pi)^k\|$;
- $V \in \|\varphi\|$ iff $(V)^k \in \|(\varphi)^k\|$.

The proof is by simultaneous induction on the structure of π and φ.

3.2.2 Programs and formulas used in the translation

The translation uses some programs and formulas that are defined in this subsection.

The first program creates two copies of the variables in a finite set $P \subseteq \mathbb{P}$ and stores that these copies have not been assigned yet:

$$\mathrm{copy}(P) = ;_{p \in P} \Big((p)^1 \leftarrow p; (p)^2 \leftarrow p; \delta_{(p)^1} \leftarrow \bot; \delta_{(p)^2} \leftarrow \bot \Big)$$

Proposition 13. For every finite $P \subseteq \mathbb{P}$, $(V, U) \in \|\mathrm{copy}(P)\|_{\mathsf{DL\text{-}PA}}$ if and only if

$$U = ((V \setminus ((P)^1 \cup (P)^2)) \cup ((V)^1 \cap (P)^1) \cup ((V)^2 \cap (P)^2)) \setminus (\delta_{((P)^1)} \cup \delta_{((P)^2)}).$$

The next program copies back the values of the i-copies of the elements of P and records whether they were modified, for $i \in \{1, 2\}$:

$$\mathrm{get}(P, i) = ;_{p \in P} \Big(p \leftarrow (p)^i; \delta_p \leftarrow \delta_p \vee \delta_{(p)^i} \Big)$$

Proposition 14. For every finite $P \subseteq \mathbb{P}$, $(V, U) \in \|\mathrm{get}(P, i)\|_{\mathsf{DL\text{-}PA}}$ if and only if

$$U = ((V \setminus P) \cup \{p \ : \ (p)^i \in V\} \cup \{\delta_p \ : \ \delta_{(p)^i} \in V \cap \delta_{(P)^i}\}.$$

The last program merges two parallel subprograms: it copies back the values of the variables in P, depending on where it got modified:

$$\mathrm{merge}(P) = ;_{p\in P}\Big(p{\leftarrow}(p \wedge \neg\delta_{(p)^1} \wedge \neg\delta_{(p)^2}) \vee ((p)^1 \wedge \delta_{(p)^1}) \vee ((p)^2 \wedge \delta_{(p)^2});\\ \delta_p{\leftarrow}\delta_p \vee \delta_{(p)^1} \vee \delta_{(p)^2}\Big)$$

Proposition 15. For every finite $P \subseteq \mathbb{P}$, $(V, U) \in \|\mathrm{merge}(P)\|_{\mathsf{DL\text{-}PA}}$ if and only if

$$U = (V \setminus (P \cup \delta_P)) \cup \\ \left\{ p \in P \ : \ \begin{array}{r} (p \in V \text{ and } \delta_{(p)^1}, \delta_{(p)^2} \notin V) \\ \text{or } (\delta_{(p)^1}, (p)^1 \in V) \\ \text{or } (\delta_{(p)^2}, (p)^2 \in V) \end{array} \right\} \cup \\ \{\delta_p \in \delta_P \ : \ \delta_p \in V \text{ or } \delta_{(p)^1} \in V \text{ or } \delta_{(p)^2} \in V)\}.$$

Finally, the following formula conditions the merging of two parallel subprograms:

$$\mathrm{Mergeable}(P) = \bigwedge_{p\in P} \Big(\Big(\delta_{(p)^1} \wedge \delta_{(p)^2}\Big) \to \Big((p)^1 \leftrightarrow (p)^2\Big)\Big)$$

3.2.3 The translation

We are going to show that $\sqcap$ and $\sqcup$ can be polynomially eliminated from DL-PPA formulas. The resulting formula is in the language of DL-PA, which will allow us to transfer complexity results. The recursive definition of the translation of formulas and programs is given in Table 3.3. It is homomorphic for all the program operators and logical operators of PDL, so we only comment the non-trivial cases: assignments, parallel composition and inclusive nondeterministic composition.

- Each assignment $p{\leftarrow}\varphi$ is translated by additionally storing that p has been assigned (via a fresh variable δ_p).
- The translation of parallel composition $\pi_1 \sqcap \pi_2$ relies on the introduction of two fresh copies $(p)^1$ and $(p)^2$ of each propositional variable p occurring in $\pi_1 \sqcap \pi_2$. Then for each program π, the program $(\pi)^1$ is obtained

by substituting each p in π by its copy $(p)^1$; similarly for $(\pi)^2$. The translation of $\pi_1 \sqcap \pi_2$ starts by making local copies for each subprogram, then executes π_1 and π_2 on these copies. Then the translation checks whether the two results can be merged. If so then the values of the relevant copies are copied back to each p (doing nothing when none of the copies has been assigned). Finally the auxiliary variables $(p)^1$ and $(p)^2$ are set to false.

- In the translation of inclusive nondeterministic composition $\pi_1 \sqcup \pi_2$, the check of compatibility followed by copying back is replaced by an exhaustive nondeterministic composition of either checking compatibility and copying back the values computed by $\pi_1 \sqcap \pi_2$, or copying back the values computed by π_1, or copying back the values computed by π_2.

$$
\begin{aligned}
f(p) &= p & f(p{\leftarrow}\varphi) &= p{\leftarrow}f(\varphi); \delta_p{\leftarrow}\top \\
f(\bot) &= \bot & f(\varphi?\,) &= f(\varphi)? \\
f(\neg\varphi) &= \neg f(\varphi) & f(\pi_1;\pi_2) &= f(\pi_1); f(\pi_2) \\
f(\varphi_1 \vee \varphi_2) &= f(\varphi_1) \vee f(\varphi_2) & f(\pi_1 \cup \pi_2) &= f(\pi_1) \cup f(\pi_2) \\
f(\langle\pi\rangle\varphi) &= \langle f(\pi)\rangle f(\varphi) & f(\pi^*) &= (f(\pi))^*
\end{aligned}
$$

$$
\begin{aligned}
f(\pi_1 \sqcap \pi_2) = {} & \mathrm{copy}(\mathbb{P}_{\pi_1} \cup \mathbb{P}_{\pi_2}); f\Big((\pi_1)^1\Big); f\Big((\pi_1)^2\Big); \\
& \mathrm{Mergeable}(\mathbb{P}_{\pi_1} \cup \mathbb{P}_{\pi_2})?\,; \mathrm{merge}(\mathbb{P}_{\pi_1} \cup \mathbb{P}_{\pi_2}); \\
f(\pi_1 \sqcup \pi_2) = {} & \mathrm{copy}(\mathbb{P}_{\pi_1} \cup \mathbb{P}_{\pi_2}); f\Big((\pi_1)^1\Big); f\Big((\pi_1)^2\Big); \\
& ((\mathrm{Mergeable}(\mathbb{P}_{\pi_1} \cup \mathbb{P}_{\pi_2})?\,; \mathrm{merge}(\mathbb{P}_{\pi_1} \cup \mathbb{P}_{\pi_2})) \\
& \cup \mathrm{get}(\mathbb{P}_{\pi_1}, 1) \cup \mathrm{get}(\mathbb{P}_{\pi_2}, 2));
\end{aligned}
$$

Table 3.3: Translation from DL-PPA to DL-PA

For example, $f(p{\leftarrow}\top \sqcap p{\leftarrow}\bot)$ is

$$(p)^1{\leftarrow}p; (p)^2{\leftarrow}p; \delta_{(p)^1}{\leftarrow}\bot; \delta_{(p)^2}{\leftarrow}\bot;$$
$$(p)^1{\leftarrow}\top; \delta_{(p)^1}{\leftarrow}\top; (p)^2{\leftarrow}\bot; \delta_{(p)^2}{\leftarrow}\top; (\delta_{(p)^1} \wedge \delta_{(p)^2}) \to ((p)^1 \leftrightarrow (p)^2)?;$$
$$p{\leftarrow}(p \wedge \neg\delta_{(p)^1} \wedge \neg\delta_{(p)^2}) \vee ((p)^1 \wedge \delta_{(p)^1}) \vee ((p)^2 \wedge \delta_{(p)^2});$$
$$\delta_p{\leftarrow}\delta_p \vee \delta_{(p)^1} \vee \delta_{(p)^2}.$$

Just as the original program, the translated program has an empty interpretation because after setting $\delta_{(p)^2}$ and $\delta_{(p)^1}$ to true and $(p)^1$ to true and $(p)^2$ to false, the test $\left(\delta_{(p)^1} \wedge \delta_{(p)^2}\right) \to ((p)^1 \leftrightarrow (p)^2)?$ is going to fail.

3.2.4 Correctness of the translation

We define the *copied variables* for the translation of programs and formulas as:

$$CP(\varphi) = \mathbb{P}_{f(\varphi)} \setminus (\mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi})$$
$$CP(\pi) = \mathbb{P}_{f(\pi)} \setminus (\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi})$$

For example, the copied variables of the above program are $CP(p{\leftarrow}\top \sqcap p{\leftarrow}\bot) = \{(p)^1, (p)^2, \delta_{(p)^1}, \delta_{(p)^2}\}$. Observe that $\mathbb{P}_{f(\pi)}$ is a subset of $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi} \cup CP(\pi)$ and that $\mathbb{P}_\pi$, $\delta_{\mathbb{P}_\pi}$, and $CP(\pi)$ are disjoint. We call $\mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi}$ the surface variables of $f(\varphi)$; similarly, $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$ are the surface variables of $f(\pi)$.

We say that two valuations V and V' *agree* on a set of propositional variables $X \subseteq \mathbb{P}$ if $V \cap X = V' \cap X$. It follows from Proposition 8 that if V and V' agree on $\mathbb{P}_\varphi$ then $V \in \|\varphi\|$ if and only if $V' \in \|\varphi\|$. This means that only the variables of φ matter when interpreting φ.

Lemma 16. Let $\delta_p \notin \mathbb{P}_\pi$ and $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$. If $\delta_p \in V$ then$\delta_p \in U$.

Proof. The only modification of the change-recording variables occurs in the translation of parallel compositions and inclusive nondeterministic compositions, via assignments of the form $\delta_p{\leftarrow}\delta_p \vee \delta_{(p)^1} \vee \delta_{(p)^2}$. Therefore the δ_p variables are never made false. □

The next lemma relies on the fact that when translating $\pi_1 \sqcap \pi_2$ and $\pi_1 \sqcup \pi_2$, the variables $\delta_{(p)^i}$ are set to false before the programs $(\pi_i)^i$ are executed:

this guarantees that their truth was caused by an assignment of $(p)^i$ that was performed during the execution of $(\pi_i)^i$. Without such an initialisation for example the translation of $f(\langle p \leftarrow \top \sqcap \top ? \rangle \top)$ would be sensitive to the original truth value of $\delta_{(p)^2}$: it would be true in the empty valuation and false in the valuation $\{\delta_{(p)^2}\}$.

Lemma 17. Let $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and $(V', U') \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$. Let U and U' agree on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$. If V and V' agree on $P \subseteq \mathbb{P}$ and $P \cap CP(\pi) = \emptyset$ then U and U' agree on P.

Proof. The only variables that $f(\pi)$ can modify are in $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi} \cup CP(\pi)$. Suppose both (V, U) and (V', U') are in $\|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and P and $CP(\pi)$ are disjoint. Then $f(\pi)$ cannot modify the variables in $P \setminus (\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi})$, and U and V must agree on $P \setminus (\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi})$. For the same reason, U' and V' must agree on $P \setminus (\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi})$. As V and V' agree on P, they trivially also agree on $P \setminus (\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi})$, and we get by transitivity of equality that U and U' agree on $P \setminus (\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi})$. As by hypothesis U and U' agree on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$, they agree on the entire set P. □

The next lemma says that only the surface variables matter when interpreting translated formulas and programs.

Lemma 18. For all formulas φ and programs π:

1. If V and V' agree on $\mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi}$ and $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ then $V' \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$;
2. If V and V' agree on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$ and $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ then there is a U' agreeing with U on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$ such that $(V', U') \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$.

Proof. The proof is by simultaneous induction on the form of φ and π and uses Lemma 17. Instead of the second item we use as induction hypothesis that if V and V' agree on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$ and $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ then $(V', U') \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ for $U' = (U \cap \mathbb{P}_{f(\pi)}) \cup (V' \setminus \mathbb{P}_{f(\pi)})$. We abbreviate that induction hypothesis by IH2, while IH1 denotes the first item in the statement of the lemma.

- For the case of formulas of the form $\langle \pi \rangle \varphi$, to prove IH1 suppose V and V' agree on $\mathbb{P}_{\langle \pi \rangle \varphi} \cup \delta_{\mathbb{P}_{\langle \pi \rangle \varphi}}$ and $V \in \|f(\langle \pi \rangle \varphi)\|_{\mathsf{DL\text{-}PA}}$. By the truth condition there is a U such that $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and $U \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. By IH2 we obtain that $(V', U') \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ for $U' = (U \cap \mathbb{P}_{f(\pi)}) \cup (V' \setminus$

$\mathbb{P}_{f(\pi)}$). By construction U' agrees with U on $\mathbb{P}_{f(\pi)}$ and therefore also on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$. Lemma 17 applies for $P = \mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi}$ because the auxiliary variables in $CP(\pi)$ are fresh: U' and U agree on $P = \mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi}$. This allows us to apply IH1, giving us $U' \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. By the truth condition for the dynamic operator it follows that $V' \in \|f(\langle\pi\rangle\varphi)\|_{\mathsf{DL\text{-}PA}}$.

- The other cases of formulas are straightforward.

- For the case of assignments, to prove IH2 suppose V and V' agree on $\mathbb{P}_{p\leftarrow\varphi} \cup \delta_{\mathbb{P}_{p\leftarrow\varphi}} = \mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi} \cup \{p, \delta_p\}$ and $(V, U) \in \|f(p{\leftarrow}\varphi)\|_{\mathsf{DL\text{-}PA}}$. We distinguish two subcases. If $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ then by IH1 $V' \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. Then $U = V \cup \{p, \delta_p\}$, and setting $U' = V' \cup \{p, \delta_p\}$ we clearly have $(V', U') \in \|f(p{\leftarrow}\varphi)\|_{\mathsf{DL\text{-}PA}}$. The subcase where $V \notin \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ is similar.

- For the case of sequential composition, suppose V and V' agree on $\mathbb{P}_{\pi_1;\pi_2} \cup \delta_{\mathbb{P}_{\pi_1;\pi_2}}$ and $(V, U) \in \|f(\pi_1;\pi_2)\|_{\mathsf{DL\text{-}PA}}$. Then there is a U_1 such that $(V, U_1) \in \|f(\pi_1)\|_{\mathsf{DL\text{-}PA}}$ and $(U_1, U) \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}}$. Applying IH2 we get $(V', U_1') \in \|f(\pi_1)\|_{\mathsf{DL\text{-}PA}}$, for $U_1' = (U_1 \cap \mathbb{P}_{f(\pi_1)}) \cup (V' \setminus \mathbb{P}_{f(\pi_1)})$. Clearly, U_1 and U_1' agree on $\mathbb{P}_{\pi_1} \cup \delta_{\mathbb{P}_{\pi_1}}$. By Lemma 17 (which applies because the auxiliary variables in $CP(\pi_1)$ are fresh) they also agree on $\mathbb{P}_{\pi_2} \cup \delta_{\mathbb{P}_{\pi_2}}$. This allows us to apply IH1, giving us $(U_1', U') \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}}$, for

$$\begin{aligned}
U' &= (U \cap \mathbb{P}_{f(\pi_2)}) \cup (U_1' \setminus \mathbb{P}_{f(\pi_2)}) \\
&= (U \cap \mathbb{P}_{f(\pi_2)}) \cup (((U_1 \cap \mathbb{P}_{f(\pi_1)}) \cup (V' \setminus \mathbb{P}_{f(\pi_1)})) \setminus \mathbb{P}_{f(\pi_2)}) \\
&= (U \cap \mathbb{P}_{f(\pi_2)}) \cup ((U_1 \cap \mathbb{P}_{f(\pi_1)}) \setminus \mathbb{P}_{f(\pi_2)}) \cup ((V' \setminus \mathbb{P}_{f(\pi_1)}) \setminus \mathbb{P}_{f(\pi_2)}) \\
&= (U \cap (\mathbb{P}_{f(\pi_2)} \cup \mathbb{P}_{f(\pi_1)})) \cup (V' \setminus (\mathbb{P}_{f(\pi_1)} \cup \mathbb{P}_{f(\pi_2)})) \\
&= (U \cap (\mathbb{P}_{f(\pi_2;\pi_1)})) \cup (V' \setminus (\mathbb{P}_{f(\pi_1;\pi_2)}))
\end{aligned}$$

It follows that $(V', U') \in \|f(\pi_1;\pi_2)\|_{\mathsf{DL\text{-}PA}}$ for $U' = (U \cap \mathbb{P}_{f(\pi_1;\pi_2)}) \cup (V' \setminus \mathbb{P}_{f(\pi_1;\pi_2)})$.

- The cases of exclusive nondeterministic composition and of finite iteration are straightforward.

- The cases of parallel composition and of inclusive nondeterministic composition are similar to that of sequential composition but a bit

more fastidious. Basically, their proof uses that in the translation of parallel composition the initial values of all the copy variables $(p)^1$ and $(p)^2$ in $CP(\mathbb{P}_\pi)$ are irrelevant because they get assigned the value of p. Similarly, the initial values of change-recording variables that are not surface variables (such as $\delta_{(p)^1}$ and $\delta_{(p)^2}$) are irrelevant because they get assigned to false.

This ends the proof. □

The second item of the above lemma implies that when V and V' agree on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$ and $(V, U) \in \|f(\pi)\|$ then there is a U' agreeing with U on $\mathbb{P}_\pi \cup \delta_{\mathbb{P}_\pi}$ such that $(V', U') \in \|f(\pi)\|$.

The next lemma says that the change-recording variables can be added to the interpretation of translated formulas and programs.

Lemma 19. For all formulas φ, programs π and finite sets of variables X such that $\delta_X \cap \mathbb{P}_\pi = \delta_X \cap \mathbb{P}_\varphi = \emptyset$:

1. If $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ then $V \cup \delta_X \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ and $V \setminus \delta_X \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$;
2. If $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ then $(V \cup \delta_X, U \cup \delta_X) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and $(V \setminus \delta_X, (U \setminus (\delta_X \cap V))) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$.

Proof. We prove by simultaneous induction on the form of φ and π that for every $p \in \mathbb{P}$:

1. If $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ then $V \cup \{\delta_p\} \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ (IH1);
2. If $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ then $V \setminus \{\delta_p\} \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ (IH2);
3. If $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ then $(V \cup \{\delta_p\}, U \cup \{\delta_p\}) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ (IH3);
4. If $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and $\delta_p \in V$ then $(V \setminus \{\delta_p\}, U \setminus \{\delta_p\}) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ (IH4).

For formulas we only give the case of the dynamic operator. Suppose $V \in \|f(\langle\pi\rangle\varphi)\|_{\mathsf{DL\text{-}PA}}$, which means that there is a U such that $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and $U \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. In order to prove that $V \cup \{\delta_p\} \in \|f(\langle\pi\rangle\varphi)\|_{\mathsf{DL\text{-}PA}}$ we use IH3 and IH1, which give us that $(V \cup \{\delta_p\}, U \cup \{\delta_p\}) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and $U \cup \{\delta_p\} \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. In order to prove that $V \setminus \{\delta_p\} \in \|f(\langle\pi\rangle\varphi)\|_{\mathsf{DL\text{-}PA}}$ we distinguish two cases.

- If $\delta_p \in V$ then by IH4 and IH2 we obtain $(V \setminus \{\delta_p\}, U \setminus \{\delta_p\}) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ and $U \setminus \{\delta_p\} \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$;
  therefore $V \setminus \{\delta_p\} \in \|\langle f(\pi)\rangle f(\varphi)\|_{\mathsf{DL\text{-}PA}}$.

- If $\delta_p \notin V$ then $V \setminus \{\delta_p\} \in \|f(\langle\pi\rangle\varphi)\|_{\mathsf{DL\text{-}PA}}$ is trivially the case (by hypothesis $V \in \|f(\langle\pi\rangle\varphi)\|_{\mathsf{DL\text{-}PA}}$).

For programs, the cases of parallel compositions and inclusive nondeterministic compositions rely on Lemma 16: in their translation, the only modifications of change-recording variables δ_p are by assignments of the form $\delta_p{\leftarrow}\delta_p \vee \delta_{(p)^1} \vee \delta_{(p)^2}$ which never make any δ_p variable false. □

The last two lemmas entail that only the variables in $\mathbb{P}_\varphi$ are relevant for the interpretation of $f(\varphi)$.

Lemma 20. For every formula φ, $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ iff $V \cap \mathbb{P}_\varphi \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$.

Proof. We prove that if V and V' agree on $\mathbb{P}_\varphi$ and $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ then $V' \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. Suppose V and V' agree on $\mathbb{P}_\varphi$ and $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. We split V up into three disjoint sets and get:

$$(V{\cap}\mathbb{P}_\varphi) \cup (V{\cap}\delta_{\mathbb{P}_\varphi)} \cup (V \setminus (\mathbb{P}_\varphi{\cup}\delta_{\mathbb{P}_\varphi})) \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}.$$

As V and V' agree on $\mathbb{P}_\varphi$ we have $V{\cap}\mathbb{P}_\varphi = V'{\cap}\mathbb{P}_\varphi$, hence:

$$(V'{\cap}\mathbb{P}_\varphi) \cup (V{\cap}\delta_{\mathbb{P}_\varphi)} \cup (V \setminus (\mathbb{P}_\varphi{\cup}\delta_{\mathbb{P}_\varphi})) \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}.$$

Then Lemma 19 (first item) tells us that the change-recording variables don't matter when interpreting formulas, therefore:

$$(V'{\cap}\mathbb{P}_\varphi) \cup (V'{\cap}\delta_{\mathbb{P}_\varphi)} \cup (V \setminus (\mathbb{P}_\varphi{\cup}\delta_{\mathbb{P}_\varphi})) \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}.$$

Finally, as this valuation agrees with V' on the surface variables $\mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi}$, by Lemma 18 (first item) we get that $V' \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. □

The above lemmas are used in the grande finale:

Theorem 21. For every formula φ, $\|\varphi\| = \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$.

Proof. We prove by simultaneous induction on the form of formulas and programs:

1. $V \in \|\varphi\|$ iff $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ (IH1);

2. If $(V, U, W) \in \|\pi\|$ then $(V{\cap}\mathbb{P}_\pi, (U{\cap}\mathbb{P}_\pi) \cup \delta_W \cup X) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ for some $X \subseteq CP(\pi)$ (IH2);

3. If $(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$ then $(V \cap \mathbb{P}_\pi, U \cap \mathbb{P}_\pi, W) \in \|\pi\|$ for some $W \subseteq \mathbb{P}$ (IH3).

We analyse the different cases of formulas and programs.

- For formulas of the form $\langle\pi\rangle\varphi$ we prove the two directions of IH1 separately.

  For the *left-to-right direction*, suppose $V \in \|\langle\pi\rangle\varphi\|$. Then there are U, W such that

  $$(V, U, W) \in \|\pi\| \text{ and } U \in \|\varphi\|$$

  . By IH2 and IH1:

  $$(V \cap \mathbb{P}_\pi, (U \cap \mathbb{P}_\pi) \cup \delta_W \cup X) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}} \text{ and } U \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$$

  for some $X \subseteq CP(\pi)$. As to the interpretation of the program $f(\pi)$, by Proposition 8 we can add the irrelevant variables $V \cap (\mathbb{P}_\varphi \setminus \mathbb{P}_\pi)$ to both elements of the tuple:

  $$((V \cap \mathbb{P}_\pi) \cup (V \cap (\mathbb{P}_\varphi \setminus \mathbb{P}_\pi)), \\ (U \cap \mathbb{P}_\pi) \cup (V \cap (\mathbb{P}_\varphi \setminus \mathbb{P}_\pi)) \cup \delta_W \cup X) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}$$

  . (The proposition applies because the sets $V \cap (\mathbb{P}_\varphi \setminus \mathbb{P}_\pi)$ and $\mathbb{P}_{f(\pi)}$ are disjoint.) Again by Proposition 8 we have $V \cap (\mathbb{P}_\varphi \setminus \mathbb{P}_\pi) = U \cap (\mathbb{P}_\varphi \setminus \mathbb{P}_\pi)$, and therefore get:

  $$((V \cap (\mathbb{P}_\varphi \cup \mathbb{P}_\pi)), (U \cap (\mathbb{P}_\varphi \cup \mathbb{P}_\pi)) \cup \delta_W \cup X) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}},$$

  that is

  $$((V \cap \mathbb{P}_{\langle\pi\rangle\varphi}, (U \cap \mathbb{P}_{\langle\pi\rangle\varphi}) \cup \delta_W \cup X) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}}. \tag{3.1}$$

  As to the interpretation of the formula $f(\varphi)$, we modify U as follows:

  – intersect U with $\mathbb{P}_{f(\varphi)}$ and then add the irrelevant variables $U \cap (\mathbb{P}_\pi \setminus \mathbb{P}_\varphi)$ thanks to Proposition 8 (similarly to what we did for programs);

  – subtract all change-recording variables of U and then add the change-recording variables δ_W thanks to Lemma 19.

By these set-theoretic operations we obtain that

$$(U \cap \mathbb{P}_{\langle\pi\rangle\varphi}) \cup \delta_W \cup (U \cap CP(\varphi)) \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}. \tag{3.2}$$

Observe that the second element of the tuple in (3.1) and the valuation in (3.2) agree on the surface variables of $\langle\pi\rangle\varphi$, and a fortiori on those of φ, i.e., on $\mathbb{P}_\varphi \cup \delta_{\mathbb{P}_\varphi}$. We can therefore apply the first item of Lemma 18: we have

$$(U \cap \mathbb{P}_{\langle\pi\rangle\varphi}) \cup \delta_W \cup X \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}. \tag{3.3}$$

Then 3.1 and 3.3 together tell us that

$$V \cap \mathbb{P}_{\langle\pi\rangle\varphi} \in \|\langle f(\pi)\rangle f(\varphi)\|_{\mathsf{DL\text{-}PA}}.$$

Applying again Lemma 19 we then get

$$(V \cap \mathbb{P}_{\langle\pi\rangle\varphi}) \cup (V \cap \delta_{\mathbb{P}_{\langle\pi\rangle\varphi}}) \in \|\langle f(\pi)\rangle f(\varphi)\|_{\mathsf{DL\text{-}PA}}.$$

As this valuation agrees with V on the surface variables of $\langle\pi\rangle\varphi$, we can apply Lemma 18 again and conclude that

$$V \in \|f(\langle\pi\rangle\varphi)\|_{\mathsf{DL\text{-}PA}}.$$

For the *right-to-left direction* of IH1, suppose $V \in \|f(\langle\pi\rangle\varphi)\|$. Then there is a U such that

$$(V, U) \in \|f(\pi)\|_{\mathsf{DL\text{-}PA}} \text{ and } U \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$$

. Applying IH1 and IH3 we get that there is a W such that

$$(V \cap \mathbb{P}_\pi, U \cap \mathbb{P}_\pi, W) \in \|\pi\| \text{ and } U \in \|\varphi\|$$

. As above we can add the irrelevant variables $V \cap (\mathbb{P}_\varphi \setminus \mathbb{P}_\pi)$ thanks to Proposition 8:

$$(V \cap \mathbb{P}_{\langle\pi\rangle\varphi}, U \cap \mathbb{P}_{\langle\pi\rangle\varphi}, W) \in \|\pi\| \text{ and } U \cap \mathbb{P}_{\langle\pi\rangle\varphi} \in \|\varphi\|$$

, from which it directly follows that $V \cap \mathbb{P}_{\langle\pi\rangle\varphi} \in \|\langle\pi\rangle\varphi\|$. The latter is equivalent to $V \in \|\langle\pi\rangle\varphi\|$ by Proposition 8.

- The other cases of formulas are straightforward.

- For the case of assignments, to prove IH2 suppose $(V, U, W) \in \|p{\leftarrow}\varphi\|$. We distinguish two subcases. If $V \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$ then $U = V \cup \{p\}$, $W = \{p\}$. By Proposition 8 we have $V{\cap}\mathbb{P}_\varphi \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$. By IH1, $V{\cap}\mathbb{P}_\varphi \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. Hence $(V{\cap}\mathbb{P}_\varphi, (V{\cap}\mathbb{P}_\varphi) \cup \{p, \delta_p\}) \in \|f(p{\leftarrow}\varphi)\|_{\mathsf{DL\text{-}PA}}$ and we have $X = \emptyset$. The subcase where $V \notin \|\varphi\|_{\mathsf{DL\text{-}PA}}$ is similar.

  To prove IH3 suppose $(V, U) \in \|f(p{\leftarrow}\varphi)\|_{\mathsf{DL\text{-}PA}}$. We again distinguish two subcases. If $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ then $U = V \cup \{p, \delta_p\}$ and therefore $U{\cap}\mathbb{P}_{p\leftarrow\varphi} = (V{\cap}\mathbb{P}_\varphi) \cup \{p\}$. By Lemma 20 we have $V{\cap}\mathbb{P}_\varphi \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. Then by IH1 we get $V{\cap}\mathbb{P}_\varphi \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$. Hence $V{\cap}\mathbb{P}_{p\leftarrow\varphi} \in \|\varphi\|_{\mathsf{DL\text{-}PA}}$, too, as well as $(V{\cap}\mathbb{P}_{p\leftarrow\varphi}, U{\cap}\mathbb{P}_{p\leftarrow\varphi}, W) \in \|p{\leftarrow}\varphi\|_{\mathsf{DL\text{-}PA}}$ for $W = \{p\}$. The subcase where $V \notin \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$ is similar.

- For the case of tests, to prove IH2 suppose $(V, U, W) \in \|\varphi?\|$. Then $V \in \|\varphi\|$, $U = V$, and $W = \emptyset$. By Proposition 8 we have $V{\cap}\mathbb{P}_\varphi \in \|\varphi\|$, and by IH1 we have $V{\cap}\mathbb{P}_\varphi \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$. Hence $(V{\cap}\mathbb{P}_\varphi, U{\cap}\mathbb{P}_\varphi, \emptyset) \in \|f(\varphi)?\|_{\mathsf{DL\text{-}PA}}$.

  To prove IH3 suppose $(V, U) \in \|f(\varphi?)\|_{\mathsf{DL\text{-}PA}}$. Then $V \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$, $U = V$. By Lemma 20 we have $V{\cap}\mathbb{P}_\varphi \in \|f(\varphi)\|_{\mathsf{DL\text{-}PA}}$, from which $V{\cap}\mathbb{P}_\varphi \in \|\varphi\|$ by IH1. Hence $(V{\cap}\mathbb{P}_\varphi, U{\cap}\mathbb{P}_\varphi, \emptyset) \in \|\varphi?\|$.

- For the case of sequential compositions, to prove IH2 suppose $(V, U, W) \in \|\pi_1; \pi_2\|$. Then there are U_1, W_1, W_2 such that $W = W_1 \cup W_2$ and

$$(V, U_1, W_1) \in \|\pi_1\| \text{ and } (U_1, U, W_2) \in \|\pi_2\|$$

  Applying IH2 twice we get

$$(V{\cap}\mathbb{P}_{\pi_1}, (U_1{\cap}\mathbb{P}_{\pi_1}) \cup \delta_{W_1} \cup X_1) \in \|f(\pi_1)\|_{\mathsf{DL\text{-}PA}},$$
$$(U_1{\cap}\mathbb{P}_{\pi_2}, (U{\cap}\mathbb{P}_{\pi_2}) \cup \delta_{W_2} \cup X_2) \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}},$$

  for some $X_1 \subseteq CP(\pi_1)$ and $X_2 \subseteq CP(\pi_2)$. Adding irrelevant variables as we did for the previous cases we get

$$(V{\cap}\mathbb{P}_{\pi_1;\pi_2}, (U_1{\cap}\mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1} \cup X_1) \in \|f(\pi_1)\|_{\mathsf{DL\text{-}PA}},$$
$$(U_1{\cap}\mathbb{P}_{\pi_1;\pi_2}, (U{\cap}\mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_2} \cup X_2) \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}}.$$

By Lemma 19 we can add the change-recording variables δ_{W_1} to the interpretation of $f(\pi_2)$ and get

$$((U_1 \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1}, (U \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1} \cup \delta_{W_2} \cup X_2) \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}}.$$

As the valuations $(U_1 \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1} \cup X_1$ and $(U_1 \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1}$ agree on the surface variables of $f(\pi_2)$ we can apply Lemma 18 and get

$$((U_1 \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1} \cup X_1, U') \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}}$$

for some U' agreeing with $(U \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1} \cup \delta_{W_2} \cup X_2$ on the surface variables of $f(\pi_2)$. It follows that

$$((U_1 \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1} \cup X_1, (U \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W} \cup X) \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}}$$

for some $X \subseteq \mathbb{P}$; more precisely, $X \subseteq X_1 \cup X_2$. Finally, together with the above

$$(V \cap \mathbb{P}_{\pi_1;\pi_2}, (U_1 \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W_1} \cup X_1) \in \|f(\pi_1)\|_{\mathsf{DL\text{-}PA}}$$

we obtain

$$(V \cap \mathbb{P}_{\pi_1;\pi_2}, (U \cap \mathbb{P}_{\pi_1;\pi_2}) \cup \delta_{W} \cup X,) \in \|f(\pi_1); f(\pi_2)\|_{\mathsf{DL\text{-}PA}}$$

for some $X \subseteq X_1 \cup X_2$.

To prove IH3 suppose $(V, U) \in \|f(\pi_1;\pi_2)\|_{\mathsf{DL\text{-}PA}}$. Then there is a U_1 such that $(V, U_1) \in \|f(\pi_1)\|_{\mathsf{DL\text{-}PA}}$ and $(U_1, U) \in \|f(\pi_2)\|_{\mathsf{DL\text{-}PA}}$. Applying IH3 twice we get

$$(V \cap \mathbb{P}_{\pi_1}, U_1 \cap \mathbb{P}_{\pi_1}, W_1) \in \|\pi_1\|,$$
$$(U_1 \cap \mathbb{P}_{\pi_2}, U \cap \mathbb{P}_{\pi_2}, W_2) \in \|\pi_2\|.$$

for some W_1 and W_2. Adding irrelevant variables as we did for the previous cases we get

$$(V \cap \mathbb{P}_{\pi_1;\pi_2}, U_1 \cap \mathbb{P}_{\pi_1;\pi_2}, W_1) \in \|\pi_1\|,$$
$$(U_1 \cap \mathbb{P}_{\pi_1;\pi_2}, U \cap \mathbb{P}_{\pi_1;\pi_2}, W_2) \in \|\pi_2\|.$$

Putting this together we obtain

$$(V \cap \mathbb{P}_{\pi_1;\pi_2}, U \cap \mathbb{P}_{\pi_1;\pi_2}, W_1 \cup W_2) \in \|\pi_1; \pi_2\|.$$

- The case of exclusive nondeterministic composition is straightforward.
- The case of finite iteration derives from the cases of exclusive nondeterministic composition and of sequential composition.
- The case of parallel composition relies on sequential composition and uses Proposition 12.

This ends the proof. □

3.2.5 Complexity results

We use the translation in order to establish complexity upper bounds. Lower bounds are imported from the fragment DL-PA.

The translation from DL-PPA to DL-PA is a polynomial transformation:

Proposition 22. The length $\ell(f(\varphi))$ of the translation of a DL-PPA formula φ is polynomial in the length $\ell(\varphi)$ of φ.

Theorem 23. The DL-PPA model checking problem of deciding, given V and φ, whether $V \in \|\varphi\|$ is PSPACE-complete.

Proof. The lower bound is due to PSPACE hardness of DL-PA model checking (Herzig, Lorini, Troquard, et al. 2011). The upper bound is obtained by Theorem 21 and Proposition 22 thanks to PSPACE membership of DL-PA (Balbiani, Herzig, Schwarzentruber, et al. 2014). □

To establish the complexity of satisfiability checking we reduce it polynomially to model checking.

Proposition 24. A DL-PPA formula φ is satisfiable iff $\emptyset \in \|\langle ;_{p \in \mathbb{P}_\varphi} (p \leftarrow \top \cup p \leftarrow \bot) \rangle \varphi\|$.

Theorem 25. The DL-PPA satisfiability checking problem is PSPACE-complete.

Proof. The lower bound is due to PSPACE hardness of DL-PA satisfiability checking (Herzig, Lorini, Troquard, et al. 2011). The upper bound follows from PSPACE membership of DL-PPA model checking via the preceding Proposition 24. □

3.3 DL-PPA applied to automated planning

Lets recall the conditions for a set of actions to be consistent. A consistent set of actions has no contradictory effects and no cross-interaction between its actions. A set of actions A is say to be free of contradictory effects in a state V if and only if for any pair of actions $\mathsf{a}_1, \mathsf{a}_2 \in A$ and for every $ce_1 \in \mathit{eff}(\mathsf{a}_1)$ and $ce_2 \in \mathit{eff}(\mathsf{a}_2)$ if $V \in \|ce_1 \wedge ce_2\|$ then $\mathit{eff}^+(ce_1) \cap \mathit{eff}^-(ce_2) = \emptyset$.

We say that two states V and V' *agree* on a formula φ if and only if either $V \in \|\varphi\|$ and $V' \in \|\varphi\|$ or $V \notin \|\varphi\|$ and $V' \notin \|\varphi\|$. Then we say that two different actions a_1 and a_2 have no cross-interaction at V if the following hold:

1. s and $\tau_{\mathsf{a}_1}(s)$ agree on $\mathit{pre}(\mathsf{a}_2)$ and on the condition $\mathit{cnd}(ce_2)$ of every conditional effect $ce_2 \in \mathit{eff}(\mathsf{a}_2)$;

2. s and $\tau_{\mathsf{a}_2}(s)$ agree on $\mathit{pre}(\mathsf{a}_1)$ and on the condition $\mathit{cnd}(ce_1)$ of every conditional effect $ce_1 \in \mathit{eff}(\mathsf{a}_1)$.

In this section, we formally define actions and sequential and parallel planning tasks within our framework DL-PPA. Then we show that a DL-PPA model checking can be used to test the solvability of such tasks.

3.3.1 Sequential planning with conditional effects

We suppose given a set of action names Act. A planning task is a triple $(V_0, \tau, \mathit{Goal})$ where:

- $V_0 \subseteq \mathbb{P}$ is a valuation (the initial state)
- $\tau : \mathit{Act} \times \mathbb{P} \to \mathbb{P}$ is a partial function modeling the actions in Act
- $\mathit{Goal} \in \mathit{Fml}$ is the goal

Each action $\mathsf{a} \in \mathit{Act}$ is described with a tuple $(\mathit{pre}(\mathsf{a}), \mathit{eff}(\mathsf{a}))$ where:

- $\mathit{pre}(\mathsf{a}) \in \mathit{Fml}$ is the precondition of a;

- $\mathit{eff}(\mathsf{a}) \in \mathit{Fml} \times 2^{\mathbb{P}} \times 2^{\mathbb{P}}$ is a set of triples ce of the form

$$(\mathit{cnd}(ce), \mathit{eff}^{+}(ce), \mathit{eff}^{-}(ce)),$$

the conditional effects of a, where $\mathit{cnd}(ce)$ is a DL-PPA formula (the condition) and $\mathit{eff}^{+}(ce)$ and $\mathit{eff}^{-}(ce)$ are sets of variables that are respectively added and deleted by a if condition ce is true.

The function τ can be defined from such action descriptions : : $\tau_{\mathsf{a}}(V)$ is defined (a is executable in state V) if and only if $V \in \|\mathit{pre}(\mathsf{a})\|$ and for all $ce_1, ce_2 \in \mathit{eff}(\mathsf{a})$ such that $V \in \|\mathit{cnd}(ce_1) \wedge \mathit{cnd}(ce_2)\|$: $\mathit{eff}^{+}(ce_1) \cap \mathit{eff}^{-}(ce_2) = \emptyset$. When τ_{a} is defined in V then:

$$\tau_{\mathsf{a}}(V) = \left(V \setminus \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ V \in \|\mathit{cnd}(ce)\|}} \mathit{eff}^{-}(ce) \right) \cup \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}) \\ V \in \|\mathit{cnd}(ce)\|}} \mathit{eff}^{+}(ce)$$

We associate to action a the DL-PPA program $\mathsf{exeAct}(\mathsf{a})$:

$$\mathsf{exeAct}(\mathsf{a}) = \mathit{pre}(\mathsf{a})? \sqcap \bigsqcap_{ce \in \mathit{eff}(\mathsf{a})} \left(\neg \mathit{cnd}(ce)? \cup \begin{pmatrix} \mathit{cnd}(ce)? \\ \sqcap \bigsqcap_{p \in \mathit{eff}^{+}(ce)} p{\leftarrow}\top \\ \sqcap \bigsqcap_{q \in \mathit{eff}^{-}(ce)} q{\leftarrow}\bot \end{pmatrix} \right)$$

The program $\mathsf{exeAct}(\mathsf{a})$ behaves like a:

Lemma 26. For every action $\mathsf{a} \in \mathit{Act}$:

1. τ_{a} is defined in V iff there are U, W such that $(V, U, W) \in \|\mathsf{exeAct}(\mathsf{a})\|$;
2. If τ_{a} is defined in V then $\tau_{\mathsf{a}}(V) = U$ iff $(V, U, W) \in \|\mathsf{exeAct}(\mathsf{a})\|$ for some W.

Proof. Let us take an arbitrary state V. $\tau_{\mathsf{a}}(V)$ is not defined iff: (1) $V \notin \|\mathit{pre}(\mathsf{a})\|$, or (2) there are $ce_1, ce_2 \in \mathit{eff}(\mathsf{a})$ and $p \in \mathbb{P}$ such that $V \in \|\mathit{cnd}(ce_1) \wedge \mathit{cnd}(ce_2)\|$ and $\mathit{eff}^{+}(ce_1) \cap \mathit{eff}^{-}(ce_2)$ contains some p. In case (1), the program fails because $\|\mathit{pre}(\mathsf{a})?\| = \emptyset$, in case (2), the program fails because $\|p{\leftarrow}\bot \sqcap p{\leftarrow}\top\| = \emptyset$.

When $\tau_{\mathsf{a}}(V)$ is defined then $V \in \|\mathit{pre}(\mathsf{a})\|$, so $(V, V, \emptyset) \in \|\mathit{pre}(\mathsf{a})?\|$. Moreover, for each $ce \in \mathit{eff}(\mathsf{a})$ such that $V \in \|\mathit{cnd}(ce)\|$ the programs $(\bigsqcap_{p \in \mathit{eff}^{+}(ce)} p{\leftarrow}\top)$ and $(\bigsqcap_{q \in \mathit{eff}^{-}(ce)} q{\leftarrow}\bot)$ are executed in parallel and all the assignments are consistent (no $p{\leftarrow}\bot$ and $p{\leftarrow}\top$ is executed in parallel). Then the parallel composition of all these programs leads, by definition, to the state

$\tau_{\mathsf{a}}(V) = U$, with $(V, U, W) \in \|\mathsf{exeAct}(\mathsf{a})\|$, where W is the set of all assigned variables in the program $\mathsf{exeAct}(\mathsf{a})$. □

We say that a state V is *reachable by a sequential plan* from a state V_0 via a set of conditional actions Act if there is a *sequential plan*, that is, a sequence of actions $\langle \mathsf{a}_1, \ldots, \mathsf{a}_m \rangle$ from Act, and a sequence of states $\langle V_0, \ldots, V_m \rangle$ with $m \geq 0$ such that $V = V_m$ and $\tau_{\mathsf{a}_k}(V_{k-1}) = V_k$ for every k such that $1 \leq k \leq m$. A planning task is *solvable by a sequential plan* if there is at least one state $V \in \|Goal\|$ such that V is reachable by a sequential plan from V_0 via Act; otherwise it is unsolvable by a sequential plan.

3.3.2 Parallel planning with conditional effects

A set of actions $A = \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}$ determines a partial function τ_A from $2^{\mathbb{P}}$ to $2^{\mathbb{P}}$: $\tau_A(V)$ is defined ($\mathsf{a}_1, \ldots, \mathsf{a}_m$ are executable in parallel in state V) if and only if:

1. for all $\mathsf{a}_i \in A$, τ_{a_i} is defined in V, and
2. the set of actions A is consistent in V.

When τ_A is defined in V then:

$$\tau_A(V) = \left(V \setminus \bigcup_{\substack{\mathsf{a} \in A, ce \in \mathit{eff}(\mathsf{a}), \\ V \in \|cnd(ce)\|}} \mathit{eff}^-(ce) \right) \cup \bigcup_{\substack{\mathsf{a} \in A, ce \in \mathit{eff}(\mathsf{a}), \\ V \in \|cnd(ce)\|}} \mathit{eff}^+(ce)$$

To every set of conditional actions we can associate a DL-PPA program that behaves exactly like the parallel execution of its elements. Given $A = \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}$, let $\mathsf{exeAct}(A)$ be the DL-PPA program

$$\mathsf{exeAct}(A) = \bigsqcap_{\mathsf{a} \in A} \left(\begin{array}{l} \mathsf{exeAct}(\mathsf{a}) \\ \sqcap \bigsqcap_{\substack{\mathsf{a}' \in A \\ \mathsf{a} \neq \mathsf{a}'}} \left(\begin{array}{l} \langle \mathsf{exeAct}(\mathsf{a}') \rangle \mathit{pre}(\mathsf{a})? \\ \sqcap \bigsqcap_{ce \in \mathit{eff}(\mathsf{a})} \left(\mathit{cnd}(ce) \leftrightarrow \langle \mathsf{exeAct}(\mathsf{a}') \rangle \mathit{cnd}(ce) \right)? \end{array} \right) \end{array} \right)$$

Lemma 27. For every finite set of actions $A = \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}$,

1. τ_A is defined in V iff there are U, W such that $(V, U, W) \in \|\mathsf{exeAct}(A)\|$;

2. If τ_A is defined in V then $\tau_A(V) = U$ iff $(V, U, W) \in \|\mathsf{exeAct}(A)\|$ for some W.

Proof. Let us take an arbitrary state V and two actions $\mathsf{a}, \mathsf{a}' \in A$. By Lemma 26, for all $i \in \{1, \ldots, m\}$ we know that $\mathsf{exeAct}(\mathsf{a}_i)$ behaves like a_i, and then is executable if $\tau_{\mathsf{a}_i}(V)$ is defined.

$\tau_A(V)$ is not defined if and only if; (1) $\tau_{\mathsf{a}}(V)$ is not defined; (2) There are $ce \in \mathit{eff}(\mathsf{a})$ and $ce' \in \mathit{eff}(\mathsf{a}')$ and $p \in \mathbb{P}$ such that $V \in \|\mathit{cnd}(ce) \wedge \mathit{cnd}(ce')\|$ and $p \in \mathit{eff}^+(ce) \cap \mathit{eff}^-(ce')$; (3) $\tau_{\mathsf{a}'}(V) \notin \|\mathit{pre}(\mathsf{a})\|$; (4) there is a conditional effect $ce = \mathit{cnd}(\mathsf{a})$ such that V and $\tau_{\mathsf{a}'}(V)$ do not agree on ce.

Case (2) happens if and only if $\mathsf{exeAct}(\mathsf{a}) \sqcap \mathsf{exeAct}(\mathsf{a}')$ fails because of the execution of the parallel composition of $p{\leftarrow}\bot$ and $p{\leftarrow}\top$; case (3) happens if and only if then the execution of $\langle \mathsf{exeAct}(\mathsf{a}') \rangle \mathit{pre}(\mathsf{a})?$ fails; case (4) happens if and only if the execution of $(\mathit{cnd}(ce) \leftrightarrow \langle \mathsf{exeAct}(\mathsf{a}') \rangle \mathit{cnd}(ce))?$ fails.

When τ_A is defined,

$$\sqcap \prod_{\substack{\mathsf{a}' \in A \\ \mathsf{a} \neq \mathsf{a}'}} \begin{pmatrix} \langle \mathsf{exeAct}(\mathsf{a}') \rangle \mathit{pre}(\mathsf{a})? \\ \sqcap \bigsqcap_{ce \in \mathit{eff}(\mathsf{a})} (\mathit{cnd}(ce) \leftrightarrow \langle \mathsf{exeAct}(\mathsf{a}') \rangle \mathit{cnd}(ce))? \end{pmatrix}$$

is defined. Then, the parallel composition of $\mathsf{exeAct}(\mathsf{a})(V)$ for every action $\mathsf{a} \in A$ leads, by definition, to the state $\tau_A(V) = U$, with $(V, U, W) = \|\mathsf{exeAct}(A)\|$, where W is the set of all assigned variables in the program $\mathsf{exeAct}(A)$. □

We say that a state V is *reachable by a parallel plan* from a state V_0 via a set of actions Act if there is a *parallel plan*, that is, a sequence $\langle A_1, \ldots, A_m \rangle$ of sets of actions $A_i \subseteq Act$ and a sequence of states $\langle V_0, \ldots, V_m \rangle$ with $m \geq 0$ such that $V = V_m$ and $\tau_{A_k}(V_{k-1}) = V_k$ for every k such that $1 \leq k \leq m$. A planning task $(Act, V_0, Goal)$ is *solvable by a parallel plan* if there is at least one state $V \in \|Goal\|$ such that V is reachable by a parallel plan from V_0 via Act; otherwise it is unsolvable by a parallel plan.

3.3.3 Solvability of bounded horizon planning tasks

Now that we have defined a parallel encoding of actions and the solvability of a planning task, we can capture the solvability of a planning task in DL-PPA

with a bounded horizon k. This problem is known to be PSPACE-complete (Bylander 1994).

Theorem 28. A planning task $(Act, V_0, Goal)$ is solvable by a sequential plan with no more than k actions if and only if:

$$V_0 \in \left\| \left\langle \left(\bigcup_{\mathsf{a} \in Act} \mathsf{exeAct}(\mathsf{a}) \right)^{\leq k} \right\rangle Goal \right\|$$

Proof. Our formula reads "there exists an execution of $(\bigcup_{\mathsf{a} \in Act} \mathsf{exeAct}(\mathsf{a}))^{\leq k}$ after which *Goal* is true." We know by Lemma 26 that $\mathsf{exeAct}(\mathsf{a})$ behaves correctly and produces the same effects as action a. The program $(\bigcup_{\mathsf{a} \in Act} \mathsf{exeAct}(\mathsf{a}))^{\leq k}$ non-deterministically chooses an action a from Act and executes the corresponding program $\mathsf{exeAct}(\mathsf{a})$, then repeats this a number of times less or equal than k. This produces a sequence of at most k actions, i.e., a sequential plan bounded by k. Therefore the formula is satisfied in the initial state if and only if there exists a sequential plan of length bounded by k after which the goal is satisfied, i.e., if and only if the planning task is solvable with a sequence of at most k actions. □

Theorem 29. A planning task $(Act, V_0, Goal)$ is solvable by a parallel plan with no more than k steps if and only if:

$$V_0 \in \left\| \left\langle \left(\bigsqcap_{\mathsf{a} \in Act} exe_{\mathsf{a}} \leftarrow \bot ; \bigsqcup_{\mathsf{a} \in Act} exe_{\mathsf{a}} \leftarrow \top ; \pi_{exe} \right)^{\leq k} \right\rangle Goal \right\|$$

where $exe_{\mathsf{a}} \notin \mathbb{P}_{\mathsf{exeAct}(\mathsf{a})}$ for all $\mathsf{a} \in Act$, and

$$\pi_{exe} = \bigsqcap_{\mathsf{a} \in Act} \left(\begin{array}{l} \neg exe_{\mathsf{a}}? \cup \\ \left(\begin{array}{l} exe_{\mathsf{a}}? \sqcap \mathsf{exeAct}(\mathsf{a}) \\ \sqcap \bigsqcap_{\substack{\mathsf{a}' \in Act \\ \mathsf{a} \neq \mathsf{a}'}} \left(\begin{array}{l} \neg exe_{\mathsf{a}'}? \\ \cup \left(\begin{array}{l} exe_{\mathsf{a}'}? \sqcap \langle \mathsf{exeAct}(\mathsf{a}') \rangle pre(\mathsf{a})? \\ \sqcap \bigsqcap_{ce \in \mathit{eff}(\mathsf{a})} \left(cnd(ce) \leftrightarrow \langle \mathsf{exeAct}(\mathsf{a}') \rangle cnd(ce) \right)? \end{array} \right) \end{array} \right) \end{array} \right) \end{array} \right)$$

Proof. The program $\bigsqcap_{\mathsf{a} \in Act} exe_{\mathsf{a}} \leftarrow \bot$ initialises a special fresh variable $exe_{\mathsf{a}} \notin$

$\mathbb{P}_{\mathsf{exeAct(a)}}$ to $\bot$, for each action $\mathsf{a} \in Act$. Then the inclusive nondeterministic composition $\bigsqcup_{\mathsf{a}\in Act} exe_{\mathsf{a}} \leftarrow \top$ chooses some non empty subset of actions $A \subseteq Act$ and executes the program $\bigsqcap_{\mathsf{a}\in A} exe_{\mathsf{a}} \leftarrow \top$. At this point, $exe_{\mathsf{a}} = \top$ iff $\mathsf{a} \in A$, and the program π_{exe} is executed. It is easily seen that, for a given choosen set of actions A, π_{exe} behaves like the program $\mathsf{exeAct}(A)$. We know by Lemma 27 that this latter program behaves correctly and produces the same effect as the parallel execution of all actions in A. The sequence $\bigsqcap_{\mathsf{a}\in Act} exe_{\mathsf{a}} \leftarrow \bot; \bigsqcup_{\mathsf{a}\in Act} exe_{\mathsf{a}} \leftarrow \top; \pi_{exe}$ is then repeated a number of times less or equal than k. This produces a sequence of at most k parallel executions of action sets, i.e., a parallel plan bounded by k. Therefore the formula is satisfied in the initial state if and only if there exists a parallel plan of length bounded by k after which the goal is satisfied, i.e., if and only if the planning task is solvable with a sequence of at most k parallel steps. □

The nondeterministic choice of a subset of the set of actions to be executed at a step could be expressed by the program $\bigcup_{A\subseteq Act} \mathsf{exeAct}(A)$. However, this would have length exponential in the number of actions.

Both in Theorem 28 and in Theorem 29, the size of the model checking problems is polynomial in the size of the planning task plus $\log k$. This relies on our compact representation of bounded programs $\pi^{\leq k}$, cf. Proposition 11.

3.4 Conclusion

We have shown how to capture parallel classical planning in an extension of DL-PA by parallel and inclusive nondeterministic composition whose model checking and satisfiability checking problems are still in PSPACE. This allows in particular to decide, within the complexity boundaries, the existence of a plan given a finite horizon.

A straightforward continuation of our work is towards parallel epistemic planning: We can take over the epistemic extension of DL-PA in terms of observational variables that was applied to sequential epistemic planning in (Cooper, Herzig, Maffre, Maris, and Régnier 2016a).

4 Making Parallel Classical Planning Epistemic

Epistemic planning is important in multiagent systems. None of these approaches investigated up to now studied the construction of parallel epistemic plans. Such plans are however particularly interesting when there is more than one agent. In this chapter we investigate how multiple agents can act in parallel in order to achieve a common goal.

In the previous chapter we used this framework in order to solve classical planning tasks with parallel plans. In that work we supposed that agents always have perfect knowledge of the current state and of the occurrence of actions. We here relax this hypothesis: an agent may fail to observe the truth value of a given propositional variable and may fail to observe some actions. Moreover, they may not know other agents' observational capabilities. The object of these capabilities can be a propositional variables, but also the other agents' visibility; in other words, we consider higher-order visibility information.

We here apply EL-O to parallel epistemic planning: we provide a reduction of EL-O-based parallel planning to classical planning, which allows us to translate planning tasks into PDDL and use classical planners. We illustrate our approach with a parallel version of the epistemic gossip problem (Cooper, Herzig, Maffre, Maris, and Régnier 2019) where n agents initially each know their secret but not the others'; agents can exchange all secrets they know during a phone call to another agent; and the goal is to achieve shared knowledge of all secrets, i.e., everybody knows every secret. A parallel solution to the epistemic gossip problem is a sequence of sets of calls.

The content of this chapter corresponds to an article published at the International Conference on Principles of Knowledge Representation and Reasoning (KR 2020) (Cooper, Herzig, Maris, Perrotin, et al. 2020).

Content of the chapter

Résumé du chapitre en français

La planification épistémique est importante dans les systèmes multi-agents. Aucune de ces approches étudiées jusqu'à présent n'a étudié la construction de plans épistémiques parallèles. De tels plans sont pourtant particulièrement intéressants lorsqu'il y a plus d'un agent. Dans ce chapitre, nous étudions comment plusieurs agents peuvent agir en parallèle afin d'atteindre un objectif commun.

Dans le chapitre précédent, nous avons utilisé ce cadre pour résoudre des tâches de planification classiques avec des plans parallèles. Dans ce travail, nous avons supposé que les agents ont toujours une connaissance parfaite de l'état actuel et de l'occurrence des actions. Nous relaxons ici cette hypothèse : un agent peut ne pas observer la valeur de vérité d'une variable propositionnelle

donnée et peut ne pas observer certaines actions. De plus, il peut ne pas connaître les capacités d'observation des autres agents. L'objet de ces capacités peut être une variable propositionnelle, mais aussi la visibilité des autres agents ; en d'autres termes, nous considérons une information de visibilité d'ordre supérieur.

Nous appliquons ici EL-O à la planification épistémique parallèle : nous fournissons une réduction de la planification parallèle basée sur EL-O à la planification classique, ce qui nous permet de traduire les tâches de planification en PDDL et d'utiliser des planificateurs classiques. Nous illustrons notre approche avec une version parallèle du problème de bavardage épistémique (Cooper, Herzig, Maffre, Maris et Régnier 2019) où n agents connaissent initialement chacun leur secret mais pas celui des autres ; les agents peuvent échanger tous les secrets qu'ils connaissent lors d'un appel téléphonique à un autre agent ; et l'objectif est de parvenir à une connaissance partagée de tous les secrets, c'est-à-dire que tout le monde connaît tous les secrets. Une solution parallèle au problème du bavardage épistémique est une séquence d'ensembles d'appels.

De la même manière que pour la planification classique parallèle du chapitre précédent il nous faut respecter l'absence d'effets contradictoires et d'interaction croisée. L'ajout de raisonnement sur les connaissances, et plus particulièrement de l'opérateur de visibilité jointe $\mathbf{JS}$, impose de revoir ces règles. Nous faisons cela avec une expansion des descriptions d'actions et des tâches de planification. Cette expansion se fait en ajoutant les causes et les conséquences des fluents de la tâche de planification. La relation de conséquence est définie comme suit. α intérêtest la cause de β – et β la conséquence de α – si $\alpha = \beta$ ou si $\alpha = \mathbf{JS}\,\alpha'$ et que β est une suite d'opérateurs de visiblité se terminant par α'. En incluant aux ensembles (effets et état initial) les conséquences de leurs atomes et aux formules (préconditions, conditions et but) les causes de leurs atomes, nous pouvons appliquer les mêmes règles qu'en planification classique pour les contraintes liées à l'exécution en parallèle des actions. En restreignant l'expansion des conséquences aux fluents pertinents pour la tâche de planification que l'on souhaite résoudre – ceux présents dans la description de la tâche avant l'expansion – la complexité du problème d'existence d'un plan parallèle à un problème de planification épistémique est PSPACE-complète.

Ce résultat nous permet de considérer l'utilisation de planificateurs pour la planification classique pour résoudre des problèmes épistémiques. Nous avons

réalisé un test de performance (*benchmark*) pour évaluer l'intérêt pratique d'une telle approche. Les résultats obtenus sont difficilement exploitables, la plupart des problèmes testés deviennent rapidement trop complexes lorsque l'on augmente leur taille. Cependant nous pensons que de futurs travaux utilisant cette approche pourraient permettre une résolution de problèmes de planification épistémique avec une efficacité proche de celle de la planification classique.

4.1 EL-O: Epistemic Logic of Observation

We recall the Epistemic Logic of Observation, abbreviated EL-O.

4.1.1 Atoms and Introspective Atoms

Let $\mathbb{P}$ be a countable set of propositional variables and let Agt be a finite set of agents. The set of *observability operators* is

$$OBS = \{\mathbf{S}_i \ : \ i \in Agt\} \cup \{\mathbf{JS}\},$$

where $\mathbf{S}_i$ stands for individual visibility of agent i and $\mathbf{JS}$ stands for joint visibility of all agents. The set of all sequences of visibility operators of length at most k is noted $OBS^{\leq k}$. Then $OBS^* = \bigcup_{k\geq 0} OBS^{\leq k}$ and $OBS^+ = \bigcup_{k\geq 1} OBS^{\leq k}$. Atoms are finite sequences of visibility operators followed by a propositional variable. The set of all atoms is

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

We use α, α', β,...to denote atoms.

The set of all introspective atoms is

$$\begin{aligned} \textit{I-Atm} = \{\sigma\, \mathbf{S}_i\, \mathbf{S}_i\, \alpha \ : \ \sigma \in OBS^* \text{ and } \alpha \in ATM\} \cup \\ \{\sigma\, \mathbf{JS}\, \alpha \ : \ \sigma \in OBS^+ \text{ and } \alpha \in ATM\}. \end{aligned}$$

The complement of *I-Atm* is the set of relevant atoms: $\textit{R-Atm} = ATM \setminus \textit{I-Atm}$.

4.1.2 Atomic Consequence

We define a relation of *atomic consequence* between visibility atoms as follows:

$\alpha \Rightarrow \beta$ if either $\alpha = \beta$, or $\alpha = \mathbf{JS}\, \alpha'$ and $\beta = \sigma\, \alpha'$ for some $\sigma \in OBS^+$.

For example, $\mathbf{JS}\, p \Rightarrow \mathbf{S}_i\, p$ and $\mathbf{JS}\, p \Rightarrow \mathbf{JS}\, \mathbf{S}_i\, p$. The relation $\Rightarrow$ is reflexive and transitive. When $\alpha \Rightarrow \beta$, we say that α is a *cause* of β and that β is a *consequence* of α. We will ensure that atomic consequences are valid implications. We note $\alpha^{\Leftarrow}$ the set of causes of α, and $\alpha^{\Rightarrow}$ the set of its consequences. Clearly, $(p)^{\Rightarrow} = (p)^{\Leftarrow} = \{p\}$ for $p \in \mathbb{P}$. Moreover, $(\mathbf{S}_i\, p)^{\Rightarrow} = \{\mathbf{S}_i\, p\}$, $(\mathbf{S}_i\, p)^{\Leftarrow} = \{\mathbf{S}_i\, p, \mathbf{JS}\, p\}$, $(\mathbf{JS}\, p)^{\Rightarrow} = \{\sigma\, p \ : \ \sigma \in OBS^+\}$, and $(\mathbf{JS}\, p)^{\Leftarrow} = \{\mathbf{JS}\, p\}$. Observe that $\alpha^{\Leftarrow}$ is always finite while $\alpha^{\Rightarrow}$ is either infinite (namely

when α starts by **JS**) or the singleton $\{\alpha\}$ (namely when α is a propositional variable or starts by some $\mathbf{S}_i$). Our formulation uses a generalisation of atomic consequence to sets of atoms $s \subseteq ATM$: $s^{\Rightarrow} = \bigcup_{\alpha \in s} \alpha^{\Rightarrow}$.

Proposition 30. The following hold for every $A, B \subseteq ATM$:

1. $(A \cup I\text{-}Atm)^{\Rightarrow} = A^{\Rightarrow} \cup I\text{-}Atm$;
2. $A^{\Rightarrow} \cap B^{\Leftarrow} = \emptyset$ iff $A^{\Rightarrow} \cap B = \emptyset$ iff $A \cap B^{\Leftarrow} = \emptyset$.

4.1.3 Language of EL-O

The language of EL-O is defined by the following grammar:

$$\varphi ::= \alpha \mid \bot \mid \neg\varphi \mid (\varphi \wedge \varphi)$$

where α ranges over ATM.

The *length* of a formula is defined recursively by:

$$\begin{aligned} \ell(\bot) &= 1 \\ \ell(p) &= 1 \\ \ell(\sigma\, p) &= \ell(\sigma) + 1 \\ \ell(\neg\varphi) &= \ell(\varphi) + 1 \\ \ell(\varphi \wedge \varphi') &= \ell(\varphi) + \ell(\varphi') + 1 \end{aligned}$$

where $\ell(\sigma)$ is the length of the finite sequence σ. For example, $\ell(\mathbf{S}_1\, \mathbf{S}_2\, p) = 3$ and $\ell(\mathbf{S}_2\, p \wedge \neg p \wedge \mathbf{JS}\, q) = 8$. If $\beta \Rightarrow \alpha$ then the length of β is less than or equal to the length of α. Moreover, the set of causes of α has at most $\ell(\alpha)$ elements: $|\alpha^{\Leftarrow}| \leq \ell(\alpha)$. It follows that the sum of the lengths of all causes of α is at most quadratic in the length of α:

Proposition 31. For every α, $\sum_{\beta\,:\,\beta \Rightarrow \alpha} \ell(\beta) \leq (\ell(\alpha))^2$.

Example 2. *In the generalised gossip problem, for $Agt = \{1, \ldots, n\}$ the set of secrets is $\mathbb{P} = \{sec_i \;:\; i \in Agt\}$. The goal is to obtain shared knowledge of depth k:*

$$Goal^{G_k} = \bigwedge_{i \in Agt} \bigwedge_{\sigma \in OBS^+, \ell(\sigma) \leq k} \sigma\, sec_i.$$

$s \models \alpha$	if	$\alpha \in s^{\Rightarrow} \cup I\text{-}Atm$
$s \models \neg\varphi$	if	$s \not\models \varphi$
$s \models \varphi \wedge \psi$	if	$s \models \varphi$ and $s \models \psi$

Table 4.1: Interpretation of formulas

Hence $Goal^{G_1} = \bigwedge_{i \in Agt} \bigwedge_{j \in Agt} \mathbf{S}_j\, sec_i$.

4.1.4 Semantics of EL-O

A *state* is a subset of the set of atoms ATM. We denote states by s, s', etc. The set of all states is $STATES = 2^{ATM}$.

The set of *relevant states* is $R\text{-}STATES = 2^{R\text{-}Atm}$.

A way of guaranteeing introspection was proposed in (Herzig, Lorini, and Maffre 2015) where formulas are interpreted exclusively in *introspectively closed states*: states that contain all introspective atoms and are closed under $\Rightarrow$, i.e., sets of atoms that equal $s \cup I\text{-}Atm^{\Rightarrow}$ for some state $s \subseteq ATM$. Such introspective states being always infinite, it is not clear how to define model checking, which requires finite states. Here we work with finite models and interpret formulas in such a way that introspection is simulated.

The truth conditions for EL-O formulas are in Table 4.1. The condition for atoms is the only non-standard one: α is true in state s if α is introspective or $\beta \Rightarrow \alpha$ for some $\beta \in s$.

Example 3. *In the initial gossiping state (in which secrets may or may not be true) every agent only knows her own secret. Therefore* $s_0^{G_1} = \{\mathbf{S}_i\, sec_i \ : \ i \in Agt\} \cup A_1$ *where* A_1 *is some subset of* $\{sec_i \ : \ i \in Agt\}$. *Then* $s_0^{G_1} \models \mathbf{S}_i\, sec_i$ *and* $s_0^{G_1} \models \bigwedge_{j \neq i} \neg \mathbf{S}_i\, sec_j$ *for every* $i \in Agt$. *Although* $s_0^{G_1}$ *does not contain* $\mathbf{S}_i\, \mathbf{S}_i\, sec_j$ *we have* $s_0^{G_1} \models \mathbf{S}_i\, \mathbf{S}_i\, sec_j$.

Given a set of states $St \subseteq STATES$, we say that a formula φ is *valid in* St if $s \models \varphi$ for every $s \in St$; when $s \models \varphi$ for some $s \in St$ then we say that φ is *satisfiable in* St. Clearly, an atom α is valid in the set of all states $STATES$ if and only if it is introspective. Moreover, atomic consequences are valid in $STATES$: if $\alpha \Rightarrow \beta$ then $\alpha \rightarrow \beta$ is valid in $STATES$.

Remark 1. When Agt is a singleton then $\mathbf{S}_i\, p \wedge \neg \mathbf{JS}\, p$ is satisfiable. While this anomaly could be taken care of by a modification of the semantics, we do not

do so for the sake of readability and content ourselves with the observation that the **JS** operator is superfluous when there is only one agent.

Proposition 32. For every $\varphi \in Fml_{\mathsf{EL\text{-}O}}$ there is a $\varphi' \in R\text{-}Fml_{\mathsf{EL\text{-}O}}$ such that $\varphi \leftrightarrow \varphi'$ is valid in $STATES$. Moreover, for every $s \in STATES$, there is a $s' \in R\text{-}STATES$ such that $s \models \varphi$ iff $s' \models \varphi$ for every $\varphi \in Fml_{\mathsf{EL\text{-}O}}$.

Classical semantics for $Fml_{\mathsf{EL\text{-}O}}$ is recovered by changing the truth condition for atoms: $s \models^{\mathsf{CPC}} \alpha$ if $\alpha \in s$.

Proposition 33. For $\varphi \in R\text{-}Fml_{\mathsf{EL\text{-}O}}$ and $s \in R\text{-}STATES$, $s \models \varphi$ iff $s^{\Rightarrow} \cap ATM(\varphi) \models \varphi$ iff $s^{\Rightarrow} \cap ATM(\varphi) \models^{\mathsf{CPC}} \varphi$.

4.2 Action Descriptions and Simple Epistemic Planning tasks

We assume that actions are deterministic and have conditional effects that are described by add- and delete-lists. Such effects are crucial in epistemic planning: when an agent performs an action then the effects on another agent's epistemic state typically depend on whether that agent sees the variables that are modified by the action.

4.2.1 Action Descriptions

An *action description* is a pair $\mathsf{a} = (pre(\mathsf{a}), \mathit{eff}(\mathsf{a}))$ where $pre(\mathsf{a})$ is a relevant formula from $R\text{-}Fml_{\mathsf{EL\text{-}O}}$ (the *precondition* of a) and

$$\mathit{eff}(\mathsf{a}) \subseteq R\text{-}Fml_{\mathsf{EL\text{-}O}} \times 2^{R\text{-}Atm} \times 2^{R\text{-}Atm}$$

is the set of *conditional effects* of a, describing which atoms the action may add or remove from the current state under additional conditions. For a triple

$$ce = (cnd(ce), \mathit{ceff}^{+}(ce), \mathit{ceff}^{-}(ce))$$

in $\mathit{eff}(\mathsf{a})$, $cnd(ce)$ is the condition of ce, $\mathit{ceff}^{+}(ce)$ are the added atoms, and $\mathit{ceff}^{-}(ce)$ are the deleted atoms. We require effects to be *consistent*: we suppose that for every $ce_1, ce_2 \in \mathit{eff}(\mathsf{a})$, if $\mathit{ceff}^{+}(ce_1) \cap (\mathit{ceff}^{-}(ce_2))^{\Leftarrow} \neq \emptyset$ then $pre(\mathsf{a}) \wedge cnd(ce_1) \wedge cnd(ce_2)$ is unsatisfiable in $\mathsf{EL\text{-}O}$. That is, we exclude actions with conditional effects $ce_1, ce_2 \in \mathit{eff}(\mathsf{a})$ and $\alpha_1 \in \mathit{ceff}^{+}(ce_1)$ and $\alpha_2 \in \mathit{ceff}^{-}(ce_2)$ such that $\alpha_1 \Rightarrow \alpha_2$. In other words, when $pre(\mathsf{a})$ and their

triggering conditions $cnd(ce_1)$ and $cnd(ce_2)$ are jointly satisfiable then two conditional effects of a cannot conflict. This in particular forbids conditional effects $ce \in \mathit{eff}(\mathsf{a})$ with $\mathit{ceff}^+(ce) \cap \mathit{ceff}^-(ce) \neq \emptyset$ and $\mathit{pre}(\mathsf{a}) \wedge \mathit{cnd}(ce)$ satisfiable.

We disregard introspective atoms in the definition of actions because they are true at every state: adding or deleting them from a state does not change what is true in that state.

Example 4. *In the original gossip problem G_1 where the goal is to obtain shared knowledge of depth 1, $\mathsf{Call}_{i,j} = (\mathit{pre}(\mathsf{Call}_{i,j}), \mathit{eff}(\mathsf{Call}_{i,j}))$ with $\mathit{pre}(\mathsf{Call}_{i,j}) = \top$ and*

$$\begin{aligned}\mathit{eff}(\mathsf{Call}_{i,j}) = \{&(\mathbf{S}_i\, sec_1, \{\mathbf{S}_j\, sec_1\}, \emptyset), (\mathbf{S}_j\, sec_1, \{\mathbf{S}_i\, sec_1\}, \emptyset),\\ &\ldots,\\ &(\mathbf{S}_i\, sec_n, \{\mathbf{S}_j\, sec_n\}, \emptyset), (\mathbf{S}_j\, sec_n, \{\mathbf{S}_i\, sec_n\}, \emptyset)\}.\end{aligned}$$

That is, a call has two conditional effects per secret: if i sees a secret then that secret becomes visible to j, and vice versa.

Example 5. *In the generalised gossip problem G_k the precondition is $\mathit{pre}(\mathsf{Call}_{i,j}) = \top$ as before, and for every $0 \leq m < k$, $\sigma_m \in OBS^{\leq m}$ and $r \in Agt$ there is a conditional effect $ce \in \mathit{eff}(\mathsf{Call}_{i,j})$ of the form:*

$$\begin{aligned}\mathit{cnd}(ce) &= \mathbf{S}_i\, \sigma_m sec_r \vee \mathbf{S}_j\, \sigma_m sec_r,\\ \mathit{ceff}^+(ce) &= \{\sigma\, \mathbf{S}_i\, \sigma_m sec_r \;:\; \sigma \in \{\mathbf{S}_i, \mathbf{S}_j\}^{\leq k-m-1}\} \cup\\ &\qquad \{\sigma\, \mathbf{S}_j\, \sigma_m sec_r \;:\; \sigma \in \{\mathbf{S}_i, \mathbf{S}_j\}^{\leq k-m-1}\}\\ &= \{\sigma\, \sigma_m sec_r \;:\; \sigma \in \{\mathbf{S}_i, \mathbf{S}_j\}^{\leq k-m}\},\\ \mathit{ceff}^-(ce) &= \emptyset,\end{aligned}$$

where $\{\mathbf{S}_i, \mathbf{S}_j\}^{\leq k-m}$ denotes the set all sequences of observability operators $\mathbf{S}_i$ and $\mathbf{S}_j$ of length at most $k-m$. Hence a call achieves common knowledge of i and j up to level k of all secrets one of them knows. The set of all actions is $Act^{G_k} = \{\mathsf{Call}_{i,j} \;:\; i,j \in Agt, i \neq j\}$. All $\mathsf{Call}_{i,j}$ satisfy our consistency condition because they have no negative effects, which makes conflicts impossible.

The *length* of an action description is

$$\ell(\mathsf{a}) = \ell(pre(\mathsf{a})) + \sum_{ce \in eff(\mathsf{a})} \begin{pmatrix} \ell(cnd(ce)) \\ +\Big(\sum_{\alpha \in ceff^{+}(ce)} \ell(\alpha)\Big) \\ +\Big(\sum_{\alpha \in ceff^{-}(ce)} \ell(\alpha)\Big) \end{pmatrix}.$$

4.2.2 Simple Epistemic Planning Tasks

A *simple epistemic planning task* is a triple

$$\mathcal{P} = (s_0, \tau, Goal)$$

where

- $s_0 \in R\text{-}STATES = 2^{R\text{-}Atm}$ is a finite state (the initial state) and
- $\tau : Act \times R\text{-}STATES \to R\text{-}STATES$ is a partial function modeling the actions in Act which is a finite set of consistent actions, and
- $Goal \in R\text{-}Fml_{\mathsf{EL\text{-}O}}$ is an $\mathsf{EL\text{-}O}$ formula without introspective atoms.

(We again disregard introspective atoms as they have no effect on the truth of a formula.)

Example 6. *The planning task that corresponds to the original gossip problem is* $G_1 = (s_0^{G_1}, Act^{G_1}, Goal^{G_1})$ *with*

- $Act^{G_1} = \{\mathsf{Call}_{i,j} \ : \ i, j \in Agt \text{ and } i \neq j\}$ *(cf. Example 4),*
- $s_0^{G_1} = \{\mathbf{S}_i \, sec_i \ : \ i \in Agt\} \cup A_1$ *for some* $A_1 \subseteq \{sec_i \ : \ i \in Agt\}$,
- $Goal^{G_1} = \bigwedge_{i \in Agt} \bigwedge_{j \in Agt} \mathbf{S}_j \, sec_i$.

The set of atoms of a simple epistemic planning task is

$$ATM(\mathcal{P}) = \left(\bigcup_{\mathsf{a} \in Act} ATM(\mathsf{a}) \right) \cup s_0 \cup ATM(Goal)$$

and its length is $\ell(\mathcal{P}) = \ell(s_0) + \ell(Goal) + \sum_{\mathsf{a} \in Act} \ell(\mathsf{a})$.

Solutions to simple epistemic planning tasks can be either sequential plans or parallel plans. We focus on the latter in the rest of the chapter.

4.3 Parallel Epistemic Planning with EL-O

A parallel plan is a sequence of *steps* each of which is a set of actions that are executed simultaneously. Actions in a step should not conflict: we start by determining the conditions of parallel executability of a set of actions in a state, following the ∀-step semantics and the notion of interference in a state of (Rintanen, Heljanko, and Niemelä 2006).

4.3.1 Semantics of a Single Action

We define the semantics of an action a in terms of a partial function τ_{a} on relevant states. The function τ_{a} is defined at s if $s \models \mathit{pre}(\mathsf{a})$. In that case we say that a is executable at s and stipulate:

$$\tau_{\mathsf{a}}(s) = \left(s \setminus \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}),\\ s \models \mathit{cnd}(ce)}} (\mathit{ceff}^{-}(ce))^{\Leftarrow} \right) \cup \bigcup_{\substack{ce \in \mathit{eff}(\mathsf{a}),\\ s \models \mathit{cnd}(ce)}} (\mathit{ceff}^{+}(ce))^{\Rightarrow}.$$

That is, if the precondition of a is satisfied then a removes negative effects of all those conditional effects *ce* that 'fire', i.e., whose triggering conditions are satisfied, plus their causes; and it adds the positive effects of *ce* plus their consequences.

4.3.2 Semantics of a Consistent Set of Actions

We use the same definition of a consistent set of action as before. A set of actions A is say to be consistent in a state s if there is no pair of actions in A with contradictory effects or cross-interaction.

Example 7. *Some set of gossiping calls* $\mathsf{Call}_{i,j}$ *can be consistent in any state. Therefore conference calls* $\{\mathsf{Call}_{i,j}, \mathsf{Call}_{i,r}\}$ *where i calls j and r at the same time can be consistent, making the parallel gossiping task solvable in fewer steps.*

One way to exclude conference calls is to replace $\mathsf{Call}_{i,j}$ *by* $\mathsf{Startcall}^{i}_{j}$ *plus a*

single Endcalls *action as follows:*

$$\begin{aligned} pre(\mathsf{Startcall}^i_j) &= free_i \wedge free_j, \\ eff(\mathsf{Startcall}^i_j) &= eff(\mathsf{Call}_{i,j}) \cup \{(\top, \emptyset, \{free_i, free_j\})\}, \\ pre(\mathsf{Endcalls}) &= \top, \\ eff(\mathsf{Endcalls}) &= \{(\top, \{free_i \ : \ i \in Agt\}, \emptyset)\}, \end{aligned}$$

and to add all $free_i$ to the initial state. Then there is no state where a set of actions with conference calls is consistent: $\mathsf{Startcall}^i_j$ *and* $\mathsf{Startcall}^i_r$ *have cross interaction at any state satisfying $free_i \wedge free_j \wedge free_r$.*

While this solution is natural (agents cannot call two agents at a time because they are no longer available once they have begun a call), splitting calls into two separate actions artificially doubles the number of steps in an optimal solution. Another possibility that avoids the Endcalls *action is to replace all* $\mathsf{Call}_{i,j}$ *by* Tcall^i_j*, with:*

$$\begin{aligned} pre(\mathsf{Tcall}^i_j) &= \top, \\ eff(\mathsf{Tcall}^i_j) &= eff(\mathsf{Call}_{i,j}) \cup \\ &\quad \{(tg_i, \emptyset, \{tg_i\})\} \cup \{(\neg tg_i, \{tg_i\}, \emptyset)\} \cup \\ &\quad \{(tg_j, \emptyset, \{tg_j\})\} \cup \{(\neg tg_j, \{tg_j\}, \emptyset)\}. \end{aligned}$$

Here any two calls involving i each toggles the value of tg_i, which makes that these calls have cross interaction at any state satisfying their preconditions.

A set of actions $A = \{\mathsf{a}_1, \ldots, \mathsf{a}_m\}$ determines a partial function τ_A from $R\text{-}STATES^{\Rightarrow}$ to $R\text{-}STATES^{\Rightarrow}$, where $R\text{-}STATES^{\Rightarrow} = \{s^{\Rightarrow} \ : \ s \in R\text{-}STATES\}$. The function τ_A is defined at s if every $\mathsf{a}_i \in A$ is executable at s and A is consistent in s. When τ_A is defined at s then:

$$\tau_A(s) = \left(s \setminus \left(\bigcup_{\mathsf{a} \in A, ce \in eff(\mathsf{a}), s \models cnd(ce)} (ceff^-(ce))^{\Leftarrow} \right) \right) \cup \left(\bigcup_{\mathsf{a} \in A, ce \in eff(\mathsf{a}), s \models cnd(ce)} (ceff^+(ce))^{\Rightarrow} \right).$$

When a_1 and a_2 are consistent in s then they can be interleaved arbitrarily: we have $\tau_{\{\mathsf{a}_1,\mathsf{a}_2\}}(s) = \tau_{\mathsf{a}_2}(\tau_{\mathsf{a}_1}(s)) = \tau_{\mathsf{a}_1}(\tau_{\mathsf{a}_2}(s))$. More generally:

Proposition 34. If $\mathsf{a} \in A$ is consistent in s with any other action in A then $\tau_A(s) = \tau_{\mathsf{a}}(\tau_{A\setminus\mathsf{a}}(s)) = \tau_{A\setminus\mathsf{a}}(\tau_{\mathsf{a}}(s))$.

4.3.3 Solvability by Parallel Plans

A state s is *reachable by a parallel plan* from a state $s_0 \in \textit{R-STATES}^{\Rightarrow}$ via a set of actions Act if there is a sequence $\langle A_1, \ldots, A_m\rangle$ of steps and a sequence of states $\langle s_0, \ldots, s_m\rangle$ with $m \geq 0$ such that $s = s_m$ and $\tau_{A_k}(s_{k-1}) = s_k$ for every k such that $1 \leq k \leq m$.

A simple epistemic planning task $(Act, s_0, Goal)$ is *solvable by a parallel plan* if there is at least one state s that is reachable by a parallel plan from $s_0^{\Rightarrow}$ via Act such that $s \models Goal$; otherwise it is unsolvable by a parallel plan. Solvability by a sequential plan is the special case where the parallel plan is required to be a sequence of singletons.

Example 8. *Let G_1' be modification of G_1 that is obtained by replacing the actions $\mathsf{Call}_{i,j}$ by Tcall_j^i of Example 7. Then G_1' can be solved in $\lceil \log_2 n \rceil$ steps of parallel calls if the number of agents n is even, and in $\lceil \log_2 n \rceil + 1$ steps if n is odd (Bavelas 1950; Landau 1954; Knödel 1975; Cooper, Herzig, Maffre, Maris, and Régnier 2019). For instance, for $n = 4$ the parallel plan $(\{\mathsf{Tcall}_2^1, \mathsf{Tcall}_4^3\}, \{\mathsf{Tcall}_3^1, \mathsf{Tcall}_4^2\})$ is a solution of G_1' with 2 steps.*

4.4 Translation into Classical Planning and Complexity

We now translate simple epistemic planning into classical planning. There, solvability by a parallel plan and by a sequential plan are equivalent under ∀-Step semantics, and both are PSPACE-complete (Bylander 1994). Our translation is polynomial, so the solvability of simple epistemic planning tasks is in PSPACE. It also gives us an encoding into PDDL, which allows us to use classical planners in sections 4.5 and 4.6. For a bounded horizon planning task we can translate into DL-PPA model checking and use the PSPACE membership result of chapter 3.

4.4.1 Two Versions of Classical Planning

There are two possibilities to define classical planning in our context. Version 1 amounts to epistemic planning restricted to the fragment of the language of EL-O without $\mathbf{S}_i$ and **JS**: none of them can occur in classical action descriptions and planning tasks. It immediately follows that EL-O-based planning is PSPACE hard because classical planning is so.

Version 2 of classical planning keeps the language of EL-O but changes the semantics by weakening the consistency condition for action descriptions: a version 2 classical planning task is a triple $\mathcal{P} = (Act, s_0, Goal)$ such that for every $ce_1, ce_2 \in \mathit{eff}(\mathsf{a})$, if $\mathit{ceff}^+(ce_1) \cap \mathit{ceff}^-(ce_2) \neq \emptyset$ then $\mathit{pre}(\mathsf{a}) \wedge \mathit{cnd}(ce_1) \wedge \mathit{cnd}(ce_2)$ is unsatisfiable in Classical Propositional Calculus CPC. We then define the partial functions $\tau_{\mathsf{a}}^{\mathsf{CPC}}$ as follows. First, $\tau_{\mathsf{a}}^{\mathsf{CPC}}$ is defined if $s \models^{\mathsf{CPC}} \mathit{pre}(\mathsf{a})$; second, the resulting state is obtained without closing under atomic causes and consequences:

$$\tau_{\mathsf{a}}^{\mathsf{CPC}}(s) = \left(s \setminus \left(\bigcup_{ce \in \mathit{eff}(\mathsf{a}), s \models^{\mathsf{CPC}} \mathit{cnd}(ce)} \mathit{ceff}^-(ce) \right) \right) \cup \left(\bigcup_{ce \in \mathit{eff}(\mathsf{a}), s \models^{\mathsf{CPC}} \mathit{cnd}(ce)} \mathit{ceff}^+(ce) \right).$$

From there we modify the definitions of consistency of a set of actions and of τ_A^{CPC} in a similar manner, removing all atomic causes and consequences of sets and requiring conditions to be satisfied classically. A classical planning task is classically solvable if a goal state is reachable from the initial state via a set of actions Act, with the difference that reachability is now defined in terms of the function τ_A^{CPC}.

4.4.2 Expansion of Planning Tasks

Let $\mathcal{P} = (Act, s_0, Goal)$ be a simple epistemic planning task. Its *expansion* is obtained by closing the initial state and the action descriptions under the

atomic causes and consequences that are relevant for $\mathcal{P}$:

$$\begin{aligned}\mathsf{Exp}(\mathcal{P}) = (&\{(pre(\mathsf{a}), \mathsf{Exp}_{\mathcal{P}}(\mathit{eff}(\mathsf{a}))) \ : \ (pre(\mathsf{a}), \mathit{eff}(\mathsf{a})) \in Act\},\\ &s_0^{\Rightarrow} \cap ATM(\mathcal{P}),\\ &\mathit{Goal}),\end{aligned}$$

where the expansion of a conditional effect is defined as:

$$\begin{aligned}\mathsf{Exp}_{\mathcal{P}}(\mathit{eff}(\mathsf{a})) = \{&(\mathit{cnd}(ce),\\ &(\mathit{ceff}^{+}(ce))^{\Rightarrow} \cap ATM(\mathcal{P}),\\ &(\mathit{ceff}^{-}(ce))^{\Leftarrow} \cap ATM(\mathcal{P})) \ : \ ce \in \mathit{eff}(\mathsf{a})\}.\end{aligned}$$

Proposition 35. Let $\mathcal{P} = (Act, s_0, \mathit{Goal})$ be a simple epistemic planning task. Then $\mathcal{P}$ is solvable if and only if its expansion $\mathsf{Exp}(\mathcal{P})$ is classically solvable.

Proof. Let

$$\mathit{R\text{-}STATES}|_{\mathcal{P}} = \{s \cap ATM(\mathcal{P}) \ : \ s \in \mathit{R\text{-}STATES}^{\Rightarrow}\}.$$

We define a semantics of actions *relative to* $\mathcal{P}$ in the following manner: if $s \in \mathit{R\text{-}STATES}|_{\mathcal{P}}$ and $A = \{\mathsf{a}_1, \ldots, \mathsf{a}_m\} \subseteq Act$ is a consistent set of actions then $\tau_A^{\mathcal{P}}(s)$ is defined iff for all $\mathsf{a} \in A$ we have $s \models pre(\mathsf{a})$, and in that case

$$\tau_A^{\mathcal{P}}(s) = \left(s \setminus \left(\bigcup_{\substack{\mathsf{a} \in A, ce \in \mathit{eff}(\mathsf{a}),\\ s \models \mathit{cnd}(ce)}} ((\mathit{ceff}^{-}(ce))^{\Leftarrow} \cap ATM(\mathcal{P}))\right)\right) \cup \left(\bigcup_{\substack{\mathsf{a} \in A, ce \in \mathit{eff}(\mathsf{a}),\\ s \models \mathit{cnd}(ce)}} \left((\mathit{ceff}^{+}(ce))^{\Rightarrow} \cap ATM(\mathcal{P})\right)\right).$$

It is easily shown that if $s \in \mathit{R\text{-}STATES}|_{\mathcal{P}}$ and A is consistent in s then $\tau_A^{\mathcal{P}}(s) \in \mathit{R\text{-}STATES}|_{\mathcal{P}}$.

By Proposition 33, for any state $s \in \mathit{R\text{-}STATES}^{\Rightarrow}$ and set of actions $A \subseteq Act$, $\tau_A(s)$ is defined iff $\tau_A^{\mathcal{P}}(s \cap ATM(\mathcal{P}))$ is defined, and in that case $\tau_A^{\mathcal{P}}(s \cap ATM(\mathcal{P})) = \tau_A(s) \cap ATM(\mathcal{P})$. We can then extend this result to any sequence of steps, i.e., to every parallel plan: for any state $s \in \mathit{R\text{-}STATES}^{\Rightarrow}$, there exists a state s' reachable from s via $\langle A_1, \ldots, A_m\rangle$ iff there exists a

state s'' that is $\mathcal{P}$-reachable from $s \cap ATM(\mathcal{P})$ via this same sequence of sets of actions, where $\mathcal{P}$-reachability is defined in the natural way following the semantics of actions relative to $\mathcal{P}$, and in that case $s'' = s' \cap ATM(\mathcal{P})$. In particular s' and s'' agree on *Goal*, and therefore $\mathcal{P}$ is solvable iff the planning task $(Act, s_0^{\vec{\Rightarrow}} \cap ATM(\mathcal{P}), Goal)$ is $\mathcal{P}$-solvable on $R\text{-}STATES|_{\mathcal{P}}$, where $\mathcal{P}$-solvability is once again defined in the natural manner following the semantics of actions relative to $\mathcal{P}$.

The expansion of $\mathcal{P}$ then 'spells out' the definition of the functions $\tau_A^{\mathcal{P}}$ for $A \subseteq Act$. Moreover, Proposition 33 tells us that for any $\varphi \in ATM(\mathcal{P})$, if $s \in R\text{-}STATES|_{\mathcal{P}}$ then $s \models \varphi$ iff $s \models^{\mathsf{CPC}} \varphi$. This gives us equivalence with classical planning. □

4.4.3 Complexity

By Proposition 31, the length of the expansion of epistemic planning tasks $\mathcal{P}$ is polynomial in the length of $\mathcal{P}$: $\ell(\mathsf{Exp}(\mathcal{P})) \leq (\ell(\mathcal{P}))^2$. Then PSPACE membership follows from Proposition 35. Hardness is the case because classical planning (version 1) is a particular case of simple epistemic planning, as we have observed in Section 4.4.1.

Proposition 36. The problem of deciding solvability of a simple epistemic planning task and its bounded horizon version are both PSPACE-complete.

4.5 Encoding into PDDL

In order to be able to use classical planners we encode simple epistemic planning tasks into the Planning Domain Definition Language PDDL McDermott et al. 1998. Fortunately, almost all planners from the 2018 International Planning Competition (IPC 2018)[1] handle conditional effects and negative preconditions, and most of them handle disjunctive preconditions.

4.5.1 Encoding of Formulas

When encoding a planning task into PDDL, some PDDL requirement flags have to be set depending on the form of conditions $cnd(ce)$ of conditional effects ce of actions as well as on the form of the formula *Goal*:

[1] `https://ipc2018-classical.bitbucket.io/`

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

For a formula φ without introspective atoms, we define a recursive function $f(\varphi)$ which returns the encoding of φ into PDDL:

$$f(\mathbf{S}_{i_1}\dots\mathbf{S}_{i_m}\,p) = \begin{cases} \texttt{(p)} & \text{if } m=0 \\ \texttt{(S-m i1 … im p)} & \text{otherwise} \end{cases}$$

$$f(\mathbf{JS}\,\mathbf{S}_{i_1}\dots\mathbf{S}_{i_m}\,p) = \begin{cases} \texttt{(JS p)} & \text{if } m=0 \\ \texttt{(JS-m i1 … im p)} & \text{otherwise} \end{cases}$$

$$f(\neg\varphi) = \texttt{(not } f(\varphi)\texttt{)}$$

$$f(\varphi_1 \wedge \varphi_2) = \texttt{(and } f(\varphi_1)\ f(\varphi_2)\texttt{)}$$

with $p \in \mathbb{P}$, $m \geq 0$, and $i_1, \dots, i_m \in \mathit{Agt}$. In words, a visibility atom $\alpha = \mathbf{S}_{i_1}\dots\mathbf{S}_{i_m}\,p$ is encoded by a special fluent with $m{+}1$ parameters. If $m = 0$, then the propositional variable p is encoded as a fluent without parameters. A visibility atom $\alpha = \mathbf{JS}\,\mathbf{S}_{i_1}\dots\mathbf{S}_{i_m}\,p$ is encoded by a special fluent with $m{+}1$ parameters. If $m = 0$ then a special fluent is encoded with the propositional variable p as unique parameter.

The formula *Goal* and the preconditions of every action are EL-O formulas and are encoded as $f(\mathit{Goal})$ etc. The initial state s_0 is encoded as a set of fluents, encoding each $\alpha \in s_0^{\Rightarrow}$ as $f(\alpha)$.

4.5.2 Encoding of Actions

For every action a and every conditional effect $ce \in \mathit{eff}(\mathsf{a})$ with $(\mathit{ceff}^{+}(ce))^{\Rightarrow} \cap \mathit{ATM}(\mathcal{P}) = \{\alpha_1, \dots, \alpha_m\}$ and $(\mathit{ceff}^{-}(ce))^{\Leftarrow} \cap \mathit{ATM}(\mathcal{P}) = \{\beta_1, \dots, \beta_\ell\}$ we add the conditional effect:

```
(when f(cnd(ce))
   (and f(α1) ... f(αm)
        (not f(β1)) ... (not f(βℓ))))
```

Example 9 (Example 4, ctd.)**.** *The action* $\mathsf{Call}_{1,2}$ *is encoded as:*

```
(: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. It can be generalised 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)))))))
```

4.6 Experimental Results

To experiment with simple epistemic planning tasks we considered some benchmark tasks using planners from the international planning competition IPC 2018. The experiments were done using three classical planners of the optimal track of IPC 2018: Planning-PDBs, Complementary1 and Complementary2.

We are in a multi-agent setting where agents execute actions simultaneously in steps. We designed our problems so that it is beneficial (in terms of minimising the number of steps) that agents cooperate and perform tasks in parallel. We want to find the shortest plan in which the agents cooperate in this way. For this, we used cost-optimal planners from the classical tracks of the competition. Given a cost function defined for all actions of the planning task, these planners return sequential plans minimizing total cost. To simulate parallel steps and ensure independence of actions in one step, we use in each experiment an EndStep action. That is, we adapt the action descriptions such as to ensure that all actions executed between two EndStep actions are applicable in parallel, and an EndStep action must occur in order to simulate the following parallel step. Moreover, we give zero cost to all actions but EndStep, therefore effectively counting the number of steps and guaranteeing that our experiments return optimal parallel plans.

The experiments all gave similar results for the problems described below, so we choose to show the results for Planning-PDBs. The results were obtained

on a GNU/Linux machine running on a 3.6 to 4.4 GHz CPU with 32 GB of RAM and a 30 minutes time limit (wall clock time). All the execution times given below are CPU times.

4.6.1 Parallel Gossip Task

We use the gossip modelling of Example 7 with the actions $\mathsf{Startcall}^i_j$ and $\mathsf{Endcalls}$, which here becomes our $\mathsf{EndStep}$.

Figure 4.1 shows the difference between parallel and sequential gossiping. We can see that with only two agents (blue and longest curve) the execution times are very similar but when, for a fixed depth, the number of agents increases then the execution time increases, too. Thus, we have fewer results for parallel gossiping than for sequential gossiping.

The largest parallel gossiping task solved under 1800s was for 5 agents and an epistemic depth of 2. This task has 125 atoms, 26 actions and 1920 conditional effects. For sequential gossiping the largest solved planning task was for 8 agents and an epistemic depth of 1. This task has 56 atoms, 64 actions and 1512 conditional effects.

4.6.2 Management Task

For the next benchmark planning task, we consider that a set of tasks has to be performed by the set of agents and that the execution of any task by an agent requires the agent to have a corresponding skill. In the beginning, the skills are split between the agents so that a particular agent may lack some of the skills required to perform particular tasks. This can limit the ability of the agents to perform tasks in parallel. However, there are also actions which allow agents to teach some of their skills to other agents.

Initially all agents are free and the state is $s_0 = \{\mathit{free}_i \,:\, i \in \mathit{Agt}\} \cup S$ for some subset S of $\{\mathbf{S}_i\, \mathit{skill}_k \,:\, i \in \mathit{Agt}, k \in \mathit{Skills}\} \cup \{\mathit{needs}_{t,k} \,:\, t \in \mathit{Tasks}, k \in \mathit{Skills}\}$. The goal is to perform all tasks: $\mathit{Goal} = \bigwedge_{t \in \mathit{Tasks}} \mathit{done}_t$.

The action descriptions are listed in the upper half of Table 4.2. For each pair of agents i,j and skill skill_k, the action $\mathsf{Teach}_{i,j,k}$ can be executed when i knows the skill she is teaching and is either free or is teaching the same skill to another agent. Its effects are that j knows the skill, that i and j are both no longer free, and that i is still available to teach skill_k to other agents. The action $\mathsf{DoTask}_{i,t,k}$ requires that i is free and knows skill skill_k and that skill_k

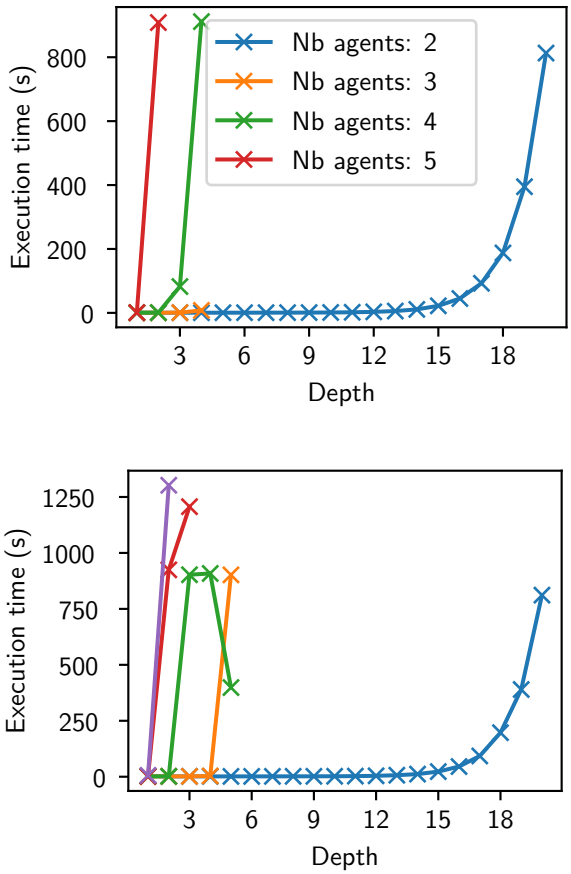

Figure 4.1: Parallel gossip: time to find an optimal parallel plan vs. epistemic depth (top) and time to find an optimal sequential plan vs. epistemic depth (bottom)

is a necessary and sufficient condition to accomplish t. Its effect is that the agent is not free and that the task is done. Finally, the $\mathsf{EndStep}$ action frees all agents (regardless of whether they were teaching, learning or performing a task).

Suppose there are n agents, n tasks and one skill $skill_k$ that is only known by agent i. Then an optimal parallel plan has two steps: first i teaches $skill_k$ to the other agents and then each agent j executes task t_j in parallel. In contrast, the optimal sequential plan is that i executes all the tasks herself, which is an n-step plan that cannot be parallelized.

Figure 4.2 shows the time needed by Planning-PDBs to find a plan, in seconds. It compares the effect of the number of tasks, the number of agents and the number of skills. The first plot shows the effect of the number of tasks with different fixed values of the number of agents and skills. We can see that this variable has almost no effect on the difficulty to find a plan. The second plot does the same with the number of agents with different fixed values for the number of tasks and skills, while in the third plot the number of skills is the only variable which varies. These latter two plots show that the number of agents and the number of skills have an effect on the complexity. However, we do not have enough values to say more about this relationship.

The number of results were limited by the increasing complexity of the planning task. The most complex planning task for which we found an optimal parallel plan involved 7 tasks, 7 agents and 6 skills: this instance has 92 atoms and 638 actions. The most complex planning task tested and which timed out, had 7 tasks, 7 agents and 7 skills: this instance has 105 atoms and 736 actions.

Remark 2. One may replace $\mathsf{Teach}_{i,j,k}$ by the action $\mathsf{ReqSkill}_{j,i,k}$ of j asking i to teach her skill $skill_k$. Then we have to add to the precondition that j does not have the skill but knows that i does:

$$pre(\mathsf{ReqSkill}_{j,i,k}) = pre(\mathsf{Teach}_{i,j,k}) \wedge \neg \mathbf{S}_j\, skill_k \wedge \mathbf{S}_j\, \mathbf{S}_i\, skill_k.$$

We note that such a precondition cannot be expressed with epistemic literals of the approach of (Muise et al. 2015).

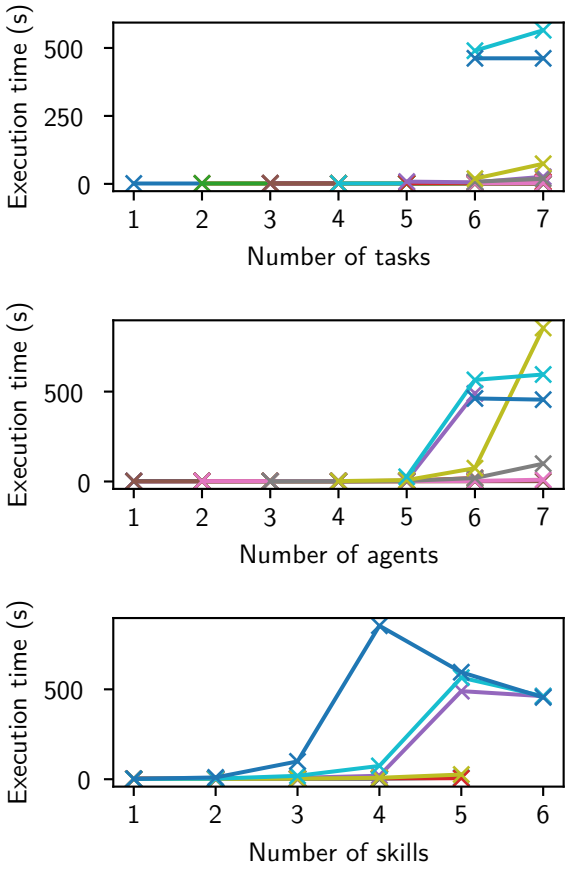

Figure 4.2: Management: time to find an optimal parallel plan vs. number of tasks, agents or skills (for different fixed values of the two other parameters)

$\mathsf{DoTask}_{i,t,k}$
$free_i \wedge \mathbf{S}_i\, skill_k \wedge needs_{t,k} \rightarrow \{(\top, \{done_t\}, \{free_i\})\}$
$\mathsf{Teach}_{i,j,k}$
$\mathbf{S}_i\, skill_k \wedge (free_i \vee teaching_{i,k}) \rightarrow \{(\top, \{\mathbf{S}_j\, skill_k, teaching_{i,k}\}, \{free_i, free_j\})\}$
$\mathsf{EndStep}$
$\top \rightarrow \{(\top, \{free_i \ : \ i \in Agt\}, \{teaching_{i,k} \ : \ i \in Agt, skillk \in Skills\})\}$
$\mathsf{DoTask}_{i,t,\ell}$
$free_i \wedge mdone_\ell \wedge \mathbf{S}_i\, mdone_\ell \wedge \neg mdone_{\ell+1} \rightarrow \{(\top, \{tdone_t\}, \{free_i\})\}$
$\mathsf{DoMeeting}_\ell$
$\bigwedge_{i \in Agt} free_i \rightarrow \{(\top, \{mdone_\ell, \mathbf{JS}\, mdone_\ell\}, \{free_i \ : \ i \in Agt\})\}$
$\mathsf{EndStep}$
$\top \rightarrow \{(\top, \{free_i \ : \ i \in Agt\}, \emptyset)\}$

Table 4.2: Action descriptions for the management task (top) and the meeting task (bottom)

4.6.3 Meetings Task

As an example of the use of common knowledge, consider a planning task which involves cooperation between different agents and which can be divided into m different stages with tasks to be performed at each step by each agent. Agents are only authorized to start stage ℓ+1 if *all* tasks of stage ℓ have been completed and all agents have common knowledge of this. The only way this can be achieved is by having a plenary meeting at the end of each stage during which each agent announces that their stage-ℓ task has been completed (action $\mathsf{DoMeeting}_\ell$).

Initially all agents are free and no meeting or task has been completed: $s_0 = \{free_i \ : \ i \in Agt\}$. The goal is for all tasks and meetings to be completed: $Goal = \bigwedge_{t \in Tasks} tdone_t \wedge \bigwedge_{\ell \in Meetings} mdone_\ell$.

The action descriptions are listed in the lower half of Table 4.2. Each stage-ℓ task requires that the agent executing it knows that meeting ℓ has taken place. To avoid having stage-ℓ tasks done at stage-ℓ' for $\ell' > \ell$, the task also requires the meeting ℓ+1 not to have taken place yet. The action $\mathsf{DoMeeting}_\ell$ of holding a meeting is executable if all the agents are free. Its

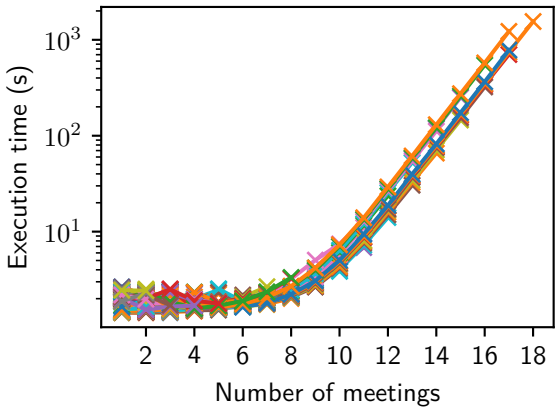

Figure 4.3: Meetings: time to find an optimal parallel plan vs. number of meetings (for different fixed numbers of tasks and agents)

effect is that all agents jointly see that the meeting has been held and that the agents are no longer free. Finally, the action EndStep frees the agents and ends the step.

For example, with 2 agents and 3 meetings (2 stages), the following is a solution plan:

$$\begin{aligned}&\langle\{\mathsf{DoMeeting}_1\},\{\mathsf{EndStep}\},\\&\{\mathsf{DoTask}_{1,1,1},\mathsf{DoTask}_{2,2,1}\},\{\mathsf{EndStep}\},\\&\{\mathsf{DoMeeting}_2\},\{\mathsf{EndStep}\},\\&\{\mathsf{DoTask}_{1,3,2},\mathsf{DoTask}_{2,4,2}\},\{\mathsf{EndStep}\},\\&\{\mathsf{DoMeeting}_3\}\rangle.\end{aligned}$$

Figure 4.3 shows the time needed by Planning-PDBs to find a plan, in seconds, relative to the number of meetings, where each plotted curve is for a fixed number of tasks and agents. We can see that the other two parameters (number of tasks and number of agents) have little to no effect on the difficulty of finding an optimal plan. In contrast, the number of meetings seems to have an exponential effect when it is greater than 10.

The largest planning task which was solved within the time limit of 1800s

involved 18 tasks, 17 meetings and two agents: this instance has 46 atoms and 16400 actions. The most complex planning tasks that we tested and that timed out had 18 tasks, 18 meetings and one agent: the planners were not able to ground this problem.

4.7 Conclusion

We have defined simple epistemic planning tasks and their solvability by a parallel plan and characterised its complexity via a polynomial translation into classical planning. This allows us to solve epistemic bounded horizon planning tasks by translating them to classical bounded horizon planning, which is known to be PSPACE-complete.

Our 'knowing whether'-based language can express more than the 'knowing-that'-based language of (Muise et al. 2015) (see Remark 2). Contrarily to (Kominis and Geffner 2015), we are not restricted to common knowledge of the initial state and public or semi-public actions, as illustrated by the gossiping task where actions are private. Moreover, none of the latter two approaches deals with common knowledge or concurrent actions. Experiments demonstrated the possibility of solving some interesting practical problems.

In future work we plan to use SAT-based planners such as Rintanen et al.'s (Rintanen, Heljanko, and Niemelä 2006), which output parallel plans and therefore do not require the EndStep action.

5 Conclusion

This thesis showed, via a running example with several variations, the gossip problem, the interest of adding both temporal and epistemic aspects to a planning problem. These temporal epistemic gossip problems can still be extended. They are one of the simplest temporal epistemic planning problems so they can be used in many different ways. We can have different durations for different calls or make calls non-deterministic: the person we want to call may not be fully available even in allowed time frames; or there might be noise in the transmission altering the communication.

The first step to temporal planning is planning with parallel plans. With that in mind, we added parallelism to a dynamic logic without increasing its complexity. DL-PPA is an interesting basis for a parallel dynamic epistemic logic in PSPACE. The logic it is based on was already extended to an epistemic logic (DEL-PAO) and such work can be used to extend DL-PPA too. There is also some work done with DL-PPA for contingent planning in (Scheck, Niveau, and Zanuttini 2020).

We also showed how to translate epistemic planning problems from EL-O to PDDL to search for a parallel plan. The work done in Chapter 4 transforms a simple form of epistemic planning to classical planning. This can be combined with approaches such as SATPLAN to translate epistemic planning problems into SAT. The benchmarks in this chapter show that more work can be done to improve the planners for epistemic planning. This approach can also be extended to temporal epistemic planning using existing temporal planners and PDDL 2.1.

All those results can be used to make tools for temporal and epistemic planning. One interesting future work could be to extend EL-O with some temporal aspects to be able to reason about the knowledge of the agents and its variation over time. This could allow one to model loss of knowledge some time after the execution of an action that added it.

An application of planning that I find interesting is about generating stories, for instance, video game scenarios. The actions of the planning tasks could be atomic plot elements. The initial state represents the world at the beginning

of the game. And the goal of the planning task is chosen so the player has fun playing the game and that some key elements of the scenario happens. However, a player does not act deterministically and could act differently from what the planner expected. On the other and, by modeling wisely the planning task and using replanning whenever the player diverges from the chosen path we may be able to have a scenario that adapts to the actions of the player. With the combination of temporal and epistemic planning, we could generate meaningful and complex stories. The knowledge of the characters can be taken into account as well as the time needed for the player to act and the world to change.

Conclusion en français

Cette thèse a montré, à travers la déclinaison d'un example, le problème du bavardage, l'intérêt d'ajouter des aspects temporels et épistémiques à un problème de planification. Ces problèmes de bavardage temporels et épistémiques peuvent encore être étendus. Puisque ils constituent l'un des problèmes de planification épistémique temporelle les plus simples, ils peuvent être utilisés de différentes manières. Nous pouvons avoir différentes durées pour différents appels ou rendre les appels non déterministes : la personne que nous voulons appeler peut ne pas être totalement disponible, même dans les intervales de temps spécifiés, ou bien il peut y avoir du bruit dans le système de communication.

La première étape vers la planification temporelle est la planification avec des plans parallèles. Dans cette optique, nous avons ajouté le parallélisme à une logique dynamique sans en augmenter la complexité. DL-PPA est une base intéressante pour une logique épistémique dynamique parallèle dans PSPACE. La logique sur laquelle elle est basée a déjà été étendue vers une logique épistémique (DEL-PAO) et un tel travail peut être utilisé pour étendre DL-PPA également. Il existe également des travaux réalisés avec DL-PPA pour la planification contingente dans (Scheck, Niveau et Zanuttini 2020).

Nous avons aussi montré comment traduire les problèmes de planification épistémique de EL-O vers PDDL pour rechercher un plan solution parallèle. Le travail effectué dans le chapitre 4 transforme une forme simple de planification épistémique en planification classique. Ceci peut être combiné avec des approches telles que SATPLAN pour traduire les problèmes de planification épistémique en SAT. Les tests de performance de ce chapitre montrent qu'il

est possible d'améliorer les planificateurs pour la planification épistémique. Cette approche peut également être étendue à la planification épistémique temporelle en utilisant les planificateurs temporels existants et PDDL 2.1.

Tous ces résultats peuvent être utilisés pour créer des outils de planification temporelle et épistémique. Un travail futur intéressant pourrait être d'étendre EL-O avec certains aspects temporels pour pouvoir raisonner sur la connaissance des agents et sa variation dans le temps. Cela pourrait permettre de modéliser la perte de connaissances un certain temps après l'exécution d'une action qui l'a ajoutée.

Une application de la planification que je trouve intéressante concerne la génération d'histoires, par exemple, des scénarios de jeux vidéo. Les actions des tâches de planification pourraient être des éléments atomiques de l'intrigue. L'état initial représente le monde au début du jeu. Et l'objectif de la tâche de planification est choisi pour que le joueur s'amuse en jouant le jeu et que certains éléments clés du scénario se produisent. Cependant, le joueur n'agit pas de façon déterministe et peut agir différemment de ce que le planificateur a prévu. D'un autre côté, en modélisant judicieusement la planification et en utilisant la replanification chaque fois que le joueur s'écarte du chemin chemin choisi, nous pouvons avoir un scénario qui s'adapte aux actions du joueur. Avec la combinaison de la planification temporelle et épistémique, nous pourrions générer des histoires intéressantes et complexes. Les connaissances des personnages peuvent être prises en compte, ainsi que le temps nécessaire pour que le joueur agisse et le monde change.

Bibliography

Akkoyunlu, E. A., K. Ekanandham, and R. V. Huber (1975). "Some Constraints and Tradeoffs in the Design of Network Communications." In: *Proceedings of the Fifth Symposium on Operating System Principles, SOSP 1975, The University of Texas at Austin, Austin, Texas, USA, November 19-21, 1975*. Ed. by James C. Browne and Juan Rodriguez-Rosell. ACM, pp. 67–74. DOI: 10.1145/800213.806523. URL: https://doi.org/10.1145/800213.806523.

Allen, James F. (1984). "Towards a General Theory of Action and Time." In: *Artif. Intell.* 23.2, pp. 123–154. DOI: 10.1016/0004-3702(84)90008-0.

Andersen, Mikkel Birkegaard, Thomas Bolander, and Martin Holm Jensen (2012). "Conditional Epistemic Planning." In: *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Toulouse, France, September 26-28, 2012. Proceedings*. Ed. by Luis Fariñas del Cerro, Andreas Herzig, and Jérôme Mengin. Vol. 7519. Lecture Notes in Computer Science. Springer, pp. 94–106. ISBN: 978-3-642-33352-1. DOI: 10.1007/978-3-642-33353-8_8. URL: https://doi.org/10.1007/978-3-642-33353-8%5C_8.

Apt, Krzysztof R., Davide Grossi, and Wiebe van der Hoek (2015). "Epistemic Protocols for Distributed Gossiping." In: *Proceedings Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2015*. Ed. by R. Ramanujam. Vol. 215. EPTCS, pp. 51–66. DOI: 10.4204/EPTCS.215. URL: https://doi.org/10.4204/EPTCS.215.5;%20https://dblp.org/rec/bib/journals/corr/AptGH16.

– (2018). "When Are Two Gossips the Same? Types of Communication in Epistemic Gossip Protocols." In: *CoRR* abs/1807.05283. URL: http://arxiv.org/abs/1807.05283.

Apt, Krzysztof R., Eryk Kopczynski, and Dominik Wojtczak (2017). "On the Computational Complexity of Gossip Protocols." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*. Ed. by Carles Sierra. ijcai.org, pp. 765–771. URL: https://doi.org/10.24963/ijcai.2017/106.

Apt, Krzysztof R. and Dominik Wojtczak (2017a). "Common Knowledge in a Logic of Gossips." In: *Proceedings Sixteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2017*. Ed. by Jérôme Lang. Vol. 251. EPTCS, pp. 10–27. URL: https://doi.org/10.4204/EPTCS.251.2.

– (2017b). "Decidability of Fair Termination of Gossip Protocols." In: *IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations*. Ed. by Thomas Eiter, David Sands, Geoff Sutcliffe, and Andrei Voronkov. Vol. 1. Kalpa Publications in Computing. EasyChair. URL: http://www.easychair.org/publications/paper/342983.

Aucher, Guillaume and Thomas Bolander (2013). "Undecidability in epistemic planning." In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*. Ed. by Francesca Rossi. AAAI Press, pp. 27–33. ISBN: 978-1-57735-633-2. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6903.

Balbiani, Philippe and Joseph Boudou (2018). "Iteration-free PDL with storing, recovering and parallel composition: a complete axiomatization." In: *J. Log. Comput.* 28.4, pp. 705–731. DOI: 10.1093/logcom/exv035. URL: https://doi.org/10.1093/logcom/exv035.

Balbiani, Philippe, Andreas Herzig, François Schwarzentruber, and Nicolas Troquard (2014). "DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE." In: *CoRR* abs/1411.7825. arXiv: 1411.7825. URL: http://arxiv.org/abs/1411.7825.

Balbiani, Philippe, Andreas Herzig, and Nicolas Troquard (2013). "Dynamic Logic of Propositional Assignments: A Well-Behaved Variant of PDL." In: *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. Ed. by Orna Kupferman. IEEE Computer Society, pp. 143–152. ISBN: 978-1-4799-0413-6. DOI: 10.1109/LICS.2013.20. URL: http://www.loa.istc.cnr.it/troquard/PAPERS/BaHeTr13lics.pdf.

Balbiani, Philippe and Dimiter Vakarelov (2003). "PDL with Intersection of Programs: A Complete Axiomatization." In: *Journal of Applied Non-Classical Logics* 13.3-4, pp. 231–276. DOI: 10.3166/jancl.13.231-276. URL: https://doi.org/10.3166/jancl.13.231-276.

Bavelas, Alex (1950). "Communication patterns in task-oriented groups." In: *The Journal of the Acoustical Society of America* 22.6, pp. 725–730.

Benevides, Mario R. F., Renata P. de Freitas, and Jorge Petrucio Viana (2011). "Propositional Dynamic Logic with Storing, Recovering and Parallel Composition." In: *Electr. Notes Theor. Comput. Sci.* 269, pp. 95–107. DOI: 10.1016/j.entcs.2011.03.008. URL: https://doi.org/10.1016/j.entcs.2011.03.008.

Benevides, Mario R. F. and L. Menasché Schechter (2014). "Propositional dynamic logics for communicating concurrent programs with CCS's parallel operator." In: *J. Log. Comput.* 24.4, pp. 919–951. DOI: 10.1093/logcom/exu001. URL: https://doi.org/10.1093/logcom/exu001.

Bolander, Thomas (2017). "A Gentle Introduction to Epistemic Planning: The DEL Approach." In: *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017*. Ed. by Sujata Ghosh and R. Ramanujam. Vol. 243. EPTCS, pp. 1–22. DOI: 10.4204/EPTCS.243.1. URL: https://doi.org/10.4204/EPTCS.243.1.

Bolander, Thomas and Mikkel Birkegaard Andersen (2011). "Epistemic planning for single and multi-agent systems." In: *Journal of Applied Non-Classical Logics* 21.1, pp. 9–34. DOI: 10.3166/JANCL.21.9. URL: https://doi.org/10.3166/jancl.21.9-34.

Bolander, Thomas, Thorsten Engesser, Robert Mattmüller, and Bernhard Nebel (2018). "Better Eager Than Lazy? How Agent Types Impact the Successfulness of Implicit Coordination." In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, pp. 445–453. ISBN: 978-1-57735-803-9. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/18070.

Bolander, Thomas, Martin Holm Jensen, and François Schwarzentruber (2015). "Complexity Results in Epistemic Planning." In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 2791–2797. URL: http://ijcai.org/Abstract/15/395.

Bylander, Tom (1994). "The Computational Complexity of Propositional STRIPS Planning." In: *Artificial Intelligence* 69.1-2, pp. 165–204. ISSN: 00043702. DOI: 10.1016/0004-3702(94)90081-7.

Cong, Sébastien Lê, Sophie Pinchinat, and François Schwarzentruber (2018). "Small Undecidable Problems in Epistemic Planning." In: *Proceedings of the*

Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. Ed. by J. Lang. ijcai.org, pp. 4780–4786. ISBN: 978-0-9992411-2-7. DOI: 10.24963/ijcai.2018/664. URL: https://doi.org/10.24963/ijcai.2018/664.

Cooper, Martin C., Andreas Herzig, Faustine Maffre, Frédéric Maris, Elise Perrotin, and Pierre Régnier (2019). "A Lightweight Epistemic Logic and its Application to Planning." In: *AI Journal*. submitted.

Cooper, Martin C., Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier (2016a). "A Simple Account of Multi-Agent Epistemic Planning." In: *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*. Ed. by Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen. Vol. 285. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 193–201. ISBN: 978-1-61499-671-2. DOI: 10.3233/978-1-61499-672-9-193. URL: https://doi.org/10.3233/978-1-61499-672-9-193.

– (2016b). "Simple Epistemic Planning: Generalised Gossiping." In: *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*. Ed. by Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen. Vol. 285. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 1563–1564. DOI: 10.3233/978-1-61499-672-9-1563. arXiv: 1606.03244.

– (2016c). "Simple epistemic planning: generalised gossiping." In: *CoRR* abs/1606.03244. URL: http://arxiv.org/abs/1606.03244.

– (2019). "The epistemic gossip problem." In: *Discrete Mathematics* 342.3, pp. 654–663. DOI: 10.1016/j.disc.2018.10.041. URL: https://doi.org/10.1016/j.disc.2018.10.041.

Cooper, Martin C., Andreas Herzig, Frédéric Maris, Elise Perrotin, and Julien Vianey (Sept. 2020). "Lightweight Parallel Multi-Agent Epistemic Planning." In: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 274–283. DOI: 10.24963/kr.2020/28. URL: https://doi.org/10.24963/kr.2020/28.

Cooper, Martin C., Andreas Herzig, Frédéric Maris, and Julien Vianey (2018). "Temporal Epistemic Gossip Problems." In: *Multi-Agent Systems - 16th*

European Conference, EUMAS 2018, Bergen, Norway, December 6-7, 2018, Revised Selected Papers. Ed. by Marija Slavkovik. Vol. 11450. Lecture Notes in Computer Science. Springer, pp. 1–14. ISBN: 978-3-030-14173-8. DOI: 10.1007/978-3-030-14174-5_1. URL: https://doi.org/10.1007/978-3-030-14174-5%5C_1.

Cooper, Martin C., Frédéric Maris, and Pierre Régnier (2014). "Monotone Temporal Planning: Tractability, Extensions and Applications." In: *J. Artif. Intell. Res.* 50, pp. 447–485. DOI: 10.1613/jair.4358.

Cushing, William, Subbarao Kambhampati, Mausam, and Daniel S. Weld (2007). "When is Temporal Planning Really Temporal?" In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. Ed. by Manuela M. Veloso, pp. 1852–1859. URL: http://ijcai.org/Proceedings/07/Papers/299.pdf.

Dean, Thomas L. and Drew V. McDermott (1987). "Temporal Data Base Management." In: *Artif. Intell.* 32.1, pp. 1–55. DOI: 10.1016/0004-3702(87)90061-0.

Dechter, Rina, Itay Meiri, and Judea Pearl (1991). "Temporal Constraint Networks." In: *Artif. Intell.* 49.1-3, pp. 61–95. DOI: 10.1016/0004-3702(91)90006-6.

Demolombe, Robert and Maria Pilar del Pozos Parra (2000). "A Simple and Tractable Extension of Situation Calculus to Epistemic Logic." In: *Foundations of Intelligent Systems, 12th International Symposium, ISMIS 2000, Charlotte, NC, USA, October 11-14, 2000, Proceedings*, pp. 515–524. URL: https://doi.org/10.1007/3-540-39963-1%5C_54.

Dimopoulos, Y., B. Nebel, and J. Koehler (1997). "Encoding Planning Problems in Nonmonotonic Logic Programs." In: *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97, Toulouse, France, September 24-26, 1997, Proceedings*. Ed. by S. Steel and R. Alami. Vol. 1348. Lecture Notes in Computer Science. Springer, pp. 169–181. ISBN: 3-540-63912-8. DOI: 10.1007/3-540-63912-8_84. URL: https://doi.org/10.1007/3-540-63912-8%5C_84.

Fikes, Richard and Nils J. Nilsson (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." In: *Artif. Intell.* 2.3/4, pp. 189–208. DOI: 10.1016/0004-3702(71)90010-5. URL: https://doi.org/10.1016/0004-3702(71)90010-5.

Fox, Maria and Derek Long (2003). "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains." In: *J. Artif. Intell. Res.* 20, pp. 61–124. DOI: 10.1613/jair.1129. URL: https://doi.org/10.1613/jair.1129;%20https://dblp.org/rec/journals/jair/FoxL03.bib.

Ghallab, Malik, Dana Nau, and Paolo Traverso (2004). *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 1558608567.

Goldblatt, Robert (1992). "Parallel Action: Concurrent Dynamic Logic with Independent Modalities." In: *Studia Logica* 51.3/4, pp. 551–578. DOI: 10.1007/BF01028975. URL: https://doi.org/10.1007/BF01028975.

Herzig, Andreas (2014). "Belief Change Operations: A Short History of Nearly Everything, Told in Dynamic Logic of Propositional Assignments." In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. Ed. by C. Baral, G. De Giacomo, and T. Eiter. AAAI Press. ISBN: 978-1-57735-657-8. URL: http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7960.

Herzig, Andreas, Emiliano Lorini, and Faustine Maffre (2015). "A Poor Man's Epistemic Logic Based on Propositional Assignment and Higher-order Observation." In: *Logic, Rationality, and Interaction - 5th International Workshop, LORI 2015 Taipei, Taiwan, October 28-31, 2015, Proceedings*. Ed. by Wiebe van der Hoek, Wesley H. Holliday, and Wen-fang Wang. Vol. 9394. Lecture Notes in Computer Science. Springer Verlag, pp. 156–168. ISBN: 978-3-662-48560-6. DOI: 10.1007/978-3-662-48561-3_13. URL: http://www.irit.fr/%7B~%7DAndreas.Herzig/P/Lori15.html.

Herzig, Andreas, Emiliano Lorini, Nicolas Troquard, and Frédéric Moisan (2011). "A Dynamic Logic of Normative Systems." In: *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*. Ed. by Toby Walsh. IJCAI/AAAI, pp. 228–233. ISBN: 978-1-57735-516-8. DOI: 10.5591/978-1-57735-516-8/IJCAI11-049. URL: http://ijcai.org/proceedings/2011.

Herzig, Andreas, Frédéric Maris, and Julien Vianey (2019). "Dynamic logic of parallel propositional assignments and its applications to planning." In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, pp. 5576–5582. DOI: 10.24963/ijcai.2019/774.

URL: https://doi.org/10.24963/ijcai.2019/774;%20https://dblp.org/rec/conf/ijcai/HerzigMV19.bib.

Herzig, Andreas, Maria Viviane de Menezes, Leliane Nunes de Barros, and Renata Wassermann (2014). "On the revision of planning tasks." In: *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*. Ed. by T. Schaub, G. Friedrich, and B. O'Sullivan. Vol. 263. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 435–440. ISBN: 978-1-61499-418-3. DOI: 10.3233/978-1-61499-419-0-435. URL: https://doi.org/10.3233/978-1-61499-419-0-435.

Hu, Guang, Tim Miller, and Nir Lipovetzky (2019). "What you get is what you see: Decomposing Epistemic Planning using Functional STRIPS." In: *CoRR* abs/1903.11777. arXiv: 1903.11777. URL: http://arxiv.org/abs/1903.11777.

Huang, Xiao, Biqing Fang, Hai Wan, and Yongmei Liu (2017). "A General Multi-agent Epistemic Planner Based on Higher-order Belief Change." In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. Ed. by Carles Sierra. ijcai.org, pp. 1093–1101. DOI: 10.24963/ijcai.2017/152. URL: https://doi.org/10.24963/ijcai.2017/152.

Hurkens, Cor AJ (Jan. 2000). "Spreading gossip efficiently." In: *Nieuw Archief voor Wiskunde* 1, pp. 208–210.

Knoblock, Craig A. (1994). "Generating Parallel Execution Plans with a Partial-order Planner." In: *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems, University of Chicago, Chicago, Illinois, USA, June 13-15, 1994*. Ed. by K.J. Hammond. AAAI, pp. 98–103. ISBN: 0-929280-56-3. URL: http://www.aaai.org/Library/AIPS/1994/aips94-017.php.

Knödel, Walter (1975). "New gossips and telephones." In: *Discrete Mathematics* 13.1, p. 95. ISSN: 0012365X. DOI: 10.1016/0012-365X(75)90090-4. URL: http://www.sciencedirect.com/science/article/pii/0012365X75900904.

Kominis, Filippos and Hector Geffner (2015). "Beliefs In Multiagent Planning: From One Agent to Many." In: *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015*. Ed. by Ronen I. Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein. AAAI Press, pp. 147–155. ISBN:

978-1-57735-731-5. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10617.

Lagniez, Jean-Marie, Daniel Le Berre, Tiago de Lima, and Valentin Montmirail (2018). "A SAT-Based Approach For PSPACE Modal Logics." In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, pp. 651–652. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/17997.

Lakemeyer, Gerhard and Yves Lespérance (2012). "Efficient Reasoning in Multiagent Epistemic Logics." In: *ECAI 2012 - 20th European Conference on Artificial Intelligence*, pp. 498–503. URL: https://doi.org/10.3233/978-1-61499-098-7-498.

Landau, Hyman G. (1954). "The distribution of completion times for random communication in a task-oriented group." In: *The Bulletin of Mathematical Biophysics* 16.3, pp. 187–201.

Le, Tiep, Francesco Fabiano, Tran Cao Son, and Enrico Pontelli (2018). "EFP and PG-EFP: Epistemic Forward Search Planners in Multi-Agent Domains." In: *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*. Ed. by Mathijs de Weerdt, Sven Koenig, Gabriele Röger, and Matthijs T. J. Spaan. AAAI Press, pp. 161–170. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17733;%20https://dblp.org/rec/conf/aips/LeFSP18.bib.

Li, Yanjun, Quan Yu, and Yanjing Wang (2017). "More for free: a dynamic epistemic framework for conformant planning over transition systems." In: *J. Log. Comput.* 27.8, pp. 2383–2410. DOI: 10.1093/logcom/exx020. URL: https://doi.org/10.1093/logcom/exx020.

Liu, Qiang and Yongmei Liu (2018). "Multi-agent Epistemic Planning with Common Knowledge." In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. Ed. by Jérôme Lang. ijcai.org, pp. 1912–1920. DOI: 10.24963/ijcai.2018/264. URL: https://doi.org/10.24963/ijcai.2018/264.

Löwe, Benedikt, Eric Pacuit, and Andreas Witzel (2011). "DEL Planning and Some Tractable Cases." In: *Logic, Rationality, and Interaction - Third International Workshop, LORI 2011, Guangzhou, China, October 10-13,*

2011. Proceedings. Ed. by Hans van Ditmarsch, Jérôme Lang, and Shier Ju. Vol. 6953. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 179–192. ISBN: 978-3-642-24129-1. DOI: 10.1007/978-3-642-24130-7_13. URL: https://doi.org/10.1007/978-3-642-24130-7_13.

Maffre, Faustine (2016). "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)." PhD thesis. Paul Sabatier University, Toulouse, France. URL: https://tel.archives-ouvertes.fr/tel-01488408.

Mali, Amol Dattatraya and Subbarao Kambhampati (1999). "On the utility of Plan-space (Causal) Encodings." In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA*. Pp. 557–563. URL: http://www.aaai.org/Library/AAAI/1999/aaai99-079.php.

Maris, Frédéric and Pierre Régnier (2008). "TLP-GP: New Results on Temporally-Expressive Planning Benchmarks." In: *20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), November 3-5, 2008, Dayton, Ohio, USA, Volume 1*. IEEE Computer Society, pp. 507–514. DOI: 10.1109/ICTAI.2008.90.

Mayer, Alain J. and Larry J. Stockmeyer (1996). "The Complexity of PDL with Interleaving." In: *Theor. Comput. Sci.* 161.1&2, pp. 109–122. DOI: 10.1016/0304-3975(95)00095-X. URL: https://doi.org/10.1016/0304-3975(95)00095-X.

McDermott, Drew V. (1982). "A Temporal Logic for Reasoning About Processes and Plans." In: *Cogn. Sci.* 6.2, pp. 101–155. DOI: 10.1207/s15516709cog0602_1. URL: https://doi.org/10.1207/s15516709cog0602_1.

McDermott, Drew V., Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins (1998). *PDDL - The Planning Domain Definition Language*. Tech. rep. Yale Center for Computational Vision and Control.

Muise, Christian, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg (2015). "Planning Over Multi-Agent Epistemic States: A Classical Planning Approach." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI

Press, pp. 3327–3334. ISBN: 978-1-57735-698-1. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9974.

Peleg, David (1987). "Concurrent dynamic logic." In: *J. ACM* 34.2, pp. 450–479. DOI: 10.1145/23005.23008. URL: https://doi.org/10.1145/23005.23008.

Rintanen, Jussi (2007). "Complexity of Concurrent Temporal Planning." In: *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS 2007, Providence, Rhode Island, USA, September 22-26, 2007*. Ed. by Mark S. Boddy, Maria Fox, and Sylvie Thiébaux. AAAI, pp. 280–287. URL: http://www.aaai.org/Library/ICAPS/2007/icaps07-036.php.

– (2015). "Discretization of Temporal Models with Application to Planning with SMT." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, pp. 3349–3355. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9428.

Rintanen, Jussi, Keijo Heljanko, and Ilkka Niemelä (2006). "Planning as satisfiability: parallel plans and algorithms for plan search." In: *Artif. Intell.* 170.12-13, pp. 1031–1080. DOI: 10.1016/j.artint.2006.08.002. URL: http://dx.doi.org/10.1016/j.artint.2006.08.002.

Scheck, Sergej, Alexandre Niveau, and Bruno Zanuttini (2020). "Knowledge Compilation for Action Languages." In: *Journées Francophone Planification, Décision et Apprentissage pour la conduite de systèmes, JFPDA 2020, Anger, France, July 3, 2020*. Ed. by Frédéric Maris.

Shin, Ji-Ae and Ernest Davis (2005). "Processes and continuous change in a SAT-based planner." In: *Artif. Intell.* 166.1-2, pp. 194–253. DOI: 10.1016/j.artint.2005.04.001.

Spalazzi, Luca and Paolo Traverso (2000). "A dynamic logic for acting, sensing, and planning." In: *J. Log. Comput.* 10.6, pp. 787–821. DOI: 10.1093/logcom/10.6.787. URL: https://doi.org/10.1093/logcom/10.6.787.

Steedman, Mark and Ronald Petrick (2007). "Planning dialog actions." In: *Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue (SIGdial 2007)*, pp. 265–272.

Stergiou, Kostas and Manolis Koubarakis (2000). "Backtracking algorithms for disjunctions of temporal constraints." In: *Artif. Intell.* 120.1, pp. 81–117. DOI: 10.1016/S0004-3702(00)00019-9.

van Ditmarsch, Hans, Jan van Eijck, Pere Pardo, Rahim Ramezanian, and François Schwarzentruber (2017). "Epistemic protocols for dynamic gossip." In: *J. Applied Logic* 20, pp. 1–31. URL: https://doi.org/10.1016/j.jal.2016.12.001.

van Ditmarsch, Hans, Davide Grossi, Andreas Herzig, Wiebe van der Hoek, and Louwe B. Kuijer (2016). "Parameters for Epistemic Gossip Problems." In: *Proc. LOFT 2016*.

Veloso, Paulo A. S., Sheila R. M. Veloso, and Mario R. F. Benevides (2014). "PDL for structured data: a graph-calculus approach." In: *Logic Journal of the IGPL* 22.5, pp. 737–757. DOI: 10.1093/jigpal/jzu011. URL: https://doi.org/10.1093/jigpal/jzu011.

Vidal, Vincent (2001). "Recherche dans les graphes de planification, satisfiabilité et stratégies de moindre engagement." PhD thesis. Université Paul Sabatier, Toulouse, France.

Les problèmes de planification multi-agents avec un raisonnement sur la connaissance des agents, appelé raisonnement épistémique, on déjà été bien étudiés. Il en est de même pour les problèmes avec des actions temporellement étendues. Cependant, les problèmes incluant à la fois raisonnement épistémique et temporel n'ont pas bénéficié d'autant d'attention. Nous considérons que la combinaison de ces aspects est nécessaire pour pouvoir pleinement représenter des problèmes que l'on rencontre dans le monde réel. C'est pourquoi nous avons cherché à les associer pour étendre les possibilités de planification.

Cette thèse présente plusieurs contributions à la planification multi-agent, temporelle et épistémique. Pour en montrer l'intérêt, nous avons créé et étudié une variante du problème du bavardage incluant des contraintes temporelles. Nous avons appelé cette famille de problèmes « temporal epistemic gossip problems » (problèmes de bavardage temporels et épistémiques). Nous montrons que le problème de l'existence d'un plan pour cette famille est NP-complet.

Étendre une logique épistémique pour lui permettre de représenter également des problèmes temporels pourrait vite mener à une complexité importante. Nous avons choisi de commencer en ajoutant du parallélisme à la logique dynamique d'assignations propositionnelles (dynamic logic of propositional assigments, DL-PA) pour créer la logique dynamique d'assignations propositionnelles parallèles (dynamic logic of parallel propositional assignments, DL-PPA), tout en conservant la complexité PSPACE-complète des problèmes de satisfiabilité et de vérification de modèle.

Nous avons ensuite tenté une autre approche, traduire un problème de planification épistémique en problème de planification classique. Le problème de planification est décrit dans une logique statique, EL-O, en définissant le problème d'existence de plan dans le méta-langage. Nous montrons que ces problèmes sont dans PSPACE via une traduction polynomiale de la planification épistémique avec EL-O vers de la planification classique. Dans le même esprit, en se basant sur notre modèle de planification épistémique nous avons créé des domaines et problèmes PDDL. Nous avons testé ces problèmes avec des planificateurs de la compétition internationale de planification (IPC) de 2018 afin de connaitre le temps nécessaire pour obtenir une solution. La comparaison des problèmes de chaque domaine nous a montré que les planificateurs actuels ne sont pas assez performants pour traiter des problèmes épistémiques avec cette approche.

Multi-agent planning problems involving reasoning over the agents' knowledge are well studied. The same is the case for problems with temporally extended actions. However, problems with both epistemic and temporal reasoning drew less attention. We consider that the combination of these two aspects is essential to fully model real world problems. That is why we wanted to combine them to extend the possibilities of planning.

This thesis contains several contributions to multi-agent, temporal and epistemic planning. To motivate such an enterprise, we designed and studied a variation of the gossip problem with temporal constraints. We called this family of problems "temporal epistemic gossip problems". We show that the solvability problem for this family is NP-complete.

Extending an epistemic logic in order to enable the modeling of temporal problems into classical planning problems may lead to a high complexity. We choose to start by adding parallelism to dynamic logic of propositional assignments (DL-PA), resulting in dynamic logic of parallel propositional assignments (DL-PPA), while keeping the PSPACE-complexity of the satisfiability and model-checking problems.

We then tried another approach, namely to translate epistemic planning problems into classical planning problems. The planning problem is first modeled in a static logic, EL-O, defining solvability in the meta-langage. We show that these problem are in PSPACE via a polynomial translation from EL-O epistemic planning to classical planning. By means of this translation we created some domains and problems in PDDL. These problems were tested with planners from the international planning competition (IPC 2018) to retrieve the amount of time for a solution. The comparison between the problems of each domain showed that more research on epistemic planning is needed.