THE UNIVERSITY OF MELBOURNE

MASTER'S THESIS

What you get is what you see: Decomposing Epistemic Planning using Functional STRIPS

Author:
Guang HU
https://orcid.org/0000-0003-3629-8040

Supervisors:
A/Prof. Tim MILLER
Dr. Nir LIPOVETZKY

A thesis submitted in fulfillment of the requirements for the degree of Master of Philosophy

in the

School of Computing and Information Systems

March 19, 2020

Declaration of Authorship

I, Guang HU, declare that this thesis titled, "What you get is what you see: Decomposing Epistemic Planning using Functional STRIPS" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Guang Hu

Date: 13th of August, 2019

THE UNIVERSITY OF MELBOURNE

Abstract

School of Engineering
School of Computing and Information Systems

Master of Philosophy

What you get is what you see: Decomposing Epistemic Planning using Functional STRIPS

by Guang HU

Epistemic planning — planning with knowledge and belief — is essential in many multi-agent and human-agent interaction domains. Most state-of-the-art epistemic planners solve this problem by compiling to propositional classical planning, for example, generating all possible knowledge atoms, or compiling epistemic formula to normal forms. It is noted that the compilations are typically exponentially larger than the original problem. However, these methods become computationally infeasible as problems grow. In addition, those methods only works on propositional variables in discrete domains.

In this thesis, we decompose epistemic planning by delegating epistemic logic reasoning to an external solver. We do this by modelling the problem using *functional STRIPS*, which is more expressive than standard STRIPS and supports the use of external, black-box functions within action models. Exploiting recent work that demonstrates the relationship between what an agent 'sees' and what it knows, we allow modellers to provide new implementations of externals functions. These define what agents see in their environment, allowing new epistemic logics to be defined without changing the planner. As a result, the capability and flexibility of the epistemic model itself are increased, as our model is able to avoid exponential pre-compilation steps and handle logics from continuous domains. We ran evaluations on well-known epistemic planning benchmarks to compare with an existing state-of-the-art planner, and on new scenarios based on different external functions. The results show that our planner scales significantly better than the state-of-the-art planner which we compared against, and can express problems more succinctly.

Preface

This thesis has been written in School of Computing and Information Systems, School of Engineering, The University of Melbourne. The whole thesis is based on a manuscript under reviewed for publication In *Artificial Intelligence: Special Issue on Epistemic Planning*. The archived version of that paper can be accessed through: https://arxiv.org/abs/1903.11777. I declare that I am the primary author and have contributed to more than 50% of that paper.

Acknowledgements

I would like to express my gratitude and appreciation to all the support I got throughout my study. First of all, I wish to thank my principal supervisor, A/Prof Tim Miller, who encourages me to pursuit my research. His institution and insights in the Epistemic Planning is remarkable, which inspires this work.

Secondly, I would like to thank Dr Nir Lipovetzky, who is my co-supervisor. As part of my research, I needed to implement my model on Nir's work. I wish to thank all his support, both practical and academic, and his faith in me.

Thirdly, I would like to thank Professor Adrian Pearce to be my advisory chair by providing insightful and constructive feedback and suggestions during my research.

Then, I would like to thank all my friends, and colleagues from the Autonomy research group for their support during my study. Especially, I wish to thank my dear friend, Xin Zhang, who provides mental support and helped me with my grammar and editing skills.

Last but not least, I would like to thank my parents for mental and financial support. My gratitude for their love and support cannot be expressed in words. This thesis would not have been possible without their help and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automated planning is a model-based approach to study sequential decision problems (Geffner and Bonet, 2013). Planning model describes the environment and the agents with concise planning languages, such as STRIPS (Fikes and Nilsson, 1971) or PDDL (Haslum et al., 2019). They then submit the description of the model to a general problem solver in order to find a sequence of actions to achieve some desired goal states. The description of the problem, in general, tracks the changes in the state of the environment. However, in many scenarios, an agent needs to reason about the *knowledge* or *beliefs* of other agents in the environment. This concept is known as *epistemic planning* (Bolander and Andersen, 2011), a research topic that brings together the knowledge reasoning and planning communities.

Epistemic logic is a formal account to perform inferences and updates about an agent's own knowledge and beliefs, including group and common knowledge in the presence of multiple agents (Hintikka, 1962). Epistemic planning is concerned about action theories that can reason not only about variables representing the state of the world, but also the beliefs and knowledge that other agents have about those variables. Thus, epistemic planning intends to find the best course of action taking into account practical performance considerations when reasoning about knowledge and beliefs (Bolander and Andersen, 2011). Bolander and Andersen (2011) first used event-based models to study epistemic planning in both single and multi agent environments, and gave a formal definition of epistemic planning problems using Dynamic Epistemic Logic (DEL) (Bolander, 2017).

There are typically two frameworks in which epistemic planning are studied. The first is to use DEL. This line of research investigates the decidability and complexity of epistemic planning and studies what type of problems it can solve (Bolander, 2014; Bolander, Jensen, and Schwarzentruber, 2015; Bolander, 2017) The second is to extend existing planning languages and solvers to epistemic tasks (Muise, Belle, et al., 2015; Muise, Miller, et al., 2015; Kominis and Geffner, 2015; Kominis and Geffner, 2017; Wan et al., 2015; Huang et al., 2017; Wu, 2018; Le et al., 2018). In this thesis, we take the latter approach.

The complexity of epistemic planning is undecidable in the general case. Thus, one of the main challenges of epistemic planning concerns computational efficiency. The dominant approach in this area relies on compilations. These solutions pre-compile epistemic planning problems into classical planning problems, using off-the-shelf classical planners to find solutions (Muise, Belle, et al., 2015; Muise, Miller, et al., 2015; Kominis and Geffner, 2015; Kominis and Geffner, 2017); or pre-compile the epistemic formulae into specific normal forms for better performance during

search (Huang et al., 2017; Wu, 2018). Such approaches have been shown to be fast at planning, but the compilation is computationally expensive. For example, from Muise, Belle, et al. (2015)'s result, the planner takes less than a second to solve a problem, but the compilation time is more than half a minute when as the problem size increases.

This thesis departs from previous approaches in two significant ways. First, we propose a model that exploits recent insights by defining what an agent knows as a function of what it "sees" (Cooper et al., 2016; Gasquet, Goranko, and Schwarzentruber, 2014). Cooper et al. (2016) define "seeing relations" for agent i as modal operators S_i that "see" whether a proposition is true, and then define knowledge K for agent i of a proposition p as $K_i p \leftrightarrow p \wedge S_i p$; that is, if p is true and agent sees p, then it knows p. Thus, the seeing modal operator is equivalent to the 'knowing whether' operator (Fan, Wang, and Ditmarsch, 2015; Miller et al., 2016). We generalise the notion of seeing relations to *perspective functions*, which are functions that determine which variables an agent sees in its environment. The domain of variables can be discrete or continuous, not just propositional. The basic implementation of perspective functions is just the same as seeing relations, however, we show that by changing the definition of perspective functions, we can establish new epistemic logics tailored to specific domains, such as Big Brother Logic (Gasquet, Goranko, and Schwarzentruber, 2014), a logic about visibility and knowledge in two-dimensional Euclidean planes, or Social-media Network, a logic about abstract "seeing" relation of who befriends with whom.

Second, we show how to integrate perspective functions within functional STRIPS models as *external functions* (Francès et al., 2017). External functions are black-box functions implemented in any programming language (in our case, C++), that can be called within action models. Epistemic reasoning is delegated to external functions, where epistemic formulae are evaluated lazily, avoiding the exponential blow-up from epistemic formulae present in other compilation-based approaches. This delegation effectively decomposes epistemic reasoning from search, and allows us to implement our approach in any functional STRIPS planner that supports external functions. Further, the modeller can implement new perspective functions that are tied to specific domains, and our model will use those functions to evaluate desired epistemic relations, effectively defining new external solvers. While perspective functions can be generic and implement e.g. Muise, Belle, et al.'s finite-depth, propositional epistemic logic Muise, Belle, et al., 2015 or Le et al. observability relations Le et al., 2018, our experience suggests that tailoring to specific domains results in more understandable and elegant models and perspective functions.

In our experiments we use a width-based functional STRIPS planner (Francès et al., 2017) that is able to evaluate the truth value of epistemic fluents with external solvers, and solve a wide range of epistemic problems efficiently, including but not limited to, nested knowledge, distributed knowledge and common knowledge. We also show how modellers can implement different perspective functions as external functions in the functional STRIPS language, enabling the use of domain-dependent epistemic

logics. Departing from propositional logic give us flexibility to encode expressive epistemic formulae concisely. We compare our approach to a state-of-the-art epistemic planner that relies on a compilation to classical planning (Muise, Belle, et al., 2015). We implement two benchmarks problems, *Corridor* (Cooper et al., 2016) and *Grapevine* (Muise, Belle, et al., 2015) to compare the computational performance, and choose another two complex domains, Big Brother Logic and Social-media Network, to examine the expressiveness of our model. The results show that, unlike in the compilation based approaches, the depth of nesting and the number of agents does not affect our performance, which means our approach avoids the exponential blow up due to our lazy evaluation of epistemic formulae.

In the following chapters we give a brief background on both epistemic logic and epistemic planning (Section 2). We then introduce a new model, the *agent perspective model* (Section 3). We discuss implementation details using a functional STRIPS planner (Section 4), and report experiments on several well-known benchmarks, along with two new scenarios to demonstrate the expressiveness of the proposed approach (Section 5).

Chapter 2

Background

In this work, we target *Multi-Agent Epistemic Planning* (MEP) problems, inspired by Muise, Belle, et al. (2015)'s definition. The formal definition is given in Section 4.2. As for preliminaries of this work, we briefly introduce the three main areas: (1) classical planning; (2) epistemic logic; and (3) epistemic planning.

2.1 Classical Planning

Planning is the model-based approach to action selection in AI, where the model is used to reason about which action an agent should do next (Geffner and Bonet, 2013). Models vary depending on the assumptions imposed on the dynamics of the world, from classical where all actions have deterministic instantaneous effects and the world is fully known, up to temporal or POMDP models, where actions have durations or belief distributions about the state of the world. Models are described concisely through declarative languages such as STRIPS and PDDL (Fikes and Nilsson, 1971; McDermott, 2000), general enough to allow the encoding of different problems, while at the same time revealing important structural information that allow planners to scale up. In fact, most planners rely on exploiting the structure revealed in the action theory to guide the search of solutions, from the very first general problem solver (Simon and Newell, 1963) up to the latest computational approaches based on SAT, and heuristic search (Rintanen, 2012; Richter and Westphal, 2010). On the other hand, declarative languages have limited the scope of planning, as certain environments representing planning models are difficult to encode declaratively, but are easily defined through simulators such as the Atari video games (Bellemare et al., 2013). Consequently, a new family of width-based planners (Lipovetzky and Geffner, 2012; Lipovetzky and Geffner, 2014; Lipovetzky and Geffner, 2017a) have been proposed, broadening the scope of planning and have been shown to scale-up even when the planning model is described through simulators, only requiring the exposure of the state variables, but not imposing any syntax restriction on the action theory (Francès et al., 2017), where the denotation of some symbols can be given procedurally through simulators or by external theories.

In this thesis we focus on epistemic planning, which considers the *classical planning model* as a tuple $\mathcal{S} = \langle S, s_0, S_G, Act, A, f, c \rangle$ where S is a set of states, $s_0 \in S$ is the initial state, $S_G \subseteq S$ is the set of goal states, Act is the set of actions, $A(s)$ is the subset actions applicable in state s, f is the transition function so that $f(a, s)$ represents the state s' that results from doing action

a in the state s, and $c(a, s)$ is a cost function. The solution to a classical planning model $\mathcal{S}$, called a plan, is a sequence of actions that maps the initial state into a goal state, i.e., $\pi = \langle a_0, \ldots, a_n \rangle$ is a plan if there is a sequence of states $s_0, \ldots, s_{n+1}$ such that $a_i \in A(s_i)$, $s_{i+1} = f(a_i, s_i)$ for $i = 0, \ldots, n$ and $s_{n+1} \in S_G$. The cost of plan π is given by the sum of action costs $c(a_i, s_i)$ and a plan is optimal if there is no plan with smaller cost.

As one of the language that can model classical planning, STRIPS can represent any classical planning problem as a tuple $P = \langle F, O, I, G \rangle$, where: F is the set of all possible facts or propositions, O the set of all actions, $I \subseteq F$ a set of all true facts in the initial situation, and $G \subseteq F$ a set of facts that needs to be true as the goal conditions. Since there is no operator cost in STRIPS, the plan is optimal if there is no plan with less operators taken.

Besides the model, a solver, which is also called a planner, plays another important role in planning. One of the most successful computational approaches to planning is heuristic search. Besides a search algorithm, the key feature which distinguishes planners is the heuristic function (Helmert and Domshlak, 2009). To achieve good performance, the heuristic functions should be as informed as possible. For example, one of the current best-performing planner, LAMA, uses a landmark-based heuristic derived from the model (Richter and Westphal, 2010) along with other delete-relaxation heuristics (Geffner and Bonet, 2013). The downside is that most heuristics require the model to be encoded in STRIPS or PDDL, but this restricts the expressiveness of the models significantly.

The standard planning languages and solvers do not support the use of procedures or external theories. One exception is the FS planner (Frances and Geffner, 2015), that supports the Functional STRIPS language (Geffner, 2000), where the denotation of (non-fluent) function symbols can be given extensionally by means of atoms, or intentionally by means of procedures. Procedures also appear as an extension of PDDL under the name of semantic attachments (Dornhege et al., 2009). The reason why procedures are not "first-class citizens" in planning languages is that there was no clear way to deal with them that is both general and effective. Recently, a new family of algorithms called BFWS(R) have been proposed in classical planning known as Width-based planning (Lipovetzky and Geffner, 2012).

The BFWS(R) family of planners (Francès et al., 2017) have been shown to scale up even in the presence of functional symbols defined procedurally.

The F-STRIPS planner using BFWS(R) has been compared over 380 classical planning problems with respect of the performance of FF* (Helmert, 2006), LAMA-11 (Richter and Westphal, 2010), and BFWS(f_5) (Lipovetzky and Geffner, 2017b) in their work (Francès et al., 2017). The results show that the F-STRIPS BFWS(R^*_G) planner performs as good as BFWS(f_5), which relies on a STRIPS model, and slightly better than LAMA-11. It is worth to mention that FF and LAMA have been the top-performing planners for last 15 years, and BFWS(f_5) has been shown to win the agile track in the last International Planning Competition 2018, and is a state-of-the-art planner for classical planning. The F-STRIPS BFWS(R) planner thus can cope with externally defined functional symbols while performing well with respect to other planners. In addition, they compare among five planners within

BFWS(R) family, each of them using different $R(s)$ functions. Since the focus of this thesis is not on evaluating performance with different $R(s)$ function, we choose the simplest BFWS(R_0) planner as our developing tool[1].

2.2 Epistemic Logic

In this section, we give the necessary preliminaries for epistemic logic – the logic of knowledge. Knowledge in a multi-agent system is not only about the environment, but also about the agents' knowledge about the environment, and of agents' knowledge of others' knowledge about the environment, and so on.

Fagin et al. (2003) provides a formal definition of epistemic logic as follows. Given a set of countable set of all primitive propositions $Prop = \{p_1, p_2, ...\}$ and a finite set of agents $Agt = \{a_1, a_2, ...\}$, the syntax for epistemic logic is defined as:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi,$$

in which $p \in Prop$ and $i \in Agt$.

$K_i\varphi$ represents that agent i knows proposition φ, $\neg$ means negation and $\wedge$ means conjunction. Other operators such as disjunction and implication can be defined in the usual way.

Fagin et al. (2003) define the semantics of epistemic logic using Kripke structures, as standard in modal logic. A Kripke structure is a tuple $M = (S, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n)$ where:

- S is a non-empty set of states;
- $\pi(s)$ is an interpretation function: $Prop \mapsto \{true \mid false\}$; and
- $\mathcal{K}_1 \ldots \mathcal{K}_n$ represents the *accessibility relations* over states for each of the n agents in Agt.

Given a state s and a proposition p, the evaluation of p over s is $\pi(s)(p)$. If and only if p is true in s, then $\pi(s)(p)$ is $true$. $\mathcal{K}_i$ for agent i is a binary relation over states, which is the key to reason about knowledge. For any pair of states s and t, if $(s,t) \in \mathcal{K}_i$, then we can say agent i cannot distinguish between t and s when in state s. In other words, if agent i is in the current real-world state s, the agent can consider t as the current state based on all the information it can obtain from state s. With this definition of Kripke structure, we can define the semantics of knowledge.

Given a state s, a proposition p, a propositional formula φ and a Kripke structure M, the truth of two basic formulae are defined as follows:

- $(M, s) \models p$ iff $\pi(s)(p)$=$true$
- $(M, s) \models K_i\varphi$ iff $(M, t) \models \varphi$ for all t such that $(s,t) \in \mathcal{K}_i$

$(M, s) \models p$ represents p is true at (M, s), which means, the evaluation of p in state s, $\pi(s)(p)$ must be true. Standard propositional logic rules define

[1]The planner is available through https://github.com/aig-upf/2017-planning-with-simulators. We used options `-driver sbfws -options bfws.rs=none`

conjunction and negation. $(M, s) \models K_i\varphi$ is defined by formula φ being true at all worlds t reachable from s via the accessibility relation $\mathcal{K}_i$. This allows knowledge to be nested; for example, K_aK_bp represents that agent a knows that agent b knows p, which means p is true at all worlds reachable by applying accessibility relation $\mathcal{K}_a$ followed by $\mathcal{K}_b$.

From these basic operators, the concept of group knowledge can be defined. For this, the grammar above is extended to:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi \mid E_G\varphi \mid D_G\varphi \mid C_G\varphi,$$

in which $p \in Prop$, $i \in Agt$, and G is a non-empty set of agents such that $G \subseteq Agt$.

$E_G\varphi$ represents that everyone in group G knows φ and $C_G\varphi$ represents that it is *commonly known* in group G that φ is true, which means that everyone knows φ, and everyone knows that everyone knows φ, *ad infinitum*. $D_G\varphi$ represents *distributed* knowledge, which means we combine the knowledge of the set of agents G such that G knows φ, even though it may be that no individual in the group knows φ.

The semantics for these group operators are defined as follows:

- $(M, s) \models E_G\varphi$ iff $(M, s) \models K_i\varphi$ for all $i \in G$;
- $(M, s) \models C_G\varphi$ iff $(M, t) \models \varphi$ for all t that are $G{-}reachable$[2] from s
- $(M, s) \models D_G\varphi$ iff $(M, t) \models \varphi$ for all t such that $(s, t) \in \cap_{i\in G}\mathcal{K}_i$

By definition, $(M, s) \models E_G\varphi$ will be true, if and only if, φ is known by all agents in G. Common knowledge in world s, $(M, s) \models C_G\varphi$ is defined as: in all worlds t, which are reachable by following the accessibility relations of all agents in G, φ is true. For distributed knowledge, $(M, s) \models D_G\varphi$ is true, if and only if, in all the possible worlds that all agents from G consider possible, φ is true. It might be easier to think in the reverse direction: if and only if, we eliminate worlds that any agent in G knows to be impossible, and φ is true in all the remaining possible worlds, then we say $D_G\varphi$ is true in (M, s).

2.2.1 Seeing and Knowledge

Recently Gasquet, Goranko, and Schwarzentruber (2014) noted the relationship between what an agent sees and what it knows. They define a more specific task of logically modeling and reasoning about cooperating tasks of vision-based agents, which they called *Big Brother Logic* (BBL). Their framework models multi-agent knowledge in a continuous environment of vision, which has great potential applications such as reasoning over cameras inputs, autonomous robots and vehicles. They introduce the semantics of their model and its extensions on natural geometric models.

In their scenario, agents are stationary cameras in a Euclidean plane $\mathbb{R}^2$, and they assume that those cameras can see anything in their sight range, and they do not block others' sight. They extend Fagin et al. (2003)'s logic by noting that, at any point in time, what an agent knows, including nested knowledge, can be derived directly from what it can see in the current

[2] We say t is $G{-}reachable$ from s if t can reach s with K steps of accessible relations, where $K \leq 1$ (Fagin et al., 2003).

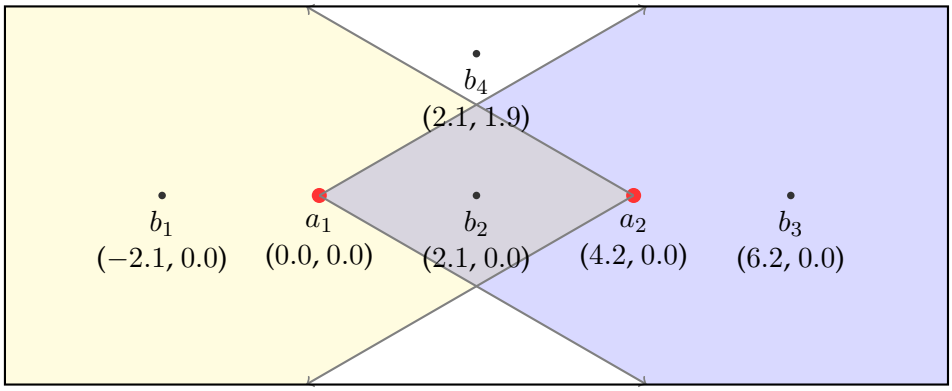


FIGURE 2.1: Example for Big Brother Logic

state. In brief, instead of Kripke frames, they define a *geometric model* as (pos, dir, ang), in which:

- $pos : Agt \to \mathbb{R}^2$
- $dir : Agt \to U$
- $ang : Agt \to [0, 2\pi)$

where U is the set of unit vectors of $\mathbb{R}^2$, the pos function gives the position of each agent, the dir function gives the direction that each agent is facing, and the function ang gives the angle of view for each agent.

A model is defined as (pos, ang, D, R), in which pos and ang are as above, D is the set of possible dir functions and R is the set of equivalence relations, one for each agent a, defined as:

$$R_a = \{(dir, dir') \in D^2 \mid \text{ for all } b \neq a, dir(b) = dir'(b)\}$$

The definition above shows the equivalence relation for agent a between the worlds (pos, dir, ang) and (pos, dir', ang), as all the agents that a can see are considered as the same.

In this context, standard propositional logic is extended with the binary operator $a \triangleright b$, which represents that "a sees b". This is defined as:

- $(pos, ang, D, R), dir \vDash a \triangleright b$ iff $pos(b) \in C_{pos(a),dir(a),ang(a)}$,

in which $C_{pos(a),dir(a),ang(a)}$ is the field of vision that begins at $pos(a)$ from direction $dir(a)$ and covers $\frac{ang(a)}{2}$ degrees in both clockwise and counter-clockwise directions.

From this, they show the relationship between seeing and knowing. For example, $K_a(b \triangleright c)$ is defined as $a \triangleright b \wedge a \triangleright c \wedge b \triangleright c$.

Figure 2.1 shows an example with two agents, a_1 and a_2, and model $((0.0, 0.0), 60°, D, R)$ and $((4.2, 0.0), 60°, D, R)$ respectively, along with four objects, b_1, b_2, b_3 and b_4. Based on the current state, for agent a_1, we have: $(pos, ang, D, R), dir \vDash a_1 \triangleright a_2$; $(pos, ang, D, R), dir \vDash a_1 \triangleright b_2$; and, $(pos, ang, D, R), dir \vDash a_1 \triangleright b_3$. In addition, a pair of directions $(0°, -5°) \in R_a$ shows that for agent a, the seeing relations stay unchanged, meaning that $((0.0, 0.0), 60°, -5°, R)$ is accessible from $((0.0, 0.0), 60°, 0°, R)$ in their logic.

Gasquet, Goranko, and Schwarzentruber (2014) also define a common knowledge operator, in a similar manner to that of Fagin et al. (2003)'s

definition based on $G{-}rearchable$ worlds. In Figure 2.1, the formula $\mathsf{C}_{\{a_1,a_2\}}a_1 \triangleright b_2$ holds by their definition, because a_1 and a_2 can both see b_2, and can both see each other. From those, we can deduce based on geometric law that a_1 can see "a_2 can see b_2" due to a_1 can see both a_2 and b_2, and a_2 can see b_2. Furthermore, from previous sentence and a_2 can see a_1, we get that a_2 can see "a_1 can see 'a_2 can see b_2' ", and etc. Thus, a common knowledge has been formed.

Cooper et al. (2016)'s idea of modelling an agent's knowledge bases on what it sees, and generalise it to seeing propositions, rather than to just to seeing other agents in a Euclidean plane. They extended Fagin et al. (2003)'s definition by adding an extra formula α that describes formula (proposition) that can be seen:

$$\begin{aligned} \alpha &::= p \mid S_i p \mid S_i \alpha \\ \varphi &::= \alpha \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i \varphi, \end{aligned}$$

in which $p \in PVar$ (the set of propositional *variables*) and $i \in Agt$. The grammar of α defines visibility relations. $S_i\alpha$ reads as "agent i sees α". Note the syntactic restriction that agents can only see atomic propositions or nestings of seeing relationships that see atomic propositions.

From this, they define knowledge using the equivalences $K_i p \leftrightarrow p \wedge S_i p$ and $K_i \neg p \leftrightarrow \neg p \wedge S_i p$. This tight correspondence between seeing and knowledge is intuitive: an agent knows p is true if p is true and the agent can see the variable p. Such a relationship is the same as the one between knowing something is true and *knowing whether* something is true (Miller et al., 2016; Fan, Wang, and Ditmarsch, 2015). In fact, in early drafts of the work available online, Cooper et al. (2016), S_i was written as KW_i, aptly named the "knowing whether" operator.

Comparing these two bodies of work, Gasquet, Goranko, and Schwarzentruber (2014) use a geometric model to represent the environment and derive knowledge from this by checking the agents' line of sight. Their idea is literally matching the phrase "Seeing is believing". However, their logic is constrained only to vision in physical spaces. While in Cooper et al. (2016)'s world, the seeing operator applies to propositional variables, and thus visibility can be interpreted more abstractly; for example, "seeing" (hearing) a message over a telephone.

This thesis generalises seeing relations to perspective functions, which are domain-dependent functions defining what agents see in particular states. The result is more flexible than seeing relations, and allows Big Brother Logic to be defined with a simple perspective function, as well as new perspective functions for other domains; for example, Big Brother Logic in three-dimensional planes, or visibility of messages on a social network.

2.3 Epistemic Planning

Bolander and Andersen (2011) introduce the concept of epistemic planning, and define it in both single agent and multi-agent domain. Their planning framework is defined in *dynamic epistemic logic* (DEL) (Van Ditmarsch, Der Hoek, and Kooi, 2007), which has been shown to be undecidable in general, but with decidable fragments (Bolander, Jensen, and Schwarzentruber,

2015). In addition, they also provided a solution to PSPACE-hardness of the plan verification problem. This formalism has been used to explore theoretical properties of epistemic planning; for example, Engesser et al. (2017) used concepts of perspective shifting to reason about other's contribution to joint goals. Along with implicit coordination actions, their model can solve some problems elegantly without communication between agents.

Since epistemic planning is formalized in DEL, there has been substantial works on DEL-based planning. However, in this thesis, our focus is on the design, implementation, and evaluation of planning tools, rather than on logic-based models of planning. Therefore, in this section, we focus on research in those works which are providing practical solutions and evaluating their model and planning algorithm on computational results on benchmarks.

A handful of researchers in the planning field focus on leveraging existing planners to solve epistemic problems. Muise, Belle, et al. (2015) proposed an approach to multi-agent epistemic planning with nested beliefs, non-homogeneous agents, co-present observation, and the ability for one agent to reason as if it were the other. Generally, they compiled an epistemic logic problem into a classical planning problem by grounding epistemic fluents into propositional fluents and using additional conditional effects of actions to enforce desirable properties of beliefs.

Muise, Belle, et al. (2015) define MEP as a tuple $\langle \mathcal{P}, \mathcal{A}, Ag, \mathcal{I}, \mathcal{G} \rangle$, where, similar to STRIPS, $\mathcal{P}$ is the set of propositions (facts), $\mathcal{A}$ is the set of actions (operators), $\mathcal{I}$ is the initial state, $\mathcal{G}$ is the goal-condition, and Ag is the set of agent. They handle epistemic relation as epistemic literals following grammar: "$\phi ::= p \mid B_i\phi \mid \neg\phi$". The literal "$B_i\phi$" reads "agent i believes ϕ". Two limitations of this approach are: a finite depth of nested beliefs; and, no disjunction. Then, they take three processes to convert their model to STRIPS and keep their solution sound and complete. Firstly, they maintain the deductive closure by removing negations and adding logical consequences of all positive effects. Then, they address the uncertainty by removing belief l of negation of an unsure effects and any other beliefs that can be used to deduce l. At last, they apply conditioned mutual awareness to conditional effects to enrich their model to handle beliefs update on different level.

They evaluate their approach on the *Corridor* (Kominis and Geffner, 2015) problem and the *Grapevine* problem, a combination of *Corridor* and *Gossip* (Herzig and Maffre, 2015). Their results show that their approach is able to solve the planning task within a typically short time, but the compilation time to generate fluents and conditional effects is exponential in the size of the original problem. In particular, the compilation time is exponential in both the number of agents and the maximum depth of epistemic relations.

Kominis and Geffner (2015) adapt methods from partially-observable planning for representing beliefs of a single agent, and convert that method to handle multi-agent setting. Their idea is to maintain the problem's Kripke structure. By their definition (adapting STRIPS), an epistemic planning problem P is a tuple $\langle A, F, I, O, N, U, G \rangle$. In their model, A is the set of agent identifiers, I is a set of possible initial states rather than just one initial state in STRIPS. Then, instead of tracking the problem by only updating actual states, they combine the Kripke structure with the state at time step

t and possible initial states by using beliefs, $B(t)$. A set of beliefs $B(t)$ contains a set of $B(s_i, t)$ for each possible initial state s_i. And in each $B(s, t)$, there are the actual state $v(s, t)$ and indistinguishable relations $r_i(s, t)$ between current state and possible initial state for each agent i. By doing so, they are able to construct the Kripke structure from the initial state for each agent.

To keep the Kripke structure during the search, they define three kinds of action sets: O, N and U. O represents all physical actions, which is the same O as in classical planning. The operator updates the actual current state s base on deterministic transition function. The action set N denotes a set of **sense** actions, which can be used to infer knowledge. The **sense** actions will iterate on each agent and remove the inconsistent belief relations according to the given formula. The last action set U is used to update beliefs according to the fact φ. The **update** will keep the possible previous state that agrees with φ and delete the rest.

Overall, they are able to use their model to encode epistemic planning problems, and can convert their model to a classical planning model using standard compilation techniques for partially-observable planning. They evaluated their model on the *Muddy Children*, *Sum* and *Word Rooms* (Kominis and Geffner, 2015) domains. As far as we can tell from their experiments, they keep the depth of the epistemic relation fixed at one and vary the number of agents or the number of rooms. Their results show that their model is able to solve all cases presented with different suitable planners. However, from their result, there is a exponential growth on the time consumption due to the increase scale of the problems. In addition, following their approach, the modeller must specify the **sense** action for each pair of (i, φ) and **update** action for each φ.

Since Kominis and Geffner (2015), and Muise, Belle, et al. (2015)'s approach the problem in a similar way (compilation to classical planning) their results are similar in general. However, the methods they used are different. Therefore, their work and results have diverse limitations and strengths. For Muise, Belle, et al. (2015)'s work, they managed to model nested beliefs without explicit or implicit Kripke structures, which means they can only represent literals, while Kominis and Geffner (2015)'s work is able to handle arbitrary formulae. Furthermore, Muise, Belle, et al. (2015)'s model does not have the strict common initial knowledge setting found in Kominis and Geffner (2015), and does not have the constraint that all action effects are commonly known to all the agents. Therefore, Muise, Belle, et al. (2015)'s model allows them to model beliefs, which might be not what is actual true in the world state, rather just model knowledge. In other words, they can handle different agents having different beliefs about the same fluent.

More recently, rather than compiling epistemic planning problems into classical planning, Huang et al. (2017) built a native multi-agent epistemic planner, and proposed a general representation framework for multi-agent epistemic problems (Huang et al., 2017). They considered the whole multi-agent epistemic planning task from a third person point of view. In addition, based on a well-established concept of belief change algorithms (both revision and update algorithms), they design and implemented an algorithm to encode belief change as the result of planning actions. Catering to their ideas and algorithms, they developed a planner called MEPK. They

evaluated their approach with *Grapevine*, *Hexa Game* and *Gossip*, among others. From their results, it is clear that their approach can handle a variety of problems, and performance on some problems is better than other approaches. While this approach is different from Kominis and Geffner (2015) and Muise, Belle, et al. (2015), it still requires a compilation phase before planning to re-write epistemic formula into a specific normal form called *alternating cover disjunctive formulas* (ACDF) (Hales, French, and Davies, 2012). The ACDF formula is worst-case exponentially longer than the original formula. The results show that this step is of similar computational burden as either Kominis and Geffner (2015) or Muise, Belle, et al. (2015). In addition, building a native planner to solve an epistemic planning problem makes it more difficult to take advantage of recent advances in other areas of planning.

Le et al. (2018) build two prototypical epistemic forward planners, named EFP and PG-EFP. They define their planning language as a tuple $(\mathcal{AG}, \mathcal{F}, A, O)$, where $\mathcal{AG}$ is the set of agent identifiers and $\mathcal{F}$ is the set of fluents. Action set A and observability statement set O compose the actions and effects. In their A, they specify preconditions and three possible effects: ontic, sensing and announcement, which work as follows: ontic effects change the state (actual world); sensing action reveals truth value of some fluent f; and, announcement action announces the truth value of some fluent f, which affects set O. In the set O, they propose two kinds of observation: fully observable actions by $observes$; and partiallyobservable by $aware_$of. Their semantics are defined by transition functions, which can handle three types of agents awareness of the execution of one action: fully, partially and oblivious. They implement those two planners based on their model with two different search algorithm, BFS and heuristic search for EFP and PG-EFP respectively. They propose the definition of *epistemic planning graph*, and use it as their main data structure in the search. As for the PG-EFP, they derive heuristic values directly from the structure of the epistemic planning graph. They compared their planner against Muise, Belle, et al. (2015)'s and Huang et al. (2017)'s solution on *Corridor*, *Collaboration-and-communication* (Kominis and Geffner, 2015), and *Assemble Line* (Huang et al., 2017). From their comparison, we find EFP does not suffer from the exponential blow up on depth of the epistemic relations, but it is affected by the length of the plan. As for PG-EFP, it does perform better than EFP on several problems, but the expressiveness is not as good as EFP.

Overall, our work is different from traditional epistemic planning approach as we do not have to convert epistemic planning problems into classical planning problems (Muise, Belle, et al., 2015; Kominis and Geffner, 2015), which means our model does not require a costly pre-compilation step. Compared to more recent work (Huang et al., 2017; Le et al., 2018), our approach works on any F-STRIPS planner. Further, our approach approach can work on continuous domains, supports common and distributed knowledge, and has no limit on the depth on epistemic formula. Moreover, the key concepts and semantics of our model are different from all current solutions in epistemic planning, although some parts of our intuition are from the latest works in epistemic logic reasoning field (Gasquet, Goranko, and Schwarzentruber, 2014; Cooper et al., 2016).

Chapter 3

A Model of Epistemic Planning using Perspectives

In this chapter, we define the syntax and semantics of our *agent perspective model*. Our idea is based on that of Big Brother Logic (Gasquet, Goranko, and Schwarzentruber, 2014) and Cooper et al. (2016)'s seeing operators. The syntax and semantics of our model are introduced, including nested knowledge for single agent and distributed knowledge and common knowledge for group of agents. Then, we show the soundness and completeness of our logic.

3.1 Language

Extending Cooper et al. (2016)'s idea of seeing propositional variables, our model is based on a model of functional STRIPS (F-STRIPS) (Francès et al., 2017), which uses variables and domains, rather than standard propositions found in classical planning. We allow agents to see variables with discrete and continuous domains.

3.1.1 Model

We define an epistemic planning problem as a tuple $(Agt, V, D, \mathcal{O}, \mathcal{I}, \mathcal{G})$, in which Agt is a set of agents, V is a set of variables, D stands for domains for each variable, in which domains can be discrete or continuous, $\mathcal{I}$ and $\mathcal{G}$ are the initial state and goal states respectively, and both of them are also bounded by V and D. Specifically, they should be assignments for some or all variables in V, or relations over these. $\mathcal{O}$ is the set of operators, with argument in the terms of variables from V.

3.1.2 Epistemic Formulae

DEFINITION 3.1.1:
Goals, actions preconditions, and conditions on conditional effects, are epistemic formulae, defined by the following grammar:

$$\varphi ::= R(v_1, \ldots, v_k) \mid \neg\varphi \mid \varphi \wedge \varphi \mid S_i v \mid S_i\varphi \mid K_i\varphi,$$

in which: R is k-arity "domain-dependent" relation symbol, which takes k grounded values and returns true or false indicating whether the relation $R(v_1, \ldots, v_k)$ with $v_1, \ldots, v_k \in V$ is true or not in the current state; $S_i v$ with $v \in V$ and $S_i\varphi$ are both visibility formulae, and $K_i\varphi$ is a knowledge formula.

3.1.3 Domain Dependent Formulae

Domain-dependent formulae do not only include basic mathematical relations, but also relational terms defined by the underlying planning language, which means it can have relations between variables. For example, based on the scenario from Figure 2.1, "$pos(a_1)$=$(0,0)$" is a true formula expressing the position of agent a_1, while "$pos(a_1)$=$pos(a_2)$" is false. In Chapter 4, we discuss the use of *external functions* in F-STRIPS, which allows more complex relations, or even customized relations, as long as they are defined in the external functions. For example, we can define a domain dependent relation in an external function to compare distance between objects, called $@far\_away(pos(i), pos(j), pos(k))$. This external function takes three coordinates as input, and returns a Boolean value, whether distance between i and j is longer than i and k. In the scenario displayed in Figure 2.1, the relation $@far\_away(pos(a_1), pos(b_4), pos(b_2))$ would be true, while relation $@far\_away(pos(a_1), pos(b_1), pos(b_2))$ would be false, since b_1 and b_2 are at the same distance to a_1. The definition of this function is delegated to a function implemented in a programming language such as C++, and the planner is unaware of its semantics. However, for the remainder of this chapter, we will ignore the existence of external functions, and return to them in our implementation chapter.

3.1.4 Visibility Formula

An important concept adapted from Cooper et al. (2016) is "seeing a proposition". Let p be a proposition, "agent i knows whether p" can be represented as "agent i sees p". Their interpretation on this is: either p is true and i knows that; or, p is false and i knows that. With higher-order observation added, it gives us a way to reason about others' epistemic viewpoint about a proposition without actually knowing whether it is true. Building on this concept, our "seeing" operator allows us to write formulae about visibility: $S_i v$ and, $S_i \varphi$.

The seeing formula represents two related interpretations: seeing a variable; or seeing a formula. The formula $S_i v$ can be understood as variable v has some value, and no matter what value it has, agent i can see the variable and knows its value. On the other hand, seeing a relation is trickier. The formula $S_v \varphi$ can be interpreted as: for formula φ, no matter whether it is true or false, agent i *knows whether* it is true or not. To make sure i knows whether φ is true or not, the evaluation for this seeing formula is simply that agent i sees all the variables in that relation.

For example, in Figure 2.1, using the notation defined in Section 3.1.1, $S_{a_1} pos(b_2)$ can be read as "agent a_1 sees variable $pos(b_2)$", and it means agent a_1 *knows* b_2's location, wherever that location is. In the case of seeing a domain-dependent formula, $S_{a_1}(far\_away, pos(a_1), pos(b_4), pos(b_2))$ can be read as "agent a_1 sees the relation $far\_away(pos(a_1), pos(b_4), pos(b_2))$", which is: "agent a_1 knows whether b_4 is farther away from a_1 than b_2".

3.1.5 Knowledge Formula

In addition to the visibility operator, our language supports the standard knowing operator K_i. Following Cooper et al. (2016)'s idea on relation between knowledge and visibility, we define knowledge as: $K_i \varphi \leftrightarrow \varphi \wedge S_i \varphi$.

That is, for i to know φ is true, it needs to be able to see φ, and φ needs to be true. In other words, if you can see something and it is true, then, you know it is true.

3.2 Semantics

Our model decomposes the planning model from the knowledge model, and as we will see in our implementation, our planner delegates the knowledge model to an external solver. Therefore, in this section, we define the semantics of that knowledge solver. The novel part of this model is the use of perspective functions, which are functions that define which variables an agent can see, instead of using full Kripke structures. From this, a rich knowledge model can be built up independent of the planning domain.

DEFINITION 3.2.1:
A model M is defined as $M = (V, D, \pi, f_1, \ldots, f_n)$.

V is a finite set of variables and D is a function that maps each variable to its (non-empty) domain. One example is $D(v_1) = \{e_1, \ldots, e_n\}$ for variable v_1. From V and D, we define a *state* $s \in \mathcal{S}$ as a set of variable assignments, denoted as $\{v_1{=}e_1, \ldots, v_k{=}e_k\}$. We use $s(v_i)$ to represent the value of v_i in state s. There are two kinds of states, namely *global state* and *local state*. A global state is a complete assignment for all variables in V. Whereas, a local state, which represents an individual agent's perspective of the global state, can be either a partial or a complete assignment. If v_i is not in a local state, $s(v_i) = null$. The set of all states (local and global) is denoted as $\mathcal{S}$. π is a set of evaluation functions, such that for $\pi_k \in \pi$, $\pi_k : R \to \mathcal{S} \to \{true \mid false\}$, where R_k is a set of atomic relational symbols of the form $R(v_1, \ldots, v_n)$. If π_k is applied to a local state in which a variable v_i occurs in $R(v_1, \ldots, v_n)$ but is not in the local state, then π_k must be evaluated to false. Finally, $f_1, \ldots, f_n : \mathcal{S} \to \mathcal{S}$ are the *agents' perspective functions* that given a state s, will return the local state from agents' perspectives.

A perspective function, $f_i : \mathcal{S} \to \mathcal{S}$ is a function that takes a state and returns a subset of that state, which represents the part of that state that is visible to agent i. These functions can be nested, such that $f_j(f_i(s))$ represents agent i's perspective of agent j's perspective, which can be just a subset of agent j's actual perspective. The following properties must hold on $f_1, \ldots, f_n$ for all $i \in Agt$ and $s \in \mathcal{S}$:

1. $f_i(s) \subseteq s$
2. $f_i(s) = f_i(f_i(s))$
3. If $s \subseteq s'$, then $f_i(s) \subseteq f_i(s')$

First, we give the semantic for propositional formulae. Let any variable be denoted as v_i, and R any k-ary domain dependent formula:

(a) $(M, s) \models R(v_1, \ldots, v_k)$ iff $\pi(R, s(v_1), \ldots, s(v_k)){=}true$

Relations are handled by the evaluation function $\pi(s)$. The relation R is evaluated by getting the value for each variable in s, and checking whether R holds or not. Other propositional operators are defined in the standard way.

Then, we have the following formal semantics for visibility, where $i \neq j$:

(b) $(M, s) \models S_i\ v$ iff $\exists\ x \in D(v)$, such that, $(v{=}x) \in f_i(s)$

(c) $(M, s) \models S_i\ R(v_1, \ldots, v_k)$ iff $\forall v \in \{v_1, \ldots, v_k\},\ (M, s) \models S_i v$

(d) $(M, s) \models S_i \neg\varphi$ iff $(M, s) \models S_i\ \varphi$

(e) $(M, s) \models S_i\ (\varphi \wedge \psi)$ iff $(M, s) \models S_i\ \varphi$ and $(M, s) \models S_i\ \psi$

(f) $(M, s) \models S_i\ S_j\ \varphi$ iff $(M, f_i(s)) \models S_j\ \varphi$

(g) $(M, s) \models S_i\ S_i\ \varphi$ is always $true$

(h) $(M, s) \models S_i\ K_j\ \varphi$ iff $(M, f_i(s)) \models K_j\ \varphi$

In (b), $S_i v$, read "Agent i sees variable v", is true if and only if v is visible in the state $f_i(s)$. That is, an agent sees a variable if and only if that variable is in its perspective of the state. Similarly in (c), an agent knows whether a domain-dependent formula is true or false if and only if it can see every variable of that formula. For example, in Figure 2.1, $S_{a_1} b_1$ is false and $S_{a_1} b_2$ is true, which is because b_2 is in a_1's perspective (blue area), while b_1 is not. The remainder of the definitions simply deal with logical operations in our language. It is worth noticing that in (d) $S_i\ \neg\varphi$ is in fact equivalent to $S_i\varphi$, because both φ and $\neg\varphi$ contain the exactly same variables. Besides, the semantic of $S_i\ \neg\varphi$ is "i knows whether $\neg\varphi$ is true or not", which is the same as semantics of $S_i\varphi$: "i knows whether φ is true or not". This effectively just defines that "seeing" a formula means seeing its variables. Furthermore, seeing a conjunction $S_i(\varphi \wedge \psi)$ in our model is equivalent as $(S_i\varphi \wedge S_i\psi)$ in (e). We can simply prove this by constructing a $(m + n)$-ary relation θ for any m-ary relation ψ and n-ary relation φ following the truth value of $\varphi \wedge \psi$. We validate soundness and completeness of our definition in Section 3.3. Disjunction works the same due to "$\psi \vee \varphi \equiv \neg(\neg\psi \wedge \neg\varphi)$".

The above items (f) and (g) are both about nested seeing relations. In the case of (f), whether $S_i S_j \varphi$ ($i \neq j$) is true is equivalent to whether $S_j\varphi$ holds in agent i's perspective of the world s. However in the case of $S_i S_i \varphi$, as noted by Cooper et al. Cooper et al., 2016, an agents always sees what it sees, to $S_i S_i \varphi$ is a validity. We also prove it in Theorem 3.3.3.

Then, the truth condition for knowledge is:

(i) $(M, s) \models K_i\ \varphi$ iff $(M, s) \models \varphi$ and $(M, s) \models S_i\varphi$

The definition as shown in (i) follows the idea in Cooper et al. (2016)'s paper on the relation between knowledge and seeing: agent i knows φ if and only if the formula is true at (M, s) and agent i sees it. Using the same example as previously, $K_{a_1} @ far\_away(pos(a_1), pos(b_2), pos(b_3))$ is false, even if a_1 does see b_2 and b_3, b_2 is not farther than b_3 to a_1. While, $K_{a_1} @ far\_away(pos(a_1), pos(b_3), pos(b_2))$ is true. In addition, combining negation semantic from seeing relation, we have $K_i\varphi \vee K_i\neg\varphi \leftrightarrow (\varphi \wedge S_i\varphi) \vee (\neg\varphi \wedge S_i\neg\varphi) \leftrightarrow S_i\varphi$, which is also similar as the idea of "knowing whether" K_w in Miller et al. (2016)'s paper.

3.3 Validation

In this section, firstly, we discuss some basic properties of our logic. Then, we prove the soundness of our model, following by showing the completeness of our model except for validity (tautology and contradiction) formula. Hereafter, we use this as an example in our proofs: a state s contains two variables x and y, and domain for both x and y is $\{1, 2, 3\}$. Therefore, all the possible states **S** in this example for our model contains s_1=$\{x$=$1\}$, s_2=$\{x$=$2\}$, s_3=$\{x$=$3\}$ or s_0=$\{\}$. This example is visualized in Figure 3.1, where k_1 and k_2 represent two Kripke structures: agent i does not know value of x in any world; and, agent i knows value of x in all three worlds, respectively.

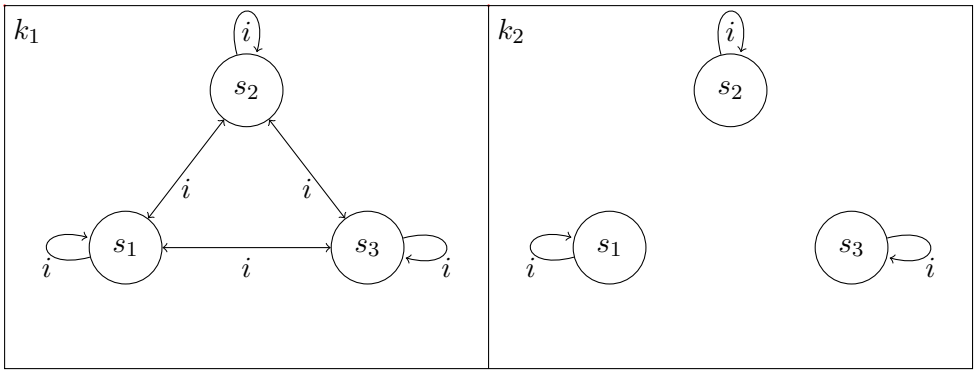


FIGURE 3.1: Two Examples of Kripke Structure

Theorem 3.3.1. The S5 axioms of epistemic logic are valid in this logic. That is, the following axioms hold:

(K)	$K_i(\varphi \to \psi)$	$\to$	$K_i\varphi \to K_i\psi$
(T)	$K_i\varphi$	$\to$	φ
(4)	$K_i\varphi$	$\to$	$K_iK_i\varphi$
(5)	$\neg K_i\varphi$	$\to$	$K_i\neg K_i\varphi$

Proof. We first consider axiom (T). By our semantics, $(M, s) \models K_i\,\varphi$ is true, if and only if, both $(M, s) \models \varphi$ and $(M, s) \models S_i\varphi$ are true. Therefore, it is trivial that axiom (T) holds.

For (K), based on our definition of knowledge, we have $(M, s) \models K_i(\varphi \to \psi)$ is equivalent to $(M, s) \models S_i(\varphi \to \psi)$ and $(M, s) \models (\varphi \to \psi)$. Then, by our semantics, we have that $(M, s) \models S_i(\varphi \to \psi)$ is equivalent to $(M, s) \models S_i\neg\varphi$ or $(M, s) \models S_i\psi$. From propositional logic, $\varphi \to \psi$ is equivalent to $\neg\varphi \vee \psi$. We combine $(M, s) \models \neg\varphi$ with $(M, s) \models S_i\varphi$ to get $(M, s) \models \neg K_i\varphi$ and similarly for ψ to get $(M, s) \models K_i\psi$, which is equivalent to $(M, s) \models K_i\varphi \to K_i\psi$ from propositional logic.

To prove (4) and (5), we use the properties of the perspective function f_i. The second property shows, a perspective function for agent i on state s converges after the first nested iteration, which means $(M, f_i(s)) \equiv (M, f_i(f_i(s)))$. Therefore, whenever $(M, f_i(s)) \models \varphi$, then φ also holds in $(M, f_i(f_i(s)))$, implying that $K_i\varphi$ holds too (4). According to (i) in our semantics, we have $K_i\neg K_i\varphi \leftrightarrow K_i(\neg S_i\varphi \vee \neg\varphi) \leftrightarrow (S_i\neg S_i\varphi \wedge \neg S_i\varphi) \vee (S_i\neg\varphi \wedge \neg\varphi)$. In combining our semantics on seeing relation (g) and (d), we have $(M, s) \models K_i\neg K_i\varphi \equiv (M, s) \models (true \wedge \neg S_i\varphi) \vee (S_i\varphi \wedge \neg\varphi)$, which

is in fact equivalent to $(M, s) \models \neg S_i\varphi \vee \neg\varphi$ and thus matches the premise "$\neg K_i\varphi \leftrightarrow \neg S_i\varphi \vee \neg\varphi$". Hence, (5) holds. □

Theorem 3.3.2. Let M be any instance of our model, there exists at least one corresponding Kripke structure M^K.

Proof. We can prove this theorem by constructing one corresponding Kripke structure M^K for any M.

Let any instance from our model be $M = (V, D, \pi, f_1, \ldots, f_n)$ and its corresponding Kripke structure $M^K = (\mathcal{S}^K, \pi^K, \mathcal{K}_1, \ldots, \mathcal{K}_n)$. As $\mathcal{S}$ in Kripke structure syntax is a set of all possible worlds (states), we create a set of propositions for all the variables $v \in V$ by taking the Cartesian product $V \times D(v)$, and then assigning true/false value to each proposition in that product. Therefore, any global (complete) state s from M can find an identical s' in M^K by assigning false value to all the propositions except those indicating assignments $(v = e)$ in s. It is trivial that the evaluation function π is identical to π^K. So, we only have to define the accessibility relations in the Kripke structures, $\mathcal{K}_1, \ldots, \mathcal{K}_n$, to represent perspective functions. Since $\mathcal{K}_i$ contains all the accessibility relations for agent i, and each relation is a pair of possible worlds (states), we now construct $\mathcal{K}_1, \ldots, \mathcal{K}_n$ by following steps:

- For each agent i, $\mathcal{K}_i$ is:
  - For each possible state s in M^K:
    1. Find i's perspective of world s by $f_i(s)$;
    2. For each possible world s' in M^K where $f_i(s) = f_i(s')$, add the pair of accessibility relation (s, s') in $\mathcal{K}_i$

For each state s, we create accessibility relations (s, s') into $\mathcal{K}_i$ by pairing s with all possible worlds s' that agree on all of the "visible" variables for agent i. In other words, agent i considers s' is possible given the current world s, as i is unsure about value of those variables i cannot "see". For example in Figure 3.1: given any state s in $\mathbf{S}$, if the agent i's perspective state $f_i(s)$ is one of s_1, s_2 and s_3 according to the current state s, then, the accessible relation of between current state s and possible world t in M^K would be added one of (s, s_1), (s, s_2), (s, s_3), respectively, because there is only one world possible when the value of x exists in $f_i(s)$. To be specific, $\mathcal{K}_i$ is $\{(s_1, s_1), (s_2, s_2), (s_3, s_3)\}$, which is visualised as k_2 in Figure 3.1. While if $f_i(s)$ is s_0, then $\mathcal{K}_i$ would be $\{(s, s_1), (s, s_2), (s, s_3)\}$ for any s from $\mathbf{S} \setminus \{s_0\}$. That is, agent i is not able to see x, which means i is not able to tell which world i is in. Therefore, $\mathcal{K}_i$ is $\{(s_1, s_1), (s_1, s_2), (s_1, s_3), (s_2, s_1), (s_2, s_2), (s_2, s_3), (s_3, s_1), (s_3, s_2), (s_3, s_3)\}$, as visualised in Figure 3.1 (k_1). □

Although we can use our model to construct a Kripke structure, the resulting Kripke structure does not contain any information that ignored by our model. Our construction above is only a full structure without any disjunctive knowledge (information that implies on a constraint on the variable value without identifying the value). We discuss this further in Section 3.5

We need to define the seeing operator before we discuss soundness of our logic on seeing relations. We adapt definition of "knowing whether" relation as our understanding of "seeing" relation in Kripke structure. In

epistemic logic, possibility relation $\mathcal{K}_i$ is usually required to be an equivalence relation on $\mathcal{S}$ (Fagin et al., 2003), which means K_i has the following properties: reflexive, symmetric and transitive.

DEFINITION 3.3.1:
(Seeing formula in Kripke structure). Let any Kripke structure be M^K, any agent be i and any formula in our grammar be φ:

- $(M^K, s) \models S_i\, v$ iff $\exists e\ such\ that\ v{=}e \in s$, and $\forall t\ such\ that\ (s,t) \in \mathcal{K}_i$, $(M^K, s) \models v{=}e \Leftrightarrow (M^K, t) \models v{=}e$.
- $(M^K, s) \models S_i\, \varphi$ iff $\forall t\ such\ that\ (s,t) \in \mathcal{K}_i$, $(M^K, s) \models \varphi \Leftrightarrow (M^K, t) \models \varphi$.

Deriving from Fan, Wang, and Ditmarsch (2015)'s definition of "knowing whether", we define the seeing operator in Kripke semantics for $S_i v$ and $S_i \varphi$. An agent i sees variable v in (M^K, s), if and only if, there exists a value e such that, $v{=}e$ is agreed be true ($v{=}e \in s$) by all the worlds that i considers possible given the current world is s. In other words, i sees v, if and only if, in all the possible worlds, v has a constant value e. The definition for $S_i\varphi$ is more intuitive: $S_i\varphi$ holds, if and only if, for all of i's possible worlds from s agree on the truth value of φ. Since $\mathcal{K}_i$ is reflexive, symmetric and transitive, $\forall t\ such\ that\ (s,t) \in \mathcal{K}_i$ should cover all the reachable worlds for agent i at state s, which is the same definition as Fan, Wang, and Ditmarsch (2015)'s "$\mathcal{M}, s \models \Delta_i\varphi \Leftrightarrow\ for\ all\ t_1, t_2\ such\ that\ s \rightarrow_i t_1, s \rightarrow_i t_2 : (\mathcal{M}, t_1 \models \varphi \Leftrightarrow \mathcal{M}, t_2 \models \varphi)$".

Theorem 3.3.3. $(M^K, s) \models S_i S_i \varphi$ is always true.

Proof. We prove this by contradiction. Assume $S_i S_i \varphi$ is false for some (M^K, s), and denote all worlds agent i consider possible at state s as $\mathcal{K}_i(s)$. Then, by Definition 3.3.1, we have:

- $(M^K, s) \models \neg S_i S_i \varphi \equiv \exists t_1, t_2\ such\ that\ (s,t_1), (s,t_2) \in \mathcal{K}_i,\ (M^K, t_1) \models S_i\varphi \wedge (M^K, t_2) \models \neg S_i\varphi$

which means there exist worlds t_1, t_2 from $\mathcal{K}_i(s)$ such that $S_i\varphi$ is true in t_1 and false in t_2. Separately, we have:

- $(M^K, t_1) \models S_i\varphi \equiv \forall t_1'\ such\ that\ (t_1, t_1') \in \mathcal{K}_i,\ (M^K, t_1) \models \varphi \Leftrightarrow (M^K, t_1') \models \varphi$ and
- $(M^K, t_2) \models S_i\varphi \equiv \exists t_2', t_2''\ such\ that\ (t_2, t_2'), (t_2, t_2'') \in \mathcal{K}_i,\ (M^K, t_2') \models \varphi \wedge (M^K, t_2'') \models \neg\varphi$

This means that for all worlds in $\mathcal{K}_i(t_1)$ agrees on value of φ, and for all worlds in $\mathcal{K}_i(t_2)$, there exist t_2', t_2'', such that φ is true in t_2' and false in t_2''. Since $\mathcal{K}_i$ is symmetric and transitive, we have $(s,t_1) \leftrightarrow (t_1,s)$ and $(t_1,s) \wedge (s,t_2) \rightarrow (t_1,t_2)$. Therefore, all of (s,t_1), (s,t_2) and (t_1,t_2) are in $\mathcal{K}_i$, which means $\mathcal{K}_i(s) \equiv \mathcal{K}_i(t_1) \equiv \mathcal{K}_i(t_2)$. Then, we have that $\forall t_1 \in \mathcal{K}_i(t_1),\ (M^K, t_1) \models \varphi \Leftrightarrow (M^K, t_1') \models \varphi$, which contradicts our earlier assertion that $\exists t_2', t_2'' \in \mathcal{K}_i(t_2),\ (M^K, t_2') \models \varphi \wedge (M^K, t_2'') \models \neg\varphi$.

Therefore, there does not exist a model (M^K, s) that makes $\neg S_i S_i \varphi$ satisfiable, meaning that $S_i S_i \varphi$ is always true. □

Theorem 3.3.4. (Soundness). Let s be the current state, M be our model and M^K be the corresponding Kripke structure defined using approach in Theorem 3.3.2. The following hold, where $i \neq j$:

(1) If $(M, s) \models S_i\ v$, then $(M^K, s) \models S_i\ v$

(2) If $(M, s) \models S_i\ R(v_1, \ldots, v_k)$, then $(M^K, s) \models S_i\ R(v_1, \ldots, v_k)$

(3) If $(M, s) \models S_i \neg\varphi$, then $(M^K, s) \models S_i \neg\varphi$

(4) If $(M, s) \models S_i\ (\varphi \wedge \psi)$, then $(M^K, s) \models S_i\ (\varphi \wedge \psi)$

(5) If $(M, s) \models S_i\ S_j\ \varphi$, then $(M^K, s) \models S_i\ S_j\ \varphi$

(6) Both $(M, s) \models S_i\ S_i\ \varphi$ and $(M^K, s) \models S_i\ S_i\ \varphi$ are always $true$.

(7) If $(M, s) \models S_i\ K_j\ \varphi$, then $(M^K, s) \models S_i\ K_j\ \varphi$

(8) If $(M, s) \models K_i\ \varphi$, then $(M^K, s) \models K_i\ \varphi$

Proof. The proof for (1) is based on our semantics for visibility of a variable v: agent i sees v in (M, s), if and only if, there exists some value e that $(v{=}e) \in f_i(s)$. Due to the process of constructing a Kripke model outlined in Theorem 3.3.2, the existing value e means $(M, s) \models S_i v$ is consistent in all states that i considers possible from $\mathcal{K}_i$. By the definition of $S_i v$ in Kripke semantics, for all the possible worlds, the value of v agrees on e if and only if $(M^K, s) \models S_i\ v$ holds. Therefore, our semantics for $S_i v$ in (M, s) holds for (M^K, s) as well.

For example in Figure 3.1: for any state s in **S**, if $(M, s) \models S_i\ v$ holds (k_2 in the figure), which means $f_i(s)$ is equal to s in any of s_1, s_2, s_3, $(M^K, s) \models S_i\ v$ will hold, as there is only one accessible relation in $\mathcal{K}_i$ for each state s which is one of (s, s_1), (s, s_2), (s, s_3), respectively, and value of v is agreed as 1, 2, 3, respectively. If $(M, s) \models S_i\ v$ is false (k_1 in the figure), which means agent i cannot see variable v and $f_i(s)$ is s_0, then, $(M^K, s) \models S_i\ v$ will not hold, as $\mathcal{K}_i$ would be $\{(s_1, s_1), (s_1, s_2), (s_1, s_3), (s_2, s_1), (s_2, s_2), (s_2, s_3), (s_3, s_1), (s_3, s_2), (s_3, s_3)\}$, and variable v does not be agreed on one value in all states.

The remaining proofs are straightforward. Since the evaluation function π is almost identical for both M and M^K, and each value in $R(v_1, \ldots, v_k)$ is the same due to (1), the result for $R(v_1, \ldots, v_k)$ is the same in both both M and M^K. Therefore, (2) in this theorem holds. Then, all remaining in M holds in M^K because (1) and (2) hold, except (6) holds as Theorem 3.3.3 and (g) in Section 3.2.

□

Theorem 3.3.5. (Completeness). Let s be the current state, M be any instance in our model and M^K be its corresponding Kripke structure constructed following steps in Theorem 3.3.2. The following hold, where $i \neq j$:

(1) If $(M^K, s) \models S_i\ v$, then $(M, s) \models S_i\ v$, except when $|D(v)| = 1$ and i cannot see v.

(2) If $(M^K, s) \models S_i\ R(v_1, \ldots, v_k)$, then $(M, s) \models S_i\ R(v_1, \ldots, v_k)$, except when $R(v_1, \ldots, v_k) \vdash \bot \vee \top$, and $\exists v_t \in \{v_1, \ldots, v_t\}, (M, s) \models \neg S_i v_t$.

(3) If $(M^K, s) \models S_i \neg\varphi$, then $(M, s) \models S_i \neg\varphi$, except when $\varphi \vdash \bot \vee \top$, and $(M, s) \models \neg S_i \varphi$.

(4) If $(M^K, s) \models S_i\ (\varphi \wedge \psi)$, then $(M, s) \models S_i\ (\varphi \wedge \psi)$, except when $(\varphi \wedge \psi) \vdash \bot$, and $(M, s) \models \neg S_i\ (\varphi \wedge \psi)$.

(5) If $(M^K, s) \models S_i\ S_j\ \varphi$, then $(M, s) \models S_i\ S_j\ \varphi$, except when $S_j\ \varphi \vdash \top$, and $(M, s) \models \neg S_i\ S_j\ \varphi$.

(6) Both $(M^K, s) \models S_i\ S_i\ \varphi$ and $(M, s) \models S_i\ S_i\ \varphi$ are always $true$.

(7) If $(M^K, s) \models S_i\ K_j\ \varphi$, then $(M, s) \models S_i\ K_j\ \varphi$, except when $K_j\ \varphi \vdash \top$, and $(M, s) \models \neg S_i\ K_j\ \varphi$.

(8) If $(M^K, s) \models K_i\ \varphi$, then $(M, s) \models K_i\ \varphi$, except when $\varphi \vdash \top \vee \bot$, and $(M, s) \models \neg K_i\ \varphi$.

Proof. Following the definition of seeing formula in Kripke structure given above, $(M^K, s) \models S_i\ v$ means for all worlds that i considers possible given the current world s, the value of v is the same. According to the steps to build corresponding M^K from M, all of the unseen variables will result in accessible worlds with all possible values. Therefore, if v is agreed on some value for all i's possible worlds given s, v must be seen by i in s, unless the domain for v contains only one value, which means in all accessible worlds v would be agreed on that one value. For example: let v be a variable with domain $\{e\}$, which means "$v = e$" is a validity. Even if agent i cannot see v, but in all possible worlds, the value of v agree on e. Therefore, $(M^K, s) \models S_i v$ holds, while $(M, s) \models S_i v$ does not. However, if the domain of v becomes $\{e, e'\}$, then, all possible worlds that accessible for i will not agree on v, because in half of the worlds v is e, while in other half, v is e'. Therefore, $v{=}e$ will be in $f_i(s)$ if $(M^K, s) \models S_i\ v$ holds and the size of v's domain is larger than 1. Then, following the definition of $(M, s) \models S_i\ v$, $v{=}e$ exists in $f_i(s)$, then (1) holds.

For example in Figure 3.1: if the $\mathcal{K}_i$ contains only (s, s_1), (s, s_2) and (s, s_3) in M^K, then there exists an assignment as $v{=}1$, $v{=}2$, $v{=}3$, respectively in $f_i(s)$ according to s, which makes $(M, s) \models S_i\ v$ hold (K_2 in the figure). If the $\mathcal{K}_i$ contains other accessible relations, such as, shown in k_1 from the figure, $\{(s_1, s_1), (s_1, s_2), (s_1, s_3), (s_2, s_1), (s_2, s_2), (s_2, s_3), (s_3, s_1), (s_3, s_2), (s_3, s_3)\}$. Then, $(M^K, s) \models S_i\ v$ does not hold as the value of v can be any of 1 or 2 or 3, as well as $(M, s) \models S_i\ v$.

Because (1) holds and $\pi(s)$ are almost identical in both M^K and M, then (2) holds. (7) is proved in the same way as in Theorem 3.3.4. The proof for (3), (5) and (8) are straightforward by using (1) and (2), given (4) holds. Therefore, we prove (4) first.

We show (4) by following the definition of the seeing operator in Kripke semantics: if $(M^K, s) \models S_i\ (\varphi \wedge \psi)$ holds, which means all worlds that i consider possible in (M^K, s) agree on the truth value of $\varphi \wedge \psi$. There are only two scenarios such agreement can be achieved: either, $(M^K, s) \models S_i\ \varphi$ and $(M^K, s) \models S_i\ \psi$ hold; or, $\varphi \wedge \psi$ is false ($\varphi \wedge \psi$ is a contradiction). If both φ and ψ can be seen by i in (M^K, s), following (2), both $(M, s) \models S_i\ \varphi$ and $(M, s) \models S_i\ \psi$ will hold, which means $S_i(\varphi \wedge \psi)$ holds. However, if $\varphi \wedge \psi$ is a contradiction, then $\varphi \wedge \psi$ is false. Following Definition 3.3.1, $(M^K, s) \models S_i\ (\varphi \wedge \psi)$ holds. However, if agent i cannot see all variables in

φ and ψ, then one of $(M, s) \models S_i\,\varphi$ and $(M, s) \models S_i\,\psi$ will not hold, which means $(M, s) \models S_i\,(\varphi \wedge \psi)$ will not hold. Therefore, (4) holds if $(\varphi \wedge \psi)$ is not a contradiction.

(3) holds as $S_i\neg\varphi \equiv S_i\varphi$, and $S_i\varphi$ holds by induction. (5), (7) and (8) are straightforward by induction. (6) holds as Theorem 3.3.3 and (g) in Section 3.2. Therefore, our model is complete for all situations except in which formulae inside seeing operators that contain non-seen variables are validities. □

By combining (3) and (4), it is straightforward to show that our model is complete for disjunctive formulae $S_i(\psi \vee \varphi)$ as long as $(\psi \vee \varphi)$ is not a validity. In addition to (3) and (4), a validity or contradiction could also arise in all the other semantics (5, 6, 8). For example on (5), $(M^K, s) \models S_jS_iv$ holds, while $(M, s) \models S_jS_iv$ does not, if $M^K \vdash S_iv$ and agent i does not see agent j sees φ.

A consequence of this theorem is that the *necessitation* inference rule is not admissible. This rule, of the form $\varphi \vdash K_i\varphi$, says that for any theorem φ, every agent should know it. The reason that Kripke semantics can handle validity is because the semantics checks whether all possible worlds agree on the truth value of the formula. Any theorem will be true in every world of the model, so also true in every world reachable by any Kripke structure. However, our model reduces reasoning on unseen variables by ignoring them in the agent's local perspective. Any validity φ containing an unseen variable will be a theorem of the logic, but $K_i\varphi$ will not hold in our model. We do not believe this is such as major limitation. First, in planning, it is neither typical nor efficient to expect models handle all valid propositions, even in propositional models. Second, if a modeller is concerned that models contain validities, they could use techniques such as resolution to check for validities and contradictions prior to planning, and encode these as constants in perspective functions, which would makes agents see the validity without seeing all its variables.

3.4 Group Knowledge

From the basic visibility and knowledge definitions, in this section, we define group operators, including distributed and common visibility or knowledge.

3.4.1 Syntax

We extend the syntax of our language with group operators:

$$\varphi ::= \psi \mid \neg\varphi \mid \varphi \wedge \varphi \mid ES_G\alpha \mid EK_G\varphi \mid DS_G\alpha \mid DK_G\varphi \mid CS_G\alpha \mid CK_G\varphi,$$

in which G is a set (group) of agents, ψ is any formula in our language for single agent defined in this chapter, and α is a variable v or formula φ.

Group formula $ES_G\alpha$ is read as: everyone in group G sees a variable or a formula α, and $EK_G\varphi$ represents that everyone in group G knows φ. DK_G is the distributed knowledge operator, equivalent to D_G in Section 2.2, while DS_G is its visibility counterpart: someone in group G sees. Finally,

CK_G is common knowledge and CS_G common visibility: "it is commonly seen".

3.4.2 Semantics

Let G be a set of agents, φ a formula, and α either a formula or a variable, then we can define the semantics of these group formula as follows:

- $(M, s) \vDash ES_G\ \alpha$ iff $\forall i \in G$, $(M, s) \vDash S_i\ \alpha$
- $(M, s) \vDash EK_G\ \varphi$ iff $(M, s) \vDash \varphi$ and $(M, s) \vDash ES_G\ \varphi$
- $(M, s) \vDash DS_G\ \alpha$ iff $(M, s') \vDash \alpha$, where $s' = \bigcup_{i \in G} f_i(s)$
- $(M, s) \vDash DK_G\ \varphi$ iff $(M, s) \vDash \varphi$ and $(M, s) \vDash DS_G\ \varphi$
- $(M, s) \vDash CS_G\ \alpha$ iff $(M, s') \vDash \alpha$, where $s' = cf(G, s)$
- $(M, s) \vDash CK_G\ \varphi$ iff $(M, s) \vDash \varphi$ and $(M, s) \vDash CS_G\ \varphi$,

in which $cf(G, s)$ is state reached by applying composite function $\bigcap_{i \in G} f_i$ until it reaches its fixed point. That is, the fixed point s' such that $cf(G, s') = cf(G, \bigcap_{i \in G} f_i(s'))$.

Reasoning about common knowledge and common visibility is more complex than other modalities. Common knowledge among a group is not only everyone in the group shares this knowledge, but also everyone knows others know this knowledge, and so on, *ad infinitum*. The infinite nature of this definition leads to definitions that are intractable in some models.

However, due to our restriction on the definition of states as variable assignments and our use of perspective functions, common knowledge is much simpler. This is based on the fact that each time we apply composite perspective function $\bigcap_{i \in G} f_i(s)$, the resulting state is either a proper subset of s (smaller) or is s. By this intuition, we can limit common formula in finite steps.

The fixed point is a recursive definition. However, the following theorem shows that this fixed point always exists (even if it is empty, in that case, there is no common knowledge), and the number of iterations is bound by the size of $|s|$, the state to which it is applied.

Theorem 3.4.1. Function $cf(G, s)$ converges on a fixed point $s' = cf(G, s')$ within $|s|$ iterations.

Proof. First, we prove convergence is stable; that is, when $s' = cf(G, s')$, further "iterations" will result in s'; that is, $s' = cf(G, cf(G, s'))$. Let $s_i = cf(G, s_i)$, where i is the number of iterations. Then, we have $s_{i+1} = cf(G, s_i) = s_i$. Since $s_{i+1} = s_i$, we have $s_{i+2} = cf(G, s_i)$, which means $s_{i+2} = s_i$. Via induction, we have that for all $k \geq i$, $s_k = s_i$. Therefore, once we reach convergence, it remains.

Next, we prove convergence within $|s|$ iterations. By the intuition and definition of the perspective functions, $f_k(s) \subseteq s$, and $s_{i+1} = (\bigcap_{k \in G} f_k(s_i))$, we have $s_{i+1} \subseteq s_i$. Then, as we prove for the first point, if $s_i = s_{i+1}$, then we have reached a fixed point and no further iterations are necessary.

Therefore, the worst case is when $s_{i+1} \subsetneq s_i$ and $|s_i| - |s_{i+1}| = 1$. There are most $|s|$ such worst case iterations until s_i converges on an empty set. Therefore, the maximum number of iterations is $|s|$. □

For each of the iterations, there are $|G|$ local states in group G that need to be applied in the generalised intersection calculation, which can be done in polynomial time, and there there are at most $|s|$ steps. So, a poly-time algorithm for function cf exists.

3.5 A Brief Note on Expressiveness

The intuitive idea about perspective functions is based on what agents can see, as determined by the current state. The relation between $t = f_i(s)$ and s corresponds roughly to accessibly relations $(s, t) \in \mathcal{K}_i$ in Kripke semantics. However, only focusing on what an agent exactly knows/sees means overlooking those variables that agents are uncertain about. Perspective functions return one partial world that the agent i is certain about, rather than a set of worlds that i considers possible. The advantage is that applying a perspective function provides us with only one state, rather than multiple states in Kripke semantics, preventing the explosion in model size. In addition, our perspective-based model allows reasoning with group knowledge, such that joint knowledge and distributed knowledge reasoning becomes set operations over the agents' local worlds.

However, the reduced complexity loses information on the "uncertain" variables. Theoretically, $t = f_i(s)$ from our model is a set intersection from all the t that $(s, t) \in \mathcal{K}_i$. This eliminates disjunctive knowledge about variables; the only uncertainty being that an agent does not see a variable. For example, in the well-known *Muddy Children*[1] problem (Fagin et al., 2003), the knowledge is not only generated by what each child can see by the others' appearance, which is modelled straightforwardly using perspective functions, but also can be derived from the questions made by their father and the response by other children. From their perspective, they would know exactly m children are dirty, which can be handled by our model, as they are certain about it. While by the k-th time the father asked and no one responds, they can use induction and get the knowledge that at least k children are dirty. By considering that there are two possible worlds, where the number of dirty children is m or $m + 1$, Kripke structures keep both possible worlds until $m+1$ steps. If we use a variable to represent the number of possible muddy children, our model cannot keep these two worlds. Therefore, although our model can handle preconditions and goals with disjunction, such as $K_i\,[(v = e_1) \vee (v = e_2)]$, it cannot *store* such disjunction in its "knowledge base". Rather, agent i knows v's value is e_1 or it knows it is e_2.

Despite this, possible worlds can be *represented* in our model, using the same approach taken by Kominis and Geffner (2015) of representing

[1] There are n children, and m of them with mud on their forehead. They can see other's forehead except their own. In order to help them find out whether themselves are muddy or not, their father can help them by asking one question to them all for k times: do you know you are muddy or not?

Kripke structures. We can then define perspective functions that implement Kripke semantics. However, this could easily result in an exponentially large model and would not add the expressiveness required for most of epistemic planning problems. For the the Muddy Children example, instead of having m (the number of dirty children) as an integer variable, we can model it as a series of propositions indicating the number of dirty children, such as $m_0, \ldots, m_n$. To model uncertain information about m, the underlying perspective function could eliminate all the propositions that agent is certain to be false. To be specific, if propositions m_3 and m_4 remain in agent i's perspective of the world, then, i knows m is either 3 or 4. Therefore, the children asking their father for the kth time will result in removal of m_k from the state until all the children only have m_m in their local state.

A comparison on expressiveness between our model and other works are listed in Table 3.1.

	Nested Knowledge	Depth	Common Knowledge	Distributed Knowledge	C
Our Model	Y	Unbounded	Y	Y	
PDKB (Muise, Belle, et al., 2015)	Y	Bounded	N	N	
K & G (Kominis and Geffner, 2015)	Y	Bounded	N	N	
MEPK (Huang et al., 2017)	Y	Unbounded	I[2]	N	
EFP & PG-EFP (Le et al., 2018)	Y	Unbounded	I[2]	N	

TABLE 3.1: Expressiveness Comparison over Epistemic Planning Approaches

There are four major points of difference that we identify. (1) Our model can handle domains in which the depth of epistemic relations are unbounded. Each level of nesting is handled by a set operation from perspective function iteratively when checking desired epistemic relations; while in other approaches, the nested epistemic relations are changed due to actions, which means they need to specify the effects on all epistemic relations in operators. However, since Kominis and Geffner (2015) and Le et al. (2018) keep Kripke structure in their approach, we are unsure about whether their approaches are practically capable of modeling unbounded domains or not. In Muise, Belle, et al. (2015)'s work, the depth also needs to be defined first as they need to generate all possible epistemic relations as atoms. (2) Reasoning about group knowledge is handled by our model using a union operation on the agent's perspective of state for distributed knowledge; and, the fixed point of intersections on nested agents' perspectives for Common Knowledge. Therefore, distributed and common knowledge result naturally from the visibility of variables. (3) Our model has the potential to handle continuous domains in both logic reasoning and problem describing. While the functional STRIPS planner we use for experiments allows only discrete variables, the external functions reason about continuous properties in the Big Brother domain. Further, our approach would work on function STRIPS planners that support continuous variables Ramirez et al., 2018. (4) Our model does not handle disjunctive knowledge, but could do

[2] 'I' means this approach can handle common knowledge indirectly, such as modeling common knowledge by public announcement (Le et al., 2018), or using a group of nested knowledge to approximate common knowledge (Huang et al., 2017).

so by modeling pairs of each variable and its all possible values as propositions, such as "$x{=}5 \vee x{=}4$". However, by doing so, we would lose the efficiency and some other expressiveness, such as continuous variables.

One possible objection is that it may be difficult to model perspective functions, because one must understand epistemic effects. However, it is important to note that in existing approaches, the modeller either needs to model epistemic effects as part of action effects, or must understand and be restricted to the assumptions in the underlying epistemic planning language; or both. Either way, the details of how actions affect knowledge must be modelled somewhere. In our case, we delegate these to perspective functions, which are more flexible than propositional approaches, because at the base case, one can implement a perspective function that has the same assumptions as any existing propositional approach. This can then be used for many domains. More details of implementing perspective functions could be found in Section 4.3.1.

To summarise Chapter 3, we have defined a new model of epistemic logic, including group operators, in which states in the model are constrained to be just variable assignments. The complexity of common knowledge in this logic is bounded by the size of the state and number of agents. In the following chapter, we show how to implement this in any F-STRIPS planner that supports external functions.

Chapter 4

Implementation

To validate our model and test its capabilities, we encode it within a planner and solve some well-known epistemic planning benchmarks. Two key aspects in planning problem solving are the planning language and solver (planner). As mentioned in Section 2.1, we use BFWS(f_5) (Francès et al., 2017) as the planner. Using F-STRIPS with external functions allows us to decompose the planning task from the epistemic logic reasoning.

In this chapter, firstly, we reinforce our intuition on implementation. Then, we introduce our F-STRIPS encoding for general MEPs.

4.1 Intuition

As mentioned in Chapter 3, our approach is built on the agents' perspectives of each state. Therefore, the key to implementing our model is to represent the state with a planning language, and change it accordingly with each action taken. Then, by only calling the related external function for evaluating the epistemic formula on request, instead of generating all truth values at the pre-compilation phase or storing entire knowledge structures (Kripke structure), the planner evaluates epistemic queries lazily. In other words, the epistemic logic reasoning task is moved from the planner to the external functions.

4.2 F-STRIPS encoding

Any classical F-STRIPS (Francès et al., 2017) problem can be represented by a tuple $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where V and D are variables and domains, $\mathcal{O}$, $\mathcal{I}$, $\mathcal{G}$ are operators, initial state and goal conditions. The set of external functions, $\mathbb{F}$, allows the planner to handle problems that cannot be defined as a propositional classical planning task, such as those whose effects are too complex to be modelled by propositional fluents, or even those whose actions and effects have some unrevealed corresponding relations. In our implementation, external functions are programmed in C++ for scalability and flexibility.

As mentioned in Section 3, our reasoning on epistemic logic is conducted in the model $M = (V, D, \pi, f_1, \ldots, f_n)$. In order to combine F-STRIPS with our model, we now give a proper definition of all the epistemic planning problems that can be handled as a tuple $(Agt, V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$ in our approach, where: Agt is a set of agent identifiers; V is a set of variables that covers the physical and the epistemic state; $\mathcal{O}$, $\mathcal{I}$ and $\mathcal{G}$ differ from their

counterparts in F-STRIPS only by adding epistemic formulae in preconditions, conditions, and goals, which will be interpreted in the following part of this section; the external functions $\mathbb{F}$ contains all the epistemic logic reasoning parts (our model).

There are two major ways to include epistemic formula in planning: using the formula as preconditions and conditions (on conditional effects) in operators $\mathcal{O}$, or using as epistemic goals in $\mathcal{G}$.

Defining $\mathcal{G}$ with desirable epistemic formulae is straightforward. For example, in Figure 2.1, if we want "agent a_1 knows a_2 sees b_1" to be true, we could simply set the goal to be $K_{a_1} S_{a_2} b_1$. However, there are some other scenarios that cannot be simply modeled by epistemic goals: temporal constraints, such as, "agent a_1 sees b_2 all the time", or, "target b_4 needs to secretly move to the other side without being seen by any other agent"; and, epistemic formulae that cannot be achieved by one state, such as, "agent a_1 needs to know values for both b_1 and b_2 (under the assumption a_1 is stationary)".

Both above scenarios can be modeled by adding epistemic formulae to $\mathcal{O}$. Temporal constraints can be inserted in the precondition of the operators directly. For example, in Figure 2.1, if the scenario is continued surveillance on b_2 over the entire plan, then the operator **turn**(a_1, d)could have that either "$S_{a_1} b_2$ after a_1 turns d degree" or "$S_{a_2} b_2$ after a_1 turns d degree" as one of the preconditions. As for the latter, we simply use a boolean *query variable* to indicate whether each desired epistemic relation is achieved or not, and update the truth value of all query variables as conditional effects in $\mathcal{O}$.

As for the encoding of $\mathcal{O}$, $\mathcal{I}$ and $\mathcal{G}$, besides the classical F-STRIPS planning parts, all the formulae are encoded as calls to external functions.

4.3 External Functions

External functions in F-STRIPS take variables as input, and return the evaluated result based on the current input and initial state. It is the key aspect that allows us to separate epistemic reasoning from planning. Therefore, unlike compilation approaches to epistemic planning that compile new formula or normal forms that may never be enquired, our epistemic reasoning uses lazy evaluation, which as we show in Chapter 5, can significantly reduce the time complexity of most epistemic planning problems.

4.3.1 Agent Perspective Functions

As briefly mentioned in Section 3.2, the perspective function, $f_i : S \to S$, is a function that takes a state and returns the local state from the perspective of agent i. Compared to the intuition of Kripke structures, our intuition is to only define which variables an agent sees. Individual and group knowledge all derive from this.

Once we have domain-specific perspective functions for all agents, or just one implementation for homogenous agents, our framework implementation takes care of the remaining parts of epistemic logic. We have implemented a library of external functions that implement the semantics of K_i, DS_G, DK_G, CS_G, and CK_G, using the underlying domain-specific

perspective functions. The modeller simply needs to provide the perspective function for their domain, if a suitable one is not already present in our library.

Example 1 In Figure 2.1, global state s covers the whole flat field. Local state $f_{a_1}(s)$ is the blue area, and $f_{a_2}(s)$ is the yellow area, which means in agent a_1's perspective, the "visible" world is the blue part, and for agent a_2, the "visible" world is only the yellow part. Furthermore, in agent a_1's perspective, what agent a_2 sees can be represented by the intersection between those two coloured areas, which is actually $f_{a_2}(f_{a_1}(s))$. The interpretation is that, agent a_1 only considers state $l = f_{a_1}(s)$ is the "global state", and inside that state, agent a_2's perspective is $f_{a_2}(l)$.

To be more specific about perspective functions, assume the global state s in the BBL example contains all variables for $\{a_1, a_2, b_1, b_2, b_3, b_4\}$ [1], such as locations, the directions agents are facing, and etc. Based on the current setup from Figure 2.1, we can implement f_i for any agent i with the following Euclidean geometric calculation given the current state is s and target is j:

$$[|\arctan(\frac{|s(y\_i) - s(y\_j)|}{s(x\_i) - s(x\_j)}) - s(dir\_i)| \leq \frac{s(ang\_i)}{2}] \bigoplus [|\arctan(\frac{|s(y\_i) - s(y\_j)|}{s(x\_i) - s(x\_j)}) - s(dir\_i)| \geq \frac{360^\circ}{} \tag{4.1}$$

The perspective function takes all agents' locations, directions and vision angles, along with variables' location as input, and returns all the variables belongs to those agents and objects that fall inside these regions. That is $f_{a_1}(s) = \{a_1, a_2, b_2, b_3\}$ and $f_{a_2}(s) = \{a_1, a_2, b_1, b_2\}$. Our library can then reason about knowledge operator.

While such an approach could be directly encoded using propositions in classical planning, we assert that the resulting encoding would be tediously difficult and error prone.

Distributed knowledge would be derived from the union over their perspectives, which is the whole world s except all variables for b_4. Both agents would know every variable in the intersection of their perspectives s'=$\{a_1, a_2, b_2\}$. Furthermore, all variables for themselves and b_2 will also be in the intersection of their perspective of s'. Since we reach a fixed point with s', we claim s' is the world a_1 and a_2 commonly knows.

However, if we alter the scenario a bit by turning a_1 180°, then the EK can be found by one level perspective functions, while CK is empty. In the new scenario, $f_{a_1}(s)$ becomes $\{a_1, b_1\}$, and EK becomes $\{a_1, b_1\}$. When we are finding the converged perspective for a_1 and a_2, $f_{a_1}(\{a_1, b_1\})$ stays the same, but the perspective for a_2 results in an empty world. The intersection over their perspective does not have a_2, which means a_2 does not exist in the perspective of at least one of the agent (a_1, in this case) from the group.

However, the above implementation of the perspective function is only useful for Euclidean planes with no obstacles. One of the novelties in this

[1] $a_1, a_2, b_1, b_2, b_3, b_4$ here are not variables. They are representations (used to simplify the example) of the group of variables that belong to that agent or that object, such as a_1 represents $x\_a_1, y\_a_1, dir\_a_1, ang\_a_1$. The same rules of representation is used in Example 2 as well.

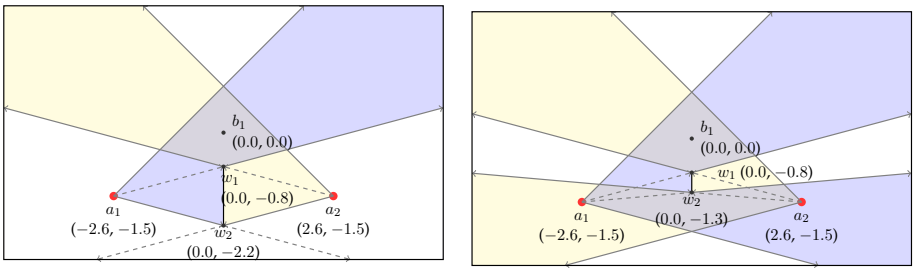
thesis is that we design our implementation to support arbitrary perspective functions, which can be provided by the modeller as new external functions. Therefore, an agent's perspective function would be able to handle any problem with a proper set of seeing rules. Basically, in our implementation, a perspective function can take any state variable from the domain model, and converts it into its agent's perspective state. Then, following the property that $f_i(s) \subseteq s$, the function applies the domain-specific seeing rules to all the variables to gain the agent's local state. From there, the semantics outlined in Chapter 3 are handled by our library.

Example 2 Example 1 does not have obstacles to block the vision of agents, which is not suitable in many domains. Therefore, we take a deeper look on how we could model two dimensional spaces with obstacles. For example, in Figure 4.1, the global state s is $\{a_1, a_2, b_1, w\}$, where w is a thick wall which blocks agent's vision.

The seeing rules in this domain are different from original BBL domain. Rather than only checking if an object is within an agent's visible range, we also need to make sure there is no object in between the agent and the object. Therefore, the perspective function for this domain contains two steps: using Formula 4.1 to check whether the agent i is facing the correct direction to see target j; and, using algebraic geometry to check whether the line of sight is obstacle-free. Let's explain this perspective function by evaluating $b_1 \in f_{a_2}(f_{a_1}(s))$ with following examples:

In Figure 4.1a, since the wall blocks the vision between agent a_1 and a_2, in standard BBL, we would have $f_{a_1}(s) = \{a_1, b_1, w\}$ and $f_{a_2}(s) = \{a_2, b_1, w\}$. But we must also check whether there is an obstacle-free line of sight. Since the wall blocks line of sight between a_1 and a_2, then a_2 must be removed from $f_{a_1}(s)$. So, in agent a_1's perspective, agent a_2's view of the world is $f_{a_2}(f_{a_1}(s)) = \emptyset$, as agent a_1 cannot see a_2.

A slightly more complex example would be in the Figure 4.1b. This is the same as the previous scenario, except that the wall is resized so that agents a_1 and a_2 can see each other. In the figure, the perspective of agent a_1 is blue. However, the wall prevents a_1 from seeing line of sight between a_2 and b. We do not have $b_1 \in f_{a_2}(f_{a_1}(s))$ if our perspective function is modelled so that when we apply f_{a_2} on b_1 in the local state $f_{a_1}(s)$, the line of site (a_2, b_1) is not fully in the blue area (a_1's perspective of the world s), which means agent a_1 cannot see if agent a_2 sees b_1.



(A) Scenario 1 (B) Scenario 2

FIGURE 4.1: Example for Big Brother Logic with Obstacle

Those examples show that we can expand to new logics by providing different implementations of f_i. From this, the logic of knowledge is provided using our implementation of the semantics in Section 3. Our code library contains perspective functions for many basic logics, but we find that tailoring perspective functions to domains results in more elegant and compact models. For example, the only difference between the external function's implementation on Corridor and Grapevine domain from Chapter 5 is that, in Corridor, the seeing rules are that an agent can "see" within the current room and the adjacent room, while in Grapevine, the agent can only "see" within the current room. Changing the perspective function results in a new domain without having to change the planning model. Moreover, to implement epistemic planning problems, e.g. , a three-dimensional Euclidean plane, we only need to modify the seeing rules of the perspective functions based on the geometric model.

From a practical perspective, this means that modellers are requested to provide: (1) a planning model that uses epistemic formulae; and (2) implementations for $f_1, \ldots, f_n$ if one does not already exist in our library. Linking the epistemic formula in the planning model to the perspective functions is delegated to our library. Existing approaches to epistemic planning specify the effects on epistemic logic directly in actions, or model the agents' Kripke structures. Instead, we delegate reasoning to external functions by applying seeing rules to all the variables in current state.

4.3.2 Domain dependent functions

Domain dependent functions are customised relations for each set of problems, corresponding to $R(v_1, \ldots, v_k)$ in Chapter 3, and can be any domain specific function that is implementable as external functions. In words, as one of the advantages of using external functions, those domain dependent formulae can be implemented to deal with complex relations, such as logics in continuous domains.

Overall, the implementation of our model is not as straightforward as other classical F-STRIPS planning problems or epistemic planning problems. However, as we demonstrate in the next section, it is scalable and flexible enough to find valid solutions to many MEPs.

Chapter 5

Experiments & Results

In this chapter, we evaluate our approach on several problems: *Corridor* (Kominis and Geffner, 2015), *Grapevine* (Muise, Belle, et al., 2015), *Big Brother Logic (BBL)* (Gasquet, Goranko, and Schwarzentruber, 2014), *Social-media Network* (*SN*) and *Gossip* (Baker and Shostak, 1972). Corridor and Grapevine are well-known epistemic planning problems, which we use to compare actual performance of our model against an existing state-of-the-art planner. BBL is a model of the Big Brother Logic in a two-dimensional continuous domain, which we use to demonstrate the expressiveness of our model. In addition, to demonstrate our model's capability of reasoning about group knowledge, we inherit the classical Gossip Problem, and create our own version, called Social-media Network. Moreover, our model has an advantage on those epistemic planning problem where the knowledge can be derived from the ontic states. However, we do investigate the capability and performance for other classical epistemic planning problem that the knowledge must be explicitly recorded by the states, such as Gossip.

The source code of our implementation along with all experiments can be found at https://github.com/guanghuhappysf128/benchmarks.

Hereafter, we assume any knowledge formula $K_a v$ is supplied with correct value e, which means its equivalent with $K_a v{=}e$, unless the value is specified.

5.1 Benchmark problems

To evaluate computational performance of our model, we compare to Muise, Belle, et al. (2015)'s PDKB planner. Their planner has been used to compare on Corridor and Grapevine domains against many others' solutions (Kominis and Geffner, 2015; Huang et al., 2017; Le et al., 2018). From their results and results from Huang et al. (2017) and Le et al. (2018), it is fair to say that PDKB is a state-of-the-art planner. In addition, to test how the performance is influenced by the problem, we created new problems that varied some of the parameters, such as the number of agents, the number of goal conditions and also depth of epistemic relations.

The PDKB planner converts epistemic planning problems into classical planning problems, which results in generating a significant number of propositions when the depth or the number of agents increase. We tried to submit the converted classical planning problem to the same planner that used by our model, BFWS(R_0) planner, to maintain a fair comparison. However, since the computational cost of the novelty check in BFWS(R_0)

planner increases with size of propositions, the planning costs was prohibitively expensive. Therefore, for comparison, we use the default *ff* planner that is used by Muise, Belle, et al. (2015).

We ran the problems with both planners on a Linux machine with 8 CPUs (Intel Core i7-7700K CPU @ 4.20GHz 8) and 16 gigabyte memory. As a matter of fact, both methods do not involve multi-threading or run in parallel, which means the performance would be the same result as if we use one CPU. We measure the number of atoms (fluents) and number of nodes generated during the search to compare the size of same problem modelled by different methods. We also measured the total time for both planners to solve the problem, and the time they take to reasoning about the epistemic relations, which correspond to the time takes to call external functions for our solution (during planning), and the time takes to convert the epistemic planning problem into classical planning problem in the PDKB solution (before planning).

5.1.1 Corridor

The corridor problem was originally presented by Kominis and Geffner (2015). It is about selective communication among agents. The basic setup is in a corridor of rooms, in which there are several agents. An agent is able to move around adjacent rooms, sense the secret in a room, and share the secret. The rule of communication is that when an agent shares the secret, all the agents in the same room or adjacent rooms would know. That is, as shown in the Figure 5.1, the blue area is the "vision" for agent b, which is also the only rule that we implemented in the agent's perspective function as seeing rule.

The goals in this domain are to have some agents knowing the secret and other agents not knowing the secret. Thus, the main agent needs to get to the right room and communication to avoid the secret being overheard.

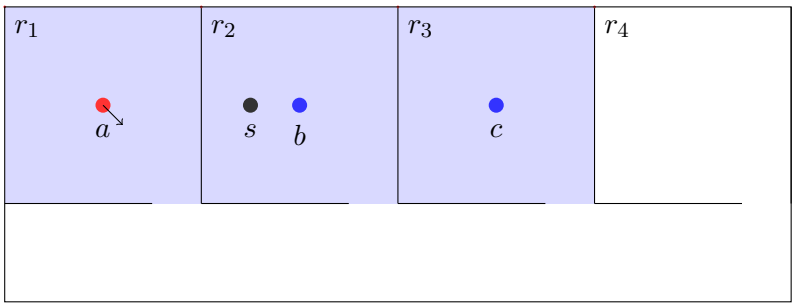


FIGURE 5.1: Corridor Domain

Examples

As in Figure 5.1, let a, b and c be three agents, and r_1, r_2, r_3 and r_4 be 4 consecutive rooms on a corridor. Initially, the secret q is in the room r_2, and agent a, b and c are located in room r_1, r_2 and r_3 respectively. And the goal is: agent a needs to sense q and reveal it to agent c without agent b knowing the secret. Then, this example can be defined as a tuple $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where

- $V = \{\ agt\_at(i),\ sct\_at,\ sct,\ K_cq,\ K_bq |\ i \in \{a, b, c\}\}$
- D : $D(agt\_at(i)) = D(sct\_at) = \{1, \ldots, 4\}$, where $i \in \{a, b, c\}$; $D(sct) = D(K_cq) = D(K_bq) = \{0, 1\}$
- $\mathcal{O}$: **move**(x, x'), **sense**(x) and **shout**(x), where $x, x' \in \{r_1, r_2, r_3, r_4\}$, and they are adjacent
- $\mathcal{I} = \{agt\_at(a) = 1,\ agt\_at(b) = 2,\ agt\_at(c) = 3,\ sct\_at = 2,\ sct = 0\}$
- $\mathcal{G} = \{K_cq = 1, K_bq = 0\}$
- $\mathbb{F} = \{$ (**@check** K_cq), (**@check** K_bq) $\} \mapsto \{true, false\}$

Here, set V and D defines the state space. There are 3 agents and one secret in this example, each should correspond to one variable to represent its location. In addition, agent a needs to know the secret first to share with others. And since we do not have to represent agent's knowledge in the state, we need a single binary variable sct to indicate disclosure of the secret.

The basic action for agent a is **move**. Agent a can move from one room to another as long as those two rooms are adjacent. And a also can perform **sense**(x) or **shout**(x) action in room x. The sensing action will set sct to 1, if $agt\_at(a)$ is equal to sct_at, which means if the a is in the same location as secret, by preforming sensing action, a acquires the secret. Action **shout** is a bit more complex. In both Kominis and Geffner (2015)'s work and Muise, Belle, et al. (2015)'s work, this action does generate knowledge by making one or several knowledge atoms true. However, as one of the advantages of our model, we don't have to generate knowledge atoms. The only effects here is the result of seeing rules. Therefore, we invoke the external functions as effects of this action, to update desired knowledge queries or visibility queries. To be specific, action **shout** can be defined as:

shout(x):
pre: $sct = 1, agt\_at(a) = x$
eff: (forall (?q - query) (when (= (**@check** ?q) 1) (= ?q 1))

To explain what this action does, we need to take a look on the goal condition first. From the example, it is obvious that the goal contains two epistemic relations: c knows q; and, b does not know q. Since in our work, we only handle knowledge rather than beliefs, we assume that knowledge will hold until it has been changed by other actions. Therefore, we can interpret those two proposition in a more systematic way: the goal can be achieved at the first time when "c knows q" become true and for all the time "b knows q" needs to be false. Then, we can define those two queries that are involved in this problem: "K_cq" and "K_bq"[1]. Therefore, the action **shout** needs to check those two queries by calling the external function **@check**, and only update the value of the query if the corresponding result is true. In other words, only the value 1 will overwrite the previous value for each query, as the knowledge can not be "forget" in our assumptions. For any

[1]The domain of all queries are $\{0, 1\}$, which indicates whether the query has been checked as true or not.

action that makes $K_b q$ true in its effects, the knowledge b knows q will be preserved, and there is no way to achieve the goal, which means that action would be excluded from the solution.

Then, by running the planner including external functions, a valid optimal plan is found:

- **move**(r_1, r_2), **sense**(r_2), **move**(r_2, r_3), **move**(r_3, r_4), **shout**(r_4).

Parameters			Our Model					PDKB			
$\|A\|$	d	$\|\mathcal{G}\|$	$\|Atom\|$	$\|Gen\|$	$\|Calls\|$	TIME(s) Calls	Total	$\|Atom\|$	$\|Gen\|$	TIME(s) Compile	Total
3	1	2	13	15	48	0.001	0.004	54	21	0.148	0.180
7	1	2	13	15	48	0.002	0.005	70	21	0.186	0.195
3	3	2	13	15	48	0.003	0.007	558	21	0.635	0.693
6	3	2	13	15	48	0.005	0.008	3810	21	5.732	6.324
7	3	2	13	15	48	0.005	0.008	5950	21	9.990	11.13
8	3	2	13	15	48	0.006	0.009	8778	21	14.14	15.68
3	4	2	13	15	48	0.006	0.009	3150	21	3.354	3.752
3	5	2	13	15	48	0.006	0.009	18702	21	25.69	29.54

TABLE 5.1: Comparison Results for the Corridor Domain

5.1.2 Grapevine

Grapevine, proposed by Muise, Belle, et al. (2015), is a similar problem to Corridor. With only two rooms available for agents, the scenario makes sharing secrets while hiding from others more difficult. The basic setup is each agent has their own secret, and they can share their secret among everyone in the same room. That is, as it can be seen from the Figure 5.2, agent a's "vision" is only in r_1. The basic actions for agents are moving between rooms and sharing his secret.

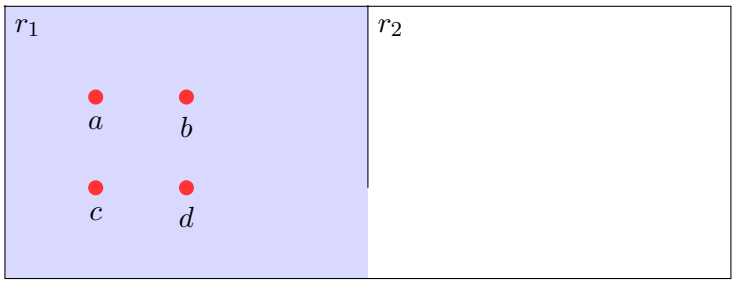


FIGURE 5.2: Grapevine Domain

Examples

Let a, b, c and d be four agents, and each of them have a secret, a', b', c' and d' respectively. There are two adjacent rooms, r_1 and r_2, and agents are able to move from one to another freely. In the initial situation, all the agents are in room r_1, and each of them only knows their own secret. The goal is to achieve a circle of known and unknown relations. To be specific, the goal conditions are: a' is known to b but not to d; b' is known to c, but not to a;

c' is known to d, but not to b; and, d' is known to a, but not to c. Then, this example can be defined as a tuple $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where:

- $V = \{\, agt\_at(i), sees(i)(j),\ K_a b',\ K_b c',\ K_c d',\ K_d a',\ K_a d',\ K_b a',\ K_c b',\ K_d c' \mid i \in \{a, b, c, d\},\ j \in \{a', b', c', d'\}\}$
- $D : D(agt\_at(i)) = \{1, 2\}$, and $D(sees(i)(j)) = D(query) = \{0, 1\}$, where $i \in \{a, b, c, d\}$, $j \in \{a', b', c', d'\}$
- $\mathcal{O}$: **move_left**(i), **move_right**(i) and **share**(i, j), where $i \in \{a, b, c, d\}$, $j \in \{a', b', c', d'\}$
- $\mathcal{I} = \{agt\_at(i) = 1,\ sees(a)(a') = 1,\ sees(b)(b') = 1,\ sees(c)(c') = 1,\ sees(d)(d') = 1 \mid i \in \{a, b, c, d\}\}$
- $\mathcal{G} = \{\, K_a b' = 1,\ K_b c' = 1,\ K_c d' = 1,\ K_d a' = 1,\ K_a d' = 0,\ K_b a' = 0,\ K_c b' = 0,\ K_d c' = 0\}$
- $\mathbb{F} = \{$ (**@check**: $?q$) $\mapsto \{true, false\}$, where $?q \in \{K_a b',\ K_b c',\ K_c d',\ K_d a',\ K_a d',\ K_b a',\ K_c b',\ K_d c'\}\}$

It is obvious that we need variables to cover agent's location, and similarly as corridor problem, we also need a set of variables $sees(i)(j)$ to indicate whether an agent can "see" the secret or not. Those seeing variable are only for keeping track of whether an agent is able to share other's secret or not. In D, location variables can be value of 1 or 2, represents room r_1 and r_2 respectively, while seeing variables and query variables only have a binary value.

As in Corridor, we use action **share**(i, j) to examine all the queries. The new knowledge is generated only when agent i shares a secret j. This secret can either be i's own secret or others secret that i knows. As long as i sees the secret, which means someone has shared the secret with i. We only update the query variables when a query becomes true. So that, all the negative queries, which are indicated by those query variables that required to be 0 in goal conditions, keep being false at all the time; while, all the positive queries will remain true after the first time they have been updated to be true.

Then, after we implement external functions[2], we can simply run the planner and get a valid optimal plan as:

- **move_right**(a), **share**(d, d'), **move_right**(d), **share**(a, a'), **share**(c, c'), **move_right**(b), **share**(b, b')

Example (Extended)

The previous example seems only a complication version of Corridor problem. Let us introduce a different example to make full use of the Grapevine problem. Consider the same situation and initial state. The only difference is for the goal state, instead of reasoning agent's knowledge about secrets, we can take a look on agent's reasoning about others' knowledge about

[2]It is worth to mention that, the only change, which we make from the external functions for Corridor domain, is to simply change the seeing rules from "same and adjacent rooms" to "same rooms".

secrets. For example, let goal condition be: "agent a knows b's secret b' "; and, "b does not know that a knows his secret". As the depth of knowledge goes further, the advantage of our model would emerge more prominently. Rather than generating all the knowledge queries with depth up to 2, we can simply replace the query variables, and use the same structure as previous examples. The only difference, $\mathcal{G}$, is listed as follows:

- $\mathcal{G} = \{K_a b' = 1, K_b K_a b' = 0\}$

And since none of the action asks knowledge queries as preconditions, we can use the same domain. The solution for this example is:

- **move_right**(a), **share**(b, b'), **move_right**(c), **share**(c, b')

Parameters			Our Model					PDKB			
$\|A\|$	d	$\|\mathcal{G}\|$	$\|Atom\|$	$\|Gen\|$	$\|Calls\|$	TIME(s) Calls	Total	$\|Atom\|$	$\|Gen\|$	TIME(s) Compile	Total
4	1	2	346	10	48	0.002	0.006	96	22	0.429	0.469
4	2	2	346	10	48	0.002	0.007	608	5	2.845	3.168
4	1	4	346	23	144	0.005	0.009	96	11	0.428	0.468
4	2	4	346	23	144	0.006	0.010	608	11	2.885	3.178
4	1	8	346	368	1200	0.040	0.047	96	529	0.381	0.455
4	2	8	346	368	1200	0.050	0.057	608	1234	3.450	4.409
4	3	8	346	368	1200	0.068	0.073	4704	14	28.66	30.72
8	1	2	546	18	48	0.003	0.010	312	5	3.025	3.321
8	2	2	546	18	48	0.003	0.010	4408	5	54.35	58.80
8	1	4	546	43	144	0.008	0.016	312	11	2.546	2.840
8	2	4	546	43	144	0.008	0.016	4408	11	55.33	59.78
8	1	8	546	1854	4528	0.238	0.268	312	2002	2.519	3.752
8	2	8	546	1854	4528	0.322	0.294	4408	4371	54.90	228.1
8	3	8	546	1854	4528	0.394	0.421	–	–	–	–

TABLE 5.2: Comparison Results for the Grapevine Domain

5.1.3 Analysis

We show the results of the problems in Table 5.1 and Table 5.2, in which $|A|$ specifies the number of agents, d the maximum depth of a nested epistemic query, $|\mathcal{G}|$ the number of goals, $|Atom|$ the number of atomic fluents, $|Gen|$ the number of generated nodes in the search, and $|Calls|$ the number of calls made to external functions. The symbol "$-$" represents there is no result within 10 minutes boundary time.

From the results, it is clear that the complexity of the PDKB approach grows exponentially on both the number of the agents and the depth of epistemic relations (the planner went over the 10-minute time boundary in the final Grapevine problem). The complexity of the pre-compilation for the PDKB planner is $O(2^{|A|*D})$, in which $|A|$ is the number of agents and D is the maximum depth of any modal formula in the modal. The search complexity is then the same as classical planning, which we model as $O(|Gen|)$), in which Gen is the set of states that are generated to solve the problem. In our approach, the number of features and depth do not have a large impact. However, epistemic reasoning in our approach (the number of calls

to the external solver), has a significant influence on the performance for our solution. Since the F-STRIPS planner we use checks each query in goal conditions at generation of each node in the search ($O(|Gen|)$), the time complexity for epistemic logic reasoning is in $O(|Gen| * |G| * |a| * |V|^2)$[3], in which G is the set of goals and V is the size of the state. Although this search problem is still an NP-hard problem, the empirical computational cost is significantly lower than the compilation in the PDKB approach in most of the test cases.

5.2 Big Brother Logic

Big Brother Logic (BBL) is a problem first discussed by Gasquet, Goranko, and Schwarzentruber (2014). The basic environment is on a two-dimensional space called "Flatland" without any obstacles. There are several stationary and transparent cameras; that is, cameras can only rotate, and do not have volume, so they do not block others' vision. In our scenario, we allow cameras to also move in Flatland.

5.2.1 Example and Encoding in F-STRIPS

Let a_1 and a_2 be two cameras in Flatland. Camera a_1 is located at $(5, 5)$, and camera a_2 at $(15, 15)$. Both cameras have an $90°$ range. Camera a_1 is facing north-east, while camera a_2 is facing south-west. There are three objects with values $o_1 = 1$, $o_2 = 2$ and $o_3 = 3$, located at $(1, 1)$, $(10, 10)$ and $(19, 19)$ respectively. For simplicity, we assume only camera a_1 can move or turn freely, and a_2, o_1, o_2 and o_3 are fixed. In order to implement that assumption, we set the locations of these to be common knowledge. Figure 5.3 visualises the problem setup.

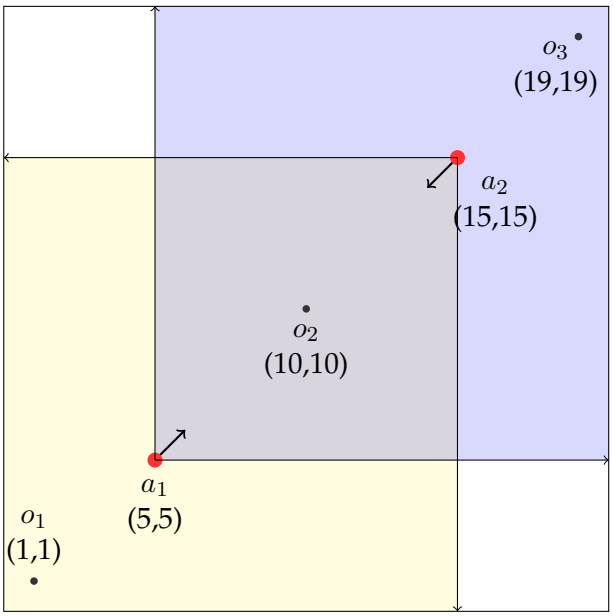


FIGURE 5.3: Example for Big Brother Logic setup

[3]In the worst case scenario, which is checking common knowledge query on a state, there are at most $|V|$ (maximum size of the state) iterations, and each iteration contains $|a|$ number of set operations on the global state or a local state (maximum $|V|$).

Let all the desired epistemic relation queries be a set of propositions Q, this problem can be represented by the tuple $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where:

- $V = \{x,\ y,\ dir,\ q \mid q \in Q\}$
- D : $D(x)$=$D(y)$=$\{-20, \ldots, 20\}$; $D(dir)$=$\{-179, \ldots, 180\}$; and, $D(q)$=$\{0, 1\}$, where $q \in Q$
- $\mathcal{O}$: **move**(dx, dy) and **turn**(d)
- $\mathcal{I} = \{x = 5,\ y = 5,\ dir = 45\}$
- $\mathcal{G} = \{q = 1\}$
- $\mathbb{F}$: (**@check** q) $\mapsto \{true, false\}$,

in which q is a goal query, which we describe later.

This is just a simple example to demonstrate our model. Variables x and y represent coordinates of camera a_1, and dir determines which way a_1 is facing. Since a_2 and all other objects are fixed, we model them in an external state handled by the external functions, which lightens the domain and reduces the state space. However, we could also model the positions of these as part of the planning model if desired. For the domain of the variables, as part of the model, although the F-STRIPS planner we use does not support using real numbers, using integers is enough to show that our model can work on continuous problems, as the external function allows us to use floating point numbers in the calculation.

We need to check the knowledge queries in the actions (precondition, conditions), or goals. Both action **move**(dx, dy) and **turn**(d) can change agents' perspectives, and therefore, can influence knowledge. To simplify the problem, instead of moving 1 or turning 1 degree per action, we can set up a reasonable boundary for actions, such as turning $[-45, 45]$ degree per action and move at speed of 2 for each direction on x and y.

5.2.2 External Functions

External functions in the BBL domain are mainly focused on checking the epistemic relations encoded as queries. The input would be the query (in the format of our language described in Chapter 3) and current states (x, y and dir are the only changing variables in this case), and the output is the evaluated truth value of the query. The seeing rules are the same as we introduced in Section 4.3.1. The formal definition of the perspective function is[4]:

BBL domain: $f_i(s) = \{v' \mid v' \in s \ \wedge i \triangleright v'\}$

Since BBL domain is in a two-dimensional continuous environment, we do not see how it could be modelled by any existing propositional planning approach, as there are an infinite number of propositions. Further, the arithmetic operators and trigonometric functions would need to be encoded.

[4]The relation "$i \triangleright v'$" reads agent i sees v', which implemented following Formula 4.1 in Subsection 4.3.1.

5.2.3 Goal Conditions

As for the goal conditions, some queries q can be achieved for the problem in Figure 5.3 without doing any actions, such as:

1. Single Knowledge query: $\neg K_{a_2} o_3$; $K_{a_1} o_3$
2. Nested Knowledge query: $\neg K_{a_1} S_{a_2} o_3$; $S_{a_1} S_{a_2} o_3$
3. Group Knowledge query: $\neg E K_{a_1,a_2} o_3$; $E K_{a_1,a_2} o_2$
4. Distributed Knowledge query: $\neg D K_{a_1,a_2}(o_1{=}3)$; $D K_{a_1,a_2}(o_1{=}1)$
5. Common Knowledge query: $C K_{a_1,a_2} o_2$; $C K_{a_1,a_2} S_{a_1} o_3$

Most of them are intuitive. From goal 2, although $S_{a_1} S_{a_2} o_3$ is true because a_1 can see a_2's location, range of vision and direction, so a_1 knows whether a_2 can see o_3, the formula $K_{a_1} S_{a_2} o_3$ is false because there is no action that a_1 can perform to make a_2 see o_3.

For goal 5, $C K_{a,b} S_{a_1} o_3$ is true without any action, because, by calling the external function, the common local state for a_1 and a_2 would be the location of all three values, both a_1 and a_2 and the value of o_2. Then, $S_{a_1} o_3$ would be evaluated true based on the common local state.

In addition, there are some query that would be achieved through valid plans:

1. $K_{a_1,a_2} o_1$: **move**$(-2,-2)$, **move**$(-2,-2)$
2. $C K_{a_1,a_2} o_1$: **move**$(-2,-2)$, **move**$(-2,-2)$
3. $S_{a_2} S_{a_1} o_1$: **move**$(-2,2)$, **move**$(-2,2)$
4. $K_{a_1} o_1 \wedge \neg K_{a_2} K_{a_1} o_1$(BBL11[5]): **move**$(-2,1)$, **move**$(-2,2)$, **move**$(-1,2)$, **move**$(0,2)$, **move**$(0,2)$, **move**$(0,2)$, **turn**$(-45)$, **turn**$(-44)$
5. $\neg K_{a_1} S_{a_2} S_{a_1} o_1 \wedge S_{a_1} o_1$(BBL12): **turn**$(-44)$, **turn**$(-45)$, **turn**$(-45)$, **move**$(1,2)$, **move**$(2,2)$, **move**$(2,2)$, **move**$(2,2)$, **move**$(2,2)$, **move**$(2,2)$, **move**$(-1,2)$

The first one is clear. There is more than one way to let both of them know value o_1, and the planner returns the optimal solution. The second one is also intuitive: to achieve common knowledge in a BBL problem, they need to both see the item and both see each other. The difference between the last two are a bit trickier. To avoid a_2 that knows whether a_1 can see o_1, the cheapest plan returned by planner was for a_1 to move out of a_2's eye sight. The last one is the most difficult to solve. Not only should a_1 see o_1, but also a_1 should know that originally a_2 cannot see that a_1 sees o_1. This is done by decomposing the query into three facts: "a_1 sees o_1";"a_2 cannot see whether a_1 sees o_1"; and, "a_1 can see that whether a_2 can see whether a_1 sees o_1".

[5]Problem index for BBL, used to refer to the result in Table 5.3 about the size of the problem.

5.2.4 Results

Table 5.3 shows the results for our problems in the BBL domain, where $|Exp|$ represents the number of nodes been expanded and $|P|$ indicates the length of the plan. A plan length of "∞" means that the problem is unsolvable – no plan exists. While the perspective function in BBL depends on a *geometric model* based on agent's position, direction and facing angle, the results show that with proper usage of our F-STRIPS planner, we can represent continuous domains. Our epistemic solver is able to reason about other the agents' epistemic states (vision) and derive plans based on these for non-trivial intricate goal that we believe would be tedious and error to encode propositionally, if possible at all given the continuous domain, demonstrating that our model can handle important problems in vision-based domains. As far as we know, there is no current epistemic planner can handle problems at this level of expressiveness. Another advantage of our model is that epistemic formulae are evaluated lazily. Instead of generating all possible combination of positions, angles and directions, and calculated visibility relation off-line, our model evaluate desired epistemic relations online during the search. The non-solvable case BBL03 shows the scale of the problem when the planner has to calculate and evaluate truth value of epistemic relations with all possible combinations of the states.

	Parameters				Performance					
	$\|A\|$	d	$\|\mathcal{G}\|$	$\|P\|$	$\|Gen\|$	$\|Exp\|$	$\|Calls\|$	TIME(s) $calls$	TIME(s) Total	Goal
BBL01	2	1	1	0	1	0	2	0.000	0.002	$K_{a_1}o_2$
BBL02	2	1	1	2	115	2	232	0.007	0.009	$K_{a_1}o_1$
BBL03	2	1	1	∞	605160	all	1210320	36.1	78.2	$K_{a_2}o_3$
BBL04	2	2	1	2	115	2	232	0.015	0.017	$K_{a_1}K_{a_2}o_1$
BBL05	2	1	1	0	1	0	2	0.000	0.002	$DK_{a_1,a_2}\{o_1,o_2,o_3\}$
BBL06	2	1	1	0	1	0	2	0.000	0.002	$EK_{a_1,a_2}o_2$
BBL07	2	1	1	2	115	2	232	0.019	0.021	$EK_{a_1,a_2}\{o_1,o_2\}$
BBL08	2	1	1	0	1	0	2	0.000	0.002	$CK_{a_1,a_2}o_2$
BBL09	2	1	1	2	115	2	232	0.048	0.050	$CK_{a_1,a_2}\{o_1,o_2\}$
BBL10	2	2	1	2	115	2	232	0.016	0.018	$K_{a_1}DK_{a,b}\{o_1,o_2,o_3\}$
BBL11	2	2	2	8	59260	7509	187332	7.650	8.382	$K_{a_1}o_1 \wedge \neg K_{a_2}K_{a_1}o_1$
BBL12	2	3	2	9	15842	592	47626	2.380	2.472	$S_{a_1}o_1 \wedge \neg K_{a_1}S_{a_2}S_{a_1}o_1$

TABLE 5.3: Experiments Results for BBL domain

5.3 Social-media Network

The *Social-media Network* (SN) domain is an abstract network in which agents can befriend each other to read their homepage, post on friend's homepage and view their friend list, and etc., based on typical social media models. We extend two-way one-time communication channels from a classical gossip problem (Cooper et al., 2016) into two-way, all-time communication channels, and add the concepts of secret messages. By decomposing secrets into messages and posting through an agent's friendship network, we model how secrets can be shared between a group of individuals not

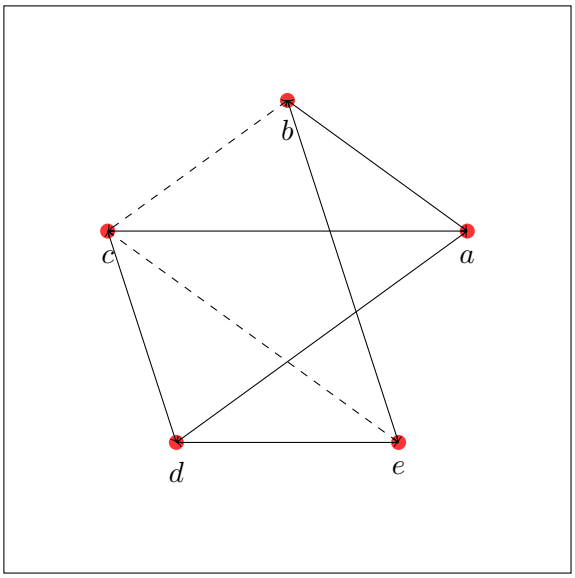


FIGURE 5.4: Example for Social-media Network

directly connected without anyone else on the network knowing the secret, and some secrets can be shared within a group excepting for some individuals. The former could be spies sharing information with each other through the resistance's personal page, and the latter could be a group arranging a surprise party for a mutual friend.

5.3.1 Example and Encoding in F-STRIPS

Let a, b, c, d, e be five agents in the SN, with friendship links shown in Figure 5.4. Their friend relations are represented by full lines between each agent. The dotted lines are for later demonstration purpose.

Let g be a friend for all agents and g wants to share a secret. We assume the social network is in g's perspective directly, and the network is fixed for simplicity. Let all the epistemic queries that we concern of be a set of propositions, Q, and p_1, p_2, p_3 as three parts of the secret P. Any problem by this setup can be represented by a tuple $(A, V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where:

- $A = \{a, b, c, d, e\}$
- $V = \{(friended\ i\ j), (post\ p)\ (q) \mid i, j \in A,\ p \in P,\ q \in Q\}$
- $D : D(friended\ i\ j) = D(q) = \{0, 1\}$, $D(post\ p) = A$, where $i, j \in A,\ p \in P,\ q \in Q$
- $\mathcal{O}$: **post**(i, p), where $i \in A,\ p \in P$
- $\mathcal{I} = \{\ (friended\ a\ b) = 1,\ (friended\ a\ c) = 1,\ (friended\ a\ d) = 1,\ (friended\ b\ e) = 1,\ (friended\ c\ d) = 1,\ (friended\ d\ e) = 1\ \}$
- $\mathcal{G}$: see below
- $\mathbb{F}$: (**@check** q) $\mapsto \{true, false\}$

The variable $(friended\ i\ j)$ represents whether i and j are friends with each other, which is a domain dependent relation, and q covers all the epistemic queries. Action (**post** $i\ p$) specifies that the message p is posted on

agent i's page. The $\mathcal{I}$ covers the friendship relations in Figure 5.4, and no message has been posted yet. Similarly, the action **post** is the only source for epistemic relation changes. Therefore, we update all desired queries Q as conditional effects.

5.3.2 External Functions

The seeing rules in SN domain is based on the friend relations. The agent is able to view all post on it's friend's homepage and also view the friend list of its friend. To be specific with this example, agent a is able to read every post on c's homepage, and a knows c is friend with a and d. With this information, a is able to deduce that any post p on a's or d's homepage is also readable for c, which in another format is "$K_a\ K_c p$", nested knowledge.

The perspective function depends on the friendship network. For example, for a full state $s = \{a, b, c, d, e, p_1\}$, assuming p_1 is posted on b's homepage, then, we have $f_a(s)$ is $\{a, b, c, d, p_1\}$; $f_d(s)$ is $\{a, c, d, e\}$; while, d's "vision" under a's perspective will be $f_d(f_a(s)) = \{a, c, d\}$, since e is not in a's perspective. Similarly, $f_e(f_a(s))$ will be empty. Formally:

$$\text{SN domain:} \quad f_i(s) = \{v' \mid v' \in s \ \wedge\ (friend\ i\ j) \wedge ((post\ v')=j \vee v'=j)\}$$

We have not seen this domain or anything similar modelled in any existing approach. The epistemic relation would be a problem for most approaches, as it involves distributed knowledge and common knowledge. The network itself could be modelled by other approaches, however, the group knowledge that we reason about depends on the network. It is not clear to us how existing approaches could compactly model the effect on knowledge when the friendship network changes. In our approach, the perspective function gives us this information by default.

5.3.3 Goal Conditions

Goals that we have tested are shown in Table 5.4. For some epistemic formulae between a and b, since they are friends, simply posting the message in any of their personal page is sufficient to establish common knowledge about the information in that post. But for the knowledge between a and e, for example, $EK_{a,e}p_1$, the message needs to be posted on the page of a mutual friend, such as agent b. In addition, since a and e are not friends, in each of their perspectives of the world, there is no information (variables) describing others. Therefore, both $EK_{a,e}EK_{a,e}p_1$ and $CK_{a,e}p_1$ are not possible without changing the network structure.

Below are some of the sample goals that we have tested, with the identifying name from Table 5.4 and the plan that achieves that goal:

- $K_a p_1$ (SN01), $K_a K_b p_1$ (SN02), $EK_{a,b}p_1$ (SN03) and $CK_{a,b}p_1$ (SN06): **post**(a,p_1)
- $CK_{a,e}p_1$ (SN07): ∞ (Unsolvable, since they are not friend.)
- $K_a\{p_1, p_2, p_3\}$ (SN08), $EK_{a,b}\{p_1, p_2, p_3\}$ (SN04) and $DK_{a,b}\{p_1, p_2, p_3\}$ (SN05):
  post(a,p_1), **post**(a,p_2), **post**(a,p_3)

The other types of goals are secretive:

- $K_a\{p_1, p_2, p_3\} \wedge \neg K_b\{p_1, p_2, p_3\}$ (SN09): **post**(a,p_1), **post**(a,p_2), **post**(c,p_3)
- $K_a\{p_1, p_2, p_3\} \wedge \neg K_b\{p_1, p_2, p_3\} \wedge \neg K_c\{p_1, p_2, p_3\}$ (SN10): **post**(a,p_1), **post**(b,p_2), **post**(c,p_3)

The aims are to share the whole secret with a while b must not know the whole secret, but can know some of it. Some parts of it, such as p_3, needs to be shared in the page that b does not have access to. Then, c must also not know the secret, the secret now needs to be posted in the way that b and c do not see some parts respectively, while a sees all the parts.

Finally, we look on those two desired scenarios in the introduction of SN:

- Sharing with a spy (SN11):
  $K_a\{p_1, p_2, p_3\} \wedge \neg K_b\{p_1, p_2, p_3\} \wedge \neg K_c\{p_1, p_2, p_3\} \wedge \neg K_d\{p_1, p_2, p_3\}$
  $\wedge \neg K_e\{p_1, p_2, p_3\}$:
  post(a,p_1), **post**(b,p_2), **post**(c,p_3)
- Surprise party (SN13):
  $\neg K_a\{p_1, p_2, p_3\} \wedge K_b\{p_1, p_2, p_3\} \wedge K_c\{p_1, p_2, p_3\} \wedge K_d\{p_1, p_2, p_3\}$
  $\wedge K_e\{p_1, p_2, p_3\}$:
  ∞

Sharing a secret to some specific individual without anyone else knowing the secret can be done with the current network. However, if we alter the problem a bit by adding a friend relation between b and c (SN12), and apply the same goal conditions as SN11, no plan would be found by the planner, because c sees everything a can see, and there is no way to share some information to a without c seeing it.

For sharing a secret surprise party for agent a among all the agents without a knowing it, the messages need to shared in such a way that a is not able to get a complete picture of the secrets. In the current setup of the problem (SN13), since a sees everything seen by c, there is no way to held a surprise party without a knowing it. However, by adding a friend relation between e and c (SN14), the planner returns with the plan: **post**(e,p_1), **post**(e,p_2), **post**(e,p_3).

5.3.4 Results

Table 5.4 shows the results for our problems in the social-media network domain. The results show that our planner is able to reason about nested knowledge and also group knowledge to achieve an intricate goal, which means our model can handle a variety of knowledge relations at same time within reasonable time complexity. In addition, this result also indicates that the time our approach takes to solve the problem directly depends on the number of external calls that the planner makes. Moreover, the number of external function calls is related to the number of nodes that planner generates and expands, which depends on the search algorithm. The test cases with length of the plan $|P|$ equal to 'infinite' (unsolvable) shows the scale of that problem with $BFWS(R_0)$ planner.

	Parameters				Performance					
	$\|A\|$	d	$\|\mathcal{G}\|$	$\|P\|$	$\|Gen\|$	$\|Exp\|$	$\|Calls\|$	TIME(s) $calls$	TIME(s) Total	Goal
SN01	5	1	1	1	16	2	42	0.002	0.004	$K_i p_1$
SN02	5	2	1	1	16	2	42	0.003	0.005	$K_i K_j p_1$
SN03	5	1	1	1	16	2	42	0.003	0.004	$EK_{i,j} p_1$
SN04	5	1	1	3	216	92	3286	0.484	0.489	$EK_{i,j}\{p_1, p_2, p_3\}$
SN05	5	1	1	3	216	92	3286	0.566	0.571	$DK_{i,j}\{p_1, p_2, p_3\}$
SN06	5	1	1	1	16	2	42	0.006	0.007	$CK_{i,j} p_1$
SN07	5	1	1	∞	216	all	7776	0.825	0.838	$CK_{i,k} p_1$
SN08	5	1	1	3	216	92	3286	0.216	0.221	$K_i\{p_1, p_2, p_3\}$
SN09	5	1	2	3	306	107	7652	0.759	0.767	$K_i\{p_1, p_2, p_3\} \wedge \neg K_j\{p_1, p_2, p_3\}$
SN10	5	1	3	3	614	189	20334	2.049	2.069	$K_i\{p_1, p_2, p_3\} \wedge \neg K_{j\wedge k}\{p_1, p_2, p_3\}$
SN11	5	1	5	3	901	265	47570	4.841	4.886	$K_i\{p_1, p_2, p_3\} \wedge \neg K_{other}\{p_1, p_2, p_3\}$
SN12[1]	5	1	5	∞	2808	all	505400	57.5	58.0	$K_i\{p_1, p_2, p_3\} \wedge \neg K_{other}\{p_1, p_2, p_3\}$
SN13	5	1	2	∞	432	all	31104	5.589	5.629	$\neg K_i\{p_1, p_2, p_3\} \wedge K_{other}\{p_1, p_2, p_3\}$
SN14[2]	5	1	2	3	418	278	19964	4.049	4.073	$\neg K_i\{p_1, p_2, p_3\} \wedge K_{other}\{p_1, p_2, p_3\}$

TABLE 5.4: Experiments Results for the Social-media Network domain

5.4 Discussion

Overall, computationally, our solution outperforms PDKB, which is state-of-the-art for epistemic planning problems. As it can be seen from the result in both *Corridor* and *Grapevine* domains, the number of agents and depth of epistemic relations do not increase the computation time as rapidly as the PDKB planner.

In the terms of expressiveness, our solution demonstrates its capability to handle variety of complex epistemic relations, such as, nested knowledge, distributed knowledge and common knowledge, or epistemic logic reasoning with continuous domain which can be found in the scenarios of *BBL* and *SN* domains.

The results show that the computational time depends heavily on how many times the external functions are called, which is actually determined by the number of generated nodes and expanded nodes. Moreover, the number of nodes involved in the search is impacted by some factors, such as, the length of the plan, the algorithm that the planner uses, and also the scale of the problem itself.

The results also show that the external solver takes up a large part of the execution time. This is a prototype implementation and this represents an opportunity for performance optimisation of our code base.

Chapter 6

Conclusions

In this work, we introduced a new epistemic planning model called the agent perspective model, driven from the intuition: "What you know is what you see". This perspective model allows us to evaluate epistemic formula, even nested, distributed or common epistemic relations, based on the simple concept of defining an agent's local state. Then, by separating the planning task from epistemic reasoning with F-STRIPS, we proposed an expressive and flexible solution for most of the epistemic planning problems without an expensive pre-compilation step. We implemented our model on well-known epistemic planning benchmarks and two new scenarios based on different perspective functions. The results not only show that our model can solve the epistemic benchmarks efficiently, but also demonstrate a variety types of epistemic relations can be handled. Our work is the first to delegate epistemic reasoning to an external solver.

For future work, there are three ways to extend our model. The first is to extend the model to beliefs rather than knowledge. The success of our model is dependent on the property $f_i(s) \subseteq s$ for perspective functions, which implies beliefs cannot be false. Extending to beliefs would increase certain expressiveness of our model. Second, we can improve our model by allowing simplified disjunctive knowledge relations, such as that proposed by Miller et al. (2016). In such way, we believe that our model have the potential to handle the partial information on variables, such as "value of x is small than 5". Finally, investigating the relation between event-model in modal logic and epistemic logic reasoning in our external functions would be another direction. Instead of seeing variables and evaluating epistemic relations lazily based on those variables, we think it would be possible to allow agents "seeing the action" from the planner and "gaining certain knowledge" from the external functions, which might lead us to a different perspective on decomposing and solving epistemic planning problems.

Bibliography

Baker, Brenda S. and Robert E. Shostak (1972). "Gossips and telephones". In: *Discrete Mathematics* 2.3, pp. 191–193. DOI: 10.1016/0012-365X(72)90001-5. URL: https://doi.org/10.1016/0012-365X(72)90001-5.

Bellemare, M. et al. (2013). "The Arcade Learning Environment: An evaluation platform for general agents." In: *JAIR* 47.

Bolander, Thomas (2014). "Seeing is Believing: Formalising False-Belief Tasks in Dynamic Epistemic Logic". In: *Proceedings of the European Conference on Social Intelligence (ECSI-2014), Barcelona, Spain, November 3-5, 2014*. Pp. 87–107. URL: http://ceur-ws.org/Vol-1283/paper%5C_14.pdf.

— (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*. 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.21.9-34. URL: https://doi.org/10.3166/jancl.21.9-34.

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.

Cooper, Martin C. et al. (2016). "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)*, pp. 193–201. DOI: 10.3233/978-1-61499-672-9-193. URL: https://doi.org/10.3233/978-1-61499-672-9-193.

Dornhege, C. et al. (2009). "Semantic Attachments for Domain-Independent Planning Systems". In: *Proc. ICAPS*.

Engesser, Thorsten et al. (2017). "Cooperative Epistemic Multi-Agent Planning for Implicit Coordination". In: *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017*. Pp. 75–90. DOI: 10.4204/EPTCS.243.6. URL: https://doi.org/10.4204/EPTCS.243.6.

Fagin, Ronald et al. (2003). *Reasoning About Knowledge*. Cambridge, MA, USA: MIT Press. ISBN: 0262562006.

Fan, Jie, Yanjing Wang, and Hans van Ditmarsch (2015). "Contingency and Knowing Whether". In: *Rew. Symb. Logic* 8.1, pp. 75–107. DOI: 10.1017/S1755020314000343. URL: https://doi.org/10.1017/S1755020314000343.

Fikes, R. and N. Nilsson (1971). "STRIPS: A new approach to the application of theorem proving to problem solving". In: *Artificial Intelligence* 1, pp. 27–120.

Frances, Guillem and Hector Geffner (2015). "Modeling and Computation in Planning: Better Heuristics for More Expressive Languages". In: *Proc. ICAPS*.

Francès, Guillem et al. (2017). "Purely Declarative Action Descriptions are Overrated: Classical Planning with Simulators". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 4294–4301. DOI: 10.24963/ijcai.2017/600. URL: https://doi.org/10.24963/ijcai.2017/600.

Gasquet, Olivier, Valentin Goranko, and François Schwarzentruber (2014). "Big brother logic: logical modeling and reasoning about agents equipped with surveillance cameras in the plane". In: *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pp. 325–332. URL: http://dl.acm.org/citation.cfm?id=2615786.

Geffner, Hector (2000). "Functional Strips: A More Flexible Language for Planning and Problem Solving". In: *Logic-Based Artificial Intelligence*. Ed. by Jack Minker. Boston, MA: Springer US, pp. 187–209.

Geffner, Hector and Blai Bonet (2013). *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. ISBN: 9781608459698. DOI: 10.2200/S00513ED1V01Y201306AIM022. URL: https://doi.org/10.2200/S00513ED1V01Y201306AIM022.

Hales, James, Tim French, and Rowan Davies (2012). "Refinement Quantified Logics of Knowledge and Belief for Multiple Agents". In: *Advances in Modal Logic* 9, pp. 317–338.

Haslum, P. et al. (2019). *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool. ISBN: 9781627057370. URL: https://ieeexplore.ieee.org/document/8681776.

Helmert, Malte (2006). "The Fast Downward Planning System". In: *J. Artif. Intell. Res.* 26, pp. 191–246. DOI: 10.1613/jair.1705. URL: https://doi.org/10.1613/jair.1705.

Helmert, Malte and Carmel Domshlak (2009). "Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?" In: *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. URL: http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/735.

Herzig, Andreas and Faustine Maffre (2015). "How to Share Knowledge by Gossiping". In: *Multi-Agent Systems and Agreement Technologies - 13th European Conference, EUMAS 2015, and Third International Conference, AT 2015, Athens, Greece, December 17-18, 2015, Revised Selected Papers*, pp. 249–263. DOI: 10.1007/978-3-319-33509-4_20. URL: https://doi.org/10.1007/978-3-319-33509-4_20.

Hintikka, Jaakko (1962). *Knowledge and Belief*. Ithaca: Cornell University Press.

Huang, Xiao et al. (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, pp. 1093–1101. DOI: 10.24963/ijcai.2017/152. URL: https://doi.org/10.24963/ijcai.2017/152.

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*. Pp. 147–155. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10617.

— (2017). "Multiagent Online Planning with Nested Beliefs and Dialogue". In: *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*. Pp. 186–194. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15748.

Le, Tiep et al. (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*. Pp. 161–170. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17733.

Lipovetzky, Nir and Hector Geffner (2012). "Width and Serialization of Classical Planning Problems". In: *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pp. 540–545. DOI: 10.3233/978-1-61499-098-7-540. URL: https://doi.org/10.3233/978-1-61499-098-7-540.

— (2014). "Width-based Algorithms for Classical Planning: New Results". In: *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pp. 1059–1060. DOI: 10.3233/978-1-61499-419-0-1059. URL: https://doi.org/10.3233/978-1-61499-419-0-1059.

— (2017a). "A Polynomial Planning Algorithm that Beats LAMA and FF". In: *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*. Pp. 195–199. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15740.

— (2017b). "Best-First Width Search: Exploration and Exploitation in Classical Planning". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. Pp. 3590–3596. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14862.

McDermott, D. (2000). "The 1998 AI Planning Systems Competition". In: *Artificial Intelligence Magazine* 21.2, pp. 35–56.

Miller, Tim et al. (2016). "'Knowing Whether' in Proper Epistemic Knowledge Bases". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Pp. 1044–1050. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12291.

Muise, Christian J., Vaishak Belle, et al. (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. Pp. 3327–3334. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9974.

Muise, Christian J., Tim Miller, et al. (2015). "Efficient Reasoning With Consistent Proper Epistemic Knowledge Bases". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pp. 1461–1469. URL: http://dl.acm.org/citation.cfm?id=2773339.

Ramirez, Miquel et al. (2018). "Integrated Hybrid Planning and Programmed Control for Real Time UAV Maneuvering". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. IFAAMAS, pp. 1318–1326.

Richter, Silvia and Matthias Westphal (2010). "The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks". In: *J. Artif. Intell. Res.* 39, pp. 127–177. DOI: 10.1613/jair.2972. URL: https://doi.org/10.1613/jair.2972.

Rintanen, J. (2012). "Planning as satisfiability: Heuristics". In: *Artificial Intelligence* 193, pp. 45–86.

Simon, HA and Allen Newell (1963). "GPS, a program that simulates human thought". In: *Computers and Thought*, pp. 279–293.

Van Ditmarsch, Hans, Wiebe van Der Hoek, and Barteld Kooi (2007). *Dynamic epistemic logic*. Vol. 337. Springer Science & Business Media.

Wan, Hai et al. (2015). "A Complete Epistemic Planner without the Epistemic Closed World Assumption". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 3257–3263. URL: http://ijcai.org/Abstract/15/459.

Wu, Zhongbin (2018). "An Improved Multi-agent Epistemic Planner via Higher-Order Belief Change Based on Heuristic Search". In: *Knowledge Science, Engineering and Management - 11th International Conference, KSEM 2018, Changchun, China, August 17-19, 2018, Proceedings, Part II*, pp. 102–116. DOI: 10.1007/978-3-319-99247-1_10. URL: https://doi.org/10.1007/978-3-319-99247-1%5C_10.