“Seen Is Believing”: Modeling and Solving Epistemic Planning Problems using Justified Perspectives

by

Guang Hu

ORCID: 0000-0003-3629-8040

A thesis submitted in total fulfillment for the degree of Doctor of Philosophy

in the
School of Computing and Information Systems
Faculty of Engineering and Information Technology
THE UNIVERSITY OF MELBOURNE

July 2025

THE UNIVERSITY OF MELBOURNE

Abstract

Faculty of Engineering and Information Technology
School of Computing and Information Systems

Doctor of Philosophy

by Guang Hu
ORCID: 0000-0003-3629-8040

Epistemic planning —– planning that incorporates knowledge and belief (knowledge that could be false) — is crucial in many multi-agent and human-agent interaction settings. However, existing approaches often struggle with scalability, particularly as the number of agents or the depth of nested epistemic relations grows. A notable exception is the state-based methods that use agent's perspective model, which focuses reasoning only on the visible part of states for agents. By delegating epistemic reasoning to an external function, this method enhances expressiveness and efficiency in solving complex epistemic planning tasks. Despite these advantages, the PWP approach has limitations, including an imprecise trade-off between efficiency and completeness and a lack of systematic modeling for beliefs, especially false beliefs.

In this thesis, we extend agent's perspective model to develop a more efficient and effective model for epistemic planning. First, we introduce multiple semantic formats with agent‘'s perspective model to clarify the balance between efficiency and completeness. Then, with the intuition that people reason unseen by retrieving their memory, we extend the original model to handle justified beliefs, resulting in the Justified Perspective (JP) model. Furthermore, we formalize the encoding, design a planner with various search algorithms, and conduct comprehensive experiments demonstrating that our approach is both more efficient and expressive than the current state-of-the-art in epistemic planning. Finally, the JP model is expanded to represent group beliefs (the final missing puzzle of the epistemic logic), including distributed beliefs and common beliefs.

Overall, in this thesis, we provide an efficient and expressive planning framework, including an (action) model-free epistemic logic reasoning model, establishing the framework's potential for broader applications.

Declaration of Authorship

I, Guang Hu, declare that this thesis titled, *"Seen Is Believing": Modeling and Solving Epistemic Planning Problems using Justified Perspectives* and the work presented in it are my own. I confirm that:

- The thesis comprises only my original work towards the Doctor of Philosophy except where indicated in the preface;
- due acknowledgement has been made in the text to all other material used; and
- the thesis is fewer than the maximum word limit (100,000 words) in length, exclusive of tables, maps, bibliographies and appendices.

Signed:
Guang Hu

Date:
Feb 2025

Preface

This thesis is submitted in total fulfilment of the requirements for the degree of Doctor of Philosophy at the University of Melbourne. The research presented here was primarily conducted at the School of Computing and Information Systems, Faculty of Engineering and IT, The University of Melbourne, under the supervision of Prof. Tim Miller and A/Prof. Nir Lipovetzky.

Below is the list of publications and manuscripts arising from this thesis. I was the principal author of all papers and contributed more than 50% on each paper. I was responsible for designing the algorithm architectures, processing data, implementation, running experiments, and analyzing the experimental results. My co-authors contributed with feedback to all these processes and the revision of the papers.

- Part of the contents of Section 2.4 in Chapter 2, which is clearly cited, has been done prior to enrolment in the degree. It has been submitted and accepted for the degree of Master of Philosophy: Guang Hu. *What you get is what you see: Decomposing epistemic planning using functional STRIPS*, Masters research thesis, University of Melbourne, 2020. URL `http://hdl.handle.net/11343/235775`.

- Part of the contents of Chapter 3 has been published in the following paper: Guang Hu, Tim Miller, and Nir Lipovetzky. *Planning with perspectives - decomposing epistemic planning using functional STRIPS*. Journal of Artificial Intelligence Research (JAIR), 75:489–539, 2022. doi: 10.1613/JAIR.1.13446. URL `https://doi.org/10.1613/jair.1.13446`.

- Part of the contents of Chapter 4 has been published in the following paper: Guang Hu, Tim Miller, and Nir Lipovetzky. *Planning with multi-agent belief using justified perspectives*. Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling, Prague, Czech Republic, July 8-13, 2023, pages 180–188. AAAI Press, 2023. doi: 10.1609/ICAPS.V33I1.27193. URL `https://doi.org/10.1609/icaps.v33i1.27193`.

- Part of the contents of Chapter 5 is currently under review by the 34th International Joint Conference on Artificial Intelligence (IJCAI) in the following paper: Guang Hu, Tim Miller, and Nir Lipovetzky. *Where common knowledge cannot be formed, common belief can - planning with multi-agent belief using group justified perspectives*. CoRR, abs/2412.07981, 2024. doi: 10.48550/ARXIV.2412.07981. URL https://doi.org/10.48550/arXiv.2412.07981.

I gratefully acknowledge Melbourne Research Scholarship for funding.

Acknowledgements

This journey has been both challenging and fulfilling, and it could not have been achieved without the support of numerous individuals.

My sincere gratitude goes to my supervisors, Professor Tim Miller and Associate Professor Nir Lipovetzky. I am truly grateful for your unwavering guidance and encouragement throughout these years. I cannot imagine better supervisors than you. Your consistent academic and emotional support has been invaluable in my development as a competent researcher with a healthy work-life balance. Tim, you have been an exceptional supervisor, with remarkable intuition and insight on Epistemic Planning, which inspires this work. You also very patient with our back-and-forth discussions on the epistemic logic, which results in this work's solid theoretical foundation. Your optimistic attitude in research is also great help. Nir, your expertise in navigating the wide literature of Planning has been invaluable. Your willingness to provide related works whenever needed and your support in understanding the rich research landscape have been truly appreciated. Moreover, your help and encouragement for me to take on my own project and supervise master students is also invaluable, which helps me built skills and experiences for my further research pursuits.

I would also like to express my gratitude to my Chairs, Professor Rajkumar Buyya and Professor Egemen Tanin. The progress meeting and high level comments gave me a better understanding of where my progress fitting in the Ph.D. lifecycle and helps ensure my finish in time. In addition, I also would like to thank Professor Adrian Pearce for providing some external perspective about my Ph.D. work and offered me help when unforeseen challenges arose.

I was fortunate to have many friendships during my Ph.D. studies in Melbourne. I receive great support from them in many ways. Xin, Ruihan, and Steven, you are like my mentors to me helping me have a smooth start of my Ph.D. study. Anubhav, our discussion about ideas and recent researches on planning are really engaging and inspire my interest on supervision student research project. Lianglu, Chenyuan, Yujing, and Chao, I really enough the time spend with you, filling with langh and joy. I really enjoy being a member of CIS with amazing people taking laughs, playing boardgame and having discussions: Thao, Lyndon, Archana, Stefan, David, Daniel, Michelle, Ronal, Abeer, Giacomo, Grady, Andrew, Jiajia, Emma, Jun, Yuansan, Jia, and Xiang.

I would like to express my heartfelt gratitude to my friends outside the school of CIS. Special thanks to Yangmengfei (Murphy), who I know him from our robotics club. We have growth a solid friendship both personal and professional through out our 5 years experiences in the club we found: Ausdroid student group for Robotics and AI. I really enjoy the after-hour time I spend in this club, which helps me gain knowledge, perspectives, and experiments on real-world multi-agent robotics applications. Participating the RoboMaster AI Challenge on ICRA (International Conference on Robotics and Automation) for 2 times is also an invaluable experiences.

Moreover, working with Murphy also inspire us to initiated student research projects supervision which is a great experiences for me. I would like to thank all the co-supervisors of those projects from 2019 to 2024: Murphy, Ruihan, Chao, Chen, and Chenyuan. In addition, I would like to express my thanks to one of the most motivative and hard-working student supervised by us, Weijia, who implemented one of our ideas, which leads to a submission to IJCAI 2025.

I am deeply grateful to the support of my family whose unwavering support always has my back. My partner, Bichen, your support and companionship provided a constant source of comfort and motivation. To my father, Dingping, and my mother, Bin, I offer my tremendous thanks for providing both financial and emotional support throughout this challenging journey. You are the ones who have given me the opportunity to chase my dream and do want I really want to do.

Thank you!

Guang Hu

Contents

List of Figures

List of Tables

Abbreviations

BBL	Big Brother Logic
BFS or **BrFS**	Breadth-First Search
DEL	Dynamic Epistemic Logic
DFS	Depth First Search
DPS	Duplication Pruning Set
FOND	Fully Observable Non-deterministic Domain
F-PDDL	Functional PDDL
F-STRIPS	Functional STRIPS
GBFS	Greedy Best-First Search
GJP	Group Justified Perspective
GPS	General Problem Solver
HNB	Have No Belief
JP	Justified Perspective Model
KB	Knowledge and Belief
KD45	Axioms **K**, **D**, **4**, and **5**
KT45	Axioms **K**, **T**, **4**, and **5**
LLM	Large Language Model
MEP	Multi-agent Epistemic Planning
NF	Normal Form
NIB	Numbers in Boxes
NMFODD	Non-Markovian Fully Observable Deterministic Domain
NMFOND	Non-Markovian Fully Observable Non-deterministic Domain
NM-F-STRIPS	Non-Markovian Functional STRIPS
PDDL	Planning Domain Definition Language
PEKB	Proper Epistemic Knowledge Bases

PDKB	Proper Doxastic Knowledge Bases
PDR	Precondition and Delete Relaxation
PK	Perspective Keys
POND	Partial-Observable Non-Deterministic
PWP	Planning with Perspective
SC	Selective Communication
SN	Social-media Network
STRIPS	STanford Research Institute Problem Solver
ToM	Theory of Mind

Symbols

$\mathcal{S}$	state space	
$\mathcal{S}$	complete state space	
Agt	a set of agents	
$Prop$	a set of propositions	
S	seeing operator	Definition 3.3
ES	uniform seeing operator	Definition 3.6
DS	distributed seeing operator	Definition 3.6
CS	common seeing operator	Definition 3.6
K	knowledge operator	Definition 3.3
EK	uniform knowledge operator	Definition 3.6
DK	distributed knowledge operator	Definition 3.6
CK	common knowledge operator	Definition 3.6
B	belief operator	Definition 4.2
EB	uniform belief operator	Definition 5.1
DB	distributed belief operator	Definition 5.1
CB	common belief operator	Definition 5.1
R	retrieval function	Definition 4.5
$\perp$	a value None	Definition 4.3
$s_\perp$	a none-value state	
$\langle\ \ \rangle$	state/sequence override function	Definition 4.1
f (Chapter 2 and 3)	perspective function	Definition 3.4
O (Chapter 4, 5 and 6)	observation function	Definition 4.3
f (Chapter 4, 5 and 6)	justified perspective function	Definition 4.6
cf (Chapter 2 and 3)	common perspective function	Definition 3.7
cO (Chapter 4, 5 and 6)	common observation function	Definition 3.7

cf (Chapter 4, 5 and 6)	common JP function	Definition 5.10
ef	uniform JP function	Definition 5.3
df	distributed JP function	Definition 5.6
C	possible sequence function	Definition 4.12
C_d	distributed possible sequence function	Definition 5.8
C_c	common possible sequence function	Definition 5.13

Chapter 1

Motivation and Introduction

Theory of Mind is fundamental to human cognition, enabling individuals to interpret and predict the behaviour of others by attributing mental states to them.

— Frith and Frith

It has been more than 70 years since Turing proposed the famous question: "*Can machine think?*", Artificial Intelligence (AI) has become a popular topic with numerous studies on its applications and theory. As mentioned by Russell and Norvig [10], AI can help humans manage the complexity of modern life. With the advance of the techniques in the field of AI, more and more intelligent agents are involved in human life. How those agents co-exist without any conflict becomes a new challenge to solve, especially when those agents do not belong to the same system. In addition, as one of the goals in AI research [11], AI applications are designed to enhance human abilities such as decision-making and problem-solving. To make intelligent agents work in a multi-agent environment or in a human-agent interaction environment, as mentioned by Breazeal [12] integrating social intelligence into AI systems is essential for creating machines that can collaborate effectively with humans.

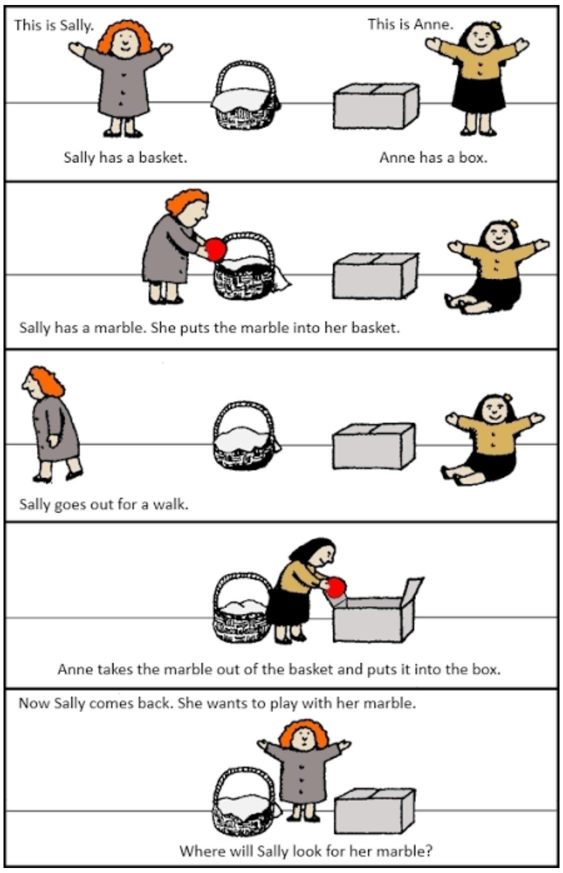

This is Sally.
This is Anne.
Sally has a basket.
Anne has a box.
Sally has a marble. She puts the marble into her basket.
Sally goes out for a walk.
Anne takes the marble out of the basket and puts it into the box.
Now Sally comes back. She wants to play with her marble.
Where will Sally look for her marble?

FIGURE 1.1: Illustration of the Sally-Anne task.

1.1 Motivation

As one fundamental concept in social intelligence, Theory of Mind (ToM) attracts the attention of many researchers. Early foundational work by Premack and Woodruff [13] framed ToM as a cognitive capacity essential for predicting and interpreting behaviour. ToM involves not only understanding someone else's mental state — including their knowledge, beliefs, and inferences — but also recognizing that they may hold beliefs different from what we know to be true. The *Sally-Anne task*, also known as the *false-belief task*, was introduced by Baron-Cohen et al. [14] and is widely used to examine individuals' ToM abilities.

Example 1.1 (The Sally-Anne Task)**.** *As shown in Figure 1.1, the setup is as follows: two individuals, Sally and Anne, are in a room with a basket and a box, both of which are covered. The story begins with Sally placing a marble in the basket before leaving for a walk. While she is away, Anne moves the marble from the basket to the box. The question is: where will Sally look for her marble when she returns?*

To pass the test, the subject should be able to come up with the conclusions that Sally will look into the basket for her marble. That is, in Sally's mindset, she believes she put the marble into the basket before she left and there is no evidence in her mind to show the location of the marble has been changed. Passing this task demonstrates the individual has a basic ability in ToM that can believe someone else holds a false belief.

In the field of AI, this concept was adapted in many different research efforts. At a high level, those researches can be divided into two categories based on the fundamental methodology used, which are: the learning-based approach and the model-based approach.

The first attempt to use a learning-based approach to model ToM is proposed by Rabinowitz et al. [15], in which they use a meta-learning approach to observe another agent's behaviour across multiple episodes, gradually inferring that agent's "policy embedding". They call the Theory of Mind neural network they trained as ToMnet. Their results demonstrate that a trained ToMnet can predict actions based on the agent's hidden or incorrect beliefs. Another indirect but relevant field is *Opponent Modeling* using Deep Reinforcement Learning [16], in which ToM was not explicitly modeled. In addition, more recently, with the development of LLMs, some of the LLM models are also capable of handling ToM tasks [17–19].

However, critics argue that the learning-based AI systems lack genuine ToM, instead relying on statistical correlations rather than causal understanding [20]. Consequently, these concerns over black-box reasoning and potential misalignment with human mental-state semantics have led researchers to consider model-based approaches — where explicit representations of beliefs, desires, and intentions enhance transparency, interpretability, and safety.

As one of the model-based approaches, Bayesian methods have emerged as a powerful framework for modeling ToM. Foundational work is proposed by Baker et al. [21]. In their framework, observers assume agents act (approximately) rationally to achieve their goals, and they use Bayesian updates to infer what the agent must believe or desire. In addition, there is much further research that uses Bayesian models [22–29]. However, as mentioned by Baker et al. [25], the computational challenge persists in scaling these models to real-world applications without approximations.

Another standard model-based approach is *Planning*, which formulates a sequence of actions to achieve objectives (goals) of the agents. Generally, planning to solve a problem has: a model to represent the problem; a language that is used to form a specific description of the problem; a solver to solve the problem automatically; a controller as the result for the agent to act. The basic model in planning is the *Classical Planning* model, in which the problem is *deterministic* and *fully-observable*. Although classical Planning itself cannot handle ToM, adaptations by adding knowledge or belief logic in the action started to appear in the late 20th century [30–33]. From this foundation, a substantial body of research on *Epistemic Planning* has developed both before and after the formal introduction of the term by Bolander and Andersen [34], continuing to expand to the present day.

1.2 Introduction

As for this thesis, the above motivation demonstrates the need for agents to reason about epistemic logic, which is essential for multi-agent and human-agent interaction applications. Compared to other methods, planning-based approaches have advantages in explainability, transparency, safety, and reliability with enough generalization and flexibility. Thus, this thesis is about epistemic planning. However, all existing approaches for epistemic planning face some drawbacks (mainly scalability), which limits the development and its potential for real-world applications (Details in Section 2.3.3). Thus, the research question that this thesis is answering is:

How can we enable expressive modelling and efficient solving of epistemic planning problems?

Throughout this thesis, firstly, we provide a comprehensive background and literature about epistemic planning. This includes the foundation for epistemic planning from both sides (planning and epistemic logic), as well as the existing research. From the literature, we pinpoint the constraints (mainly scalability) of existing epistemic planning methods that hinder the advancement of the research and its potential for application. In addition, we identify the first attempt to use a fundamentally different method by Hu [4] to model and solve epistemic planning problems, which is known as the agent's perspective model.

Hu's intuition is that people know things by observation. Then, they reason about knowledge by generating agents' perspectives (observations), which are effectively local states representing the observable part of the world for each agent. They claim that for any epistemic domain, the modeler just has to define the domain-dependent perspective function, which is the seeing rules of this domain, while their semantics do the reasoning parts about agents' (nested) knowledge. With this agent's perspective model, they are able to reason about nested knowledge and group knowledge by forming corresponding perspectives using set operations. In addition, they used a non-standard planning language to separate the epistemic reasoning from planning. By doing so, they can lazily evaluate epistemic relations by an external reasoner when solving epistemic planning problems. Moreover, their framework can work with any planner that supports external functions.

We found their solution is promising for bridging the gap between epistemic planning and its real-world application due to the efficiency and expressiveness of their framework. Therefore, this thesis digs deeper into the agent's perspective model to investigate the properties (soundness and completeness) their model traded for efficiency. Then, we formalize our findings and propose new semantics to balance between efficiency and these properties.

Then, this thesis goes beyond the agent's perspective model as it only handles knowledge. Goldman [35] claims that people believe something they have seen, until they see evidence to suggest otherwise. Following Goldman's justified belief, instead of only modeling agents' perspectives (observations), we propose a new model, namely the Justified Perspective (JP) Model, that reasons about individual agents' nested epistemic logic, including both knowledge and belief. Agents' justified perspectives are formed by composing their current observation with their past observations. That is, we reason about agents' epistemic relations using the sequence of states rather than just the current state. We show the formalization of the JP model as well as its semantics and axiomatic system. Moreover, we also provide a problem encoding with a widely used planning language and adopt some existing algorithms to search the state sequences. With comprehensive experiments, this thesis shows the JP model is an expressive and efficient framework to solve single-agent (nested) epistemic planning problems.

Further, this thesis extends the JP model to handle group (nested) beliefs. Differing

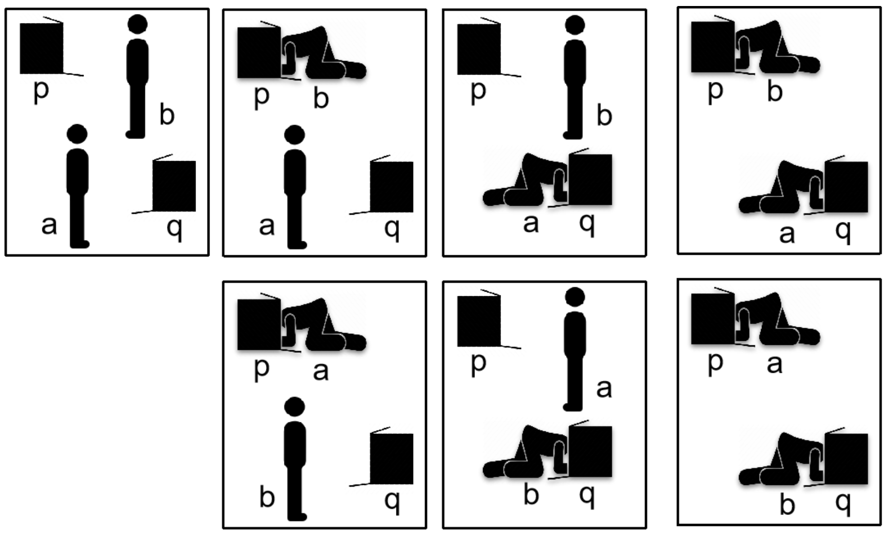


FIGURE 1.2: Illustration of the example states in the Number In Boxes domain.

from agents' knowledge, where it cannot be false since it is derived from the global state (ground truth), agents can hold false beliefs. Thus, agents' group beliefs cannot simply be handled by merging their justified perspectives, like in the agent's perspective model. In this thesis, we show how we "safely" merge agents' justified perspectives for the distributed belief, and how we efficiently handle agents' common belief by forming a fix-point set of justified perspectives.

To better illustrate our work, we proposed a complex version of the Sally-Anne Task that suits in planning as follows.

Example 1.2 (Numbers In Boxes)**.** *As shown in Figure 1.2, there are two agents, a and b, and two numbers, p and q, hidden in separate boxes. The value of each number ranges from 0 to 99.*

The agents do not know the value of the numbers unless they peek into the corresponding box. Each box can only be peeked at by one agent at a time.

The numbers can be modified (incremented or decremented by 1) by a hidden third agent, without a or b noticing. Consequently, neither a nor b would know that a number has been changed unless they are peeking at it or peeking afterwards.

This example domain is the Numbers in Boxes (NIB) domain. The problems in this domain can be identified by describing the initial situation and desirable goal conditions. An example problem can be described by assuming the initial value for p and q are

4 and 6 and no agent is peeking at any box, and the goal conditions could be both boxes being peeked at by an agent (any of the two example states on the right side of Figure 1.2) and $p \times q < 10$. The goal conditions determine whether the problem instance belongs to a classical planning problem or an epistemic planning problem. If the goal conditions contain epistemic relations (knowledge or beliefs), then the problem instance needs to be modeled as an epistemic planning problem [1] rather than a classical planning problem [2]. Throughout this thesis, the above example is used to demonstrate the approaches encountered as background or related work, as well as the approaches proposed by this thesis.

[1]The formal definitions of epistemic planning problem is given in Section 2.3.1.1.

[2]An exception is the methods that model and solve epistemic planning problems by converting it into a classical planning problem. For those approaches, we still consider them as solving epistemic planning problem instead of classical planning problem.

Chapter 2

Epistemic Planning

It is this that we omitted in our investigation of the nature of virtue, when we said that only knowledge can lead to correct action, for true opinion can do so also.

—Plato

In this chapter, we provide the background from both directions of Epistemic Planning, which are Classical Planning and Epistemic Logic. Then, we review the two traditional directions in Epistemic Planning. At the end, we raise the research question and outline of this thesis.

2.1 Automated Planning

Planning is the model-based approach to action selection in artificial intelligence, where the model is used to reason about which actions an agent should take to achieve some objective, such as reaching a goal [37]. The concept initially appeared in General Problem Solver (GPS) [38]. Although their intuition is to design a system that mimics human problem-solving processes, which is breaking down the complex task into simpler subproblems and addressing them systematically, it is still providing groundwork for subsequent developments in AI planning. That is, decomposition of goals and strategic sequencing of actions to achieve those goals. This idea has been further investigated by

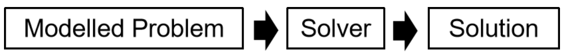


FIGURE 2.1: Illustration of planning process to solve a problem instance.

Fikes and Nilsson [39], who developed a new problem-solving program, namely STRIPS (STanford Research Institute Problem Solver). They illustrate the "planning" idea from GPS with the process of: modelling the problem (with STRIPS language), solving the problem and finding a solution (As shown in Figure 2.1).

There are many types of planning, including but not limited to: *classical planning*, *conditional planning*, *temporal planning*, *generalized planning*, *hierarchical planning*, and *epistemic planning*. Despite their differences, all of these follow a similar overall process.

The fundamental distinction between these variations lies in the type of problems that each model handles. These models differ based on the assumptions made about the dynamics of the world. For example, classical planning models assume deterministic, instantaneous effects of actions and complete knowledge of the world, whereas temporal models account for actions with durations, and partially-observable Markov-decision-process models represent uncertainty through belief distributions over possible states of the world.

As noted by Ghallab et al. [40], the conceptual model is not intended to be operational. Instead, problems that can be represented by these models are concisely described using declarative languages such as STRIPS [39] and PDDL (Planning Domain Definition Language) [41]. These languages are sufficiently general to encode a variety of problems while simultaneously revealing structural information that enables planners to scale to large and complex problems. Once a problem instance is described, the planner applies search algorithms to find a solution. The agent then follows this solution to complete the task. The nature of the solution varies depending on the problem type: for classical planning, it is typically a plan (a sequence of actions); while for more complex problems, such as those involving non-deterministic actions, the solution may be a policy, which maps every possible state to the best action.

2.1.1 Planning Assumptions

When it comes to planning assumptions, classical planning, as the foundational model of planning, is the paradigm with the most fundamental assumptions that simplify the planning problem. The assumptions for classical planning have been outlined and discussed in various works [10, 37, 39, 40]. The most systematic analysis on planning assumptions is done by Ghallab et al. [40]. These assumptions include, but are not limited to, the following: the planning domain is finite, discrete (instantaneous), deterministic, fully observable, static, Markovian, with restricted goals, considered as single-agent, uses sequential plans, and operates within a closed-world assumption.

Assumption 1 (Finite)**.** The set of all possible states and the set of all possible actions for a classical planning problem need to be finite. Without this assumption, theoretically, any algorithm that is used to solve this problem can neither guarantee the completeness nor termination of itself [42]. In addition, in its application, even with the finite assumption, classical planning problems can be computationally challenging (PSPACE-complete) [43]. Thus, a finite number of possible states and possible actions is a well-accepted assumption in classical planning.

Assumption 2 (Discrete)**.** Classical Planning models problems where both state and action are discrete. Discrete state space can be ensured by a finite number of possible states, while discrete action is not only about the finite number of actions. This also ensures that the application of the action is instantaneous, which is discussed when comparing to Temporal Logic in planning [44]. That is, in classical planning, states transition from one to another instantly by applying an action.

Assumption 3 (Deterministic)**.** All actions in classical planning need to be deterministic. That is, in a given state, the outcome of performing an (available) action is singular and predictable. This allows the planner to use logical reasoning to determine the outcomes of action sequences.

Assumption 4 (Fully Observable)**.** The "fully observable" assumption posits that the agent has complete and accurate knowledge of the current state of the environment at all times. Similarly, this assumption is originally from STRIPS [39]. It is formed by the fact that the initial state is fully observable, as well as all actions' effects, which indicate that every state in the problem-solving process is fully observable.

Assumption 5 (Static)**.** The "static" assumption refers to the premise that the environment remains unchanged except as a direct result of the acting agent's actions. The assumption originates from STRIPS [39] in 1971. In STRIPS, a state is modeled as a set of propositions, which currently hold, and actions (operators) are used to transition between states. The framework assumes that no external events or environmental dynamics modify the state unless explicitly specified by the defined actions.

Assumption 6 (Markovian)**.** The Markovian assumption is an important concept in planning, stating that the next state resulting from an action depends solely on the current state and the action executed, not on the sequence of previous states or actions. While in classical planning this property is often implicit, only a few works have mentioned it [45]. With the above assumptions (Assumption 1 and Assumption 3), this assumption is guaranteed. That is, for any non-Markovian problem with a finite number of possible states and a finite number of possible actions, the number of possible state sequences without repeats is also finite. Then, we can model the original problem by modeling each state sequence as a state in the new model, which makes the problem become Markovian.

Assumption 7 (Restricted Goals)**.** The goal states for the problem need to be specified. The objective of solving a problem is to select and apply actions to reach one of the states that are considered as goal states. This assumption explicitly avoids extended goals, such as states or transition constraints when solving the problem or utility functions.

Assumption 8 (Single-Agent)**.** The "single agent" assumption in classical planning needs some clarification. It restricts the problem to scenarios where only one decision-making entity is responsible for finding a sequence of actions to achieve the goal. Originally, in STRIPS [39], this assumption also requires the acting agent to be the same entity as the decision-making one. However, this limitation seems unnecessary. When the problem, which this entity is trying to solve, contains multiple acting agents, who follow the instructions (plan) generated by that decision-making entity, the plan generated is still in the single-agent setting. The explicit discussion on this can be found in the latest edition (Edition 4) of Russell and Norvig's book [10]. They distinguish whether an entity should be modeled as an agent by whether its performance measure (goals) completely or partially conflicts with others. For example, in modeling a multi-agent pathfinding problem (using the simplest version), in which each agent has a goal position to visit in a grid and they cannot be at the same location at the same time, the modeler could consider this problem as single-agent if they consider the whole agent group as the "single agent". That is, this "single agent" has to choose one move for each acting agent without collision as one action in the modeled problem.

Assumption 9 (Sequential Plan)**.** A solution to the problem should be a plan, which is a finite linearly ordered sequence of actions. This assumption is consistent with Assumption 1, Assumption 3 and Assumption 8.

Assumption 10 (Closed World)**.** The ”closed-world” assumption in classical planning posits that any fact not explicitly stated is considered false. This was first formalised by Fikes and Nilsson [39]. They defined the state as a set of propositions (they called well-formed formula) and the initial state is all the propositions that are initially true. In addition, the state transition is done by operators, which contain a set of preconditions and effects, where effects contain a list of propositions that become true (need to be added to the state) and a list of propositions that become false (need to be removed from the state). Their initial state aligns with the closed world assumption, and their operator mechanism ensures the assumption holds all the time.

Compared to the list of assumptions in Ghallab et al. [40], we omitted the assumption of “offline planning”, as all the works covered in this thesis are in the offline setting. While we cover a few new assumptions, as the main research topic of this thesis, Epistemic Planning, is: in a multi-agent (or human-agent interaction) environment (Assumption 8); and, one’s belief depends on what happened in the past (Assumption 6); and, one’s perspective of the world could follow the closed-world or possible-worlds assumption (Assumption 10).

2.1.2 Planning Models

The planning model determines the type of problems that could be represented. We start by explaining the basic planning model, *Classical Planning Model*, and then briefly mention other planning models.

2.1.2.1 Classical Planning Model

Classical Planning Model, also known as a state-space model, is the widely accepted theoretical ground for many different variations of planning. It was originated from Mathematics (state space) and adapted by researchers studying AI about robotic control [46]. Then, the model is used by Ghallab et al. [40] as the basic theoretical ground

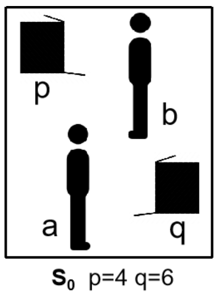


FIGURE 2.2: Example initial state in the NIB domain.

for planning. A formal definition is found in Geffner and Bonet [37]'s work which is commonly used by the current planning community. Here, we begin with Geffner and Bonet's definition, as it is somewhat simpler to comprehend than other definitions.

Definition 2.1 (State-Space Model)**.** Following all assumptions in Section 2.1.1, a classical planning problem instance P can be defined by a tuple as follows:

$$P = (\mathcal{S}, s_0, \mathcal{S}_G, A, t, c), \text{ where:}$$

- $\mathcal{S}$ is the set of all possible states.
- s_0 is the initial state of the problem instance that $s_0 \in \mathcal{S}$.
- $\mathcal{S}_G$ is the set of goal states that $\mathcal{S}_G \subseteq \mathcal{S}$.
- A is the set of all action, where the applicable action for a given state s is $A(s) \subseteq A$.
- $t : \mathcal{S} \times A \to S$ is the deterministic transition function that returns the next state s' when applying action a in state s
- $c : \mathcal{S} \times A \to \mathbb{R}_{\geq 0}$ is the cost function that determines the cost $c(s, a)$ (a real number that is larger or equal to 0) of applying action a in state s.

In the planning paradigm, a plan (solution) is a sequence of actions $\vec{a} = [a_0, \ldots, a_n]$ that could transition the initial state s_0 to one of the goal states $s_g \in S_G$. That is, $t(\ldots t(s_0, a_0) \ldots, a_n) = s_g$. A plan is optimal if and only if there does not exist another plan that has less cost $(\Sigma_{i \in \{0,\ldots,n\}} c(s_i, a_i))$. The concept of the state space in the above model might be clear to any AI researchers, but it might be too abstract to the reader without relative background. Thus, we use the following example to explain how it works.

Example 2.1 (NIB Example for Classical Planning)**.** *Following the same domain as described in Example 1.2, we specify the initial state and goals. Initially, as depicted in*

Figure 2.2, no agents are peeking into the boxes and the values of the numbers p and q are 4 and 6, respectively. The goal is to ensure that both numbers, p and q, are being peeked at, regardless of which agent is looking at which number.

One way to model the above example is:

$$\mathcal{S} = \left\{ \begin{pmatrix} \mathrm{pt}_a, & \mathrm{pt}_b, \\ \mathrm{value}_p, & \mathrm{value}_q \end{pmatrix} \middle| \begin{array}{c} \mathrm{pt}_a, \mathrm{pt}_b \in \{0,1,2\}, \mathrm{pt}_a \times \mathrm{pt}_b \notin \{1,4\}, \\ \mathrm{value}_p, \mathrm{value}_q \in \{0,\ldots,99\} \end{array} \right\}$$

Each state in the NIB domain can be represented by 4 elements: pt_a and pt_b are the postures of the agents, where 0, 1 and 2 are postures that are standing, peeking at the box containing p and peeking at the box containing q; $\mathrm{pt}_a \times \mathrm{pt}_b \notin \{1,4\}$ ensures both agents will not peek into the same box; value_p and value_q represent values of p and q. With the above state space, the initial state is $s_0 = (0,0,4,6)$. The goal states are $\mathcal{S}_G = \big\{(1,2,x,y) \mid x,y \in \{0,\ldots,99\}\big\} \cup \big\{(2,1,x,y) \mid x,y \in \{0,\ldots,99\}\big\}$, which is a set union of: all states that a sees p and b sees q; and, all states that a sees q and b sees p. The returned set of available actions by the action function for any given state is:

$$A\big((\mathrm{pt}_a, \mathrm{pt}_b, \mathrm{value}_a, \mathrm{value}_b)\big) = \begin{cases} \emptyset & \text{base case} \\ \cup\{\text{peek(a,p)}\} & \text{if } \mathrm{pt}_a = 0, \mathrm{pt}_b \neq 1 \\ \cup\{\text{peek(a,q)}\} & \text{if } \mathrm{pt}_a = 0, \mathrm{pt}_b \neq 2 \\ \cup\{\text{peek(b,p)}\} & \text{if } \mathrm{pt}_a \neq 1, \mathrm{pt}_b = 0 \\ \cup\{\text{peek(b,q)}\} & \text{if } \mathrm{pt}_b \neq 2, \mathrm{pt}_b = 0 \\ \cup\{\text{return(a,p)}\} & \text{if } \mathrm{pt}_a = 1 \\ \cup\{\text{return(a,q)}\} & \text{if } \mathrm{pt}_a = 2 \\ \cup\{\text{return(b,p)}\} & \text{if } \mathrm{pt}_b = 1 \\ \cup\{\text{return(b,q)}\} & \text{if } \mathrm{pt}_b = 2 \\ \cup\{\text{increment(p)}\} & \text{if } \mathrm{value}_p < 99 \\ \cup\{\text{decrement(p)}\} & \text{if } \mathrm{value}_p > 0 \\ \cup\{\text{increment(q)}\} & \text{if } \mathrm{value}_q < 99 \\ \cup\{\text{decrement(q)}\} & \text{if } \mathrm{value}_q > 0 \end{cases}$$

At last, the transition function is straightforward in that one value of the state will be changed by the applied action accordingly. A valid plan is one of [peek(a,p),peek(b,q)], [peek(a,q),peek(b,p)], [peek(b,p),peek(a,q)] and [peek(b,q),peek(a,p)].

As mentioned by Ghallab et al. [40], the classical planning model is important in planning as one of the common motivations in most of the scientific research when facing a complex question is to make restrictive assumptions. With reasonable assumptions, a simplified problem can be worked out with general models (languages) and approaches.

2.1.2.2 Other Planning Models

As mentioned in its definition (Definition 2.1), the classical planning model follows all assumptions in Section 2.1.1, as it provides the basics for all planning directions. Those directions are found by relaxing one or many of those assumptions. In this thesis, we only introduce the relevant ones.

The deterministic assumption (Assumption 3) receives much attention. In contrast to deterministic actions, the alternatives are non-deterministic, probabilistic, and stochastic actions. The classical planning domains with non-deterministic actions are named as Fully-Observable Non-deterministic Domain (FOND) in Ghallab et al. [40]'s book. The problem in FOND can be represented by a tuple that $P = (\mathcal{S}, s_0, \mathcal{S}_G, A, t, c)$, where the transition function becomes $t : \mathcal{S} \times A \to \mathcal{P}(\mathcal{S})$. That is, one action could have a set of possible next states. A solution for any problem instances in FOND would be a policy (a function that inputs a state and returns an action) instead of a plan.

From the domain in FOND, Brafman and Giacomo [47] introduced the Non-Markovian Fully Observable Non-deterministic Domain (NMFOND) by also relaxing the Markovian assumption (Assumption 6). A problem instance in NMFOND is often defined by a tuple $P = (\mathcal{S}, s_0, \mathcal{S}_G, A, t, c)$, where the transition function $t : \mathcal{S}^+ \times A \to \mathcal{S}$ and the goal becomes a set of state sequences $S_G \subset \mathcal{S}^+$.

2.1.3 Planning Language

Planning languages are the bridge between the human modelers and the planning system. Although the planning model defines the theoretical ground of the planning problems, the planning language is required as a practical tool to encode problems within the corresponding planning model.

2.1.3.1 STRIPS

As the earliest language that can model classical planning problems, STRIPS [39] represents a classical planning problem as a tuple:

$$P = \langle F, O, I, G \rangle, \text{ where:}$$

- F is the set of all possible facts (positional variables, also named as fluents).
- O is the set of all operators.
- $I \subseteq F$ is a set of all true facts in the initial situation.
- $G \subseteq F$ is a set of facts that need to be true as the goal conditions.

The state representation in STRIPS is a subset of F, where all propositions that hold true are included in the set, while those that do not hold are excluded, in alignment with the closed-world assumption. Each operator o in STRIPS consists of three components:

- $pre(o)$: The set of propositions (preconditions) that must be true for the action to be applicable.
- $add(o)$: The set of propositions that become true as a result of the action.
- $del(o)$: The set of propositions that become false as a result of the action.

State transitions occur by applying an operator, transforming one set of propositions into another. Since STRIPS does not account for customized operator costs, a plan is optimal if there is no plan with fewer operators taken.

Using Example 2.1, the set of all facts can be expressed as:

$$F = \left\{ \begin{array}{l|l} \text{peeking}(x, y), \text{standing}(x), & x \in \{a, b\}, y \in \{p, q\} \\ \text{free}(y), \text{being\_peeked(y)}, \text{val}(y, z) & z \in \{0, \ldots, 99\} \end{array} \right\}$$

The propositions peeking(x, y) indicate whether agent x is peeking into the box containing y. The propositions standing(x) signify that agent x is in a standing position (not peeking into any box), while the propositions free(y) denote that the box containing y is available for an agent to peek into. Finally, the propositions val(y, z) are true if the current value of the number y is z. Overall, any state in the NIB domain can be represented with a F of size $4 + 2 + 2 + 200 = 208$.

The initial state can be represented by:

$$I = \{\text{standing}(a), \text{standing}(b), \text{free}(p), \text{free}(q), \text{val}(p, 4), \text{val}(q, 6)\}$$

The set of goal propositions requires some discussion. Intuitively, "having both boxes to be being peeked at" means one of the peeking(a, p) and peeking(b, p) is true, and one of the peeking(a, q) and peeking(b, q) is true. However, the goal conditions in STRIPS need to be one set of propositions. Thus, "$\{\text{peeking}(a, p), \text{peeking}(b, q)\}$ or $\{\text{peeking}(b, p), \text{peeking}(a, q)\}$" would not be a valid G. On the second thought, free(x) becomes false when someone is peeking at it, which means $\{\neg\text{free}(p), \neg\text{free}(q)\}$ could be the goal conditions. However, STRIPS does not allow negation in its goal condition. Therefore, we need an additional set of propositions to serve as goal propositions, $G = \{\text{being\_peeked}(p), \text{being\_peeked}(q)\}$.

All peek operators O_{peek} can be represented by:

$$O_{\text{peek}} = \left\{ \begin{array}{l|c} \text{peek}(x, y) & \\ \text{pre: } \{\text{free}(y), \text{standing}(x)\} & x \in \{a, b\} \\ \text{add: } \{\text{peeking}(x, y), \text{being\_peeked(y)}\} & y \in \{p, q\} \\ \text{del: } \{\text{free}(y), \text{standing}(x)\} & \end{array} \right\}$$

The peek operators O_{peek} represent actions where an agent x peeks into a box containing the number y. The preconditions, standing(x) and free(y), ensure that agent x is in a standing position and that no one is peeking into the box containing y (ensuring the box can only be peeked at by one agent at a time). Once the operator is performed, agent x is peeking at y (making peeking(x, y) true), while the agent is no longer in a standing position and the box is being peeked at. The return operators O_{return} are just the peek operators swapping the set precondition and delete set to the add set and vice versa.

All increment operators O_{inc} can be represented by:

$$O_{\text{inc}} = \left\{ \begin{array}{l} \text{increment}(y, z, z') \\ \text{pre: } \{\text{val}(y, z)\} \\ \text{add: } \{\text{val}(y, z')\} \\ \text{del: } \{\text{val}(y, z)\} \end{array} \;\middle|\; \begin{array}{l} y \in \{p, q\} \\ z, z' \in \{0, \ldots, 99\} \\ z' = z - 1 \end{array} \right\}$$

This example action is straightforward. It is worth mentioning that the boundary of y is controlled by not having operators, such as increment$(y, 99, 100)$, in O_{inc}.

Overall, O is the union of O_{peek} (size of 4), O_{return} (size of 4), O_{inc} (size of 200) and O_{dec} (size of 200). An example plan for the example goal propositions G would be $[\text{peek}(a, p), \text{peek}(b, q)]$.

2.1.3.2 Planning Domain Definition Language (PDDL)

In addition to STRIPS, the *Planning Domain Definition Language* PDDL [41] is commonly used to model planning problems. The original PDDL [1] was proposed by McDermott et al. [48] in 1998, for the first planning competition at the Artificial Intelligence Planning and Scheduling (AIPS) [49] conference in 2000. The language is originally designed to model all sorts of planning problems instead of just the classical planning problems. A trimmed version of PDDL (known as PDDL 1.2) was introduced by Bacchus [50] at the second AIPS planning competition by pruning unused features and focusing on the classical planning problems.

As mentioned by Haslum et al. [41], although PDDL is intended to be a common modeling language, it is important to recognize that it is not a standard. Thus, there is no "formal" definition of the PDDL. By "formal", we mean that there is no complete and unambiguous formalization of the syntax or semantics for all of PDDL. Practically, to our knowledge, there does not exist an "uniform" planner that supports all of PDDL. Besides, different planner implementers have different interpretations of the ambiguous parts of the PDDL. Therefore, in this thesis, we only provide the preliminary on the parts that are relevant to what we used.

[1] McDermott et al. referred the version of PDDL as "PDDL 0.0", while in most of the work, it has been referred as "PDDL 1.0".

A standard PDDL instance contains two files: a domain file and a problem file, both using the extension ".pddl". The domain file specifies the descriptions of propositions (predicates) and operators (actions). The action description covers the parameters, preconditions, and effects (add and delete set of propositions, similar to operators in STRIPS). The problem gives the objects, initial state, and goal conditions. One of the key advantages of PDDL compared to STRIPS is that the domain file is reusable. Thus, any problem instances from the same domain share one domain file, while the predicates and actions are grounded by the objects in each paired problem file. In addition, the domain file also provides a list of "requirements", which contains some features that are supported by some planners but not all of them, such as: :typing, negative-preconditions.

Syntactically, PDDL is not case-sensitive, and all expressions are enclosed in matching brackets. In 1998, McDermott et al. [48] valid term names "are strings of characters beginning with a letter and containing letters, digits, hyphens (-) and underscores (_)". However, in many planners, hyphens are disallowed in term names, as they are used for a different feature named "typing". Most of the key words start with a colon, except "define", "domain" and "problem" in the file header.

An example is provided using the same problem instance in Example 2.1. The domain file is shown in Code Example 2.1 and the problem file[2] is shown in Code Example 2.2.

```
(define (domain NIB)

    (:requirements :strips :typing :negative-preconditions)

    (:types
        agent num value
    )

    (:predicates
        (peeking ?a - agent ?n - num)
        (standing ?a - agent)
        (free ?n - num)
        (value ?n - num ?v - value)
        (increasing ?v1 ?v2 - value)
        (decreasing ?v1 ?v2 - value)
    )

    (:action peek
        :parameters (?a - agent ?n - num)
        :precondition (and
            (standing ?a)
            (free ?n)
        )
        :effect (and
```

[2]Some lines are omitted for readability. The complete problem file can be found in Appendix B.1.

```
                (peeking ?a ?n)
                (not (free ?n))
                (not (standing ?a))
            )
        )

        (:action return
            :parameters (?a - agent ?n - num)
            :precondition (and
                (peeking ?a ?n)
            )
            :effect (and
                (free ?n)
                (standing ?a)
                (not (peeking ?a ?n))
            )
        )

        (:action increment
            :parameters (?n - num ?v1 ?v2 - value)
            :precondition (and
                (value ?n ?v1)
                (increasing ?v1 ?v2)
            )
            :effect (and
                (value ?n ?v2)
                (not (value ?n ?v1))
            )
        )

        (:action decrement
            :parameters (?n - num ?v1 ?v2 - value)
            :precondition (and
                (value ?n ?v1)
                (decreasing ?v1 ?v2)
            )
            :effect (and
                (value ?n ?v2)
                (not (value ?n ?v1))
            )
        )
)
```

Code Example 2.1: PDDL Domain: NIB

```
(define (problem NIB_example)
    (:domain NIB)
    (:objects
        a b - agent
        p q - num
        v0 v1 v2 v3 v4 v5 v6 v7 v8 v9
...     ; Placeholder for skipped object declarations
        v90 v91 v92 v93 v94 v95 v96 v97 v98 v99 - value
    )

    (:init
        (standing a)
        (standing b)

        (free p)
        (free q)
17
18         (value p v4)
19         (value q v6)
20
21         (increasing v0 v1)(decreasing v1 v0)
22         (increasing v1 v2)(decreasing v2 v1)
23 ...     ; Placeholder for skipped predicate declarations
24         (increasing v97 v98)(decreasing v98 v97)
25         (increasing v98 v99)(decreasing v99 v98)
26
27
28     )
29
30     (:goal (and
31             (not (free p))
32             (not (free q))
33         )
34     )
35 )
```

Code Example 2.2: PDDL Problem: NIB

The above PDDL example follows PDDL 1.2 syntax. It modeled the problem instance from Example 2.1 following a very similar way as in STRIPS language. The state representation for the PDDL is described by the set of facts (predicates) that are true in it, which is the same as STRIPS. Everything other predicate that is not in the current state is assumed to be false, which follows the closed world assumption (Assumption 10). Specific to the above example, :predicates, :types and :objects form the set of all facts (same as F is STRIPS). Since all PDDL 1 only models propositions (predicates), the values of two numbers p and q need to be enumerated as a set of propositions, which are covered by Line 13 in the domain file and Line 3-9 in the problem file.

PDDL differs mainly from STRIPS in that certain components, specifically the domain file, of a problem instance in PDDL can be applied to describe other instances within the same domain. This offers greater generalizability in PDDL compared to STRIPS. Moreover, as outlined in :requirements, PDDL is versatile enough to accommodate features such as ":negative-preconditions", which broadens the scope of problems it can model beyond the classical planning problems.

Although there are many modifications of the language, including but not limited to: PDDL 2.1 [51], PDDL + [52], PDDL 2.2 [53], PDDL 3.0 [54] and PDDL 3.1 [55]. Here, we only discuss the modifications that are relevant to this thesis, which are PDDL 2.1.

As depicted in earlier examples, both PDDL 1 and the STRIPS framework represent problems through facts, which are propositional variables. This implies that any variable's value in a problem's state is strictly *true* or *false*. Nevertheless, to effectively model real-world challenges—where constraints and objectives frequently entail numeric reasoning—numeric variables become essential. Therefore, PDDL 2.1, introduced by Fox and Long [51], incorporates instances of "functions" [3]. An illustration of how the NIB problem instance (Example 2.1) is modeled in PDDL 2.1 is provided in Code Example 2.3 and Code Example 2.4.

```
(define (domain NIB)

    (:requirements :strips :typing :negative-preconditions)

    (:types
        agent num value
    )

    (:predicates
        (peeking ?a - agent ?n - num)
        (standing ?a - agent)
        (free ?n - num)
    )

    (:functions
        (value ?n - num) - value
    )

    (:action peek
        :parameters (?a - agent ?n - num)
        :precondition (and
            (standing ?a)
            (free ?n)
        )
        :effect (and
            (peeking ?a ?n)
            (not (free ?n))
            (not (standing ?a))
        )
    )

    (:action return
        :parameters (?a - agent ?n - num)
        :precondition (and
            (peeking ?a ?n)
        )
        :effect (and
            (free ?n)
            (standing ?a)
            (not (peeking ?a ?n))
        )
    )

```

[3]Here, "functions" diverges from its mathematical definition; it resembles variables acquiring a numeric value.

```
    (:action increment
        :parameters (?n - num)
        :precondition (and

        )
        :effect (and
            (assign (value ?n) (+ (value ?n) 1))
        )
    )

    (:action decrement
        :parameters (?n - num)
        :precondition (and

        )
        :effect (and
            (assign (value ?n) (- (value ?n) 1))
        )
    )
)
```

Code Example 2.3: PDDL 2.1 Domain: NIB

```
(define (problem NIB_example)
    (:domain NIB)
    (:objects
        a b - agent
        p q - num
    )

    (:init
        (standing a)
        (standing b)

        (free p)
        (free q)

        (= (value p) 4)
        (= (value q) 6)

    )

    (:goal (and
            (not (free p))
            (not (free q))
        )
    )

    (:bounds
        (value - int[0..99])
    )
)
```

Code Example 2.4: PDDL 2.1 Problem: NIB

As shown in the example, PDDL 2.1 keeps the original predicate definition from PDDL, and the value of the numbers p and q are modeled as functions. In addition, PDDL 2.1

adapts expressions and conditions over numeric values, including arithmetic and comparison operators, such as $-$, $>$, etc. Any example action updates numeric value can be found between Line 44 to 52 in the domain file, where the variable is assigned its origin value plus one. In contrast to Code Example 2.2, the modeler is not required to specify the values and their relations by detailing them through propositions.

2.1.3.3 Functional STRIPS (F-STRIPS)

However, declarative languages like STRIPS and PDDL 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 [56]. Thus, an extension of STRIPS, *Functional STRIPS* (F-STRIPS) has been proposed by Geffner [57] by introducing first-class function symbols to STRIPS, which provides additional flexibility in modeling planning problems. Although there are some mature ideas about integrating functions in both STRIPS [58–62] (using constant symbols and their domains of interpretation) and PDDL [48], F-STRIPS is different from those in two ways: allowing nesting in functions; and allowing customized representation functions.

Any problem modeled by F-STRIPS can be represented as $P = (\mathcal{L}_F, \mathcal{O}_F, \mathcal{I}_F, \mathcal{G}_F)$, where $\mathcal{L}_F$ is the language, $\mathcal{O}_F$ is the set of operators, and $\mathcal{I}_F$ and $\mathcal{G}_F$ are formulae representing the initial state and goal conditions. The language $\mathcal{L}_F$ is defined by declaring the fluents and their domains, while the operators are defined by using representation functions, including the standard representation functions, such as "+" , "=", and ">", and customised representation functions defined by the modeler. The states are represented by complete assignments, which means each fluent is assigned a value. Moreover, the operators define the transitions, where the precondition decides their availability, and the effects indicate the assignments that will be updated.

The potential of customised representation functions of F-STRIPS did not get fully explored until Francès et al. [63]'s work. Their novel concept involves employing F-STRIPS model domains that do not use a declarative action representation. This approach is particularly useful for domains where the relations are too complex to be represented by declarative planning language or involve even black-box relations (a simulator, such as Atari video games [56]). They treated those relations as *external functions* with a special symbol "@".

In their formalization, any classical F-STRIPS problem can be represented by a tuple $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where V and D are variables (named as functions in the language) 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 contain the above expressive relations. The external function can be either implemented intuitively in any programming language, or obtained from the simulation.

2.1.4 Planner

Besides the problem model and representations, a solver, which is also called a planner, plays another important role in planning by applying algorithms, usually search algorithms, to generate a solution for the modeled problem.

2.1.4.1 Search Algorithms

The basic search algorithms used are the blind (uninformed) search algorithms, such as Breadth First Search (BFS), Depth First Search (DFS), Iterative Deepening (ID) and Uniform-cost Search, etc. Those algorithms have their own advantages and limitations. The performance measures for those search algorithms include completeness, optimality, time complexity, and space complexity. BFS ensures completeness, given the state space of the problem is finite, and also guarantees optimality, given the costs of all actions are uniform. An extension of BFS is Uniform-cost Search [64] (akin to the Dijkstra algorithm [65], also known as Best First Search in some work [66]), which is optimal with any positive action costs. DFS is neither complete nor optimal, but it has significantly lower space complexity ($O(b * D)$) in comparison with BFS ($O(b^d)$), where b is the branching factor, d is the optimal depth, and D is the maximum depth. ID is a combination of BFS and DFS, which also inherits the advantages from both algorithms: the same completeness and optimality from BFS; and the same space complexity from DFS.

One of the most successful computational approaches to planning is the heuristic (informed) search [67]. It employs a heuristic function to estimate the cost from the given state to achieve goals. The standard heuristic search algorithms include A* [68], Greedy Best-First Search (GBFS) and Weighted A* [69]. The heuristic function distinguishes different planners [70]. A few properties are defined to analyse the performance of the heuristic functions, which are: admissibility, safety, goal-awareness, and consistency. In addition, to achieve good performance, the heuristic functions should be as informed as possible. Using the planner LAMA as an example, the heuristic it uses is a landmark-based heuristic derived from the model [71] along with other delete-relaxation heuristics [72]. This makes LAMA be one of the state-of-the-art planners in the planning competition (won the international planning competition twice in 2008 and 2011). The limitation of the heuristic functions is that most efficient heuristics require the model to be encoded in STRIPS or PDDL (following the classical planning assumptions, such as Assumption 3 and 6) since those heuristic functions explore action's determinism and Markovian from their declarative representations. This restricts the expressiveness of the models significantly.

2.1.4.2 Variations in Planners

The standard classical planning languages and solvers do not support the use of procedures or external theories. As introduced in Section 2.1.3.3, the first theoretical research that solves this problem is from Geffner [57]'s F-STRIPS language, where the denotation of (non-fluent) function symbols can be given using external functions. In addition, Dornhege et al. [73] proposed an extension of the PDDL language (PDDL/M) that uses a similar idea called semantic attachments. They apply this idea by integrating with existing heuristic search-based planners. Their approach is widely used for robotic motion planning [74–77]. Planning Modulo Theories were introduced by Gregory et al., an idea inspired by SAT Modulo Theories [79], where specialized theories were integrated too with a heuristic search planner.

The reason why functions 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. Most planning approaches ground all functions, which allows them to convert the problem to a classical propositional planning problem that can be solved using a classical planner, but recently, a new family of algorithms called BFWS have been proposed as a new width-based planning [80]. They show their width-based approaches are as efficient as most of the state-of-the-art heuristic search approaches.

2.2 Epistemic Logic

At the beginning of this section, we give some necessary logic operators, notations, and axioms for epistemic logic. Then, we discuss the preliminaries of one part in epistemic logic, which is "knowledge". Knowledge requires the system to model and reason about the actual environment and the agents' knowledge about this environment, and of agents' knowledge of others' knowledge about this environment, and so on. In addition, we give the preliminaries for the other type of epistemic logic – belief. Similarly, belief can be nested arbitrarily. Differing from knowledge, belief is less restricted.

For handling both belief and knowledge, we follow the most fundamental and widely used approach, the one that is using *Kripke Structure*. Firstly, in this chapter, we show how the Kripke Structure is used to handle knowledge, group knowledge, as well as belief and group belief. In addition, we also give background on the difference between knowledge and belief.

At the end, we show an extended epistemic logic model, namely *Dynamic Epistemic Logic*, which is designed to formalize the changing of the state and Kripke structure.

It is worthy to mention to the reader that in this chapter, we used the terms "single knowledge" (or "individual knowledge") and "single belief" as the differentiation from the terms "group knowledge" and "group belief", while they could contain multiple agents in the full formula. For example, we consider: "agent a knows agent b knows φ" is a single-knowledge relation, as in each knowledge nesting level, there is only one agent; "agent a knows φ and agent b knows φ" is a conjunction of two single-knowledge relations; "agent a and agent b knows φ" is a group-knowledge relation that a and b as a group, and each agent in this group uniformly knows φ.

2.2.1 Preliminary for Knowledge in Epistemic Logic

The fundamental models for epistemic logic are based on classical logic. Following classical logic, the fundamental epistemic logic is also a propositional logic. Thus, given the set of all propositions $Prop = \{p_1, \dots\}$, the basic language $\mathcal{L}_K(Prop)$ used in epistemic logic can be defined as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K\varphi, \text{ where:}$$

K is a general knowledge operator that is going to be replaced by K_i or K_G later (once single-knowledge or group-knowledge is introduced). The interpretation of $K\varphi$ is that φ is known.

Other logic operators are included, such as $\equiv$, and $\rightarrow$. The material implication operator in this thesis is denoted as $\rightarrow$, which is also commonly represented by $\supset$ or $\Rightarrow$ in other works. The disjunction operator $\vee$ is omitted as classical epistemic logic follows **De Morgan's Laws** from classical logic ($A \vee B \equiv \neg(\neg A \wedge \neg B)$). Other logic operators, such as $\Box$, are not relevant in this thesis. Thus, those are omitted.

The semantics for all formulae in language $\mathcal{L}_K(Prop)$ are the same as in classical logic, except for the knowledge operator, which is going to be explained later in this section.

The fundamental axioms and rules we listed below are referred from Gochet and Gribomont [81]'s book chapter and Fagin et al. [3]'s book. The combinations of these axioms or rules provide a family of sound and complete axiomatic systems for different variations of epistemic logics.

Definition 2.2 (Epistemic Logic Benchmark Statements)**.** Here are listed 7 intuitive axioms and two inference rules as follows:

P: (Tautology Property)	**Classical tautologies are valid**
K: (Knowledge Property)	$\big(K\varphi \wedge K(\varphi \rightarrow \psi)\big) \rightarrow K\psi$
T: (Distribution Property)	$K\varphi \rightarrow \varphi$, $\varphi \rightarrow \neg K\neg\varphi$
B: (Brouwerian Property)	$\varphi \rightarrow K\neg K\neg\varphi$
4: (Positive Introspection Property)	$K\varphi \rightarrow KK\varphi$
5: (Negative Introspection Property)	$\neg K\varphi \rightarrow K\neg K\varphi$
D: (Consistency Property)	$\neg K\ false$
MP: (Modus Ponens)	$\big(\varphi \wedge (\varphi \rightarrow \psi)\big) \rightarrow \psi$
KG: (Knowledge Generalization)	$\varphi \rightarrow K\varphi$

Both Axiom P and Rule MP are intuitive. Since they hold in classical logic, they must also hold in epistemic logic as well. Axiom K is the "epistemic version" of Rule MP: if φ and $\varphi \rightarrow \psi$ are known, then ψ must also be unknown. While Axioms T, B, 4, 5, and D are valid in some formal systems but not in all.

The intuition for Axiom T is straightforward: if φ is known, then it must be true; and, if φ holds, then not φ must not be known. Axioms 4 and 5 indicate known (4) or not known (5) relations should be known. Combining Axioms T and 5, we have Axiom B. Axiom D represents that knowledge is consistent, which means a contradiction or impossible cannot be unknown.

As mentioned by Gochet and Gribomont, some of the above axioms and rules are widely accepted and intuitively valid. Thus, those must hold in any appropriate formal system, while others are more controversial. Some would be relaxed from valid to simply satisfiable in other systems.

Historically, Axiom P and both inference rules are commonly accepted by most of the logic systems, including all epistemic logic systems discussed in this subsection. With this premise, the axiomatic system is named after other significant axioms, excluding Axiom P, Rule MP, and Rule KG. Here, we listed some common axiomatic systems: **KT4** (also known as **S4**), **KT45** (also known as **S5**) and **KD45**. The details are given in the later parts of this section.

2.2.2 Kripke Structure

The standard and foundational epistemic logic model is defined following the Kripke structure. We follow the formal definition from Fagin et al. [3]'s book.

Definition 2.3 (Kripke Structure)**.** Let *Pro* be a finite set of propositions and *Agt* be a finite set of agents (k of them), a Kripke structure is a tuple:

$$M = (W, \pi, \mathcal{K}_0, \ldots, \mathcal{K}_k), \text{ where:}$$

- W is a non-empty set of all possible worlds;
- π is an interpretation function such that $\pi(w) : \mathit{Prop} \to \{\mathit{true}, \mathit{false}\}$ defines which propositions are true and false in world $w \in W$;
- $\mathcal{K}_1, \ldots, \mathcal{K}_k$ represents the *accessibility relations* over worlds for each of the k agents in *Agt*.

Given a world [4] w and a proposition p, the evaluation of p over w is $\pi(w)(p)$. p is true in w if and only if $\pi(w)(p)$ is *true*. $\mathcal{K}_i$ for agent i is a binary relation [5] over worlds. For any pair of worlds v and w, if $(w, v) \in \mathcal{K}_i$, then we say that agent i cannot distinguish between v and w when in world w. In other words, the world v and w are equivalent to agent i if and only if $(w, v) \in \mathcal{K}_i$. For example, an agent throws a coin and covers it in his/her hand. Before the coin is revealed, others cannot distinguish between the coin being head up or tail up.

In addition, they also discussed the five constraints for these accessibility relations as follows.

Definition 2.4 (Epistemic Logic Benchmark Statements)**.** For any accessibility relation $\mathcal{K}$ in a given Kripke structure $M = (W, \pi, \mathcal{K}_0, \dots, \mathcal{K}_k)$, 5 common constraints of $\mathcal{K}$ can be defined as:

Reflexive:	$\forall w \in W$, we have $(w, w) \in \mathcal{K}$
Symmetric:	$\forall w, w' \in W$, we have that $(w, w') \in \mathcal{K}$ if and only if $(w', w) \in \mathcal{K}$
Transitive:	$\forall w, w', w'' \in W$, we have if $(w, w'), (w', w'') \in \mathcal{K}$, then $(w, w'') \in \mathcal{K}$
Euclidean:	$\forall w, w', w'' \in W$, we have if $(w, w'), (w, w'') \in \mathcal{K}$, then $(w', w'') \in \mathcal{K}$
Serial:	$\forall w \in W$, we have $\exists w' \in W$ such that $(w, w') \in \mathcal{K}$

The above five constraints are not independent. As proposed by [3]: if $\mathcal{K}$ is reflexive and Euclidean, then $\mathcal{K}$ is symmetric and transitive; if $\mathcal{K}$ is symmetric and transitive, then $\mathcal{K}$ is Euclidean. In addition, if $\mathcal{K}$ is reflexive, then $\mathcal{K}$ is serial.

With the definition of Kripke structures and constraints on the accessibility relations, the semantics for knowledge and group knowledge can be formed. It is worthy to mention here, the definition of knowledge and belief by [3] follows the same semantics, while the difference lies in the properties of the equivalence relations $\mathcal{K}_i$, which will be covered in the later parts of this section.

[4]In some of the literature, worlds are used interchangeably with the word "state", but in this thesis, we use "state" with a slightly different meaning from world, which will be explained later (after Definition 3.2).

[5]In some work, this binary relation was also refereed to *Equivalence relation*. However, it can be called as equivalence relation only if it met some desired constraints (as discussed below).

2.2.3 Kripke Structure for Knowledge

The Kripke structure that can model agents' knowledge is the same as in Definition 2.3. Fagin et al. [3] discussed different combinations of accessibility relation constraints from Definition 2.4, which form 4 different Kripke structures: M^r (reflexive), M^{rt} (reflexive and transitive), M^{rst} (reflexive, symmetric and transitive) and M^{elt} (Euclidean, serial and transitive).

They chose M^{rst} as the Kripke structure to reason about the semantics of knowledge. In other words, any accessibility relation $\mathcal{K}$ in M^{rst} is reflexive, symmetric, and transitive. They also call these relations *equivalence relations*. In addition, they proved that for any formulae in the language $\mathcal{L}_K(Prop)$, **S5** (KT45 in Definition 2.2) is a sound and complete axiomatization.

2.2.3.1 Semantics for Single-Knowledge

Let signature Σ contain a 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 language $\mathcal{L}_K(\Sigma)$ (in the form of BNF) is defined as:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi, \text{ where:}$$

$p \in Prop$ 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. This definition allows arbitrary nesting on the knowledge operator, such as, K_aK_bp representing agent a knows that agent b knows p.

Then, they give the semantics for this knowledge language.

Definition 2.5 (Semantics for Single Knowledge with Kripke Structure)**.** Given a Kripke structure $M = (W, \pi, \mathcal{K}_0, \ldots, \mathcal{K}_k)$ and the current world w, the truth value of any formula in language $\mathcal{L}_K(\Sigma)$ can be defined as:

(a) $(M, w) \models p$ iff $\pi(w)(p) = true$

(b) $(M, w) \models K_i\varphi$ iff $(M, v) \models \varphi$ for all v such that $(w, v) \in \mathcal{K}_i$

The notions $(M, w) \vDash p$ from item (a) read as "(M, w) satisfies p". That is, the proposition p evaluated by π is true given the current world w and the model M. $(M, w) \vDash K_i\varphi$ is defined by formula φ being true at all worlds v reachable from w via the accessibility relation $\mathcal{K}_i$. Besides, conjunction and negation are defined by standard propositional logic rules.

This semantics allows knowledge to be nested as well. For example, K_aK_bp means p is true at all worlds reachable by applying the accessibility relation $\mathcal{K}_a$ followed by $\mathcal{K}_b$. To be specific, $(M, w) \vDash K_aK_bp$ is true if and only if: $(M, v) \vDash K_bp$ for all v such that $(w, v) \in \mathcal{K}_a$, which means $(M, v') \vDash p$ for all v' such that $(v, v') \in \mathcal{K}_b$, for all v such that $(w, v) \in \mathcal{K}_a$. This idea generalises to an arbitrary level of nested knowledge.

With the semantics for knowledge, they are able to prove that **S5** is a sound and complete axiomatization for the Kripke structure $M = (W, \pi, \mathcal{K}_0, \ldots, \mathcal{K}_k)$ discussed at the beginning of this section (Section 2.2.3).

Now, let us elaborate the Kripke structure and its knowledge semantics by using the following example (from Example 1.2). First, we construct the Kripke structure $M = (W, \pi, \mathcal{K}_0, \ldots, \mathcal{K}_k)$. The set of all possible worlds W follows the state-space in the classical planning model in Example 2.1 (Section 2.1.2.1). To show the accessibility relation for each agent, we use the following notation to represent some subsets of all possible worlds:

$$W_{i,j,x,y} = \{(i, j, x, y) \mid (i, j, x, y) \in W\}, \text{ where:}$$

any i, j, x, y can be replaced by an underscore "_", which matches any value in the possible worlds. For example, $W_{0,0,\_,\_}$ represents a set of states in which $\text{pt}_a = \text{pt}_b = 0$. The possible worlds in this example can be divided using $W_{i,j,\_,\_}$, where $(i, j) \in \{(0, 0), (0, 1), (0, 2), (1, 0), (2, 0), (1, 2), (2, 1)\}$. That is, the possible worlds are divided into 7 sets according to agents' postures (as shown in Figure 1.2). Then, the accessibilities in the Kripke structure for knowledge can be constructed as follows:

- For $W_{0,\_,\_,\_}$, a binary relation for any two possible worlds in $W_{0,\_,\_,\_}$ that contains the same value of pt_b will be added into $\mathcal{K}_a$ (That is, every combination of size two in each set $W_{0,0,\_,\_}$ and $W_{0,1,\_,\_}$ will be added to $\mathcal{K}_a$); and, respectively the same for $W_{\_,0,\_,\_}$ on $\mathcal{K}_b$;

- For $W_{1,\_,\_,\_}$, a binary relation for any two possible worlds in $W_{1,\_,\_,\_}$ that contain the same pt_b and value_p will be added into $\mathcal{K}_a$; and, respectively the same for $W_{\_,1,\_,\_}$ on $\mathcal{K}_b$;
- For $W_{2,\_,\_,\_}$, a binary relation for any two possible worlds in $W_{2,\_,\_,\_}$ that contain the same pt_b and value_q will be added into $\mathcal{K}_a$; and, respectively the same for $W_{\_,2,\_,\_}$ on $\mathcal{K}_b$;

From the above construction, the accessibility relations $\mathcal{K}_a$ and $\mathcal{K}_b$ are trivially reflexive, symmetric, and transitive. The evaluation function π is just standard, following classical logics.

Then, using the above Kripke structure, the semantics of knowledge can be discussed. In order to do so, we need to have a few example epistemic goals in language $\mathcal{L}_K(\Sigma)$.

Example 2.2 (NIB Example Knowledge Formulae)**.** *Following Example 1.2, the example epistemic formulae are listed as follows:*

1. $K_a(p{=}4)$
2. $K_aK_b(p{=}4)$
3. $K_a(p \times q{=}4)$
4. $K_a(p \times q \leq 99^2)$
5. $K_aK_b(p \times q \leq 99^2)$

For each of the above epistemic formulae, their truth value can be evaluated using the semantics from Definition 2.5 as follows:

1. $(M, w) \models K_a(p{=}4)$ holds if and only if $w \in W_{1,\_,4,\_}$. Based on the construction of the M, all the worlds that agent a considers possible ($\forall(w, v) \in \mathcal{K}_a$), contain the same value of value_p as in w. That is, the value of p is 4 in every state that agent a considered possible. Thus, $(M, w) \models K_a(p{=}4)$ holds. While, for any $w' \notin W_{1,\_,4,\_}$, then it is either a knows p is a different value ($w' \in W_{1,\_,x,\_}$, where $x \neq 4$), or a does not know the value of p ($w' \in W_{i,\_,\_,\_}$, where $i \neq 1$).

2. $(M, w) \models K_a K_b(p=4)$ will never hold in the given Kripke structure. Following the above conclusion, $(M, w) \models K_a(p=4)$ holds if and only if $w \in W_{1,\_,4,\_}$. Thus, $(M, w) \models K_a K_b(p=4) \equiv \ \forall w' \in W_{1,\_,4,\_}, (M, w') \models K_b(p=4)$. Since the number p cannot be peeked by both agent at the same time, which means $w' \notin W_{\_,1,4,\_}$. Thus, $K_b(p=4)$ does not hold in any w', which means $(M, w) \models K_a K_b(p=4)$ will never hold.

3. $(M, w) \models K_a(p \times q=4)$ will also never hold in the given Kripke structure. This is trivial if $w \in W_{0,\_,\_,\_}$. If $w \in W_{1,\_,k,\_}$, where $W_{1,\_,k,\_} \subset (W_{\_,\_,x,y},\ where\ x \times y=4)$. Following the above conclusion, $(M, w) \models K_a(p = k)$ holds if and only if $w \in W_{1,\_,k,\_}$. There exists $(w, v) \in \mathcal{K}_a$ such that the value of q in v is not equal to $4 \div k$. Thus, $(M, w) \models K_a(p \times q=4)$ does not holds for all $w \in W_{1,\_,k,\_}$. Similar, $(M, w) \models K_a(p \times q=4)$ does not holds for all $w \in W_{2,\_,\_,k}$.

4. Both $(M, w) \models K_a(p \times q \leq 99^2)$ and $(M, w) \models K_a K_b(p \times q \leq 99^2)$ holds trivial as $(M, w) \models (p \times q \leq 99^2)$ holds for all $w \in W$.

In the above example, it seems less intuitive that agent a cannot have the knowledge that $p \times q$ is 4 even if $p \times q$ actually is 4. This happens because the above Kripke structure is constructed only based on the current state without any other implicit information nor the state transitions that update the knowledge. How the knowledge evolves will be covered in the later part of this section as well as in this thesis.

2.2.3.2 Semantics for Group-Knowledge

Following the same intuition, the concept of group-knowledge can be defined. For this, with the same signature Σ above, the grammar of the language ($\mathcal{L}_{GK}(\Sigma)$) is extended to:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi \mid E_G\varphi \mid D_G\varphi \mid C_G\varphi, \text{ where:}$$

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

$E_G\varphi$ represents that everyone in the group G knows φ and $C_G\varphi$ represents that it is *commonly known* in the 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 if all agents in G pooled their knowledge together, they would know φ, even though it may be that no individual in the group knows φ.

Following the similar semantics as the single knowledge, they give the semantics for the group knowledge operators as follows.

Definition 2.6 (Semantics for Group Knowledge with Kripke Structure)**.** Given a Kripke structure $M = (W, \pi, \mathcal{K}_0, \ldots, \mathcal{K}_k)$ and the current world w, the truth value of any formula in language $\mathcal{L}_{GK}(\Sigma)$ can be defined as:

$$\begin{aligned}
(M,w) \vDash E_G\varphi \quad &\text{iff} \quad (M,w) \vDash K_i\varphi \text{ for all } i \in G \\
(M,w) \vDash C_G\varphi \quad &\text{iff} \quad (M,v) \vDash \varphi \text{ for all } v \text{ that are } G\textit{-reachable} \\
(M,w) \vDash D_G\varphi \quad &\text{iff} \quad (M,v) \vDash \varphi \text{ for all } v \text{ such that } (w,v) \in \textstyle\bigcap_{i\in G}\mathcal{K}_i
\end{aligned}$$

By definition, $(M,w) \vDash E_G\varphi$ holds if and only if φ is known by all agents (uniformly) in G. This can also be rewritten in a form that is similar to $D_G\varphi$, which is: $(M,w) \vDash E_G\varphi$ holds if and only if $(M,v) \vDash \varphi$ for all v such that $(w,v) \in \bigcup_{i\in G}\mathcal{K}_i$.

World v is G*−reachable* from w if w can reach v within k steps of accessible relations, or for some k where $k \geq 1$. Common knowledge $(M,w) \vDash C_G\varphi$ holds if and only if in all worlds v that are G*−reachable* by following the accessibility relations of all agents in G, φ is true.

For distributed knowledge, $(M,w) \vDash D_G\varphi$ holds if and only if in all worlds that all agents from G agree are possible, φ is true. It might be easier to think in the reverse direction: we say $D_G\varphi$ is true in (M,w) 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. That is, when "pulling" all agents' knowledge together, there might be some accessibility relations from $\mathcal{K}_i$ that are disapproved by another agent j. This might be less intuitive when one considers that pulling knowledge together would gain more knowledge for the group. This is because the more binary relations in the evaluation, the more uncertainty of the knowledge. If any agent i knows everything about the given world w, then, the w in $\mathcal{K}_i$ would only connect to itself ($(w,w) \in \mathcal{K}_i$).

Now, let us elaborate on the group knowledge semantics with the same Kripke structure constructed in Section 2.2.3.1. Some group knowledge formulae are given in the following example.

Example 2.3 (NIB Example Group Knowledge Formulae)**.** *Following Example 1.2, the example group knowledge formulae are listed as follows, where group G contains both agent a and b:*

1. $E_G(p{=}4)$
2. $D_G(p \times q{=}4)$
3. $C_G(pt_a{=}1)$

For each of the above epistemic formulae, their truth value can be evaluated using the semantics from Definition 2.6 as follows:

- $(M, w) \vDash E_G(p{=}4)$ will never hold in the given Kripke structure. $(M, w) \vDash K_a(p{=}4)$ will not hold for any $w \in W_{i,\_,\_,\_}$ where i is not 1, and $(M, w) \vDash K_b(p{=}4)$ will not hold for any $w \in W_{\_,j,\_,\_}$ where j is not 1. The union of the above two sets forms all possible worlds W. Thus, $(M, w) \vDash E_G(p = 4)$ will never hold for any $w \in W$.
- $(M, w) \vDash D_G(p = 4)$ holds if and only if $w \in W_{1,2,x,y} \cup W_{2,1,x,y}$, where $x \times y = 4$. Using $w' \in W_{1,2,1,4}$ as example, based on current world w', the accessibility relations in $\mathcal{K}_a$ is the set $\mathcal{K}_a(w') = \{(w', v) \mid v \in W_{1,2,1,\_}\}$, while the accessibility relations in $\mathcal{K}_b$ is the set $\mathcal{K}_b(w') = \{(w', v) \mid v \in W_{1,2,\_,4}\}$. Based on group semantics, $(M, w) \vDash D_G(p{=}4)$ is equivalent to $(M, v) \vDash (p{=}4)$ for all v such that $(w, v) \in \bigcap_{i \in G} \mathcal{K}_i$. The intersection $\mathcal{K}_a(w') \cap \mathcal{K}_b(w')$ is $\{(w', w') \mid w' = (1, 2, 1, 4)\}$. Thus, we have $(M, v) \vDash (p = 4)$. Following the same process for all the other combinations of x and y, we have $(M, w) \vDash D_G(p = 4)$ holds if and only if $w \in W_{1,2,x,y} \cup W_{2,1,x,y}$, where $x \times y{=}4$.
- $(M, w) \vDash C_G(\text{pt}_a{=}1)$ if and only if $w \in W_{1,\_,\_,\_}$. We consider this by all posture of agent b. If b is not peeking, given an example state $w' = (1, 0, x, y) \in W_{1,0,\_,\_}$, the reachable possible worlds by b's accessibility relations is $\mathcal{K}_b(w') = W_{1,0,\_,\_}$. The reachable possible worlds for a based on the possibles worlds from b ($\mathcal{K}_b(w')$) is also $W_{1,0,\_,\_}$. The above process has converged on the set $W_{1,0,\_,\_}$. It ends in the same converged set if agent a's accessibility relations is evaluated first. Thus, the set of all $G-$*reachable* possible worlds is $W_{1,0,\_,\_}$, and pt_a is 1 in all of them, which means $(M, w) \vDash C_G(\text{pt}_a{=}1)$ holds.

2.2.4 Preliminary for Belief in Epistemic Logic

The relation between knowledge and belief has been discussed by both philosophers and logicians. The first intuition is from *Meno* by Plato [36]. As the epigraph aptly quotes, Plato claimed knowledge is true opinion (belief). Later on, in *Theaetetus* [82], he disproved the above idea by extending it to that knowledge is true opinion combined with a definition or a rational explanation. That is, one knows that a proposition p holds if and only if: 1), p is true; 2), one believes p holds; and, 3), one is justified in believing p. His idea is further explained by defining the necessary and sufficient conditions for knowledge by Ayer [83] and Chisholm [84].

However, in 1963, Gettier [85] proposed a counter-example to their definition of the knowledge, which shows that even with necessary and sufficient conditions, the justified true belief is still not enough to generate knowledge. This is caused by a person can justifiably believe a proposition for the wrong reason. In order to avoid this issue, one could add one condition to the above Plato's definition: the reason for the third condition (one is justified in believing p) holds should be the same as the reason the first condition holds. However, this results in those two conditions (first and third) no longer being independent, which indicates the connections between knowledge and belief are difficult to analyse. Thus, we define the belief operator here first.

Using a similar format from the grammar of the knowledge operator in Language $\mathcal{L}_K(Prop)$, the grammar for the language ($\mathcal{L}_B(Prop)$) that contains the belief operator for a set of propositions $Prop$ can be defined as:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \land \varphi \mid B\varphi, \text{ where:}$$

$p \in Prop$. This language is simply replacing the knowledge operator K with the belief operator B.

In general, the key distinction between knowledge and belief lies in that knowledge must reflect the actual world, whereas belief does not have this requirement. As shown in the following definition (Definition 2.2a), the axiomatic system defined for knowledge (Definition 2.2) can be used for belief by replacing K with B, and removing Distribution Property Axiom **T** as well as Brouwerian Property Axiom **B**.

Definition 2.2a (Epistemic Logic Benchmark Statements for Belief)

Here are listed 5 intuitive axioms as follows:

P: (Tautology Property)	**Classical tautologies are valid**
K: (Knowledge Property)	$(B\varphi \wedge B(\varphi \rightarrow \psi)) \rightarrow B\psi$
4: (Positive Introspection Property)	$B\varphi \rightarrow BB\varphi$
5: (Negative Introspection Property)	$\neg B\varphi \rightarrow B\neg B\varphi$
D: (Consistency Property)	$\neg B\ false$

The removal of Axiom **T** stems from the fact that agents may hold beliefs that are not necessarily true, while Axiom **B** is omitted because agents may also believe in their own beliefs, regardless of their veracity. Axiom **D** serves as the axiom ensuring consistency, which prevents agents from believing in impossibilities, such as contradictions. In certain studies [86–88], an alternative version of the Consistency Property Axiom **D** is used: $B\varphi \rightarrow \neg B\neg\varphi$, emphasizing that an individual's beliefs should remain consistent rather than merely possible.

2.2.5 Kripke Structure for Belief

With the axiomatic system (in Definition 2.2a) for language $\mathcal{L}_B(Prop)$, we can now give the classical definition for belief by using Kripke structures.

The signature of the model is the same as in Section 2.2.3, while the grammar of the language is also the same, except the language $\mathcal{L}_K(\Sigma)$ and $\mathcal{L}_{GK}(\Sigma)$ were replaced by language $\mathcal{L}_B(\Sigma)$ – which is $\mathcal{L}_K(\Sigma)$ by replacing the knowledge operator K with belief operator B– and $\mathcal{L}_{GB}(\Sigma)$, where E, D and C become EB, DB and CB respectively. The representation of the Kripke structure stays the same (Definition 2.3). The difference lies in the requirements for the accessibility relation.

As mentioned in Fagin et al. [3]'s book, the Kripke structure that models belief is M^{elt}, which means the accessibility relations in M^{elt} must be *Euclidean*, *serial*, and *transitive* (from Definition 2.4), rather than equivalent relations (reflexive, symmetric, and transitive) for knowledge. Then, the semantics for single belief and group belief are the same as in Definition 2.5 and Definition 2.6 respectively. They also provide a proof that **KD45** is a sound and complete axiomatization for M^{elt}.

2.2.6 Knowledge Versus Belief

Gochet and Gribomont [81] points out that analysing concepts of knowledge and belief in isolation is not very promising. The relation between knowledge and belief is complicated, as briefly mentioned when we introduced belief earlier. There are many research [89–91] analyses on the connection and difference between knowledge and belief. Here, we only show the relevant works to this thesis.

Although the Kripke structure can be used to model knowledge and belief, it is trickier when both knowledge and belief are evaluated in the same model. Thus, Kraus and Lehmann [92] use two types of accessibility relation $\equiv_i$ and $\approx_i$ to represent indistinguishability for agent i on knowledge and belief respectively. Similarly, as introduced earlier, they required $\equiv_i$ to be an equivalence relation (reflexive, symmetric, and transitive), while $\approx_i$ should be Euclidean and serial, but it is not necessarily symmetric and reflexive. Then, following the intuition that "It is easier to believe something than to know it.", they proposed that for any two possible worlds w and w', $w \approx_i w' \to w \equiv_i w'$. That is, $K_i\varphi \to B_i\varphi$ (Axiom **KB1**), which is proved by Gochet and Gribomont [81]. Then, Gochet and Gribomont also proposed another intuitive "bridge axiom" named Axiom **KB2**, which is $B_i\varphi \to K_iB_i\varphi$. To sum up, they proposed an axiomatic system as evaluation for systems that handle both knowledge and belief as follows.

Definition 2.7 (Axioms for KB)**.** Presented are 10 axioms:

K: (Knowledge)	$\big(K_i\varphi \land K_i(\varphi \to \psi)\big) \to K_i\psi$
T: (Knowledge)	$K_i\varphi \to \varphi$, $\varphi \to \neg K_i\neg\varphi$
4: (Knowledge)	$K_i\varphi \to K_iK_i\varphi$
5: (Knowledge)	$\neg K_i\varphi \to K_i\neg K_i\varphi$
K: (Belief)	$\big(B_i\varphi \land B_i(\varphi \to \psi)\big) \to B_i\psi$
D: (Belief)	$\neg B_i\ false$
4: (Belief)	$B_i\varphi \to B_iB_i\varphi$
5: (Belief)	$\neg B_i\varphi \to B_i\neg B_i\varphi$
KB1:	$K_i\varphi \to B_i\varphi$
KB2:	$B_i\varphi \to K_iB_i\varphi$

In addition, Voorbraak [93] raised a theorem as shown in Theorem 2.8, which claimed agents believe they know φ do know it.

Theorem 2.8 (The unwanted axiom)**.** $B_i K_i \varphi \rightarrow K_i$

The above axiom might be intuitive in the agents' local view, but globally it is counter-intuitive. One should have the ability to believe to know a false proposition.

However, as proven by Gochet and Gribomont [81], the above theorem holds for the axiomatic system provided in Definition 2.7. Since the proof is based on three axioms: **KB1**, **D** and **5**, they claimed that one of these three axioms must be removed to avoid having Theorem 2.8 hold in a logic system. For instance, some researchers [93, 94] chose to drop Axiom **KB1** forming "Objective Knowledge", while others [95, 96] removed Axiom **D** which causes an agent's beliefs to be inconsistent ($B_i\varphi \wedge B_i\psi \not\rightarrow B_i(\varphi \wedge \psi)$). In addition, Axiom **5** was questioned by Lenzen [97]. As an agent's knowledge is always consistent with the true world, they claim that one cannot – by mere introspection – ascertain whether they know something. Furthermore, Williamson [98] proved that with an additional Axiom **KB3** ($B_i\varphi \rightarrow B_i K_i \varphi$), the axiomatic system in Definition 2.7 results in another unwanted axiom, Axiom $\mathbf{\Omega}$, where $B_i\varphi \rightarrow \varphi$. This issue was fixed by Halpern [99], in which they limited Axiom **KB1** to be objective (non-modal formulae). They proved with this restriction, the axiomatic system mentioned above is sound and complete.

Overall, it is common for a logic system to follow **S5** for knowledge (as described in Section 2.2.3) and **KD45** for belief, as well as using Axiom **KB1** and Axiom **KB2** as the bridge between knowledge and belief.

2.2.7 Knowledge & Belief Updates

While Kripke structures can effectively model knowledge and belief, they follow a "fixed" methodology. This means that the assessment of epistemic relations relies on the specified (input) world (state), like a "snapshot" of the dynamic reality. This can be elaborated further with the well-known Muddy Children [3, 100] example.

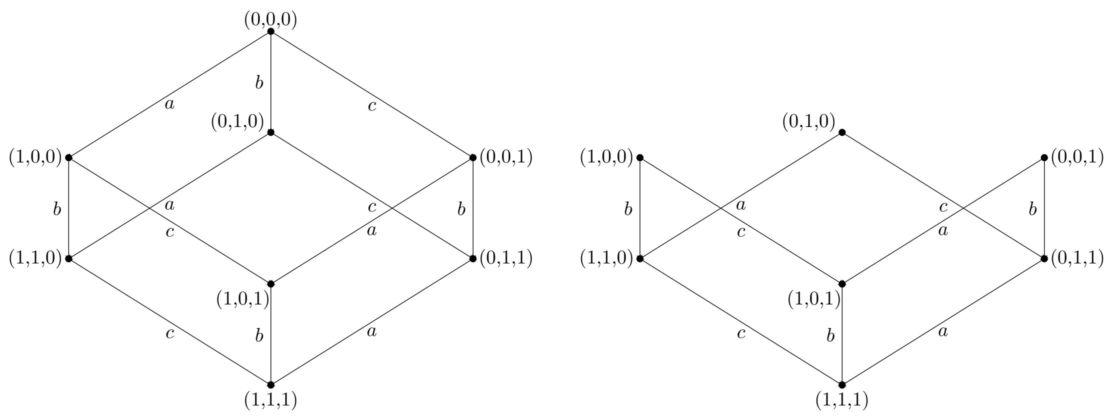
Example 2.4. *There are n children, and m of them with mud on their forehead. They can all see each other's foreheads, but not their own. They will announce immediately when they know whether they are muddy or not. To help them find out whether they themselves are muddy or not to get themselves cleaned, their teacher can help. Firstly, the teacher told them: "At least one of you has mud on your forehead." ($m > 0$). Then,* *the teacher will repeatedly ask one question to them all: do you know whether you are muddy or not? The process stops when everyone knows whether they have mud on their forehead.*

This has been proven that the teacher has to ask for $m-1$ times for the muddy children to announce they know that they are muddy, thereby allowing the clean children to recognize that they are clean. For example, when m is 1, it is trivial that the only muddy child will notice no one else has mud on their forehead after the teacher's statement, which means himself/herself is muddy since there is at least one child who is muddy ($m > 0$). So that, the muddy child will announce immediately, while the clean children stay quiet. Right after this announcement, others will know their forehead is clean by induction.

When there are two children who are muddy ($m = 2$), both muddy children see there is one child who is muddy and they cannot see their own forehead. In the meantime, the clean children see two muddy children and are also unsure about their own cleanness. The teacher is able to help them by asking the question once. Then, both muddy children will know immediately they are muddy, because if they are not, the other muddy child would make the announcement before the teacher asking the question. Since the other muddy child did not announce, which means that child sees another muddy child, and everyone else is clean, the conclusion is straightforward that both muddy children will know they are muddy. Once they made the announcement, the remaining clean children will know the identity of the two muddy children (if there were 3, then both of them would remain silent), which indicates they are clean.

The above induction is captured by Kripke structures in Fagin et al. [3]'s book. They used an example with 3 children ($n = 3$), namely a, b, and c, and used a tuple (p_a, p_b, p_c) to represent the state, where p_a, p_b, p_c are propositional variables with a value of 0 or 1, representing whether each child is muddy or not. That is, a tuple $(0, 0, 0)$ means all three children are clean, while a tuple $(0, 1, 1)$ represents that child a is clean and both b and c are muddy.

The visualization can be found in Figure 2.3. Each node represents a state, and each edge represents an accessibility relation $\mathcal{K}_i$ with agent index i as it label.



a: Scenario 1 **b:** Scenario 2

FIGURE 2.3: Examples for Muddy Children using Kripke Structure [3], where the self-loops are omitted.

Initially (as shown in Figure 2.3a), children are able to observe others except themselves, signifying their inability to distinguish between possible worlds, where the only distinction is the variable representing their own cleanliness. Then, once the teacher made the statement ($m > 0$), all the children would consider the world $(0, 0, 0)$ as not possible. Therefore, the world and its related accessibility relations have been removed and formed a new Kripke structure as shown in Figure 2.3b. Although the Kripke structure can capture the epistemic relations in the above example, it evaluates epistemic formulae with the input of a static world. In other words, it cannot handle the changes in between. Despite numerous studies [101–104] in Epistemic Logic that address the updates of knowledge and belief, this thesis highlights only the most pertinent one, namely Dynamic Epistemic Logic.

2.2.7.1 Public Announcement in Dynamic Epistemic Logic (DEL)

As discussed in Ditmarsch et al. [105], Dynamic Epistemic Logic (DEL) refers to a collection of extensions of epistemic logic that incorporate dynamic operators for reasoning about information dynamics. Since utilizing DEL is not the central focus of this thesis, we will limit our discussion to the most renowned and frequently cited version of DEL, as presented by Ditmarsch et al. [105], which is a version that mainly covers the changes of knowledge. As DEL is widely used in Epistemic Planning, which is the most relevant concept that is going to be introduced in the forthcoming section (Section 2.3), here, we only show the intuitive idea about updating the epistemic model with a formula φ rather than delving into the planning-centric notion of epistemic actions, which is also the reason we chose to present Ditmarsch et al.'s work. In addition, we make some adjustments to the notion to make it consistent with other parts of this section.

DEL keeps the Kripke structure as introduced in Section 2.2.2 and follows the same semantics (including language $\mathcal{L}_K(Prop)$) in Section 2.2.3.1. Moreover, it incorporates the language and semantics for the common knowledge operator "C" described in Section 2.2.3.2. Given the signature of their model $\Sigma = (Prop, Agt)$, the formalization of the language $\mathcal{L}_{KC[]}(\Sigma)$ is presented by the BNF as follows:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi \mid C_G\varphi \mid [\varphi]\varphi, \text{ where:}$$

$p \in Prop$, $i \in Agt$ and $G \subseteq Agt$. To capture the changes between two epistemic states (instances of Kripke structure), they proposed formula $[\varphi]\psi$ to represent after the state updated with φ, formula ψ holds, as shown in Figure 2.3. Then, they added the semantics for the newly proposed update operator $[]$ as:

$$(M, w) \vDash [\varphi]\psi \text{ iff } (M, w) \vDash \varphi \text{ implies } M \mid \varphi, w \vDash \psi, \text{ where:}$$

$M \mid \varphi$ represents the model $M = (W, \pi, \mathcal{K}_0, \ldots, \mathcal{K}_k)$ is updated by φ into $M' = (W', \pi, \mathcal{K}'_0, \ldots, \mathcal{K}'_k)$, where:

$$W' = \{w \mid (M, w) \vDash \varphi\} \tag{2.1}$$

$$\mathcal{K}'_i = \mathcal{K}_i \cap (W' \times W) \tag{2.2}$$

With their definition, the changes in between Kripke structure M_1 (Figure 2.3a) and M_2 (Figure 2.3b) from Example 2.4 can be formally represented, which is $M_1 \mid m > 0 = M_2$. That is, the initial Kripke structure, including the accessibility relation for every child, has been updated by their teacher's first statement $m > 0$. Since $\big(M, (0,0,0)\big) \nvDash m > 0$, the world space in the updated Kripke structure does not contain world $(0,0,0)$. In addition, all the accessibility relations that contain $(0,0,0)$ are removed in the updated Kripke structure as well.

Ditmarsch et al. [105] named this φ as the effect of a public announcement action, which is aligned with the teacher's statement and questions in the muddy children example.

In addition, they also provided other types of epistemic actions, which are going to be mentioned in Section 2.3.1.

2.3 Reasoning with Epistemic Logic in Planning

Epistemic Planning is a combination of Automated Planning, Epistemic Logic, and Knowledge Representation and Reasoning. Its distinction from classical planning primarily involves the representation of the agents' epistemic logic and integrated epistemic logic reasoning into the planning. As mentioned by Bolander [106], Belle et al. [107], existing research in epistemic planning is divided into *syntactic* and *semantic* approaches.

2.3.1 Semantic Approaches

Semantic approaches, which are also termed as *model-theoretical* approaches, necessitate the usage of a theoretical epistemic logic model as their foundation, generally the Kripke model (introduced in Section 2.2.2) and evaluate agents' epistemic formulae according to this model. Although the Kripke structure can model agents' knowledge or belief in a static world, as noted in Section 2.2.7, it is complemented by event-based models [3] like DEL (detailed in Section 2.2.7.1) for dynamic updates. The DEL is designed to handle the changes between Kripke structures, which are often caused by the actions done by the agents, which is aligned with automated planning. Thus, the concept of epistemic planning is formalised in DEL by Bolander and Andersen [34], Bolander [106].

2.3.1.1 The DEL Approach

In their work, Bolander and Andersen [34] are the first to give a definition of the Epistemic Planning Problem. Their definition is extended from Ghallab et al. [40]'s definition of the classical planning problem, which is very similar to Geffner and Bonet [37]'s definition (Definition 2.1). Bolander and Andersen and Ghallab et al. differ from Geffner and Bonet's definition by separating the planning task into a planning domain and a problem instance.

In their work, a signature Σ is defined as $\Sigma = (Prop, Agt)$, where $Prop$ is a collection of propositions and Agt a collection of agents. The language $\mathcal{L}_{KC}(\Sigma)$ in their model is an expansion of $\mathcal{L}_{KC[]}(\Sigma)$ achieved by omitting the update operator "[]". That is, a standard modal logic language with knowledge operator K and common knowledge operator C. Then, they used the same Kripke structure (in Definition 2.3), denoting it as M, and semantics (in Definition 2.5 and C in Definition 2.6). Then, they gave the definition of the epistemic planning domain and epistemic planning problem instances as follows.

Definition 2.9 (Epistemic Planning Domain)**.** Given a signature $\Sigma = (Prop, Agt)$ and its language $\mathcal{L}_{KC}(\Sigma)$, an epistemic planning domain is a restricted state-transition system $\mathcal{D} = (S, A, \gamma)$, where:

- S is a finite state space of $\mathcal{L}_{KC}(\Sigma)$
- A is a finite set of actions of $\mathcal{L}_{KC}(\Sigma)$
- γ is defined by: $\gamma(s,a) = \begin{cases} s \otimes a & \text{if } a \text{ is applicable in } s \\ \text{undefined} & \text{otherwise} \end{cases}$

They claim if all states and actions are from $\mathcal{L}_K(\Sigma)$ (without common knowledge operator), then the domain is called an epistemic planning domain without common knowledge. If $|Agt| = 1$, then it is called a single-agent [6] epistemic planning domain.

Definition 2.10 (Epistemic Planning Problem)**.** Given an epistemic planning domain $\mathcal{D} = (S, A, \gamma)$ (including its signature $\Sigma = (Prop, Agt)$ and language $\mathcal{L}_{KC}(\Sigma)$) and the Kripke structure that models this domain as M, an epistemic planning problem instance in this domain can be defined as a tuple $P = (\mathcal{D}, s_0, \Phi_g)$, where:

- s_0 is the initial state, a member of S
- Φ_g is a set of formula in $\mathcal{L}_{KC}(\Sigma)$, called goal formulae. The set of goal states S_G is $S_G = \{s \mid s \in S \wedge \forall M, s \vDash \phi\}$

Different from Ditmarsch et al. [105]'s work, Bolander and Andersen differentiated the global (epistemic) state and local (epistemic) state, and performed event updates on all of them. They used the notion (M, W_d) to represent an epistemic state in $M = (W, \pi, \mathcal{K})$,

[6]Note: their usage of the term is different from ours as discussed and in Assumption 8 from Section 2.1.1. In addition, it is also different from our interpretation of "single-knowledge" or "single-belief" at the beginning of Section 2.2.

where $W_d \subseteq W$ and $\mathcal{K}$ are functions that map Agt to its accessibility relations set. A singleton W_d represents the global epistemic state, while given any global epistemic state $(M, \{w\})$, an agent i's local epistemic state would be $(M, \{w' \mid (w, w') \in \mathcal{K}(i)\})$.

Then, they took the concept of the event model and the event executions [104] to formalize their own *event model*, which is also known as *update model* or *action model*, as follows.

Definition 2.11 (Event Model)**.** An event model for language $\mathcal{L}_{KC}(\Sigma)$ is a tuple $\mathcal{E} = (E, \mathcal{K}_E, pre, post)$, where:

- E is a finite (non-empty) set of events
- $\mathcal{K}_E : Agt \to \mathcal{P}(E \times E)$ is a function that assigns accessibility relation to each agent.
- $pre : E \to \mathcal{L}_{KC}(\Sigma)$ is a function that assigns a precondition to each event.
- $post : E \to \mathcal{L}_{KC}(\Sigma)$: assign each event a postcondition.

This event model is very similar to the actions in classical planning, except the indistinguishable relation function $\mathcal{K}_E$. For any agent $i \in Agt$, $\mathcal{K}_E(i)$ returns a set of binary relations. Each of these relations $(e_1, e_2) \in \mathcal{K}_E(i)$ represents that the agent i cannot tell the difference between action e_1 and action e_2 that has taken place. Similarly, they differentiated local and global epistemic actions as a pair of $(\mathcal{E}, E_d)$, where $E_d \subseteq E$. The global epistemic action is a singleton E_d, while given any global epistemic action $(\mathcal{E}, \{e\})$, an agent i's local epistemic action would be $(\mathcal{E}, \{e' \mid (e, e') \in \mathcal{K}_E(i)\})$.

The core idea of the DEL framework is the *product update*, which keeps both possible worlds and possible actions.

Definition 2.12 (Product Update)**.** Given an epistemic state (M, W_d) and an epistemic action $(\mathcal{E}, E_d)$, where $M = (W, \pi, \mathcal{K})$ and $\mathcal{E} = (E, \mathcal{K}_E, pre, post)$, the product update is defined as the epistemic state $(M, W_d) \otimes (\mathcal{E}, E_d) = ((W', \pi, \mathcal{K}'), W'_d)$, where:

- $W' = \{(w, e) \in W \times E \mid M, w \vDash pre(e)\}$
- $\mathcal{K}'(i) = \{\big((w, e), (w', e')\big) \in W' \times W' \mid (w, w') \in \mathcal{K}(i), (e, e') \in \mathcal{K}_E(i)\}$

- $W'_d = \{(w,e) \in W' \mid w \in W_d, e \in E_d\}$

When evaluating the semantics on (w,e), they apply the action e on the world w to generate a new world, and apply the evaluation function π in the newly generated world. In addition, they allow nesting on the events. For example, from a world w applying action a_1 and a_2 would be $((w,a_1),a_2)$. They also provide a few common properties of the actions, including epistemic or ontic, public or private. For example, the teacher's statement in the muddy children example (Example 2.4) is a public announcement.

At the end, as one of their contributions, they showed that based on their formalization, single-agent ($|Agt| = 1$) epistemic planning is decidable, whereas multi-agent epistemic planning is undecidable (even without common knowledge) due to the exponential growth on the $\mathcal{K}_E$ related to the depth of epistemic formulae, the number of agents, and the number of events. This conclusion shows how complex the problem of multi-agent epistemic planning is, as well as the limitations of the DEL-based approaches.

2.3.1.2 Planning Aspects on the DEL-Based Approaches

The DEL-based formalism has been used to explore the theoretical properties of epistemic planning in many research works [1, 108–115]. From the planning perspectives, the DEL-based approach can be implemented by either developing a customised planning language and planner for the DEL-based formalism and solving the problems directly or encoding the problems into classical planning problems and solving them by a normal classical planning planner. In addition, as complexity results shown by Bolander and Andersen [34], efficiency becomes one of the main issues that are targeted by a lot of work to improve the practicability of the DEL-based formalism.

The former implementation is straightforward. Baral et al. [116] (originally Baral et al. [117]) defined an action language m$\mathcal{A}^*$ that represents and reasons about the effects of actions for epistemic planning. Similar to others, they divide actions in epistemic planning into different types based on the effects of the action, including *world-altering actions* (also known as ontic actions in some works [118]), sensing actions, and announcement actions. The semantics of m$\mathcal{A}^*$ follows the one in Kripke structure, and they use pointed Kripke structures to represent the states of the actual worlds and the states of agents' knowledge and beliefs. Furthermore, other languages are proposed based on m$\mathcal{A}^*$, including $m\mathcal{A}^p$ by Fabiano et al. [119] and E-PDDL by Fabiano et al. [120].

Le et al. utilize m$\mathcal{A}^*$ to develop two epistemic forward planners, **EFP** and **PG-EFP**. They defined their planning problem as a tuple $(Agt, \mathcal{F}, A, O)$, where Agt is the set of agent identifiers and $\mathcal{F}$ is the set of fluents. The actions and effects in their model are determined by the set of all actions A and the observability statement set O. They specify preconditions and three possible effects for the actions, which are ontic, sensing, and announcement. Ontic actions change the state (actual world), while sensing actions reveal the truth value of some fluent f. Announcement actions announce the truth value of some fluent f, which affects the set O. In the set O, they propose two kinds of observations: fully observable actions by *observes*; and partially observable 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 their model on two planners, EFP and PG-EFP, with breadth-first search and heuristic search respectively. They propose the definition of an *epistemic planning graph*, and use it as their main data structure in the search. As for PG-EFP, they derive heuristic values directly from the structure of the epistemic planning graph. They compared their planners against Muise et al. [87]'s and Huang et al. [7]'s solutions on *Corridor*, *Collaboration-and-communication* [115], and *Assembly Line* [7]. From their comparison, we find EFP does not suffer from the exponential blow-up on the 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.

The latter approach of epistemic planning using DEL-formalism has first been explored by Kominis and Geffner [115]. Similarly, to improve the planning efficiency, their model only captures a fragment of the DEL by using the intuition of the belief state from the partially observable planning. Their approach also maintains the problem's Kripke structure. By their definition (adapting STRIPS), an epistemic planning problem P is a tuple $\langle Agt, F, I, O, N, U, G\rangle$. In their model, Agt 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 the 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. They define three kinds of action sets, O, N and U to maintain and update the Kripke structure during the searching process. Set O represents all physical (ontic) actions, which is the same O as in classical planning. 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, which they adapt from Levesque [121]. 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.

They convert epistemic planning problems to classical planning problems using standard compilation techniques for partially-observable planning. As far as we can tell from their experiments in this work [115], 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. Furthermore, they extended their work to handle nested belief in a multi-agent setting [118]. In addition, in this extension, they perform planning from the perspective of each agent following planning methods in modeling and solving Partial-Observable Non-Deterministic (POND) problems.

2.3.2 Syntactic Approaches

Another direction of modeling and solving the epistemic planning problems is the syntactic approach. The syntactic approach represents and reasons about agents' knowledge and beliefs using sets of true epistemic formulae, namely "knowledge-bases". It starts with a knowledge-base and updates it according to the action sequence. Some earliest works in knowledge-base planning are proposed by Petrick and Bacchus [122]. Since there is no commonly used theoretical model for the syntactic approaches, we only show a few examples in this thesis. Similarly to the semantic approach, research works that implemented the syntactic approach have two directions: either converting the epistemic planning problems into classical planning problems [1, 87, 88, 123–127]; or developing their own language and planner to model and solve epistemic planning problems [7, 122, 128–130].

Many works with the syntactic approach (knowledge base) focus on converting the epistemic planning problems to the classical planning problems. So that, they can incorporate the state-of-the-art classical planner to achieve efficiency. Muise et al. [88] (originally Muise et al. [87]) proposed an approach to model and solve multi-agent epistemic planning problems by generating all effects (belief updates) of the epistemic action in the pre-compilation phase. This is done by grounding epistemic fluents into propositional fluents and using additional conditional effects of actions to enforce desirable properties of beliefs. They define an instance of Multi-agent Epistemic Planning (MEP) problems as a tuple $P = (F, A, Agt, I, G)$, where, similar to STRIPS (Section 2.1.3.1), F is the set of propositions (facts), A is the set of actions (operators), I is the initial state, G is the set of goal conditions, and Agt is the set of agents. The epistemic formulae (literals) their model handles following this grammar: "$\phi ::= p \mid B_i\phi \mid \neg\phi$". The literal "$B_i\phi$" reads as "agent i believes ϕ". Their model is restricted to epistemic formulae with a predefined finite depth of nested beliefs and excludes disjunctions.

They take three processes to convert their model to STRIPS and ensure those processes keep their solution sound and complete. In the first step, they remove negations and add logical consequences of all positive effects when applying an action to maintain deductive closure. Then, the beliefs about the negation of an unobservable effect (including other beliefs that can deduce this unobservable effect) are removed to handle uncertainty. At last, they handle different level belief updates by using conditional effects to cover the mutual awareness. They evaluate their approach on benchmarks, including Corridor [115] domain, Grapevine domain (a combination of *Corridor* and *Gossip* [131]), and Selective Communication [132]. Their results show that their approach is able to model and solve epistemic planning problems within a typically short time, but the compilation time to generate fluents and conditional effects (converting the epistemic planning problem into a classical planning problem) is exponential in both the number of agents and the maximum depth of epistemic relations.

Rather than using classical planners, Huang et al. [7] built a native multi-agent epistemic planner and proposed a general representation framework for multi-agent epistemic problems. They consider the multi-agent epistemic planning problems from a third-person point of view. They implement a planner, **MEPK**, to handle their representation following well-established belief revision and update algorithms. Although this approach is different from Kominis and Geffner [115] and Muise et al. [87] as they have their own encoding and an epistemic planner, it still requires a compilation phase before planning to re-write epistemic formulae into a specific normal form called *Alternating Cover Disjunctive Formulae* (ACDF) [133]. The ACDF formula is worst-case exponentially longer than the original formula. Their result indicates that their approach suffers from a similar computational burden as either Kominis and Geffner or Muise et al..

2.3.3 Discussion of the Existing Epistemic Planning Works

Epistemic planning plays a pivotal role as a bridge between theoretical advancements in epistemic logic and their practical applications in real-world scenarios. As briefly mentioned in Section 2.3.1 and Section 2.3.2, a significant limitation of existing approaches in epistemic planning is their lack of practicability, which continues to limit progress in the field.

Firstly, this limitation is evident even within the context of modeled complex instances in epistemic planning domains rather than solving them, where the term "complex" refers to scenarios involving a large number of agents and highly nested epistemic relations. While existing approaches are theoretically capable of handling such instances, their practical applicability is limited.

In particular, constructing and maintaining the Kripke structure for a complex problem instance with any semantic approach is often infeasible for a modeler. This challenge is amplified when changes involve deeply nested epistemic relations. Similarly, correctly generating all epistemic action effects during the pre-compilation phase is also challenging when involving deeply nested epistemic relations for the syntactic approach. Thus, **scalability** is an issue when modeling those complex instances. Besides, even if those problem instances are properly modelled, the feasibility of solving them is also concerning. The semantic approach must preserve Kripke structures, including accessibility relations, and update them with event-based models. Intuitively speaking, as agents observe more (epistemic) actions, the possible worlds they consider should narrow until all target epistemic formulae are validated by the semantics used. Conversely, the syntactic method uses a knowledge base to represent agents' knowledge and belief. Intuitively speaking, as more (epistemic) actions are performed, this knowledge-base should expand to a big set until all goal epistemic formulae are in it. Both approaches require a costly data structure at some stage.

Despite the scalability, the **efficiency** [7] is another limitation when solving them. It is inefficient to reason about knowledge over the possible worlds, as it is exponential to the maximum depth of epistemic relations. Specifically, it is the product of the number of possible worlds for each agent that appears in the given epistemic relations. In the syntactic approach, although there is no time taken for evaluating epistemic formulae when planning, updating the whole knowledge-base is still exponential, as the number of possible action effects is the number of agents to the power of the maximum depth of epistemic relations.

In addition, **generalizability** – the ease with which these approaches can be adapted to model new epistemic planning domains – remains a significant concern. Whether using the semantic approach or the syntactic approach, current approaches require modelers to possess a deep understanding of both epistemic logic and the targeting domain. More specifically, modelers are required to specify how each action affects an agent's individual and nested knowledge or beliefs, often to a highly detailed level. This manual effort creates a substantial barrier to the adoption of epistemic planning.

Last but not least, **expressiveness** remains a notable limitation in current epistemic planning approaches, a challenge inherited from classical planning. Declarative planning languages are not inherently designed to handle complex functions or dynamics, such as those governed by physical laws. For instance, consider modeling the concept of a line of sight. While this is intuitive for humans and straightforward to implement in imperative programming languages such as Python or C++, representing it in a planning language is often infeasible and impractical. This lack of expressiveness restricts the ability of epistemic planning to model and solve problems that require such nuanced representations, further limiting its applicability to real-world scenarios.

Overall, the current epistemic planning approaches have a significant gap between theoretical work and real-world applications because of the limitations: scalability, efficiency, generalizability, and expressiveness. The first attempt to address the above limitations on modeling and solving epistemic is done by Hu [4]. They proposed a state-based approach that is able to reason about epistemic logics with efficiency. In addition, they decomposed epistemic planning by using an external function (implemented apart from planning) to handle epistemic logic reasoning. By their approach, they are able to use

[7]The efficiency we discussed here is only about the efficiency in reasoning about epistemic logic. The efficiency of the search algorithms when planning is not relevant to content of this thesis.

any programming language to model epistemic logic rather than using a declarative planning language. The details can be found in the following section.

2.4 State-based Epistemic Planning

In this section, we introduced the first attempt on a state-based approach that models and solves epistemic planning problems which is done by Hu [4]. In their work, they show intuitively how the knowledge can be derived from state by observation (from their *Naïve* semantics) and how the epistemic logic reasoning can be done by an external solver. This approach derives the agent's knowledge (including group knowledge) from the state directly without explicitly maintaining the changes in the Kripke structure. In addition, since this approach reasons on the states only, it is an action-model-free approach that can work with simulators or even in other techniques beyond planning.

2.4.1 Background

The general background has been introduced in previous sections in this chapter, including Planning (Section 2.1), Epistemic Logic (Section 2.2) and Epistemic Planning (Section 2.3). Some additional background about seeing relation specific to their work [4] is provided here:

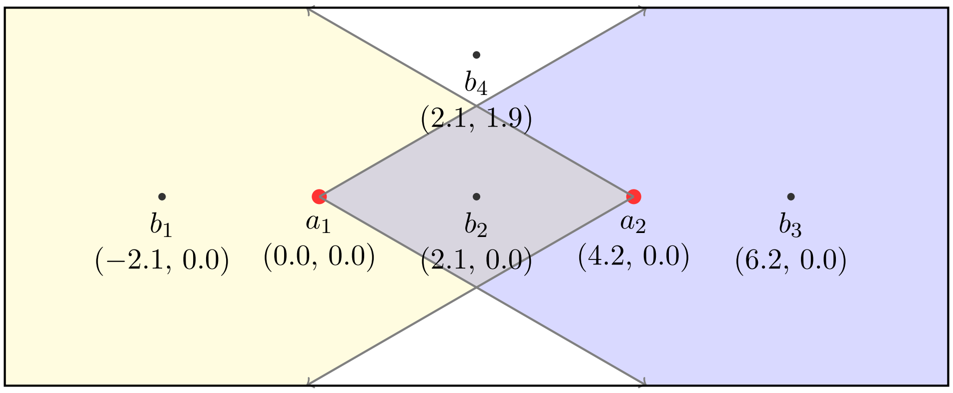


FIGURE 2.4: Example for Big Brother Logic.

Quoted text: " Gasquet et al. [2] 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, named *Big*

Brother Logic (BBL). Their framework models multi-agent knowledge in a continuous environment of vision, which has many potential applications such as reasoning over camera inputs, autonomous robots, and vehicles. They introduce the semantics of their model and its extensions on natural geometric models.

In their work, the agents (stationary cameras) are 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. [3]'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 world. 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. Those functions are defined for every 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), that if in two direction functions that all agents except a have the exact same directions, then those two direction functions are indistinguishable to a.

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:

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

> 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.
>
> Figure 2.4 shows an example with two agents, a_1 and a_2, and model $((0.0, 0.0), 60^\circ, D, R)$ and $((4.2, 0.0), 60^\circ, D, R)$ respectively, along with four objects, b_1, b_2, b_3, and b_4. Based on the current world, 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 $(pos, ang, D, R), dir \vDash a_1 \triangleright b_3$.
>
> From this, Gasquet et al. show the relationship between seeing and knowing. For example, $K_a(b \triangleright c)$ is defined as $a \triangleright b \wedge a \triangleright c \wedge b \triangleright c$.
>
> They also define a common knowledge operator, in a similar manner to that of Fagin et al.'s definition based on $G{-}reachable$ worlds. In Figure 2.4, the formula $\mathrm{C}_{\{a_1,a_2\}} a_1 \triangleright b_2$ 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 the laws of geometry that a_1 can see "a_2 can see b_2" as a_1 can see both a_2 and b_2, and a_2 can see b_2. Furthermore, from the previous statement, and a_2 can see a_1, we get that a_2 can see "a_1 can see that a_2 can see b_2", etc. Thus, some common knowledge has been established. "

In addition to Gasquet et al.'s work, another source of inspiration for Hu's research is from Cooper et al. [1]. Cooper et al. generalise seeing relations from the visibility in the actual world to the abstract ideas of seeing propositions.= This means that agents can see properties about the world, and can see and know whether other agents see these properties. This is flexible enough to model, for example, whether agents see the same value of a traffic light or whether agents see that others see the same value of a traffic light. Thus, Hu [4] includes Cooper et al.'s formulization as follows:

> ***Quoted text:*** " Cooper et al. [1] add an extra type of formulae α that describes

> formulae (propositions) that can be seen:
>
> $$\begin{aligned} \alpha &::= p \mid S_i\alpha \\ \varphi &::= \alpha \mid \varphi \wedge \varphi \mid \neg\varphi \mid K_i\varphi, \end{aligned}$$
>
> in which $p \in Prop$ (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 nesting of seeing relationships that see atomic propositions.
>
> From this, they note the equivalences $K_ip \leftrightarrow p \wedge S_ip$ and $K_i\neg p \leftrightarrow \neg p \wedge S_ip$. To be specific, they disallow $S_i\neg p$. This tight correspondence between knowing and seeing 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 [122, 123, 130]. "

Comparing these two bodies of work, Gasquet et al. use a geometric model to represent the environment and derive knowledge from this by checking the agents' line of sight. Their idea formalises the notation that "seeing is believing". However, their logic is focused only on agents' visions in physical spaces.While in Cooper et al.'s world, the seeing operator applies to propositional variables or formulae (even includes epistemic formulae), and thus visibility is more abstract, such as "seeing" (hearing) a message over a telephone (instantaneous effects). This connection between seeing and knowing is similar to the idea of sensing actions in partially-observable planning [6, 73–77, 115, 118, 119, 126], as seeing/sensing generates new knowledge. However, sensing actions are actions, whereas the idea of 'seeing' is a relation over properties of states.

Differing from all the works above, Hu [4] generalises seeing relations by defining the perspective functions, which are domain-dependent functions defining what agents see (the seeing rules) in particular worlds. The result is more flexible than seeing relations. It allows epistemic logic (knowledge), such as BBL, to be defined and reasoned with a simple perspective function.

2.4.2 Agent's Perspective Model

Now, we introduce one of the core ideas in Hu [4]'s work, which is the Agent's perspective model. By following F-STRIPS (using functional variables rather than propositions), their model is able to handle problems with both discrete and continuous domains.

They defined any epistemic planning problem that can be handled by their model as follows:

> ***Quoted text:*** " We defined an epistemic planning problem as a tuple $(Agt, V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, 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 set respectively, and both of them are also bounded by V and D. Specifically, the initial state should be a complete assignment for all V, while the goal states set is a set of complete states that satisfy goal conditions (assignments or relations between variables' values). $\mathcal{O}$ is the set of operators, with arguments in the terms of variables from V. The $\mathbb{F}$ denotes the external functions. "

2.4.2.1 Language

Then, they defined the language of their model as follows:

> ***Quoted text:*** "
>
> **Definition 2.13.** 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 "ontic" relation symbol, which takes k ground 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.

> ”

With this definition, they divide all formulae in their model into three categories: ontic formulae [8], visibility formulae, and knowledge formulae.

Ontic formulae do not only include basic mathematical relations, but also relational terms defined by the underlying planning language, which means they can have relations between variables. They gave an example as follows:

> ***Quoted text:*** “ For example, from Figure 2.4, “$pos(a_1) = (0, 0)$” is a true formula expressing the position of agent a_1, while “$pos(a_1) = pos(a_2)$” is false. Since they used F-STRIPS, which allows more complex customized relations, as long as they are defined in the external functions. For example, we can define an ontic 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 the distance between i and j is longer than i and k. From the same scenario in Figure 2.4, $@far\_away(pos(a_1), pos(b_4), pos(b_2))$ would be true, while $@far\_away(pos(a_1), pos(b_1), pos(b_2))$ would be false, since b_1 and b_2 are at the same distance to a_1. ”

Their intuition is that this function can be defined and implemented in any programming language, such as C++, as the external functions, and the planner is unaware of its semantics.

As for seeing relations, their intuition is from “seeing a proposition” [1]. They gave an example as follows:

> ***Quoted text:*** “ Using a proposition p as an example, “agent i knows whether p” can be represented as “agent i sees p”. The seeing formula represents two related interpretations: either p is true and i knows that; or, p is false and i knows that. With higher-order observation added, this intuition provides them

[8]From their discussion, we believe the naming is not accurate. Thus, we rename it as ontic formulae onwards.

a way to reason about others' epistemic viewpoints about a proposition without actually knowing whether it is true. Building on this concept, their seeing formula represents two related interpretations: $S_i v$ (seeing a variable) and, $S_i \varphi$ (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. 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 simplified by them to that agent i sees all the variables in φ. For example, in Figure 2.4, $S_{a_1} pos(b_2)$ can be read as "agent a_1 sees variable $pos(b_2)$", and it represents whether agent a_1 *knows* b_2's location, wherever b_2 locates. In the case of seeing an ontic formula, let φ be $far\_away(pos(a_1), pos(b_4), pos(b_2))$. Then, $S_{a_1} \varphi$ 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". "

Following Cooper et al. [1]'s idea on defining knowledge based on visibility, they 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.

2.4.2.2 Model

Their model decomposes the planning model from the epistemic logic reasoning model. As discussed later in their implementation, their planner reasons about epistemic logic by external functions. Therefore, we introduce their model and semantics of their external epistemic logic (knowledge) reasoning solver. The novel part of this model is the use of perspective functions, which are functions that define the seeing rules of the epistemic planning domain, instead of using full Kripke structures. From this, a rich knowledge model can be built up independent of the planning process.

Quoted text: "

Definition 2.14 (Quoted from Hu [4]). A model M is defined as $M = (V, D, \pi, f_1, \ldots, f_n)$, in which symbols are explained as follows:

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, $s(v_i) = \mathit{null}$. The set of all states (local and global) is denoted as $\mathcal{S}$. π is a set of evaluation functions, such that for $\pi_k \in \pi$, $\pi_k : R \to \mathcal{S} \to \{\mathit{true} \mid \mathit{false}\}$, 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 $i \in \mathit{Agt}$ 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 $f_i(s) \subseteq f_i(s')$

"

The model they defined follows the F-STRIPS language, which is also very similar to first-order Kripke structure for knowledge in Fagin et al. [3]'s book. The difference lies in the perspective function $f_0, \ldots, f_k$. First-order Kripke structures still keep the possible worlds and their accessibility relation $\mathcal{K}_i$, while Hu uses perspective function to define the semantics of the epistemic relations in their language as below. Then, they gave their semantics with an explanation as follows:

Quoted text: "

Definition 2.15 (Semantics for Single Knowledge with Perspective Function [4])**.** Given an agent's justified perspective model $M = (V, D, \pi, f_1, \ldots, f_k)$, the semantics of their language is defined as:

(a) $(M, s) \vDash R(v_1, \ldots, v_k)$ iff $\pi(R, s(v_1), \ldots, s(v_k))$=*true*

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

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

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

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

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

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

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

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

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.

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 2.4, $S_{a_1} b_1$ 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 exactly the same variables. Besides, the semantics of $S_i\ \neg\varphi$ is "i knows whether $\neg\varphi$ is true or not", which is the same as the 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 to $(S_i\varphi \wedge S_i\psi)$ 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 $\varphi \wedge \psi$. 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. [1], an agent always sees what it sees, to $S_i S_i \varphi$ is a validity.

The definition as shown in (i) follows the idea in Cooper et al. [1]'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, $K_{a_1} @\mathit{far\_away}(\mathit{pos}(a_1), \mathit{pos}(b_2), \mathit{pos}(b_3))$ 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, $K_{a_1} @\mathit{far\_away}(\mathit{pos}(a_1), \mathit{pos}(b_3), \mathit{pos}(b_2))$ is true. In addition, combining negation semantics from the 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. [123]'s paper. "

2.4.2.3 Validation

They also validated their single knowledge semantics. Firstly, they discuss some basic properties of their logic. Then, they prove the soundness of their model, followed by showing the completeness of their model for logically separable formulae. In the following parts of this section, they use the following example: a state s contains one variable x, and the domain for x is $\{1, 2, 3\}$. Therefore, all global states in this example for our model contain $s_1 = \{x=1\}$, $s_2 = \{x=2\}$, $s_3 = \{x=3\}$. The state space is formed as $\mathcal{S} = \mathcal{S}_c \cup \{s_{empty}\}$, where $s_{empty} = \{\}$. This example is visualized in Figure 2.5, where k_1 and k_2 represent two Kripke structures: agent i does not know the value of x in any world; and agent i knows the value of x in all three worlds, respectively.

They claim their semantics follows S5 axioms and proves it as follows:

Quoted text: "

Theorem 2.16 (Quoted from Hu [4]). *The S5 axioms of epistemic logic are valid in this logic. That is, the following axioms hold:*

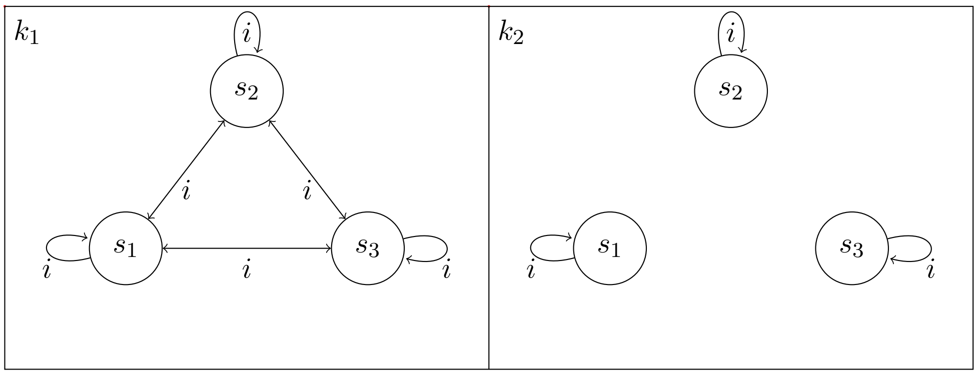


FIGURE 2.5: Two Examples of Kripke Structure [4]

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

"

They proved the above theorem as follows:

Quoted text: "

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 $(M, s) \models K_i(\varphi \rightarrow \psi)$ is equivalent to $(M, s) \models S_i(\varphi \rightarrow \psi)$ and $(M, s) \models (\varphi \rightarrow \psi)$. Then, by our semantics, we have that $(M, s) \models S_i(\varphi \rightarrow \psi)$ is equivalent to $(M, s) \models S_i\neg\varphi$ or $(M, s) \models S_i\psi$. From propositional logic, $\varphi \rightarrow \psi$ 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 $(M, s) \models K_i\varphi \rightarrow K_i\psi$ 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 $(M, f_i(s)) \vDash \varphi$, 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 $(M, s) \vDash K_i\neg K_i\varphi \equiv (M, s) \vDash (true \wedge \neg S_i\varphi) \vee (S_i\varphi \wedge \neg\varphi)$, which is in fact equivalent to $(M, s) \vDash \neg S_i\varphi \vee \neg\varphi$ and thus matches the premise "$\neg K_i\varphi \leftrightarrow \neg S_i\varphi \vee \neg\varphi$". Hence, (5) holds. □

"

Then, they proved the soundness and completeness of their semantics by constructing a corresponding Kripke structure. They proposed a theorem (Theorem 2.17) to prove that for every instance in their model M, there is a corresponding Kripke structure M^K. Their theorem and proofs are provided here:

Quoted text: "

Theorem 2.17 (Quoted from Hu [4])**.** *Let M be any instance of the agent's perspective 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 $M = (V, D, \pi, f_1, \dots, f_n)$ and its corresponding Kripke structure $M^K = (\mathcal{S}^K, \pi^K, \mathcal{K}_1, \dots, \mathcal{K}_n)$. 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, $\mathcal{K}_1, \dots, \mathcal{K}_n$, 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 $\mathcal{K}_1, \dots, \mathcal{K}_n$ 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". $\square$

"

As they mentioned, although the above theorem holds, the corresponding Kripke structure does contain more information that their semantics deliberately choose to ignore. Their construction above is only a full structure without any imperfect information (information that implies a constraint on the variable value without identifying the value). Using example from Figure 2.5, k_1 is a full structure without any imperfect information. If agent i knows whether the value of x is not 1, then we remove the bidirectional edges between s_1 and s_2 and between s_1 and s_3. Thus, in the Kripke structure, we can have something such as $K_i x > 1$ while it cannot be directly modeled by their model.

Then, since "seeing operator" (from their model) is not defined in Kripke structure, they need to define it first before reasoning about its soundness and completeness of their semantics. Their definition is presented as follows.

Quoted text: "

Definition 2.18 (Quoted from Hu [4]). (Seeing formula in Kripke structure). Let any Kripke structure be M^K, any agent be i and any formula in our grammar be φ:

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

- $(M^K, s) \models S_i\ \varphi$ iff $\forall(s,t) \in \mathcal{K}_i$, $(M^K, s) \models \varphi \Leftrightarrow (M^K, t) \models \varphi$.

"

Their definition of seeing operator in Kripke semantics for $S_i v$ and $S_i\varphi$ was derived from Wan et al. [129]'s definition of "knowing whether". Their explanation about the above definition is given as:

Quoted text: " 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 to 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 all of i's possible worlds from s agree on the truth value of φ. "

With the definition of the seeing operator in Kripke structure, then, they can reason about soundness and completeness of their semantics. But first, they proved item (g), which is the special case in semantics, holds in the model also holds in the constructing Kripke structure (as proposed in the following theorem).

Theorem 2.19. $(M^K, s) \models S_i S_i\varphi$ *is always true.*

The proof is given as follows:

Quoted text: "

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 2.18, 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:

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

This means that for all worlds in $\mathcal{K}_i(t_1)$ agree on the 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 $\forall t_1 \in \mathcal{K}_i(t_1),\ (M^K, t_1) \ \vDash\ \varphi \Leftrightarrow (M^K, t_1') \ \vDash\ \varphi$, which contradicts our earlier assertion that $\exists t_2', t_2'' \in \mathcal{K}_i(t_2),\ (M^K, t_2') \ \vDash\ \varphi \wedge (M^K, t_2'') \ \vDash\ \neg\varphi$.

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. □

"

Then, they proposed a theorem for the soundness in general cases in their semantics as follows:

Quoted text: "

Theorem 2.20 (Quoted from Hu [4])**.** *(Soundness). Let s be the current state, M be our model, and M^K be the corresponding Kripke structure defined using the approach in the proof of Theorem 2.17. The following hold:*

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

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

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

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

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

(6) Both $(M, s) \ \vDash\ S_i\ S_i\ \varphi$ *and* $(M^K, s) \ \vDash\ S_i\ S_i\ \varphi$ *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* $(M^K, s) \models K_i\ \varphi$

"

They prove the above theorem as follows:

Quoted text: "

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 $(v{=}e) \in f_i(s)$. 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 2.5: for any state s in $\mathbf{S}$, 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 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 2.19 and item (g) in Definition 2.15. □

"

Then, they proposed a theorem for the completeness of their semantics and showed the proof as follows.

Quoted text: "

Theorem 2.21 (Quoted from Hu [4])**.** *(Completeness). Let s be the current state, M be any instance in our model, and M^K be its corresponding Kripke structure constructed following the steps in the proof for Theorem 2.17. All the following hold for any formula in our language excluding* tautology *and* contradiction*:*

(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 $(M, s) \models \neg K_i\ \varphi$.

"

They prove the above theorem as follows:

Quoted text: "

Proof. Following the definition of seeing formula in Kripke structure given above, $(M^K, s) \vDash S_i\ v$ 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, $(M^K, s) \vDash S_i v$ holds, while $(M, s) \vDash S_i v$ 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 $(M^K, s) \vDash S_i\ v$ holds and the size of v's domain is larger than 1. Then, following the definition of $(M, s) \vDash S_i\ v$, $v{=}e$ exists in $f_i(s)$, then (1) holds.

For example in Figure 2.5: 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 $(M, s) \vDash S_i\ v$ 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, $(M^K, s) \vDash S_i\ v$ does not hold as the value of v can be any of 1 or 2 or 3, as well as $(M, s) \vDash S_i\ v$.

Because (1) holds and $\pi(s)$ are almost identical in both M^K and M, then (2) holds. Item (7) is proved in the same way as in Theorem 2.20. 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 $(M^K, s) \vDash S_i\ (\varphi \wedge \psi)$ 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, $(M^K, s) \vDash S_i\ \varphi$ and $(M^K, s) \vDash S_i\ \psi$ 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 $(M, s) \vDash S_i\ \varphi$ and $(M, s) \vDash S_i\ \psi$ 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 2.5, $(M^K, s) \vDash S_i\ (\varphi \wedge \psi)$ holds. However, if agent i cannot see all variables in φ and ψ, then one of $(M, s) \vDash S_i\ \varphi$ and $(M, s) \vDash S_i\ \psi$ will not hold, which means $(M, s) \vDash S_i\ (\varphi \wedge \psi)$ will not hold. Therefore, (4) holds if $(\varphi \wedge \psi)$ is not a contradiction.

Item (3) holds as $S_i \neg\varphi \equiv S_i\varphi$, and $S_i\varphi$ holds by induction. Items (5), (7) and (8) are straightforward by induction. Item (6) holds as Theorem 2.19 and item (g) in Definition 2.15. Therefore, our model is complete for all situations except in which formulae inside seeing operators that contain non-seen variables are validities. □

"

They explained the reason that Kripke semantics can handle tautologies and contradictions is that the semantics checks whether all possible worlds agree on the truth value of the formula, while their model reduces reasoning on uncertain (unseen) variables by ignoring them in the agent's local perspective. They claimed they could handle tautologies and contradictions by checking formulae using resolution. But it is an NP-hard problem to solve, and they believe it would be unnecessary for most problems.

2.4.2.4 Agent's Perspective Model for Group Knowledge

Based on the foundation and similar intuition of their single-agent's knowledge semantics, they defined language and semantics for group operators, including distributed and common visibility or knowledge.

They extend the syntax of their 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 a single agent defined in this, and α is a variable v or formula φ. In addition, this explained their language by:

> ***Quoted text:*** " 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, 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". "

They defined their group semantics as follows:

> ***Quoted text:*** "
>
> **Definition 2.22** (Quoted from Hu [4])**.** Let G be the set of agents, φ a formula, and α either a formula or a variable, the semantics of these group formulae can be defined as follows:
>
> - $(M, s) \models ES_G\ \alpha$ iff $\forall i \in G$, $(M, s) \models S_i\ \alpha$
> - $(M, s) \models EK_G\ \varphi$ iff $(M, s) \models \varphi$ and $(M, s) \models ES_G\ \varphi$
> - $(M, s) \models DS_G\ \alpha$ iff $(M, s') \models \alpha$, where $s' = \bigcup_{i \in G} f_i(s)$
> - $(M, s) \models DK_G\ \varphi$ iff $(M, s) \models \varphi$ and $(M, s) \models DS_G\ \varphi$
> - $(M, s) \models CS_G\ \alpha$ iff $(M, s') \models \alpha$, where $s' = cf(G, s)$
> - $(M, s) \models CK_G\ \varphi$ iff $(M, s) \models \varphi$ and $(M, s) \models CS_G\ \varphi$,
>
> in which $cf(G, s)$ is the state reached by applying the composite function $\bigcap_{i \in G} f_i$ until it reaches its fixed point. That is, the fixed point s' such that $cf(G, s') = cf(G, \bigcap_{i \in G} f_i(s'))$.
>
> "

They explained their group seeing and knowledge semantics and proposed a theorem to show their fix-point state can be found in polynomial time (specifically in at most s steps).

> ***Quoted text:*** " Reasoning about common knowledge and common visibility is more complex than other modalities. Common knowledge among a group is not only that everyone in the group shares this knowledge, but also that 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 the 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.
>
> 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 are at most $|s|$ steps. So, a poly-time algorithm for function cf exists.
>
> "

2.4.2.5 A Brief Note on Expressiveness

At the end of their model section, they discussed the difference and links between their model and the models that use Kripke structure. Firstly, they explained their intuition and the connection and difference between their model and first-order Kripke structure [3]. Then, they showed their model has the capability to model any problem that is modeled by Kripke structure. That is, reducing from first-order logic into propositional logic.

As they discussed, although this could model disjunction (same as Kominis and Geffner [115]), this could easily result in an exponentially large model and would not add the expressiveness required for most of the epistemic planning problems. They also summarised the expressiveness of their approach with others in Table 2.1.

	Depth	CK	DK	Continuous Domains	Disjunctive Knowledge
Perspective Model	Unlimited	Y	Y	Y	Possible
PDKB [88]	Bounded	N	N	N	N
K & G [115]	Bounded	N	N	N	Y
MEPK [7]	Bounded	I	N	N	N
EFP & PG-EFP [6]	Bounded	I	N	N	N

Table 2.1: Expressiveness Comparison over Epistemic Planning Approaches [4].

They summarise the differences between their approach and others as the following four major points:

Quoted text: " (1) Our model can handle domains in which the depth of epistemic relations is unbounded. Each level of nesting is handled by a set operation from the 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 Le et al. [6], Kominis and Geffner [115] keeps the Kripke structure in their approach, we are unsure about whether their approaches are practically capable of modeling unbounded domains or not. In Muise et al. [88]'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 [134].
>
> (4) Our model does not handle disjunctive knowledge, but could do so by modeling pairs of each variable and all its 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 modeled 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. "

2.4.3 Implementation & Experiments

To validate their model and test its capabilities, they encoded it within a planner and solved some well-known epistemic planning benchmarks. They used BFWS(f_5) [63] as the planner and used F-STRIPS with external functions, which allow them to decompose the planning task from the epistemic logic reasoning.

In this section, they explained the F-STRIPS encoding for their model and the implementation of the agent's perspective function.

2.4.3.1 F-STRIPS Encoding

They explained their intuition that, with the perspective model, they only need a planning language to describe the ontic states and how it changes. Then, for every epistemic relation reasoning, they used the external functions from F-STRIPS. That is, the epistemic logic reasoning task is moved from the planner to the external functions.

They show how they model the problem with F-STRIPS as follows:

> ***Quoted text:*** " Our reasoning 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 and goals; the external functions $\mathbb{F}$ contain all the epistemic logic reasoning parts (our model). "

Then, they showed how the epistemic formulae can be integrated into the planning language: by in planning action's precondition; or, by in goal conditions (in the format of external function calls). They explained this by using the example in Figure 2.4 as follows:

> ***Quoted text:*** " Defining $\mathcal{G}$ with desirable epistemic formulae is straightforward. For example, in Figure 2.4, 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 2.4, 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 $\mathcal{O}$. "

2.4.3.2 Agent's Perspective Function

As they mentioned in their model definition (Definition 2.14), the perspective function, $f_i : S \rightarrow S$, is a function that takes a state and returns the local state as the perspective of agent i. Compared to the intuition of Kripke structures, their intuition is to only define which variables an agent sees. Individual and group knowledge all derive from agents' perspectives.

They have developed a library of external functions that implement the semantics of S_i, K_i, DS_G, DK_G, CS_G, and CK_G, using the underlying domain-specific perspective functions. In addition, with the planner from Francès et al. [63], they provide an epistemic planning framework that the model simply needs to provide the perspective function for their domain (if a suitable one is not already present in their library).

They show two examples based on BBL to demonstrate their implementation. The first one follows the example in Figure 2.4, where the blue area, yellow area and their intersection represent agent a_1's, agent a_2's and their common local perspectives of the global world. The implementation of the perspective function in this example follows Euclidean geometric calculation. Given the current state is s, the agent is i and target is j, whether i sees j is the evaluation of the following equation:

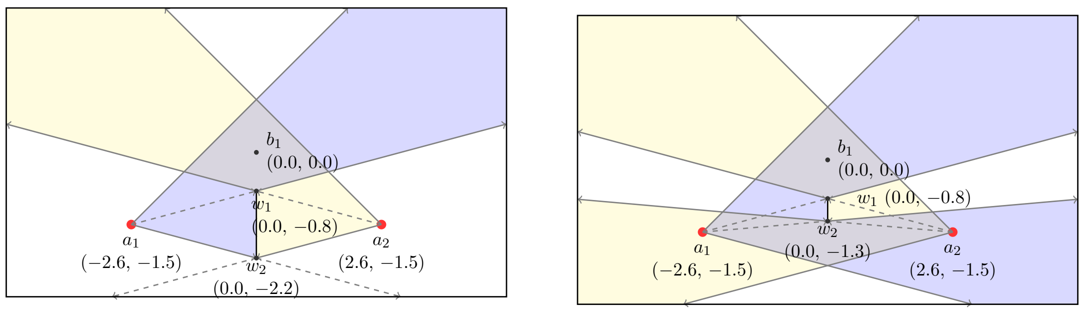
$$\left|\arctan\left(\frac{|s(y_i) - s(y_j)|}{s(x_i) - s(x_j)}\right) - s(dir_i)\right| \leq \frac{s(ang_i)}{2} \bigoplus \left|\arctan\left(\frac{|s(y_i) - s(y_j)|}{s(x_i) - s(x_j)}\right) - s(dir_i)\right| \geq \frac{360^\circ - s(ang_i)}{2} \quad (2.3)$$

Following the above seeing relation evaluation, their perspective function for this example takes the current state (including all agents' locations, directions, and vision angles, along with all other variables' locations; and it could be a local state when the epistemic relation evaluating has nesting) as input, and returns all the variables belonging to those agents and variables that fall inside these regions.

Their second example is a complicated version of the first one, in which they take obstacles into consideration. They show two scenarios with different sizes of the wall as shown in Figure 2.6. They explained this example as:

Quoted text: " In Figure 2.6a, 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 2.6b. 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 2.6: Examples for Big Brother Logic with Obstacle [4].

To sum up, those examples show that they can expand to new logics by providing different implementations of f_i. From this, the logic of knowledge is provided using their implementation of the semantics in Definition 2.15. That is, the modeller only needs to provide: a classical planning model that uses epistemic formulae, and implementation for $f_1, \ldots, f_k$ for each agent to specify how the seeing rule works in the modeled domain. Their library is able to use those perspective functions to evaluate the truth value of the epistemic formulae when solving the modeled epistemic planning problem.

2.4.3.3 Experiments

At last, authors demonstrated their approach by modeling and conducting experiments on various example domains, including benchmark domains such as Corridor and Grapevine, as well as challenging domains like BBL, Social Media Networks, and Gossip.

Their results show that the agent's perspective model approach outperforms the state-of-the-art PDKB approach [87] on the benchmark domains. Additionally, they showcased the expressiveness of their method by successfully modeling and solving problem instances in the challenging domains.

2.5 Research Questions and Thesis Outline

2.5.1 Research Questions

Recall that, in Chapter 1, we proposed our research question:

How can we enable expressive modelling and efficient solving of epistemic planning problems?

As discussed in Section 2.3.3, the current approaches in epistemic planning face limitations on their scalability, efficiency, generalizability, and expressiveness. Although Hu [4]'s approach is an initial effort to address these limitations, it presents some problems: 1) their semantics are neither sound nor complete; and, 2) their model only models knowledge (both individual nesting knowledge and group nested knowledge).

Thus, the main research question of this thesis can be broken down into a few less-general research questions as follows:

RQ1 : How can we use the perspective model to define a fragment of sound and complete epistemic planning model? Is this model more efficient than existing epistemic planning tools?

RQ2 : How do we extend this into a single coherent model that allows mixing of belief and knowledge?

RQ3 : How do we extend this into a group coherent model that allows mixing of group belief and group knowledge?

The details about how each research question is addressed in this thesis can be found in the outline of this thesis (in the following section).

2.5.2 Thesis Outline

The starting point and intuition of this work is from the master thesis by Hu [4] (Section 2.4), in which they proposed a novel state-based approach to model and solve epistemic planning problems with efficiency and expressiveness. We revise their work by proposing a formal definition with a clarification of their model and naïve semantics. In addition, we propose two forms of definition on its semantics and show their soundness and completeness. Then, we extend their model to handle agent's beliefs as well as group beliefs, including common and distributed beliefs. In addition, we also formalised the planning model, planning language and viable search algorithms. Furthermore, we did large-scale experiments on one distinct domain to show the performance of the search algorithms as well as how the epistemic planning problem instances change through altering features of those instances. The detailed content breakdown of this thesis is given as follows.

To address research question **RQ1**, in Chapter 3, we revise how the knowledge (group knowledge) is modeled by the agent's justified perspective model with premature semantics (based on Hu [4]'s work). The issue of Hu's model, mainly from its premature semantics, is that it is neither sound nor complete. Thus, we propose two new forms of semantics and show their soundness and completeness along with the proofs. At the end, we clarify the implementation of this approach, show the experiments and results, and conclude this approach with some discussions.

To address research question **RQ2**, in Chapter 4, we discuss the motivation for raising a new model, namely *Justified Perspective* (JP) model,to handle both knowledge and beliefs instead of just knowledge. In addition, we provide the background by showing this difference from an epistemic logic level. Then, we formally give our definition of the JP model and two forms of semantics, followed by a demonstration of its expressiveness and efficiency with some epistemic planning benchmarks and a primitive implementation. At last, we formalize the planning models for the JP model as well as providing definitions of two planning languages (a theoretical one and a practical one) for modeling corresponding epistemic planning domains. Since the planning models we proposed do not follow the assumptions of classical planning, especially the Markovian assumption (Assumption 6), some mechanisms of classical search algorithms are not applicable, such as duplication elimination and most of the heuristic functions. Thus, we designed our planner with an adapted duplication elimination mechanism and search algorithms that still work with the Non-Markovian assumption. Finally, we apply the JP model on benchmarks to show its efficiency and expressiveness and conduct extensive experiments on the chosen domain with varying instance settings, including the number of agents, the number of desired epistemic relations, and their depth.

To address research question **RQ3**, in Chapter 5, we extend the JP model to handle group beliefs. Compared to the relation between the single-agent's knowledge and belief, which effectively is that belief is past or present knowledge, the relation between group knowledge and belief is trickier. The group belief can be formed even if the group knowledge has never occurred. Then, we proposed the *Group Justified Perspective* (GJP) Model, which models group beliefs, including distributed beliefs and common beliefs. Similarly, at last, we show the expressiveness and efficiency of the GJP model through some epistemic planning domains with primitive implementation.

At the end, in Chapter 6, we summarize our work and discuss its contribution, as well as potential future directions.

Chapter 3

Planning with Perspectives: Decomposing epistemic planning with perspective

All men by nature desire to know. An indication of this is the delight we take in our senses, for even apart from their usefulness they are loved for themselves; and above all others the sense of sight.

—Aristotle and Aristotle

In this chapter, we firstly revise the first state-based approach on modeling and solving the epistemic planning problem. Then, we pointed out critical shortcomings in that work. Especially, their Naïve semantics is neither sound nor complete. Thus, we propose two new forms of semantics, *Complete Semantics* and *Ternary Semantics* and show the proofs for the soundness and completeness. At the end, we revise their implementation with more example domains and performed experiments based on ternary semantics.

3.1 A Revision on Planning with Perspective (PWP)

In this section, we revise the syntax and semantics of the agent perspective model from Section 2.4, including distributed and common knowledge. We named this model Planning with Perspective (PWP) Model for future referencing.

3.1.1 Signature

First of all, we need to define the signature of the PWP model, which specifies the "vocabulary" of our logic model.

Definition 3.1 (Signature)**.** A *signature* Σ is described by the tuple $\Sigma = (Agt, V, D, \mathcal{R})$, with Agt being a finite set of agent identifiers (of size k), V as a finite set of variables (of size m) such that $Agt \subseteq V$ and $m \leq k$, implying agent identifiers serve as variables. Furthermore, D denotes the set of all domains, where each D_{v_i} corresponds to a possibly infinite domain of constant symbols for each variable $v_i \in V$. Lastly, $\mathcal{R}$ denotes a finite collection of predicate symbols. Domains can be discrete or continuous.

To demonstrate the PWP model in the following parts of this thesis, we provide an example signature using the NIB domain.

Example 3.1. *The signature of the given example NIB domain (Example 1.2) can be represented by* $\Sigma^{NIB} = (Agt, V, D, \mathcal{R})$*, where:*

- $Agt = \{a, b\}$[1]
- $V = \{peeking_{ij}, j \mid i \in \{a,b\}, j \in \{p,q\}\}$
- $D = \{D_{peeking_{ij}}, D_j \mid i \in \{a,b\}, j \in \{p,q\}\}$*:*
    - $D_{peeking_{ap}} = D_{peeking_{aq}} = D_{peeking_{bp}} = D_{peeking_{bq}} = \{true, false\}$
    - $D_p = D_q = \{0, \ldots, 99\}$
- $\mathcal{R}$*: Includes all logical relation predicate symbols, such as "$>$" or "≤ 1".*

As previously noted, the PWP model is state-based; herein, we offer the formalization and associated notations for its states.

[1]In this example, the agent identifiers can be any of $peeking_{i,p}$ or $peeking_{i,q}$. For readability, we still use a and b to represent agent identifiers instead of $peeking_{a,p}$ and $peeking_{b,p}$

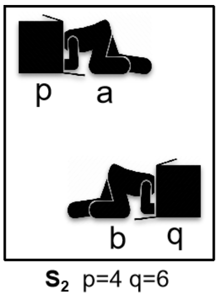


FIGURE 3.1: State s_2 in NIB domain.

Definition 3.2 (State)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, any state s in the PWP can be represented as a set of variable assignments $(V \to D)$, mapping the variables v to a value in their domains.

The set of all valid states is denoted as the state-space $\mathcal{S}$, while a *complete state* is a complete set of assignments for all variables in V $(s \in S \wedge |s| = |V|)$, while a *local state* is a partial set of assignments (some variables may not be assigned). The set of all complete states is denoted as $\mathcal{S}_c$. We use $s(v)$ to represent the value of v in s. If variable v is not in the local state s, then $s(v) = null$. The set of all models is denoted $\mathcal{M}$.

Following the above signature example (Example 3.1) for the NIB domain, its state space $\mathcal{S}^{NIB}$ and complete state space $\mathcal{S}_c^{NIB}$ can be represented as follows:

$$\mathcal{S}^{NIB} = \{s \mid s \subset s', s' \in \mathcal{S}_c^{NIB}\} \text{ , where:}$$

$$\mathcal{S}_c^{NIB} = \left\{ \begin{array}{l|l} peeking_{ap} = n_1, peeking_{aq} = n_2, & n_1, n_2, n_3, n_4 \in \{true, false\}, \\ peeking_{bp} = n_3, peeking_{bq} = n_4, & n_1 \uparrow n_2, n_2 \uparrow n_4, n_1 \uparrow n_3, n_2 \uparrow n_4, \\ p = i, q = j & i, j \in \{0, \ldots, 99\} \end{array} \right\}$$

In the above representation, "$\uparrow$" is used as "NAND" (not both). Those NAND relations ensure: neither a nor b can peek at both p and q at the same time; and, neither p nor q can be peeked by both a and b at the same time (as it is outlined in Example 1.2). The initial state described in Figure 2.2 is the state s_0 in the above state space, where:

$$s_0 = \left\{ \begin{array}{l} peeking_{ap} = false, peeking_{bp} = false, p = 4, \\ peeking_{aq} = false, peeking_{bq} = false, q = 6 \end{array} \right\}$$

From the initial state, after agent a performed action "**(peek a p)**" and agent b performed action "**(peek b q)**" (as shown in Figure 3.1), the new state (denoted as s_2)

can be represented by:

$$s_2 = \left\{ \begin{array}{l} peeking_{ap} = true, peeking_{bp} = false, p = 4, \\ peeking_{aq} = false, peeking_{bq} = true, q = 6 \end{array} \right\}$$

3.1.2 Language

With the signature in place, we can now define the language permitted in the PWP model by specifying its grammar. While this definition aligns with Definition 2.13, the use of the signature clarifies the structure of the language as outlined below.

Definition 3.3 (Language $\mathcal{L}_K(\Sigma)$)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, the language $\mathcal{L}_K(\Sigma)$ is defined by the grammar:

$$\varphi ::= r(V_r) \mid \neg\varphi \mid \varphi \wedge \varphi \mid S_i v \mid S_i \varphi \mid K_i \varphi,$$

in which $r \in \mathcal{R}$, terms $V_r \subseteq V$, $i \in Agt$, and $v \in V$.

A n-ary predicate symbol $r \in \mathcal{R}$ becomes a proposition by taking a set of variable V_r as input. In addition, intuitively speaking, only the ontic parts of the world should be modeled by our model as variables in the state. Thus, normally, $r(V_r)$ could be any propositional ontic relation (exceptions will be discussed in Section 3.3). $S_i\varphi$ is a visibility formula that means agent i sees the truth value of formula φ, while $S_i v$ is a visibility formula that means agent i sees the value of the variable v. $K_i\varphi$ is a knowledge formula that means agent i knows formula φ holds. Operators $\neg$ and $\wedge$ are defined in the standard way. We call a formula with no conjunction a *modal literal*. The function $vars(\varphi)$ returns all variables in φ.

The ontic relations that can be modeled by the PWP model have been discussed in Section 2.4.2.1. Here, we just clarify some intuitions about the relation between seeing formulae and knowledge formulae. The important concept in this logic, adapted from Cooper et al. [1] and [2], is "seeing a proposition". Let φ be a proposition, "agent i knows whether φ" can be represented as "agent i sees φ". The interpretation on this is: either φ is true and i knows that; or, φ is false and i knows that. With higher-order observations added, it gives agent i the ability to reason about whether other agents know whether proposition φ is true, without i knowing whether φ is true itself; e.g. $S_i S_j \varphi$.

We include $K_i\varphi$ in the grammar, but in fact, it is simply shorthand and can be defined as:

$$K_i\varphi \quad \leftrightarrow \quad \varphi \wedge S_i\varphi$$

That is, agent i knows φ if agent i sees φ and also φ is true. Similarly, if agent i knows φ, then it means that it is true (because it is knowledge, not belief), and that they must be able to see that it is true. This definition of knowledge is consistent with the relationship between knowledge and seeing identified by Cooper et al. [1].

Consider the example Big Brother Logic domain in Figure 2.4 and assume $value(b_1)$ is $false$ and all objects' ($b\_$) positions are commonly known to all agents. The formula $S_{a_2} value(b_1)$ can be read as "agent a_2 sees variable $value(b_1)$", and it means agent a_2 *knows* b_1's value, whatever that value is. The formula $K_{a_2}\ value(b_1) = false$ can be read as "agent a_2 knows variable $value(b_1)$ is $false$", which represents a_2 knows b_1's value is $false$. Further, agent a_1 does not know b_1's value, so we can say $\neg K_{a_1} K_{a_2}\ value(b_1) = false$. However, with the seeing relation, the formula $K_{a_1} S_{a_2}\ value(b_1)$ holds, since both $S_{a_1} S_{a_2}\ value(b_1)$ and $S_{a_2}\ value(b_1)$ hold.

3.1.3 PWP model

Now, we can give a formal definition of the PWP model as follows.

Definition 3.4. Given the signature $\Sigma = (Agt, V, D, \mathcal{R})$, a model M is defined as $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$.

- Agt, V and D are from the given signature Σ;
- π is an interpretation function $\pi : \mathcal{S} \times \mathcal{R} \to \{true, false\}$ that determines whether the predicate $r(V_r)$ is true in s. π is undefined if any of its arguments t_i is a variable in V that is not also in s.
- Finally, $f_1, \ldots, f_k$ are the agents' *perspective functions*, one for each agent in Agt. 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.

The following properties must hold on f_i 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 $f_i(s) \subseteq f_i(s')$

The above definition of the model is almost the same as in Definition 2.14. We added the domain for each variable v as D_v.

The definition of the perspective function requires some discussion. We can only provide a high-level definition as the perspective function is domain-specific. Each agent has their own domain-specific perspective function f_i that, for any given state in this domain, the perspective function returns a subset of the given state containing all the assignments that are visible to agent i.

For example, given a state $s = \{v_1 = e_1, v_2 = e_2\}$, then $f_1(s) = \{v_2 = e_2\}$ specifies that agent 1 cannot see variable v_1 or, by definition, its value, but can see variable v_2 and its value. These functions can be nested, such that $f_2(f_1(s))$ represents agent 1's perspective from agent 2's perspective, which can be just a subset of agent 1's actual perspective.

We provide 3 meaningful properties as constraints for the modeler to develop their own domain-specific perspective function. Property (1) ensures that each agent can only see true values of variables. Later, we see that this ensures that knowledge is always true. Property (2) ensures that an agent sees what it sees. Property (3) is a monotonicity constraint.

Using the NIB example (Example 1.2), the perspective functions for agent a and agent b are identical. Given a state $s \in \mathcal{S}^{NIB}$, the perspective function for agent i (either a or b in this example) is:

$$f_i^{NIB}(s) = \begin{cases} \{peeking_x = s(peeking_x) \mid x \in \{ap, aq, bp, bq\}\} & \text{base case} \\ \cup\{p = s(p)\} & \text{if } peeking_{ip} = true \\ \cup\{q = s(q)\} & \text{if } peeking_{iq} = true \end{cases}$$

That is, agents see whether each other is peeking into the box and they see the value of the number if they are peeking. Using the example states s_0 and s_2 (in Section 3.1.1), we have:

$$f_a(s_0) = f_b(s_0) = \left\{ \begin{array}{l} peeking_{ap} = false, peeking_{bp} = false, \\ peeking_{aq} = false, peeking_{bq} = false \end{array} \right\}$$

$$f_a(s_2) = \left\{ \begin{array}{l} peeking_{ap} = true, peeking_{bp} = false, \\ peeking_{aq} = false, peeking_{bq} = true, \\ p = 4 \end{array} \right\}, \ f_b(s_2) = f_a(s_2) \setminus \{p = 4\} \cup \{q = 6\}$$

3.1.4 Naïve Semantics for Individual Knowledge

Given the new definition of the model, we first formalised the following definition of semantics for language $\mathcal{L}_K(\Sigma)$, which we called the *Naïve* semantics. The intuition of this semantics is from Definition 2.15.

Definition 3.5 (Naïve Semantics for PWP model on $\mathcal{L}_K(\Sigma)$)**.** Given a PWP model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$, the naïve semantics of the language $\mathcal{L}_K(\Sigma)$ is defined as:

(a) $(M, s) \models r(V_r)$ iff $\pi(s, r(V_r)) = true$

(b) $(M, s) \models \phi \land \psi$ iff $(M, s) \models \phi$ and $(M, s) \models \psi$

(c) $(M, s) \models \neg\varphi$ iff $(M, s) \not\models \varphi$

(d) $(M, s) \models S_i v$ iff $v \in f_i(s)$

(e) $(M, s) \models S_i \varphi$ iff $(M, f_i(s)) \models \varphi$ or $(M, f_i(s)) \models \neg\varphi$

(f) $(M, s) \models K_i \varphi$ iff $(M, s) \models \varphi \land S_i \varphi$

The semantics for relational terms and propositional operators are straightforward, but the semantics for seeing is worth discussion. The semantics for $S_i v$, which means agent i sees variable v, is defined by stating that agent i sees v iff v is in the domain of state $f_i(s)$. The semantics for $S_i\varphi$ is defined as: either φ is true from agent i's perspective, or $\neg\varphi$ is true from agent i's perspective.

Compared to the original semantics from Hu [4] (Definition 2.15), our new naïve definition is simpler, clearer, and has less explicit treatments, such as $S_i S_j \varphi$ or $S_i \neg\varphi$. With the recursive grammar of the language (φ in the above definition could be any formula in $\mathcal{L}_K(\Sigma)$, including a seeing formula or knowledge formula), this new definition can also be nested arbitrarily. The semantics of: (a), the ontic predicates; (b) seeing a variable; and, (i) knowing a formula in the original naïve semantics are item (a), (d) and (f) in this definition respectively. Item (c), and (e)-(h) from Definition 2.15 are modeled recursively by item (e) in this definition. The remaining item, item (d), from the original definition is modeled by item (e) in this definition. Intuitively, seeing a formula means either knowing the formula is true or knowing the formula is false, which indicates $S_i\varphi \equiv S_i\neg\varphi$.

Let us illustrate the naïve semantics with more details using the following example.

Example 3.2. *In accordance with the NIB example, consider the state space $\mathcal{S}^{NIB}$ and perspective functions f_i^{NIB}. Two states, s_0 and s_2, are used as illustrative examples (see Section 3.1.1). Epistemic formulae [2] from Example 2.2 serve as the epistemic relation for evaluation.*

The evaluation of those formulae is as follows:

1. $K_a(p{=}4)$:
   - $M, s_0 \models K_a(p{=}4)$ does not hold because of $M, s_0 \not\models S_a(p{=}4)$;
   - $M, s_2 \models K_a(p{=}4)$ holds because $M, s_2 \models S_a(p{=}4)$ and $M, s_2 \models (p{=}4)$.
2. $K_aK_b(p{=}4)$: Both are $false$ since $M, s_0 \not\models K_b(p{=}4)$ and $M, s_2 \not\models K_b(p{=}4)$.
3. $K_a(p \times q{=}4)$: Both are $false$ since $S_a(p \times q{=}4)$ does not hold in neither states (a does not see q).
4. $K_a(p \times q \leq 99^2)$: Both are $false$ for the same reason as Item 3.
5. $K_aK_b(p \times q \leq 99^2)$: Same as above.

However, the naïve semantics suffers from two problems (hence the name 'naïve'), both related to the problem of having local states. First, the semantics are ill-defined. For $S_i\varphi$, $f_i(s)$ can be a local state, which is only a partial assignment of variables. If a

[2] For clarity, throughout this thesis, we display $r(V_r)$ in its typical notation. For instance, $(p{=}4)$ from $K_a(p{=}4)$ denotes the predicate $=4([p])$; similarly, $(p \times q{=}4)$ corresponds to the predicate $\times{=}4([p, q])$.

variable v is not visible in $f_i(s)$, then any proposition that uses v will default to false. It is reasonable to say that if an agent cannot see a variable in a predicate, then it cannot see (prove) the truth value of that predicate. However, this causes problems with formulae such as seeing tautologies (item 4 and item 5 in the above example) or contradictions. For example, the formula $M, s \vDash S_i\big(v = e \vee \neg(v = e)\big)$ will evaluate to false if v is not in $f_i(s)$. However, $v = e \vee \neg(v = e)$ is clearly a tautology, so agent i should always see that it is true.

Second, the semantics of $\neg\varphi$ uses a closed-world assumption. However, when s is a local state (partial), for any formula φ that refers to a variable not in s, we should be unable to prove φ or $\neg\varphi$. Defining $\neg\varphi$ as the inability to prove φ means that $S_i\varphi$ is a tautology: either φ or $\neg\varphi$ will always be true.

3.1.5 Naïve Semantics for Group knowledge

The signature and model for modeling and reasoning group knowledge are the same as described above. In addition, the grammar of the language is the same as defined in Section 2.4.2.4. We copy the grammar here for reference.

Definition 3.6 (Language $\mathcal{L}_{GK}(\Sigma)$)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, the language $\mathcal{L}_{GK}(\Sigma)$ is defined by the grammar:

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

in which ψ is $r(V_r)$ and $r \in \mathcal{R}$, G is a set (group) of agents ($G \subseteq Agt$), and α[3] is a variable v or formula φ.

Group formula $ES_G\alpha$ is read as: everyone in group G *uniformly* sees a variable or formula α, and $EK_G\varphi$ represents that everyone in group *uniformly* G knows φ. We named ES and EK as uniform seeing operator and uniform knowledge operator respectively. DK_G is the distributed knowledge operator, equivalent to D_G in Section 2.2.3.2, while DS_G is its visibility counterpart: someone in group G sees, or the group sees by merging the view of sight from each member. Finally, CK_G is common knowledge and CS_G common visibility: "it is commonly known" and "it is commonly seen" respectively.

[3]We use α for simplicity

As with the equivalence $K_i\varphi \leftrightarrow \varphi \land S_i\varphi$, we define the following equivalences:

$$\begin{aligned} EK_G\varphi &\leftrightarrow \varphi \land ES_G\varphi \leftrightarrow \textstyle\bigwedge_{i\in G} K_i\varphi, \\ DK_G\varphi &\leftrightarrow \varphi \land DS_G\varphi, \\ CK_G\varphi &\leftrightarrow \varphi \land CS_G\varphi. \end{aligned}$$

The group semantics in Definition 2.22 needs formalization and clarification. In order to do so, firstly, we need to propose a formal definition for the common perspective function cf.

Definition 3.7 (Common Perspective Function)**.** Given a group of agents G and the current state s, the common observation of the group can be defined as:

$$cf(G,s) = \begin{cases} s & \text{if } s = \bigcap_{i\in G} f(s) \\ cf(G, \bigcap_{i\in G} f(s)) & \text{otherwise.} \end{cases}$$

This is a recursive function, while the base case is when the input is equal to the intersection of all agents' local states. This means the common observation of this group G has converted into a partial state (could be empty). The variables that are not visible to any agent in the group G are filtered out until the remaining set becomes a fixed point set. That is, every variable in the set is commonly seen by the group G.

Then, we give the definition of the Naïve Semantics for Group Knowledge as follows.

Definition 3.8 (Naïve Semantics for PWP model on $\mathcal{L}_{GK}(\Sigma)$)**.** Given a PWP model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$, the naïve semantics of the language $\mathcal{L}_{GK}(\Sigma)$ is defined as:

(g) $(M,s) \models ES_G\alpha$ iff for all $i \in G$, $(M,s) \models S_i\alpha$

(h) $(M,s) \models EK_G\varphi$ iff $(M,s) \models (\varphi \land ES_G\varphi)$

(i) $(M,s) \models DS_Gv$ iff $v \in \bigcup_{i\in G} f_i(s)$

(j) $(M,s) \models DS_G\varphi$ iff $(M,s') \models \varphi$ or $(M,s') \models \neg\varphi$, where $s' = \bigcup_{i\in G} f_i(s)$

(k) $(M,s) \models DK_G\varphi$ iff $(M,s) \models (\varphi \land DS_G\varphi)$

(l) $(M,s) \models CS_Gv$ iff $v \in cf(G,s)$

(m) $(M,s) \models CS_G\varphi$ iff $(M,s') \models \varphi$ or $(M,s') \models \neg\varphi$, where $s' = cf(G,s)$

(n) $(M,s) \models CK_G\varphi$ iff $(M,s) \models (\varphi \land CS_G\varphi)$

where items (a)-(f) are inherited from Definition 3.5, and α represents a variable v or a formula φ.

The above definition of the semantics is similar to that defined in Hu [4]. This definition follows the intuition that knowledge can be derived from an agent's observation, while in the case of group knowledge, it becomes group observation. For the uniform observation of group G, it can be reasoned by either checking each agent's individual perspective or obtaining the intersection of all agents' perspectives ($\bigcap_{i \in G} f_i(s)$). While for the distributed observation of group G, the set union operator is needed to "pull" all agents' observations together. Since all local states are a subset of the global state at a given timestamp, there is no conflict on those assignments. Here, the conflict means given an input state s and any two agents i and j, no variable v that is in $f_i(s)$ and also in $f_j(s)$ makes $f_i(s)(v) \neq f_j(s)(v)$, since $f_i(s)(v) = s(v)$ and $f_j(s)(v) = s(v)$.

Compared to other modalities, the group's common observation and knowledge are more complex to reason about. Common knowledge in a group is not only that 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. Therefore, many researchers choose to add a specific notation and definition for common knowledge that sets apart from their definitions of agent's individual (nested) knowledge.

However, in our perspective logic, common knowledge is much simpler. This is based on the fact that each time we apply the composite perspective function $\bigcap_{i \in G} f_i(s)$ (to get uniform perspective of a group), the resulting state is either a proper subset of s or s itself. By this intuition, we can evaluate common visibility/knowledge in a bounded number of steps.

The fixed point is a recursive definition. However, the following theorem shows that this fixed point always exists, and the number of iterations is bounded by the size of $|s|$, the state to which it is applied.

Theorem 3.9. *Function $cf(G, s)$ (in Definition 3.7) converges to a fixed point $s' = cf(G, s')$ within $|s|$ iterations.*

Proof. In each iteration of cf, either $\bigcap_{i\in G} f_i(s) = s$ or $\bigcap_{i\in G} f_i(s) \subsetneq s$ because of the property that $f_i(s) \subseteq s$. If the former, we have reached the fixed point. For the latter, a maximum of $|s|$ such iterations are possible, by which point the fixed point has been reached, even if it is empty, in which there is no common knowledge. □

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 are at most $|s|$ steps. So, a poly-time algorithm for function cf exists.

Analogous to the individual naïve semantics, we illustrate the group naïve semantics with an example.

Example 3.3. *In accordance with the NIB example, consider the state space $\mathcal{S}^{NIB}$ and perspective functions f_i^{NIB}. One state s_2 (as all formulae do trivially not hold for s_0) is used as an illustrative example (see Section 3.1.1). Group epistemic formulae from Example 2.3 serve as the epistemic relation for evaluation. We list them here as the language in that example is different from ours ($\mathcal{L}_{GK}$):*

1. $EK_G(p{=}4)$
2. $DK_G(p \times q{=}4)$
3. $CK_G(peeking_{ap}{=}true)$

Item 1 does not hold due to $M, s_2 \not\models K_b(p{=}4)$. This can be reasoned by evaluating either $\forall i \in G, M, s_2 \models K_i(p{=}4)$ or $M, \bigcap_{i\in G} f_i(s_2) \models (p{=}4)$. The intersection $f_a(s_2) \cap f_b(s_2)$ is $\{peeking_{ap}{=}true, peeking_{bp}{=}false, peeking_{aq}{=}false, peeking_{bq}{=}true\}$, which means $M, \bigcap_{i\in G} f_i(s_2) \models (p{=}4)$ does not hold.

Item 2 does not hold due to $M, s_2 \not\models (p \times q{=}4)$. However, if $q = 1$, then Item 2 will hold because of $f_a(s_2) \cup f_b(s_2) = s_2$ and $M, s_2 \models (p \times q{=}4)$.

Item 3 holds. As mentioned in reasoning about Item 1, the intersection of $f_a(s_2)$ and $f_b(s_2)$, namely s', is $\{peeking_{ap}{=}true, peeking_{bp}{=}false, peeking_{aq}{=}false, peeking_{bq}{=}true\}$, which is not the same as input s_2. In the next iteration, we have $f_a(s') = f_b(s') = f_a(s') \cap f_b(s') = s'$. Thus, the common perspective of group G with the given state s_2 has converged on s'. We have $M, s' \models (peeking_{ap} = true)$, indicating $M, s_2 \models CS_G(peeking_{ap}{=}true)$. With $M, s_2 \models (peeking_{ap}{=}true)$, Item 3 holds.

3.1.6 Discussion

Now, we discuss the contribution and limitations of Hu [4]'s work, in which we can raise the motivation for proposing new semantics and proofs.

3.1.6.1 Remarks

In their work, they introduced a novel epistemic logic reasoning model in planning called agent's perspective model (Section 2.4.2, which we refined in previous parts of this section) driven from the intuition: "what you know is what you see". This perspective model allows them to evaluate epistemic relation formulae (knowledge), including nested, distributed, and common epistemic relations, based on the simple concept of defining an agent's local state. Then, by using F-STRIPS, they separated the planning task from epistemic relation reasoning (with an external function), which is an expressive and flexible solution for most of the epistemic planning problems. Compared to the Kripke structure based approach, their approach does not require specifying explicitly how epistemic formulae are updated as each action affects, which is done by the external function automatically with a given perspective function. In addition, compared to the pre-compilation approach, their approach allows lazy evaluation on epistemic relations without an expensive pre-compilation step.

Overall, their work is the first to: (1) reason about knowledge only based on the observable parts of the world; (2) separate epistemic logic reasoning from planning by delegating epistemic reasoning to an external solver.

3.1.6.2 Limitations

First of all, the semantics defined in Definition 2.15 and Definition 2.22 are neither sound nor complete. As we mentioned when refining their semantics in Section 3.1.4, this is caused by the closed world assumption (Assumption 10). The modeled problem follows this closed world assumption at the level of global perspective (global states), while this assumption no longer holds in the agent's local states, as the agent's local perspectives could be partial states.

In addition, the generalisability of their perspective (observation) function is another controversy that needs to be explained with more examples (Section 3.4). Besides, as one of their main contributions, their planning process of the problem is novel and valid, but needs more clarification with detailed examples (Section 3.5 and Section 3.6).

At last, their approach works well when modeling epistemic planning domains that epistemic relations can be reasoned from all (ontic) variables from the state, while for domains that contain unbounded epistemic relations in the action precondition, even with their approach, it still cannot be modeled. One typical example is the Gossip domain [136]. This requires some further work to model and solve those domains following their intuitions (Section 3.6.4).

3.2 Complete Semantics

In this section, we provide a *Complete* semantics of our model. As the name indicates, the complete semantics is both sound and complete. Similarly, as in Hu [4]'s work, the key part of the semantics is the use of states of the form $\{v_1 = e_1, \ldots, v_m = e_m\}$, rather than possible worlds found in Kripke semantics, and the use of perspective functions rather than Kripke relations. All preliminaries are given in Section 3.1, where a signature $\Sigma = (Agt, V, D, \mathcal{R})$ is defined in Definition 3.1 and a model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$ is defined using perspective functions f_i.

3.2.1 Complete Semantics for PWP model on Knowledge

First of all, since all states in the PWP model are sets of assignments, we need to define a state override function $\langle\ \ \rangle$ to locate all states that are consistent with the given state.

Definition 3.10 (State Override Function)**.** A state override function $s'\langle s\rangle : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ for a given state s overrides a state s' is defined as:

$$s'\langle s\rangle = s \cup \{v = s'(v) \mid v \in s' \wedge v \notin s\}$$

Intuitively, overriding a state s' with state s means that the new state is the same as $s \cup s'$, but if v has a value in both s and s', the value in s is used. Using this, we can extend the naïve semantics in Definition 3.5 into a semantics that is both sound and complete, but has exponential time complexity.

Definition 3.11 (Complete Semantics for PWP model on Knowledge)**.** Given a PWP model $M = (Agt, V, D, \pi, f_1, \dots, f_k)$, the complete semantics of the language $\mathcal{L}_K(\Sigma)$ is defined as:

(a) $(M, s) \vDash r(V_r)$ iff $\pi(s, r(V_r)) = true$

(b) $(M, s) \vDash \phi \wedge \psi$ iff $(M, s) \vDash \phi$ and $(M, s) \vDash \psi$

(c) $(M, s) \vDash \neg\varphi$ iff $(M, s) \nvDash \varphi$

(d) $(M, s) \vDash S_i v$ iff $v \in f_i(s)$ or $|D_v| = 1$

(e) $(M, s) \vDash S_i \varphi$ iff $\forall g \in \mathcal{S}_c, \big(M, g\langle f_i(s)\rangle\big) \vDash \varphi$ or, $\forall g \in \mathcal{S}_c, \big(M, g\langle f_i(s)\rangle\big) \vDash \neg\varphi$

(f) $(M, s) \vDash K_i \varphi$ iff $(M, s) \vDash \varphi \wedge S_i \varphi$

The seeing operator S needs some clarification. Based on the truth value of the seeing formula defined in Definition 2.18, $S_i v$ is true if and only if: either, agent i syntactically sees v, which is v in agent i's perspective (observation) of the given state; or, agent i semantically sees v, which follows the closed-world assumption (Assumption 10). The latter condition triggers when v is not in $f_i(s)$. If and only if v is consistent (has the same value) in all worlds agent i considers possible, which effectively means v only has 1 possible value, we have agent i semantically sees v. As for seeing a formula $S_i\varphi$, the effect of evaluating φ (and $\neg\varphi$) under $g\langle f_i(s)\rangle$ for every g in the complete-state space means that φ (and $\neg\varphi$) is evaluated under $f_i(s)$, but quantifying over every possible value for variables not in $f_i(s)$. This complete semantics solves the issues with the naïve semantics.

The time complexity of the naïve semantics is $\Theta(n \times |\varphi|)$, in which n is the maximum depth of a nested query in φ, and $|\varphi|$ is the size of the formula. However, for the complete semantics for $S_i\varphi$, we need to iterate over all $\mathcal{S}_G$ global states, meaning the worst-case complexity is $\Theta(n \times |\mathcal{S}_G|)$. Note that for models with infinite domains (e.g. continuous variables), $\mathcal{S}_G$ is infinite. Of course, in practice, we need only iterate over any variables in φ that are not in $f_i(s)$, and we can also re-write the formula into CNF and solve for any unreferenced variables.

As already noted, the naïve semantics is unsound and incomplete; however, the complete semantics is both sound and complete. Using item 4 ($K_a(p \times q \leq 99^2)$) and item 5 ($K_aK_b(p \times q \leq 99^2)$) from Example 3.2 as examples, both do not hold in naïve semantics due to agent a not seeing q in neither s_0 nor s_2. However, with the complete semantics, in all the worlds ($\forall g \in \mathcal{S}_c, g\langle f_a(s)\rangle$) agent a considers possible given the current state s (works for any reachable state, not only s_0 and s_2), we have $M, g \vDash (p \times q \leq 99^2)$. Thus, $M, s \vDash S_a(p \times q \leq 99^2)$ and $M, s \vDash (p \times q \leq 99^2)$, which results in item 4 holding. Item 5 also holds and can be proved in the same way.

Both example and theorem above show that, unlike the naïve semantics, the complete semantics handles epistemic formulae that hold only because of the closed-world assumption. The example above shows that, unlike the naïve semantics, the complete semantics handles epistemic formulae that hold only because of the closed-world assumption. In other words, the complete semantics is sound and complete. This is straightforward to show by simply defining Kripke structures corresponding to our models, which is also aligned with Theorem 2.17.

For each model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$, we can map to a corresponding Kripke structure $M' = (\mathcal{W}, \pi, \mathcal{R}_1, \ldots, \mathcal{R}_n)$. First, we map states to worlds: each global state $g \in \mathcal{S}_c$ corresponds to a world in $\mathcal{W}$. Second, perspective functions are mapped to Kripke relations: given a perspective function $f_i(s)$, the corresponding Kripke relation $\mathcal{R}_i$ can be constructed by taking each global state g and its corresponding world w, and defining $(u, w) \in \mathcal{R}_i$ for every $u \in \mathcal{W}$ such that u and w agree on all variables in $f_i(s)$. Effectively, this means that for any variable $v \in f_i(g)$, all reachable worlds in $\mathcal{R}_i(w)$ will agree on v, and for any variable $v \notin f_i(g)$, there will be at least one reachable world for every $e \in D_v$. So, an agent can either know the value of a variable, or know nothing about the value of the variable.

Therefore, the set of reachable worlds $\mathcal{R}_i(w)$ corresponds to the set of states $\{g\langle f_i(s)\rangle \mid g \in \mathcal{S}_G\}$, which is precisely the set of states that are evaluated in the semantics of $S_i\varphi$. Given that the Kripke-based semantics for $K_i\varphi$ assesses all reachable worlds in $\mathcal{R}_i(w)$, and given the equivalence $S_i\varphi \leftrightarrow (K_i\varphi \vee K_i\neg\varphi)$, our complete semantics are sound and complete.

3.2.2 Complete Semantics for PWP model on Group Knowledge

Then, we can extend the complete semantics from $\mathcal{L}_K(\Sigma)$ to $\mathcal{L}_{GK}(\Sigma)$.

Definition 3.12 (Complete Semantics for PWP model on Group Knowledge)**.** Given a PWP model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$, the complete semantics of the language $\mathcal{L}_{GK}(\Sigma)$ is defined as:

(g)	$(M, s) \models ES_G\alpha$	iff	for all $i \in G$, $(M, s) \models S_i\alpha$
(h)	$(M, s) \models EK_G\varphi$	iff	$(M, s) \models (\varphi \wedge ES_G\varphi)$
(i)	$(M, s) \models DS_Gv$	iff	$v \in \bigcup_{i\in G} f_i(s)$ or $\|D_v\| = 1$
(j)	$(M, s) \models DS_G\varphi$	iff	Let s' be $\bigcup_{i\in G} f_i(s)$, $\forall g \in \mathcal{S}_c$, $(M, g\langle s'\rangle) \models \varphi$ or $(M, g\langle s'\rangle) \models \neg\varphi$
(k)	$(M, s) \models DK_G\varphi$	iff	$(M, s) \models (\varphi \wedge DS_G\varphi)$
(l)	$(M, s) \models CS_Gv$	iff	$v \in cf(G, s)$ or $\|D_v\| = 1$
(m)	$(M, s) \models CS_G\varphi$	iff	Let s' be $cf(G, s)$, $\forall g \in \mathcal{S}_c$, $(M, g\langle s'\rangle) \models \varphi$ or $(M, g\langle s'\rangle) \models \neg\varphi$
(n)	$(M, s) \models CK_G\varphi$	iff	$(M, s) \models (\varphi \wedge CS_G\varphi)$

where items (a)-(f) are inherited from Definition 3.11, and α represents a variable v or a formula φ.

From the complete semantics above, the definition for uniform seeing and knowledge formulae are the same as in naïve semantics (Definition 3.8). This is because the evaluation of each individual $S_i\alpha$ handles possible values (Definition 3.11). For group seeing a variable in items (i) and (l), similarly as in individual semantics, if the set of all possible values of v only contains one value, then the group semantically sees v. Items (j) and (m) consider all possible values for the unobserved variables. And, if formula φ is consistent in all of them, which means φ holds in all possible worlds, or φ does not hold in all possible worlds, we have corresponding group seeing relation.

3.3 Ternary Semantics

In this section, we show how to implement this logic using a *ternary* logic semantics. This semantics aims to overcome the weaknesses of the naïve semantics, while providing a polynomial-time complexity for entailment. All preliminaries are given in Section 3.1, where a signature $\Sigma = (Agt, V, D, \mathcal{R})$ is defined in Definition 3.1 and a model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$ is defined using perspective functions f_i.

We take the concept from Levesque [137] for reasoning about knowledge bases with incomplete information, in which they used the Kleene [5]'s three-valued logic. The truth value in this logic are 1 (true), 0 (false), or $\frac{1}{2}$ (unknown), in which $\frac{1}{2}$ is interpreted as: unable to be proved as either true or false. In our semantics, proposition statements about variables that are not in a local state are given the value $\frac{1}{2}$. Like Levesque, we prove that the semantics are complete for a wide class of formulae based on *logically separable formula*.

Following the notation by Levesque, we define the semantics using a function $T \in (\mathcal{M} \times \mathcal{S}) \times \mathcal{L} \to \{0, 1, \frac{1}{2}\}$, which takes the knowledge base (a model and state pair (M, s) where s can be local or global) and a formula φ in the given language, and returns 1 for true, 0 for false, and $\frac{1}{2}$ for unknown. In order to systematically handle unknown seeing relation in a partial state, we request one variable in V acts as the agent's identifier, which means agents' identifiers are parts of the state. For example, in the BBL domain, this identifier could be the variable representing locations of each agent or the variables representing agents' facing directions.

In addition, following Levesque, our semantics also use the three-value truth table as shown in Table 3.1.

	$q = 1$	$q = \frac{1}{2}$	$q = 0$	$\neg p$
$p = 1$	1	$\frac{1}{2}$	0	0
$p = \frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
$p = 0$	0	0	0	1

TABLE 3.1: Three-valued truth table for two proposition p and q in Kleene's logic.

3.3.1 Ternary Semantics for PWP model on Knowledge

Now, we can define the ternary semantics for individual observation and knowledge (language $\mathcal{L}_K(\Sigma)$).

Definition 3.13. Given a PWP model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$ and the current state s, Function T for the language $\mathcal{L}_K(\Sigma)$ is defined as (omitted M for readability):

$$\begin{array}{llcl}
\text{(a)} & T[s, r(V_r)] & = & 1 \text{ if } \pi(s, r(V_r)) = true \\
 & & & 0 \text{ if } \pi(s, r(V_r)) = false \\
 & & & \frac{1}{2} \text{ otherwise} \\
\text{(b)} & T[s, \phi \wedge \psi] & = & \min(T[s, \phi], T[s, \psi]) \\
\text{(c)} & T[s, \neg\varphi] & = & 1 - T[s, \varphi] \\
\text{(d)} & T[s, S_i v] & = & \frac{1}{2} \text{ if } i \notin s \text{ or } v \notin s \\
 & & & 0 \text{ if } v \notin f_i(s) \\
 & & & 1 \text{ otherwise} \\
\text{(e)} & T[s, S_i \varphi] & = & \frac{1}{2} \text{ if } T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2} \text{ or } i \notin s \\
 & & & 0 \text{ if } T[f_i(s), \varphi] = T[f_i(s), \neg\varphi] = \frac{1}{2} \\
 & & & 1 \text{ otherwise} \\
\text{(f)} & T[s, K_i \varphi] & = & T[s, \varphi \wedge S_i \varphi]
\end{array}$$

The definitions of (d) and (e) deserve some discussion. For (d), we cannot reason about whether agent i sees variable v or not if at least one of the following holds: v is not visible in the current state, or the agent i is not visible in the current state. In both cases, $T[s, S_i v]$ is $\frac{1}{2}$. Otherwise, $T[s, S_i v]$ is 1 or 0 depending on whether v is in i's perspective $f_i(s)$ or not respectively.

As for (e), $T[s, S_i \varphi] = \frac{1}{2}$ in a local state s if and only if not both φ's visibility and agent i's observability can be evaluated. In short, we cannot prove that i sees the value of φ if we cannot prove φ ourselves; or alternatively, we cannot see if i sees φ if the agent itself cannot reason about i's visibility. If we reflect on the definition of $(M, s) \vDash S_i \varphi$, we note that any evaluation of $S_i \varphi$ is done in a global state that is 'anchored' by the 'for all $g \in \mathcal{S}_c$' in the complete semantics. This first part of the definition handles this for local states.

The second part of the definition says that $T[s, S_i\varphi] = 0$ when $T[f_i(s), \varphi] = T[f_i(s), \neg\varphi] = \frac{1}{2}$. What this means is that $S_i\varphi$ is false when neither φ nor $\neg\varphi$ can be proved in $f_i(s)$, because one or more variables in φ are not visible. In fact, we need only test one of these: if φ cannot be proved ($T[s, \varphi] = \frac{1}{2}$), then by definition, $T[s, \neg\varphi] = 1 - T[s, \varphi] = \frac{1}{2}$ as well. Note that in both the first and second parts, not all variables in φ need to be visible for φ or $\neg\varphi$ to be proved. For example, $S_i(v = 1 \vee u = 2)$ where $f_i(s) = [u = 2]$. Even though v is not visible, the truth value of $(v = 1 \vee u = 2)$ can be seen because $u = 2$ can be proved.

Finally, the third part of the definition says that $T[s, S_i\varphi] = 1$ if neither of the first two cases holds. So, if either φ or $\neg\varphi$ can be proved in state $f_i(s)$ (one of them returns 0 or 1), then φ can be seen, as in the complete semantics; otherwise, it cannot be seen.

Definition 3.14. (Soundness and completeness, adapted from Levesque [137]) Consider a function $h : (\mathcal{M} \times \mathcal{S}) \times \mathcal{L} \rightarrow \{0, 1, \frac{1}{2}\}$. Then:

- h is *sound* iff for every $M \in \mathcal{M}, s \in \mathcal{S}$, and $\varphi \in \mathcal{L}$, if $h[(M, s), \varphi] = 1$ then $(M, s) \models \phi$ and if $h[(M, s), \varphi] = 0$ then $(M, s) \models \neg\phi$;
- h is *complete* iff for every $M \in \mathcal{M}, s \in \mathcal{S}$, and $\varphi \in \mathcal{L}$, if $(M, s) \models \phi$ then $h[(M, s), \varphi] = 1$ and if $(M, s) \not\models \phi$ then $h[(M, s), \varphi] = 0$.

Clearly the function T is incomplete compared to the complete semantics, because it returns $\frac{1}{2}$ for some queries. However, in the remainder of this section, we show that this logic is sound, and we characterise precisely when the logic is complete.

First, we introduce the following lemma.

Lemma 3.15. *Given a formula $\varphi \in \mathcal{L}$, then:*

- *if $T[s, \varphi] = \frac{1}{2}$, then there exists a global state $g \in \mathcal{S}_c$, such that $T[g\langle s\rangle, \varphi] \neq \frac{1}{2}$;*
- *if $T[s, \varphi] = 1$, then for all global states $g \in \mathcal{S}_c$, $T[g\langle s\rangle, \varphi] = 1$; and*
- *if $T[s, \varphi] = 0$, then for all global states $g \in \mathcal{S}_c$, $T[g\langle s\rangle, \varphi] = 0$.*

Effectively, this lemma means that T returns 0 or 1 for any global state. If T cannot prove φ is true or false, then it must be due to reference to a variable that is not visible in some partial state, as this is the only way that $\frac{1}{2}$ is introduced by T. By 'completing' the state, we make φ (or $\neg\varphi$) provable. The latter two propositions state that, once φ is proved, even in a partial state, adding more information cannot change the outcome.

Theorem 3.16. *(Soundness of T). Let $M \in \mathcal{M}$ be a model, $s \in \mathcal{S}$ be a (local or global) state, and $\varphi \in \mathcal{L}$ a formula. If $T[(M,s),\varphi] = 1$ then $(M,s) \vDash \varphi$, and if $T[(M,s),\varphi] = 0$ then $(M,s) \nvDash \varphi$.*

Proof. We prove this inductively on the structure of φ.

Case (a): The case of $r(t_1,\ldots,t_k)$ is straightforward as the semantics of T and $\vDash$ are both defined using π. The only case where they disagree is then $T[s, r(t_1,\ldots,t_k)] = \frac{1}{2}$, which can only happen when s is a local state.

Case (b): Assume $T[s, \varphi \wedge \psi] = 1$. Therefore, $T[s,\varphi] = 1$ and $T[s,\psi] = 1$ from the definition of T. By induction, $(M,s) \vDash \varphi$ and $(M,s) \vDash \psi$. Therefore, from the definition of $\vDash$, we have that $(M,s) \vDash \varphi \wedge \psi$.

Now, assume $T[s,\varphi \wedge \psi] = 0$. Therefore, $T[s,\varphi] = 0$ or $T[s,\psi] = 0$ from the definition of T. By induction, $(M,s) \nvDash \varphi$ or $(M,s) \nvDash \psi$. Therefore, from the definition of $\vDash$, we have that $(M,s) \nvDash \varphi \wedge \psi$. This holds even if either $T[s,\varphi] = \frac{1}{2}$ or $T[s,\psi] = \frac{1}{2}$, and the other is 0. That is, provided that one of φ or ψ evaluates to 0, we know that $\varphi \wedge \psi$ evaluates to 0 irrelevant of the other.

Case (c): Assume $T[s,\neg\varphi] = 1$. Therefore, $T[s,\varphi] = 0$ from the definition of T. By induction, $(M,s) \nvDash \varphi$ and therefore from the definition of $\vDash$ we have that $(M,s) \vDash \neg\varphi$. The case for $T = 0$ is just the reverse.

Case (d): The definitions of $S_i v$ for T is follows the same definition in $\vDash$. The only case they disagree is when $T[s, S_i v] = \frac{1}{2}$, which can only happen in a local state.

Case (e): Assume $T[s, S_i\varphi] = 1$. Therefore, from the definition of T, we have that $T[s,\varphi] \in \{0,1\}$ (recall that $T[s,\neg\varphi] = 1 - T[s,\varphi]$ by definition), and $T[f_i(s),\varphi] \in \{0,1\}$ (so $T[f_i(s),\neg\varphi] = 1 - T[f_i(s),\varphi]$). From Lemma 3.15, this implies that for all $g \in \mathcal{S}_c$, $T[g\langle f_i(s)\rangle, \varphi] \in \{0,1\}$ or for all $g \in \mathcal{S}_c$, $T[g\langle f_i(s)\rangle, \neg\varphi] \in \{0,1\}$. By induction, this means that for all $g \in \mathcal{S}_c$, either $(M, g\langle f_i(s)\rangle) \vDash \varphi$ or $(M, g\langle f_i(s)\rangle) \vDash \neg\varphi$. Therefore, from the definition of $\vDash$, we have that $(M,s) \vDash S_i\varphi$.

Now, assume $T[s, S_i\varphi] = 0$. Therefore, from the definition of T, we have that $T[f_i(s), \varphi] = T[f_i(s), \neg\varphi] = \frac{1}{2}$. From Lemma 3.15, it must be that there exists $g \in \mathcal{S}_c$, such that $T[g\langle f_i(s)\rangle, \varphi] \in \{0, 1\}$ and there exists $g \in \mathcal{S}_c$, such that $T[g\langle f_i(s)\rangle, \neg\varphi] \in \{0, 1\}$. By induction, this means that there exists $g \in \mathcal{S}_c$, such that $(M, g\langle f_i(s)\rangle) \vDash \varphi$ and there exists $g \in \mathcal{S}_c$, $(M, g\langle f_i(s)\rangle) \vDash \neg\varphi$. Therefore, from the definition of $\vDash$, we have that $(M, s) \nvDash S_i\varphi$.

□

Next, we characterise when the logic is complete. To show this, we first introduce the concept of logical separability.

Definition 3.17. (Logical separability) Adapted from Levesque [137], a set of formulae Γ is *logically separable* iff for every satisfiable set of literals L, if $L \cup \Gamma$ is unsatisfiable, then $L \cup \{\varphi\}$ is unsatisfiable for some literal $\varphi \in \Gamma$.

This property captures whether there are any joint logical relations hidden in a set of formulae. Intuitively, given a logically-separable set of formulae, we cannot infer anything new by combining the formula in that set than we can from those items individually.

A *contradiction* is a simple example of a non-logically-separable formula. For example, let Γ be $\{p, \neg p\}$, and L be a singleton set containing any proposition q other than p or $\neg p$.

Clearly, both $\{p, q\}$ and $\{q, \neg p\}$ are satisfiable, which means Γ is not logically separable.

Definition 3.18. (Normal form $\mathcal{NF}$, adapted from [138]) We define the normal form $\mathcal{NF} \subseteq \mathcal{L}$ as the smallest set of formulae where each formula $\varphi \in \mathcal{L}$ adheres to the following grammar:

$$\varphi ::= r(t_1, \ldots, t_k) \mid \neg\varphi \mid \varphi \land \varphi' \mid S_i v \mid S_i\psi,$$
$$\psi ::= r(t_1, \ldots, t_k) \mid \neg\psi \mid \psi \land \psi',$$

where the set $\{\varphi, \varphi'\}$ is logically separable. This represents a normal form in which non-separable formulae are only permitted outside of S_i operators, and S_i operators cannot be nested. For any query, such as $S_i\varphi$ in $\mathcal{NF}$, φ must be non-separable.

Theorem 3.19. *(Completeness of T). Let $M \in \mathcal{M}$ be a model, $s \in \mathcal{S}$ be a (local or global) state, and $\varphi \in \mathcal{NF}$. Then, if $(M, s) \vDash \varphi$ then $T[(M, s), \varphi] = 1$, and if $(M, s) \nvDash \varphi$ then $T[(M, s), \varphi] = 0$.*

Proof. We prove this inductively on the structure of φ.

Case (a): As with soundness, the case of $r(t_1, \ldots, t_k)$ is straightforward as the semantics of T and $\vDash$ are both defined using π, and they disagree only when $T[s, r(t_1, \ldots, t_k)] = \frac{1}{2}$.

Case (b): Assume that $(M, s) \vDash \varphi \wedge \psi$. From the definition of $\vDash$, we have that $(M, s) \vDash \varphi$ and $(M, s) \vDash \psi$. By induction and that $\varphi \wedge \psi \in \mathcal{NF}$, we have that $T[s, \varphi] = 1$ and $T[s, \psi] = 1$. Therefore, from the definition of T, we have that $T[s, \varphi \wedge \psi] = 1$.

Now, assume that $(M, s) \nvDash \varphi \wedge \psi$. From the definition of $\vDash$, we have that $(M, s) \nvDash \varphi$ or $(M, s) \nvDash \psi$. By induction and that $\varphi \wedge \psi \in \mathcal{NF}$, we have that $T[s, \varphi] = 0$ or $T[s, \psi] = 0$. There, from the definition of T, we have that $T[s, \varphi \wedge \psi] = 0$.

If $\varphi \wedge \psi \notin \mathcal{NF}$, then the completeness does not hold because there are cases when, for example, $(M, s) \nvDash \varphi$ but $T[s, \varphi] = \frac{1}{2}$; for example, if $\varphi \equiv p \wedge \neg p$, but p is not visible in s.

Case (c): Assume $(M, s) \vDash \neg\varphi$. From the definition of $\vDash$, we have that $(M, s) \nvDash \varphi$. By induction, this means that $T[s, \varphi] = 0$, and therefore from the definition of T, we have that $T[s, \neg\varphi] = 1$. The case for $(M, s) \nvDash \neg\varphi$ is just the reverse.

Case (d): Similar to soundness, the definitions of $S_i v$ for T is follows the same definition in $\vDash$. The only case they disagree is when $T[s, S_i v] = \frac{1}{2}$.

Case (e): Assume that $(M, s) \vDash S_i\varphi$. Note that $\varphi \in \mathcal{NF}$, therefore it must be that s is global for the case S_i. From the definition of $\vDash$, we have that either for all $g \in \mathcal{S}_c$, $(M, g\langle f_i(s)\rangle) \vDash \varphi$ or for all $g \in \mathcal{S}_c$, $(M, g\langle f_i(s)\rangle) \vDash \neg\varphi$. By induction and $\varphi \in \mathcal{NF}$, we have that for all $g \in \mathcal{S}_c$, $T[g\langle f_i(s)\rangle, \varphi] = 1$ or for all $g \in \mathcal{S}_c$, $T[g\langle f_i(s)\rangle, \neg\varphi] = 1$. If one of these two expressions hold for all $g \in \mathcal{S}_c$, then they must also hold for all $g\langle s\rangle$ because $f_i(s) \subseteq s$. Therefore, either $T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2}$, in which case $T[s, S_i\varphi] = \frac{1}{2}$; or $T[s, \varphi] \in \{0, 1\}$, in which case $T[s, S_i\varphi] = 1$. In this first instance, if $T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2}$, then s must be a partial state, in which case, $S_i\varphi$ must be occurring within another S_j operator, so $S_i\varphi \notin \mathcal{NF}$.

Now, assume that $(M, s) \not\vDash S_i\varphi$. From the definition of $\vDash$, we have that there exists $g \in \mathcal{S}_c$, such that $(M, g\langle f_i(s)\rangle) \vDash \varphi$ and there exists $g \in \mathcal{S}_c$, such that $(M, g\langle f_i(s)\rangle) \vDash \neg\varphi$. By induction and $\varphi \in \mathcal{NF}$, we have that there exists $g \in \mathcal{S}_c$, such that $T[g\langle f_i(s)\rangle, \varphi] = 1$ and there exists $g \in \mathcal{S}_c$, such that $T[g\langle f_i(s)\rangle, \neg\varphi] = 1$. From Lemma 3.15 and $\varphi \in \mathcal{NF}$, it must be that $T[f_i(s), \varphi] = T[f_i(s), \neg\varphi] = \frac{1}{2}$. Therefore, from the definition of T, we have that $T[s, S_i\varphi] = 0$; therefore case (e) and the theorem hold. □

Definition 3.20. (Normal form $\mathcal{NF}^+$) We define the normal form $\mathcal{NF}^+ \subseteq \mathcal{L}$ as the smallest set of formula where each formula $\varphi \in \mathcal{L}$ adheres to the following grammar:

$$\varphi ::= r(t_1, \ldots, t_k) \mid \neg\varphi \mid \varphi \wedge \varphi' \mid S_i v \mid S_i\psi,$$
$$\psi ::= r(t_1, \ldots, t_k) \mid \neg\psi \mid \psi \wedge \psi' \mid S_i v \mid S_i\psi,$$

where $\{\psi, \psi'\}$ is logically separable. This is the same as $\mathcal{NF}$, except that seeing operators can be nested.

Theorem 3.21. *(Soundness and completeness of T in global states). Let $M \in \mathcal{M}$ be a model, $g \in \mathcal{S}_c$ be a global state, and $\varphi \in \mathcal{L}$ be a formula in $\mathcal{NF}^+$. Then, if $T[(M, g), \varphi] = 1$ then $(M, s) \vDash \varphi$, and if $T[(M, g), \varphi] = 0$ then $(M, s) \not\vDash \varphi$; and if $(M, s) \vDash \varphi$ then $T[(M, s), \varphi] = 1$, and if $(M, s) \not\vDash \varphi$ then $T[(M, s), \varphi] = 0$.*

Proof. This is a small extension to the proofs of Theorems 3.16 and 3.19, which prove the case for local and global states in $\mathcal{NF}$. Thus, we just need to prove the case for nested seeing operators, which is the only difference between $\mathcal{NF}$ and $\mathcal{NF}^+$. The proof for Theorem 3.16 already holds for this. However, not in the proof for completeness where if $T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2}$, then $T[s, S_i\varphi] = \frac{1}{2}$, but that this can only occur in a local state, which implies $S_i\varphi$ must be within another seeing operator. For the global case, however, we have that $T[g, \varphi] = T[g, \neg\varphi] = \frac{1}{2}$. From the definition of T, this can only occur if φ refers to a variable not in g, which is not possible because g is global. Therefore, the theorem holds. □

Finally, we discuss the potential implementation of our ternary semantics. Classical planning languages do not support ternary propositional logic. However, as proven in Theorem 3.21, our semantics is complete and sound for global states. Therefore, for a global state, our semantics always returns true or false; and never returns $\frac{1}{2}$. This admits a wide class of formulae suitable for many planning tasks, which are a superset of admissible formulae in planning languages such as PDDL. That is, epistemic relations are modelled as propositions in the planning language, and their truth values are reasoned externally with our ternary semantics (including value $\frac{1}{2}$ when nested relations are reasoned.) Therefore, the completeness and soundness of solving a planning task is equivalent with the completeness and soundness of our ternary semantics.

3.3.2 Ternary Semantics for PWP model on Group Knowledge

Then, extending the ternary semantics to handle the group operators (language $\mathcal{L}_{GK}(\Sigma)$), we defined T as follows.

Definition 3.22. Given a PWP model $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$, G be a group of agents, and the current state s, function T for the language $\mathcal{L}_{GK}(\Sigma)$ is defined:

$$
\begin{array}{llcl}
\text{(g)} & T[s, ES_G\alpha] & = & min(\{T[s, S_i\alpha] \mid i \in G\}) \\
\text{(h)} & T[s, EK_G\varphi] & = & T[s, \varphi \wedge ES_i\varphi] \\
\text{(i)} & T[s, DS_Gv] & = & \frac{1}{2} \text{ if } v \notin s \text{ or } \forall i \in G, i \notin s \\
 & & & 0 \text{ if } v \notin \bigcup_{i\in G} O_i(s) \\
 & & & 1 \text{ otherwise} \\
\text{(j)} & T[s, DS_G\varphi] & = & \frac{1}{2} \text{ if } T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2} \text{ or } \forall i \in G, i \notin s \\
 & & & 0 \text{ if } T[\bigcup_{i\in G} O_i(s), \varphi] = T[\bigcup_{i\in G} O_i(s), \neg\varphi] = \frac{1}{2} \\
 & & & 1 \text{ otherwise} \\
\text{(k)} & T[s, DK_G\varphi] & = & T[s, \varphi \wedge DS_G\varphi] \\
\text{(i)} & T[s, CS_Gv] & = & \frac{1}{2} \text{ if } v \notin s \text{ or } \exists i \in G, i \notin s \\
 & & & 0 \text{ if } v \notin cf(G, s) \\
 & & & 1 \text{ otherwise} \\
\text{(j)} & T[s, CS_G\varphi] & = & \frac{1}{2} \text{ if } T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2} \text{ or } \exists i \in G, i \notin s \\
 & & & 0 \text{ if } T[cf(G, s), \varphi] = T[cf(G, s), \neg\varphi] = \frac{1}{2} \\
 & & & 1 \text{ otherwise} \\
\text{(k)} & T[s, CK_G\varphi] & = & T[s, \varphi \wedge CS_G\varphi]
\end{array}
$$

where the model M is omitted in representation for readability; α can be any formula φ from the language or any variable $v \in V$; and cf is the common perspective function defined in Definition 3.7.

The items (a)-(f) are inherited from the ternary semantics in Definition 3.13.

Theorem 3.23. *(Soundness and completeness of T for group operators) Let $M \in \mathcal{M}$ be a model, $g \in \mathcal{S}_c$ be global state, and $\varphi \in \mathcal{L}$ be a formula in $\mathcal{NF}^+$. Then, if $T[(M,g),\varphi] = 1$ then $(M,s) \vDash \varphi$, and if $T[(M,g),\varphi] = 0$ then $(M,s) \nvDash \varphi$; and if $(M,s) \vDash \varphi$ then $T[(M,s),\varphi] = 1$, and if $(M,s) \nvDash \varphi$ then $T[(M,s),\varphi] = 0$.*

Proof. We prove this inductively on the structure of φ.

Soundness, case (f): Assume $T[s, ES_G\alpha] = 1$. From the definition of T, the minimum of all $T[s, S_i\alpha]$ for $i \in G$ is 1, which means that $T[s, S_i\alpha] = 1$ for all $i \in G$. By induction, this means that for all $i \in G$, $(M,s) \vDash S_i\alpha$. Therefore, from the definition $\vDash$, we have that $(M,s) \vDash ES_G\alpha$.

Now, assume $T[s, ES_G\alpha] = 0$. This means that for some $i \in G$, $T[s, S_i\alpha] = 0$. By induction, we have that $(M,s) \nvDash S_i\alpha$ for some $i \in G$. Therefore, from the definition of $\vDash$, we have that $(M,s) \nvDash ES_G\alpha$.

Completeness, case (f): Assume $(M,s) \vDash ES_G\alpha$. From the definition of $\vDash$, this means that for all $i \in G$, $(M,s) \vDash S_i\alpha$. By induction, we have that for all $i \in G$, $T[s, S_i\alpha] = 1$. Clearly, the minimum of $T[s, S_i\alpha]$ for any $i \in G$ is 1, therefore, we have that $T[s, ES_G\alpha] = 1$.

Now, assume $(M,s) \nvDash ES_G\alpha$. From the definition of $\vDash$, this means that there exists an $i \in G$, such that $(M,s) \nvDash S_i\alpha$. By induction, this means that there exists an $i \in G$, such that $T[s, S_i\alpha] = 0$. If $T[s, S_i\alpha] = 0$ for at least one $i \in G$, then the minimum $T[s, S_i\alpha]$ must be 0, therefore, we have that $T[s, ES_G\alpha] = 0$.

Cases (g)-(j) are all straightforward mappings from the proofs of Theorems 3.16, 3.19, and 3.21. The unknown relation which is caused by agent now becomes none of agent's observability (g, h) and all agents' observability (i, j) respectively. Besides the unknown relation, the structure of proofs is identical, with just the replacement of $f_i(s)$ with $\bigcup_{i \in G} f_i(s)$ and $cf(G,s)$ respectively. $\square$

3.4 Representing Existing Epistemic Logic Models using Perspective Function

In this section, we show the expressiveness of our logic by using it to represent several well-known epistemic logics.

3.4.1 Kripke Semantics

We can simulate Kripke semantics as follows. The set of variables $V = W$, where W is the set of worlds in a Kripke model. Therefore, a state s represents the set of possible worlds. The domain of variables is not relevant. The perspective function $f_i(s)$ returns the set of possible worlds according to agent i, so it is just equivalent to $\mathcal{K}_i$. The evaluation function $\pi(s)(r(t_1, \ldots, t_k))$ is then just defined as being true if and only if $\forall w \in s$, $r(t_1, \ldots, t_k))$ holds in the world corresponding to w.

The downside of this is that while the complexity is still polynomial in the number of states, the number of states is exponentially larger than the set of propositions (or variables) in the underlying problem, which is as difficult to solve as if using Kripke semantics. Instead, using a domain-specific representation would often be more suitable.

The reader may have noted that if s is a global state, then s only contains one world and if the perspective function $f_i(s)$ returns the set of possible worlds according to agent i, then the property $f_i(s) \subseteq s$ on perspective functions is violated. A trick around this is to use the set of *impossible* worlds in the state representation. That is, a global state is $s = \{w\} \cup \neg W$, where the w represent the one actual world and $\neg W$ represents the set of all impossible worlds [4].

Using the same example from Figure 2.5, where $V = \{x\}$ and $D_x = \{1, 2, 3\}$. For simplicity, we use x_1 to represent a world that is equivalent to the state $\{x = 1\}$ in our original model. The impossible worlds are represented by $\{\neg x_1, \neg x_2, \neg x_3\}$. Using x_1 as the initial world as an example, the initial state is $s = \{x_1, \neg x_1, \neg x_2, \neg x_3\}$. The logic works as follows:

- **If agent a has no knowledge of x:** $f_a(s) = \{\neg x_1, \neg x_2, \neg x_3\}$.

[4]It is just the possible worlds with an impossible indicator each.

- **If agent a knows exact x:** $f_a(s) = \{\neg x_2, \neg x_3\}$, which means the only world a thinks is possible is x_1.
- **If agent a knows something about x:** using $x < 3$ as example, $f_a(s) = \{\neg x_3\}$.
- **If agent a knows $x < 3$ and b knows $x > 1$:** $f_a(s) = \{\neg x_3\}$ and $f_b(s) = \{\neg x_1\}$, while their distributed knowledge would still be the union $\{\neg x_2, \neg x_3\}$.

The evaluation function on the impossible worlds (local perspective) is defined as $\forall \neg w \notin s$, $r(V_r)$ holds in w (note the $\in$ replaced by $\notin$ and the corresponding relation between $\neg w$ and w).

3.4.2 Proper Epistemic Knowledge Bases (PEKBs) and Cooper et al. [1]'s Seeing Logic

PEKBs [87, 88, 138] and Cooper et al. [1]'s seeing logic are closely related, and our logic can represent both using the same representation as Cooper et al.'s.

In this representation, V is the set of all modal literals up to a maximum depth of k. If $k = 2$, there is just one proposition p, and two agents i and j, then $V = \{p, S_ip, S_jp, S_iS_ip, S_iS_jp, S_jS_ip, S_jS_jp\}$. A state s represents the set of propositions that are true. Since the domains of all variables are not relevant (it can be any domain that contains more than one value), we only use variable names in states in this example for simplicity. The perspective function $f_i(s) = \{\alpha, S_i\alpha \mid S_i\alpha \in s\}$. The reader might find this is less intuitive and might lead to unwanted common knowledge between agents in domains like *the Byzantine Generals* domain. We ensure it is not the case and provide an example in Appendix A. The evaluation function $\pi(s)(p)$ is true if and only if $p \in \text{dom}(s)$.

For Cooper et al. [127]'s $JS\alpha$ operator, which means that all agents jointly see literal α (the operator CS_G in our logic), we can use the same encoding by adding a variable $JS\alpha$ for each literal α. We then define $f_i(s) = \{\alpha, S_i\alpha \mid S_i\alpha \in s\} \cup \{\alpha, JS_i\alpha \mid JS\alpha \in s\}$.

A compact way to represent this logic is to have one variable ip for each proposition p. The domain of each variable is a bit vector that represents each value in $\{p, S_ip, S_jp, S_iS_ip, S_iS_jp, S_jS_ip, S_jS_jp, JSp\}$. An example can be found in Section 3.6.4.

In their PEKB-based planner, Muise et al. [87] introduce the concept of 'always known' propositions, which are propositions that are common knowledge in a problem. This reduces the size of the compiled classical planning problem because these propositions do not have to be expanded.

Common knowledge can be represented in the PWP approach by ensuring that any commonly-known variable is included in all agents' perspective functions; or at least, all agents who are part of the group. For any agent $a \in G$, where G is the group who commonly know some proposition about variables $v_1, \ldots v_n$, we can define $f_a(s)$ as:

$$f_a(s) = \{v \mid a \triangleright v, v \in s\} \cup \{v_1, \ldots, v_n\}, \text{ where } a \triangleright v \text{ means } a \text{ sees } v.$$

Any propositions about the variables $v_1, \ldots v_n$ are commonly known by all agents in $a \in G$ because they are part of the fixed point for cf.

3.4.3 Big Brother Logic (BBL) by Gasquet et al. [2]

In BBL, the set of variables V is $x_i, y_i \in \mathbb{R}$, $dir_i \in U$ and $ang_i \in (0, 2\pi)$ for each agent i where U is the set of unit vectors for $\mathbb{R}^2$. These variables represent the Cartesian coordinates, the direction the agent is facing, and its angle of vision, respectively. The perspective function is defined as:

$$f_i(s) = \big\{v{=}s(v) \in s \mid i \triangleright v\big\} \cup \big\{v{=}s(v) \mid v \in \{x_j, y_j, ang_j\},\ j \in Agt\big\}, \text{ where:}$$

$i \triangleright v$ is defined as in Section 2.4.1 and e is the value of v in s. $i \triangleright v$ can be implemented using the following, assuming that (x_j, y_j) represents the location of the target agent j:

$$\begin{gathered}\Big(|\arctan(\tfrac{|s(y_i)-s(y_j)|}{s(x_i)-s(x_j)}) - s(dir_i)| \leq \tfrac{s(ang_i)}{2}\Big) \\ \vee \\ \Big(|\arctan(\tfrac{|s(y_i)-s(y_j)|}{s(x_i)-s(x_j)}) - s(dir_i)| \geq \tfrac{360^\circ - s(ang_i)}{2}\Big)\end{gathered} \tag{3.1}$$

Therefore, perspective function $f_i(s)$ takes all agents' locations, directions, and vision angles, and returns all the variables that belong to those agents that fall inside these regions. The set on the right side of the set union operator in the perspective function captures that the locations and angles (the angular range) of all agents are common knowledge, as in the original Big Brother Logic. Therefore, for all agents j, we have that $x_j, y_j, ang_j \in f_i(s)$, and therefore $x_j, y_j, ang_j \in f_i(s) \subseteq cf(G, s)$.

As we outline in the following section, in our planning framework, perspective functions are implemented using external functions in F-STRIPS. In our case, the external functions are implemented in C++. This brings flexibility, such as the ability to implement the expression in Equation 4.1. If it were possible to encode this function using propositions in classical planning, we assert that the resulting encoding would be difficult, error-prone, and hard for a reader to understand. However, implementing the above in C++ is straightforward for the modeler to implement and straightforward for a reader to understand.

3.5 Implementation

In this section, we define the problems that can be modelled by our PWP model, provide an encoding for a working planner, and show some examples. Two key aspects in planning are the planning language and solver (planner). The encoding we provided is a combination of F-STRIPS and PDDL (as dicussed in Section 2.1.3.3), and the planner used is the BFWS planner by Francès et al. [63]. Using this encoding with external functions allows us to decompose the planning task from the epistemic logic reasoning.

The intuition behind the PWP model is that the action model is specified using a planning language, and queries specified in epistemic logic are implemented as F-STRIPS *external functions*. External functions are functions that can be called from within an F-STRIPS model, but whose semantics are defined external to the PDDL model and can be implemented in languages outside of the planning language.

3.5.1 F-STRIPS Encoding

External functions are arbitrary functions that can be written in any language. Thus, verifying the correctness and termination of the external function is the task of the modeller. In our implementation, external functions are programmed in C++ for scalability and flexibility.

To show how to implement 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; $D = D_{v_1} \cup \ldots \cup D_{v_m}$ is the domain of variables; $\mathcal{O}$, $\mathcal{I}$ and $\mathcal{G}$ differ from their counterparts in F-STRIPS only by adding epistemic formulae in preconditions and goals, which will be interpreted in the following part of this section; and $\mathbb{F}$ are the set of external functions.

In the PWP approach, the main external functions are of the form $(@check\ ?v_1\ \ldots ?v_n\ ?q)$, where q is the epistemic relation and $v_1, \ldots, v_n$ are variables. These evaluate the truth value for q based on the given current state. For readability, we represent the validation of the epistemic relation by using $(@check\ ?q)$ only for the remainder of the thesis, omitting the variables. In the implementation, the modeller decides which variables are needed as the input to the external functions.

The core logic implemented in the external function formalizes a PWP model instance $M = (Agt, V, D, \pi, f_1, \ldots, f_k)$ (as defined in Definition 3.4). Then, it uses the inputs to construct the current state s and the formula needed to be evaluated φ. It applies the ternary semantics (Definition 3.22) based on M, s, and φ, and returns the ternary value as output.

There are two major ways to embed epistemic formulae in a planning problem: using the formulae as preconditions and conditions (on conditional effects) in operators $\mathcal{O}$; or using the formulae as epistemic goals in $\mathcal{G}$. Defining preconditions and goals with desirable epistemic formulae is straightforward. For example, in Figure 2.4, if we want "agent a_1 knows a_2 sees b_1" to be true, we could simply set the goal to be $K_{a_1} S_{a_2} b_1$.

An important part of the modelling is to represent the state with a planning language, and update it accordingly with each action taken. Particularly important is to update the state with information that is sufficient to determine what each agent sees, such as the position, direction, and angle in Big Brother Logic. For non-visual domains, using encodings similar to the PEKB encoding in Section 3.4.2 is possible as this is a general encoding. In this case, the update will need to ensure that the relevant seeing variables are updated correctly.

3.5.2 External Functions

External functions in F-STRIPS take variables as input, and return a result based on that input. This is the key aspect that allows us to separate epistemic reasoning from planning.

Recall from Section 3.1 that each agent i has a perspective function, $f_i : \mathcal{S} \rightarrow \mathcal{S}$, which takes a state and returns the local state as agent i's perspective. In most problems we have modeled, the perspective function is the same for all agents, but the framework allows each agent to have its own perspective function implementation.

Given f_i for each agent, our library of external functions has implementations for K_i and S_i, their group knowledge counterparts, and propositional logic operators. The modeller simply needs to provide the perspective functions for their domain, if a suitable one is not already present in our library.

As an example, consider the Big Brother Logic domain. Here, the state of the world includes the x-y coordinates of each agent, the direction they are facing, and the angle they can see. The perspective function in Equation 4.1 is implemented in C++. The semantics of the epistemic logic are implemented as external functions and remain the same regardless of the perspective function that is used.

When an epistemic formula needs to be evaluated, the planner calls the epistemic logic external function. In other words, the epistemic logic reasoning task is moved from the planner to the external functions. The underlying planner has no concept of epistemic logic and simply uses its search algorithm to find the goal. In addition, the external function allows us to deploy *lazy evaluation*. That is, instead of generating all truth values for all epistemic queries at the pre-compilation phase or storing entire knowledge structures, the planner evaluates them only when they appear in the searching process.

Despite having general representations such as PEKBs (in Section 3.4.2), in our experience, using a domain-specific perspective function results in shorter, more elegant models that are more straightforward to specify and verify. We give the external function for the BBL domain here, and several examples in the following section (Section 3.6).

Example 3.4. *External function for BBL domain is explained as follows.*

In Figure 2.4, 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 area, and for agent a_2, the "visible" world is only the yellow area. 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 $l = f_{a_1}(s)$ as the "global state", and inside that state, agent a_2's perspective is $f_{a_2}(l)$.

Specifically, assume the global state s in the BBL example contains all variables for $\{a_1, a_2, b_1, b_2, b_3, b_4\}$[5], such as locations, the directions agents are facing, and etc. Based on the current setup, we can implement f_i for any agent i with the Euclidean geometric calculation given in Equation 4.1. By applying this perspective function on all variables in the given state (could be a local state when evaluating nested perspective), we can filter out all unseen variables for the agent and get its perspective based on the given state.

Then, for any epistemic query φ, such as $S_{a_1}b_2$, the external function (@*check* ?φ) takes all variables $\{a_1, a_2, b_1, b_2, b_3, b_4\}$ and the query φ as input. By applying the above perspective function f_{a_1} on the given state, we can retrieve agent a_1's perspective $\{a_1, a_2, b_2, b_3, b_4\}$ over the current state. Since b_2 is in the perspective, the external function will return 1, which means (@*check* ?φ) will be evaluated as 1 (true) by the F-STRIPS planner. Let another query ψ be $S_{a_1}b_1$. Following the exact same approach, since b_1 is not in agent a_1's perspective, the external function will return 0 (false), so ψ is false.

3.5.3 Expressiveness

Now, we discuss the expressiveness of our planning framework. The intuitive idea about the agent's perspective model is based on what agents can see, as determined by applying the seeing rules (perspective function f) on the current state. The relation between $t = f_i(s)$ and s corresponds roughly to Kripke accessibility relations $(s, t) \in \mathcal{R}_i$. However, rather than generating a set of worlds that i considers possible, the perspective function

[5] $a_1, a_2, b_1, b_2, b_3, b_4$ here in the set are not variables. They are simplified representations of the group of variables that belong to that agent or that object, such as a_1 represents $x\_a_1, y\_a_1, dir\_a_1, ang\_a_1$.

only returns the one partial world that the agent is certain about. This advantage (one state, rather than multiple states) prevents the explosion in model size.

However, the reduced complexity loses information on "uncertain" variables. That is, variables that an agent has some information about, but not complete information. Theoretically, $f_i(s)$ is equivalent to $\bigcap_{t \in W, (s,t) \in \mathcal{R}_i} t$ from Kripke semantics. This eliminates disjunctive knowledge about variables; the only uncertainty being that an agent does not see a variable. For example, in the Muddy Children problem (Example 2.4), the knowledge is not only generated by what each child can see by the others' appearance, which is modeled straightforwardly using perspective functions, but also can be derived from the questions made by their teacher 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 teacher 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 for $m+1$ 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".

Despite this, we can still represent the muddy children problem in our logic. Instead of m representing the number of dirty children, 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 the 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 teacher for the kth time will result in the removal of m_k from the state until all the children only have m_m in their local state. This is similar to how Kripke semantics are encoded in Section 3.4.1 but also shows that being able to customize perspective functions to specific domains can be useful.

A comparison on expressiveness between our model and other approaches is given in Table 3.2. This table is similar to Table 2.1; the difference is some clarification we made. The explanation of the detailed differences is the same as in Section 2.4.2.5.

	Nested	Depth	*CK*	*DK*	Continuous Domains	Disjunctive Knowledge
PWP Model	Y	U	Y	Y	Y	Y/N
Muise et al. [87]	Y	B	N	N	N	N
Kominis and Geffner [115]	Y	B	N	N	N	Y
Huang et al. [7]	Y	U	I	N	N	Y
Le et al. [6]	Y	U	I	N	N	Y

TABLE 3.2: Expressiveness Comparison over Epistemic Planning Approaches, where CK and DK represent whether the model supports common and distributed knowledge. 'I' means this approach can handle common knowledge indirectly, such as modeling common knowledge by public announcement [6], or using a group of nested knowledge to approximate common knowledge [7]. For depth, 'U' means no bound on the depth of queries, while 'B' means there is a fixed bound.

3.6 Experiments & Results

Now, we evaluate our approach on several domains: Corridor [115], Grapevine [87], BBL [2], Social-media Network (SN) and Gossip [136]. Corridor, Grapevine, and Gossip are well-known epistemic planning problems, which we use to compare the actual performance of our PWP model against two state-of-the-art approaches in epistemic planning. BBL is a model of the Big Brother Logic in a two-dimensional continuous domain, which we use to demonstrate the expressiveness of PWP. The Social-media Network problem demonstrates group knowledge operators, modeling information sharing over a digital social network platform. PWP has an advantage in those epistemic planning problems where knowledge can be derived from the ontic states. We also evaluate PWP on problems in which agents can have 'memory' about knowledge, such as the canonical 'Gossip' domain.

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

3.6.1 Benchmarks

In this section, we briefly describe the corridor and grapevine problems, which are benchmark problems that we use to compare against Muise et al. [87]'s epistemic planner, which is currently the state-of-the-art in epistemic planning.

Corridor domain was originally presented by Kominis and Geffner [115]. It models selective communication among agents. The setup is several agents (3-7) located in a corridor of rooms, and there is a secret in one of the rooms. Only one agent is able to move between rooms, sense the secret, 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 then know the secret. The goals in this domain are to have some agents knowing the secret and other agents not knowing the secret. The perspective function is simply that a secret variable is 'visible' to an agent (which models it hearing the secret) if they are in the same room or adjacent rooms when the secret is shared.

Grapevine, proposed by Muise et al. [87], is a similar problem to Corridor. With only two rooms available for agents, the scenario makes sharing secrets while hiding from others more difficult. In this domain, each agent has their own secret. The agents can move between two rooms and share their own secret or others' secrets once they know them. Since there are only two rooms, the secret is only shared within the room. The emphasis of this domain is on sharing one's secret with others without being noticed. This is the same as in the Corridor domain, except we change the seeing rules so that an agent sees a variable if and only if they are in the same room when the secret is shared.

3.6.1.1 Encoding

Both the Corridor and Grapevine domains are modeled similarly to standard propositional planning problems. The only difference is that the locations for movable agents are modeled by functions (variables in the BWFS planner) rather than propositions, which increases the readability and flexibility for the external functions. The desired epistemic formulae are modeled by Boolean query 'indicators'. Each of the indicators is a Boolean variable that records the truth value for an epistemic formula which is in the format of a JSON string. For example, a query entry in the Grapevine domain '`{"query_info":{"id":"p1","query":"ck a,b sct_a:value:2"}}`' represents the common knowledge of agent a and b that the value of a's secret is 2 ($CK_{a,b}\ sct_a = 2$). This separates the epistemic language from F-STRIPS. The truth values for query indicators can be modified by conditional effects in actions, such as `shout` in Corridor and `share` in Grapevine. For example, in those actions, all query indicators are evaluated by calling external functions. We only update the corresponding indicators if the epistemic formulae hold in the current state. An example action `shout` is listed as below:

```
action  shout(x)
  prec  sct=1, loc(a)=x
  effs  (forall (?q - query) (when (= (@check ?q) 1) (assign (fact ?q) 1))
```

The conditional effects assign the truth value to each query `?q` to record its value. That is, for any positive epistemic relations, its query variable should be 1 when checking the goal state, while for any negative epistemic relations, its query variable should keep being 0. While this is somewhat inelegant, it would be straightforward to take any existing epistemic planning language and compile it into this format.

3.6.1.2 External Functions

The input of the **@check** function would be the location of each agent and the query itself. The agent's perspective function for Corridor and Grapevine is similar. The visibility of secrets for both domains depends on the location of the agent whose perspective is modeled. Therefore, both rules take the location of the speaking agent and the hearing agent, and return all variables whose locations are the same location (for the Grapevine domain); or the locations are the same or in adjacent rooms (for the Corridor domain). Given the function $loc(i)$ that returns the location of an agent using the rooms as a sequence of numbers, we can define this formally as follows:

$$\text{Corridor domain:} \quad f_i(s) = \{v' \mid v' \in s \ \wedge \ |loc(v') - loc(i)| \leq 1\}$$
$$\text{Grapevine domain:} \quad f_i(s) = \{v' \mid v' \in s \ \wedge \ loc(v') = loc(i)\}.$$

3.6.1.3 Results

To evaluate the computational performance of PWP, we compare it to Muise et al. [87]'s PDKB planner. Their planner has been used to compare on the Corridor and Grapevine domains against many other solutions [6, 115]. From their results and results from Huang et al. [7] and Le et al. [6], it is fair to say that PDKB is a state-of-the-art planner. Although the PDKB approach is for belief, rather than knowledge, it can still be used as a suitable baseline for problems in which the agent's belief cannot be incorrect, and thus can simulate knowledge for these domains. In addition, to test how the performance is influenced by the problem, we create new problems that varied some of the parameters, such as the number of agents, the number of goal conditions, and also the depth of epistemic relations.

The PDKB planner converts an epistemic planning problem into a classical planning problem, which generates a significant number of propositions when the depth of epistemic relations or the number of agents increases. We tried to submit the converted classical planning problems to the same planner that is used by our PWP model, the BFWS(R_0) planner, to maintain a fair comparison. However, in this domain, there was not a significant performance difference with respect to the original planner used by Muise et al. [87], the *FF* planner.

We ran the problems with both planners on a Linux machine with 8 CPUs (Intel Core i7-7700K CPU @ 4.20GHz $\times$ 8) and 16 gigabytes of memory. We measured the number of atoms (fluents) and the number of nodes generated during the search to compare the size of the same problem modeled by different methods. We also measured the total time for both planners for solving the problems, and the time they take for reasoning about the epistemic relations, which corresponds to the time taken to call external functions for our solution (during planning), and the time it takes to convert the epistemic planning problems into classical planning problems in the PDKB solution (before planning).

Table 3.3 shows the results for the Corridor and Grapevine problems, in which $|Agt|$ 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 that there is no result within a 10-minute time limit. In the Grapevine tests, to eliminate any influence from the different lengths of the plan on the computation time, we increase the depth of the goal while keeping the solution the same. Therefore, with the same number of agents and size of the goal condition, the problems have the same solution. Evidence of this is that the number of search nodes generated and the number of external function calls remains static across problems.

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 $O(2^{|Agt|\cdot D})$, in which $|Agt|$ is the number of agents and D is the maximum depth of any modal formula in the

Parameters			PWP					PDKB			
$\|Agt\|$	d	$\|\mathcal{G}\|$	$\|Atom\|$	$\|Gen\|$	$\|Calls\|$	TIME(s) Calls	TIME(s) Total	$\|Atom\|$	$\|Gen\|$	TIME(s) Compile	TIME(s) Total
Corridor											
3	1	2	15	8	24	0.001	0.002	54	21	0.148	0.180
7	1	2	15	15	72	0.002	0.004	70	21	0.186	0.195
3	3	2	15	8	24	0.002	0.004	558	21	0.635	0.693
6	3	2	15	15	72	0.006	0.007	3810	21	5.732	6.324
7	3	2	15	15	72	0.007	0.008	5950	21	9.990	11.130
8	3	2	15	15	72	0.008	0.009	8778	21	14.140	15.680
3	4	2	15	8	24	0.003	0.004	3150	21	3.354	3.752
3	5	2	15	8	24	0.002	0.003	18702	21	25.690	29.540
Grapevine											
4	1	4	358	23	144	0.003	0.005	96	11	0.428	0.468
4	2	4	358	23	144	0.005	0.007	608	11	2.885	3.178
4	1	8	370	270	2144	0.044	0.048	96	529	0.381	0.455
4	2	8	370	270	2144	0.077	0.079	608	1234	3.450	4.409
4	3	8	370	270	2144	0.136	0.138	4704	14	28.660	30.720
8	1	2	600	18	24	0.001	0.006	312	5	3.025	3.321
8	2	2	600	18	24	0.001	0.007	4408	5	54.350	58.800
8	1	4	606	43	144	0.005	0.011	312	11	2.546	2.840
8	2	4	606	43	144	0.009	0.014	4408	11	55.330	59.780
8	1	8	618	1068	4448	0.158	0.171	312	2002	2.519	3.752
8	2	8	618	1068	4448	0.257	0.269	4408	4371	54.900	228.100
8	3	8	618	1068	4448	0.460	0.466	–	–	–	–

TABLE 3.3: Results for the Corridor and Grapevine domain.

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. Using PWP, 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. Since the F-STRIPS planner we use checks each query in goal conditions at the generation of each node in the search ($O(|Gen|)$), the time complexity for epistemic logic reasoning is in $O(|Gen| \cdot |G| \cdot |Agt| \cdot |V|^2)$, in which G is the set of goals and V is the size of the state[6].

Although the search part of the problem is still NP-hard, the empirical computational cost of epistemic reasoning is significantly lower than the compilation in the PDKB approach in most of the test cases. In fact, using our encoding, none of the problems

[6]In the worst case, we need to check common knowledge on a state, there are at most $|V|$ (maximum size of the state) iterations, and each iteration contains $|Agt|$ amount of set operations on the global state or a local state (maximum $|V|$).

exceed even half a second, while for the PDKB approach, many do, some running for close to a minute.

3.6.2 Big Brother Logic

Big Brother Logic (BBL) is a problem first discussed by Gasquet et al. [2]. The basic environment is in 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.

3.6.2.1 Encoding

Figure 3.2 visualizes the problem setup. 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 a 90° range. Camera a_1 is facing north-east, while camera a_2 is facing south-west. There are three objects with values $o_1 = e_1$, $o_2 = e_2$, and $o_3 = e_3$, 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. The locations of these stationary objects and agents are common knowledge.

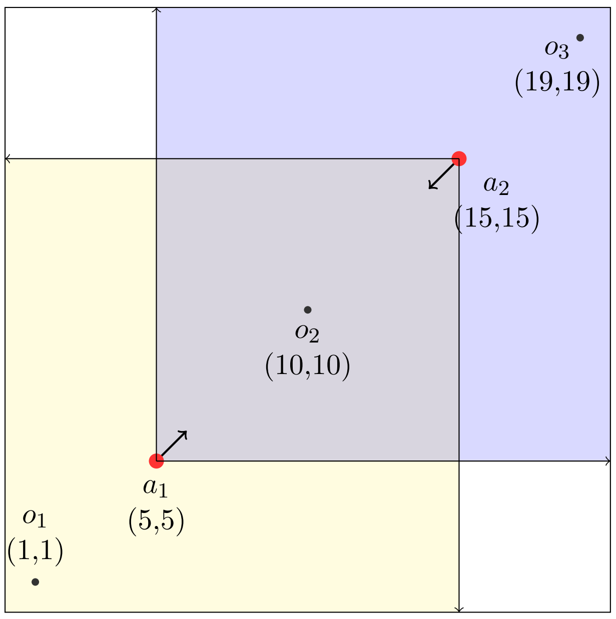


FIGURE 3.2: Example for Big Brother Logic setup.

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:

- $V = \{x, y, dir, q\}$ for $i \in Agt$;
- $D : D(x) = D(y) = \{-20, \dots, 20\}$, $D(dir) = \{-180, \dots, 180\}$, and $D(q) = \{0, 1\}$, where $q \in Q$;
- $\mathcal{O}$: $\texttt{move}(dx, dy)$ and $\texttt{turning}(d)$, where $dx, dy \in \{-2, \dots, 2\}$ and $d \in \{-45, \dots, 45\}$;
- $\mathcal{I} = [x = 5,\ y = 5,\ dir = 45]$;
- $\mathcal{G} = \{q = 1\}$; and
- $\mathbb{F}$: $(\textbf{@check}\ q) \mapsto \{true, false\}$;

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

We need to check the knowledge queries in the actions (precondition) or goals. Both action $\texttt{move}(dx, dy)$ and action $\texttt{turning}(d)$ can change all of agents' perspectives, and therefore, can influence knowledge.

3.6.2.2 External Functions

Inputs to the external functions would be the query (in the format of our language $\mathcal{L}_{GK}\Sigma$ described in Section 3.6.1.1) and current state (x, y and dir are the only changing variables in this case). The output is the evaluated truth value of the query. The perspective function is similar to the one in Equation 4.1, except that because the angle and position of all agents except a_1 are known, it can be simplified to just:

$$\text{BBL domain:} \quad f_i(s) = \{v' = s(v') \mid v' \in s \land i \triangleright v'\} \cup \{v = s(v) \mid v \in \{x_b, y_b, ang_b, dir_b\}\}$$

Since the BBL domain is in a two-dimensional continuous environment, encoding in other epistemic planners would not be straightforward. First, a propositional approach could not be taken because there are an infinite number of propositions corresponding to the continuous variables in the domain. Second, the arithmetic operators and trigonometric functions would need to be encoded propositionally, which we believe would prove tedious and error-prone.

3.6.2.3 Goal Conditions

As for the goal conditions, some queries q can be achieved for the problem in Figure 3.2 without executing any actions because they hold in the initial state, such as the following, assuming that o_1, o_2, and o_3 have values e_1, e_2, and e_3 respectively:

1.	Single Knowledge query:	$K_{a_1} o_3 = e_3 \land \neg K_{a_2} o_3 = e_3$
2.	Nested Knowledge query:	$S_{a_1} S_{a_2} o_3 \land \neg K_{a_1} S_{a_2} o_3$
3.	Group Knowledge query:	$EK_{a_1,a_2} o_2 = e_2 \land \neg EK_{a_1,a_2} o_3 = e_3$
4.	Distributed Knowledge query:	$DK_{a_1,a_2} o_1 = e_1 \land \neg DK_{a_1,a_2} o_1 = e_3$
5.	Common Knowledge query:	$CK_{a_1,a_2} o_2 = e_2 \land CK_{a_1,a_2} S_{a_1} o_3$

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 a_2 cannot see o_3.

For goal 5, $CK_{a_1,a_2} S_{a_1} o_3$ holds in the initial state because 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, $S_{a_1} o_3$ holds based on the common local state.

In addition, there are some queries that can be achieved through valid plans:

1.	$EK_{a_1,a_2} o_1 = e_1$:	`move`$(-2,-2)$, `move`$(-2,-2)$
2.	$CK_{a_1,a_2} o_1 = e_1$:	`move`$(-2,-2)$, `move`$(-2,-2)$
3.	$S_{a_1} S_{a_2} o_1$	`move`$(-2,2)$, `move`$(-2,2)$
4.	$S_{a_1} o_3 \land \neg S_{a_2} S_{a_1} o_3$:	`move`$(-2,1)$, `move`$(-2,2)$, `move`$(-2,2)$, `move`$(-2,2)$, `move`$(-2,2)$, `move`$(-2,2)$
5.	$\neg K_{a_1} S_{a_2} S_{a_1} o_3 \land S_{a_1} o_3$:	`move`$(-2,1)$, `move`$(-2,2)$, `move`$(-2,2)$, `move`$(-2,2)$, `move`$(-2,2)$, `move`$(-2,2)$, `turning`(-45)

The first plan 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 plan 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 next two is not straightforward. To avoid a_2 seeing whether a_1 can see o_1, the cheapest plan returned by the 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_3, but also a_1 should know that originally a_2 cannot see that a_1 sees o_3. This is done by decomposing the query into three facts: "a_1 sees o_3"; "a_2 cannot see whether a_1 sees o_3"; and, "a_1 can see that whether a_2 can see whether a_1 sees o_3".

3.6.2.4 Results

Table 3.4 shows the results for our problems in the BBL domain, where $|Exp|$ represents the number of nodes expanded and $|p|$ indicates the length of the plan. The length of a plan is "∞" means that the problem instance is unsolvable – no plan exists. The perspective function in this domain depends on a *geometric model* based on the agent's position, direction, and facing angle (as defined in Equation 2.3). This shows that with proper usage of our F-STRIPS planner, we can represent continuous domains.

Our epistemic solver is able to reason about other agents' epistemic states (vision) and derive plans based on these for non-trivial goals that we believe would be tedious and error-prone to encode propositionally, if possible at all given the continuous domain. As far as we know, there is no current epistemic planner that can handle problems at this level of expressiveness.

Moreover, this expressiveness bridges the gap between high-level abstract planning spaces and low-level motion spaces, which has great potential for application in hybrid-planning [134].

3.6.3 Social-media Network

The *Social-media Network* (SN) domain is an abstract network based on typical social media platforms, in which agents can befriend each other to read their page, post on

	Parameters				Performance					Goal
	$\lvert Agt\rvert$	d	$\lvert\mathcal{G}\rvert$	$\lvert p\rvert$	$\lvert Gen\rvert$	$\lvert Exp\rvert$	$\lvert Calls\rvert$	TIME(s) *calls*	TIME(s) Total	
BBL01	2	1	1	0	1	0	2	0.000	0.001	$K_{a_1}o_2$
BBL02	2	1	1	2	115	2	232	0.007	0.009	$K_{a_1}o_1$
BBL03	2	1	1	∞	605160	*all*	1210320	39.822	87.126	$K_{a_2}o_3$
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	0.018	0.020	$EK_{a_1,a_2}\{o_1,o_2\}$
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	0.034	0.037	$CK_{a_1,a_2}\{o_1,o_2\}$
BBL10	2	2	1	2	115	2	232	0.026	0.028	$K_{a_1}DK_{a,b}\{o_1,o_2,o_3\}$
BBL11	2	2	2	6	4559	120	17807	0.592	0.620	$S_{a_1}o_3 \wedge \neg S_{a_2}S_{a_1}o_3$
BBL12	2	3	2	7	5254	127	30196	0.969	1.011	$S_{a_1}o_3 \wedge \neg K_{a_1}S_{a_2}S_{a_1}o_3$

TABLE 3.4: Results for the BBL domain.

a friend's page, and view their friend list. The observation in this domain is evaluated based on the communication channels constructed as the friendship networks. This communication channel is a two-way and all-time communication, which is an extension of the two-way one-time communication channels from the Gossip domain [125, 136]. In addition, to make it more challenging, we add sharing a secret without it being fully revealed to an agent or a group of agents as one of the objectives. 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 directly connected without anyone else on the network knowing the secret, and some secrets can be shared within a group excepting some individuals. The former could be spies sharing information with each other through the resistances' personal pages, and the latter could be a group arranging a surprise party for a mutual friend.

3.6.3.1 Example and Encoding in F-STRIPS

There are five agents, a, b, c, d, and e, with their friendship networks as shown in Figure 3.3. Their friend relations are represented by full lines between each agent. The dotted lines are referenced later for illustration purposes.

Assume there is another agent g (the global agent), who is friended with all agents. This agent g is the only acting agent in this domain, which means the friendship network is fixed. In addition, to make it more challenging, agent g is only allowed to post on one

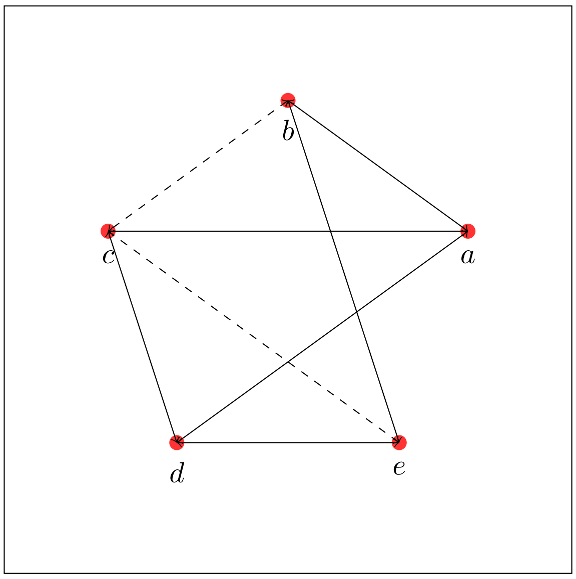


FIGURE 3.3: Example for Social-media Network.

of the other agents' home pages. Let the epistemic queries be the set of propositions Q, and p_1, p_2, p_3 be three parts of the secret P. Any problems in this setup can be represented by a tuple $(A, V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where:

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

The variable $(\mathit{friended}\ i\ j)$ represents whether i and j are friends with each other. Action $(\texttt{post}\ i\ p)$ specifies that the message p is posted on agent i's page. The initial state $\mathcal{I}$ represents the friendship relations in Figure 3.3, with no message posted yet. Similarly, the action `post` is the only source for epistemic relation changes.

3.6.3.2 External Functions

Each agent is able to view all posts on their friends' pages and also view the friend list of their friends. In this example, agent a is able to read every post on agent c's homepage, and a knows c is friended 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_aK_cp".

The perspective function depends on the friendship network. For example, consider the global state $s = \{a, b, c, d, e, (post\ p_1) = b\}$, where for simplicity, a, b, c, d, e represents whether the respective agent's page is visible, p_1 is a social media post from b, and the friend relationship is as shown in Figure 3.3. We have $f_a(s) = \{a, b, c, d, (postp_1) = b\}$; $f_d(s) = \{a, c, d, e\}$; and d's perspective in a's perspective of world s 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. We formally define the perspective function as:

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

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 and is straightforward to implement in C++.

3.6.3.3 Goal Conditions

Goals that we tested are shown in Table 3.5. For some epistemic formulae between a and b, since they are friends, simply posting the message on either of their personal pages is sufficient to establish common knowledge about the information in that post. But for goals about the shared knowledge between a and e (they are not befriended with each other), 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 each other.

Therefore, neither $EK_{a,e}EK_{a,e}p_1$ nor $CK_{a,e}p_1$ is possible without changing the network structure.

Some goals are secretive:

1. Goal: $K_a(p_1 \wedge p_2 \wedge p_3) \wedge \neg K_b(p_1 \wedge p_2 \wedge p_3)$

   Plan: `post`(a,p_1), `post`(a,p_2), `post`(c,p_3)

2. Goal: $K_a(p_1 \wedge p_2 \wedge p_3) \wedge \neg K_b(p_1 \wedge p_2 \wedge p_3) \wedge \neg K_c(p_1 \wedge p_2 \wedge p_3)$

   Plan: `post`(a,p_1), `post`(b,p_2), `post`(c,p_3)

The aims are to share the whole secret $(p_1 \wedge p_2 \wedge p_3)$ with a without b knowing the whole secret — it can know at most two out of three propositions p_1, p_2, and p_3. Some parts of it, such as p_3, need to be shared in the page that b does not have access to. In the second example, agent c must also not know the secret, the secret now needs to be posted in a way that b and c do not see some parts respectively, while a sees all the parts.

Finally, we look into those two desired scenarios in the introduction of SN for sharing with a spy (goal 3) and organizing a surprise party for agent a (goal 4):

3. Goal: $K_a(p_1 \wedge p_2 \wedge p_3) \wedge \neg K_b(p_1 \wedge p_2 \wedge p_3) \wedge \neg K_c(p_1 \wedge p_2 \wedge p_3) \wedge$
   $\neg K_d(p_1 \wedge p_2 \wedge p_3) \wedge \neg K_e(p_1 \wedge p_2 \wedge p_3)$

   Plan: `post`(a,p_1), `post`(b,p_2), `post`(c,p_3)

4. Goal: $\neg K_a(p_1 \wedge p_2 \wedge p_3) \wedge K_b(p_1 \wedge p_2 \wedge p_3) \wedge K_c(p_1 \wedge p_2 \wedge p_3) \wedge$
   $K_d(p_1 \wedge p_2 \wedge p_3) \wedge K_e(p_1 \wedge p_2 \wedge p_3)$

   Plan: unsolvable

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 by adding a friend relation between b and c, and apply the same goal conditions as above, 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 be shared in such a way that a is not able to get a complete picture of the secrets. In the setup of the problem from Figure 3.3, since a sees everything seen by c, there is no way to hold a surprise party without a knowing it. However, by adding a friend relation between e and c (SN14 in Table 3.5), the planner returns with the plan: `post`(e,p_1), `post`(e,p_2), `post`(e,p_3).

3.6.3.4 Results

	Parameters				Performance					
	$\|Agt\|$	d	$\|\mathcal{G}\|$	$\|p\|$	$\|Gen\|$	$\|Exp\|$	$\|Calls\|$	TIME(s) *calls*	TIME(s) Total	Goal
SN01	5	1	1	1	16	2	84	0.003	0.004	$K_i p_1$
SN02	5	2	1	1	16	2	84	0.005	0.006	$K_i K_j p_1$
SN03	5	1	1	1	16	2	84	0.005	0.006	$EK_{i,j} p_1$
SN04	5	1	1	3	216	92	6572	1.007	1.015	$EK_{i,j} Ps$
SN05	5	1	1	3	216	92	6572	1.048	1.056	$DK_{i,j} Ps$
SN06	5	1	1	1	16	2	84	0.008	0.009	$CK_{i,j} p_1$
SN07	5	1	1	∞	216	*all*	15552	1.030	1.050	$CK_{i,k} p_1$
SN08	5	1	1	3	216	92	6572	0.420	0.429	$K_i Ps$
SN09	5	1	2	3	232	93	13288	0.815	0.829	$K_i Ps \land \neg K_j Ps$
SN10	5	1	3	3	565	175	37644	2.484	2.530	$K_i Ps \land \neg K_{j,k} Ps$
SN11	5	1	5	3	816	251	90100	5.720	5.810	$K_i Ps \land \neg K_{other} Ps$
SN12	5	1	5	∞	2160	*all*	777600	53.191	54.004	$K_i Ps \land \neg K_{other} Ps$
SN13	5	1	2	∞	432	*all*	62208	9.809	9.895	$\neg K_i Ps \land K_{other} Ps$
SN14	5	1	2	3	216	92	13144	2.436	2.454	$\neg K_i Ps \land K_{other} Ps$

TABLE 3.5: Results for the Social-media Network domain.

Table 3.5 shows the result for our problems in the social-media network domain, where Ps represents $(p_1 \land p_2 \land p_3)$ and $K_g\varphi$ means $\bigcap_{i\in g} K_i\varphi$. The results show that our PWP model can handle a variety of knowledge relations at the same time within reasonable time complexity. Although we acknowledge that the lengths of the plans are not long by comparison to some classical planning benchmarks, it is clear that the computational burden comes from the epistemic reasoning. In addition, our results show the correlation between the number of expanded/generated nodes and the number of external function calls, which correlate with each other as well as total time.

3.6.4 Gossip

The Gossip problem is a canonical epistemic planning problem proposed by Baker and Shostak [136]. The original version contains a group of people, with each knowing a secret. They can communicate with each other by telephone. At each call, they will learn what each other knows at that moment, including direct knowledge about a secret and nested knowledge about others' knowledge. The key problem is: what is the minimum number of telephone calls that have to be performed before everyone knows all the secrets? We also experiment with other goals, such as everyone knowing that everyone knows all the secrets.

Different from the previous epistemic planning problems we experiment with in this thesis, the knowledge generated in the gossip problem depends on the current knowledge of each agent, rather than just the current world state itself. As such, we need to encode this knowledge in our state. We demonstrate two different encodings with a simple example (suppose there are three agents a, b and c, each of them has their own secret a', b' and c' respectively): one similar to the PEKB encoding in Section 3.4.2, and a novel encoding based on actions.

3.6.4.1 State-based Approach Encoding

The Gossip problem can be represented by a tuple $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where:

- $V \ = \ \{I_s\}$
- $D \ : \ dom(I_s) = \{0, \ldots, 2^{|Agt|^{|d|+1}} - 1\}$
- $\mathcal{O} \ : \ \texttt{call}(x, y)$, where $x, y \in Agt$
- $\mathcal{I} \ =$ discussed below
- $\mathcal{G} \ =$ discussed below
- $\mathbb{F} = \ (\textbf{@check}\ I_s\ q) \mapsto \{true, false\}$

The problem contains one variable, I_s, and the domain is a set of bit vectors of size $|Agt|^{|d|+1}$, in which d is the maximum depth of nested knowledge. Each bit in the vector represents a single proposition, such as $S_i p$, as outlined in Section 3.4.2. I_s represents all knowledge about secret s. In our implementation, the set is described by a large binary integer. To query a seeing formula, we simply look at the bit at the corresponding index in the bit vector I_s.

For the `call` operator, we implement the handling of 'seeing update' to an external function, which is more compact and elegant than encoding directly in the actions, which

would be equivalent to the encoding outlined by Muise et al. [87]. Therefore, the external function would update the state based on the current state and the current action. Consider the example of update in Table 3.6, where the number of agents is 3 and the depth is 2, and I_s represents the truth value of:
$\{K_a a', K_a b', K_a c', K_a K_b a', K_a K_b b', K_a K_b c', K_a K_c a', K_a K_c b', K_a K_c c', K_b K_a a', K_b K_a b', K_b K_a c', K_b a', K_b b', K_b c', K_b K_c a', K_b K_c b', K_b K_c c', K_c K_a a', K_c K_a b', K_c K_a c', K_c K_b a', K_c K_b b', K_c K_b c', K_c a', K_c b', K_c c'\}$.

Index	Action	I_s in binary	I_s in decimal
0	Initial State	100000000000010000000000001	67117057
1	`call`(a,b)	110110000110110000000000001	113467393
2	`call`(a,c)	111110111110110000111110111	132080119
3	`call`(b,c)	111110111111111111111111111	132120575
4	`call`(a,b)	111111111111111111111111111	134217727

TABLE 3.6: An example for S_i updating.

The size of the state space depends on the number of possible epistemic relations, which is bound by $|Agt|^d$. Although this approach is naïve, the computational complexity of the solution would be the same as the approach proposed by Muise et al. [87]. However, we found a limitation when we experimented with this: the grounding of actions by the planner was prohibitively expensive, in some cases running out of memory.

Can we do better? It seems unnecessary to store propositions that are never used to solve the problem. Therefore, we propose another approach, which takes advantage of the F-STRIPS planning language.

3.6.4.2 Action-based Approach Encoding

The intuition behind the action-based encoding is that we can calculate the epistemic *effects* of actions using external functions. In this solution, we store the sequence of actions (calls) that have been made, and then calculate the epistemic state from this sequence.

The Gossip problem can be described as a tuple [7], $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where

- $V = \{A_s\}$

[7]The bound of the D depends on the length of the plan, while the maximum length of the plan, $|p|$, is bound by $(d+1) \cdot (|Agt| - 1)$ according to Cooper et al. [125]'s proof.

- D : $dom(A_s) = \{0, \ldots, (|Agt| \cdot (|Agt| - 1)/2)^{|p|} - 1\}$
- $\mathcal{O}$: `call`(x, y'), where $x, y \in Agt$
- $\mathcal{I} = \{A_s = 0\}$
- $\mathcal{G}$ = discussed as below
- $\mathbb{F} = \{$ (**@check** A_s q) $\} \mapsto \{true, false\}$

The set of variables V in this approach is a bit vector (represented as an integer) used to record the action sequence that the planner has applied to reach the current expanding node. Since Cooper et al. [125] proves that even with one-way communication, any gossip problem with $|Agt|$ and d depth can be solved with $(d+1)(|Agt|-1)$ calls, we know that this is the upper bound of plan length. Using the same example as above, for a gossip problem based on the above setup and depth of 2, the domain for A_s is from 0 to 3^6. Therefore, the initial state would be $A_s = 0$, since no one has made any call yet.

The effect of the action is encoded using an external function $\Gamma : S \times A \to S$, which is a visibility update function. The planner calls $\Gamma(A_s, a)$, where A_s is the bit vector representing the history h of actions so far, and a is the current action. Then, $\Gamma(A_s, a)$ returns a new bit vector A'_s that represents $h \cdot \langle a \rangle$ — the concatenation of h and a.

For an epistemic query, the perspective function applies 'actions' encoded in A_s to calculate the current epistemic state.

We implemented two versions of this. The *Full* implementation naïvely implements the scheme above. The *Relative* implementation takes advantage of the ability to parameterise perspective functions f_i by only encoding A_s with propositions that are relevant for the epistemic goal formula. For example, consider the epistemic goal $K_c K_b a'$. This would result in $|Agt|^{|d|+1} = 27$ epistemic relations in the Full encoding. When generating epistemic formula, we start with the secret first. Since any epistemic formula related to b' or c' will be irrelevant to the query, we need not encode any epistemic relations about those secrets. So, the maximum number of epistemic relations at level 1 is $|Agt|$, because with one secret a' and $|Agt|$ agents, the greatest number of epistemic formulae that can be generated is in the case of each agent sees that secret.

Iteratively, we do the same for the next level, from $|Agt|$ amount of formulae, we select the one, $K_b a'$ $(1/|Agt|)$. There are at most $|Agt|$ new epistemic formulae that can be generated to know this one formula. With each depth, we drop all the old formulae except the one relative formula and generating $|Agt|$ amount of new epistemic formulae. Therefore, the complexity for a single modal literal would be in $O(|Agt| \cdot |depth|)$. The worst case is all agents knowing all agents' secrets, and nested up to the level of *depth*, which would be equivalent to the Full representation.

In our experiments, we compare these three methods — state-based, action-based (full) and action-based (relative) — with the baseline of Cooper et al. [125]'s method using their tool for generating their Gossip Generator [8]. Their generator compiles Gossip problems into classical planning problems, but is not a general epistemic planning tool. However, this is a suitable baseline as it allows us to evaluate solving Gossip problems using a state-of-the-art approach. To compare the performance directly, we use the $BFWS(R_0)$ planner to generate the results.

	Parameters			State			Action (full)			Action (relative)			Gossip Generator		
	d	$\lvert g\rvert$	$\lvert p\rvert$	$\lvert calls\rvert$	TIME(s) *calls*	Total	$\lvert calls\rvert$	TIME(s) *calls*	Total	$\lvert calls\rvert$	TIME(s) *calls*	Total	$\lvert p\rvert$	TIME(s) *Compile*	Total
G1-3	2	1	2	12	0.00	0.00	12	0.00	0.00	44	0.00	0.00	2	0.00	0.01
G2-3	2	9	4	126	0.00	0.00	102	0.00	0.01	989	0.02	0.02	2	0.00	0.02
G3-3	3	1	3	M	M	M	12	0.00	0.01	266	0.00	0.01	1	0.00	0.06
G4-3	3	27	5	M	M	M	422	0.37	0.37	3644	0.11	0.12	5	0.00	0.06
G5-3	4	1	4	M	M	M	34	0.06	0.07	724	0.01	0.02	3	0.01	0.26
G6-3	4	81	6	M	M	M	1625	15.54	15.55	13624	0.69	0.72	7	0.01	0.24
G7-3	5	1	4	M	M	M	38	0.43	0.43	724	0.01	0.02	4	0.04	1.76
G8-3	5	243	7	M	M	M	4845	333.90	334.00	47194	3.58	3.67	11	0.04	0.96
G1-4	2	1	2	–	–	–	–	–	–	18	0.00	0.00	2	0.01	0.05
G2-4	2	16	7	–	–	–	–	–	–	1935	0.06	0.07	7	0.01	0.06
G3-4	3	1	3	–	–	–	–	–	–	26	0.00	0.00	3	0.01	0.43
G4-4	3	64	7	–	–	–	–	–	–	11848	0.63	0.66	10	0.01	0.26
G5-4	4	1	3	–	–	–	–	–	–	104	0.00	0.01	3	0.10	2.60
G6-4	4	256	9	–	–	–	–	–	–	54711	4.49	4.61	14	0.10	1.57
G7-4	5	1	4	–	–	–	–	–	–	484	0.02	0.03	4	0.25	31.82
G8-4	5	1024	11	–	–	–	–	–	–	286288	38.82	39.40	22	0.25	9.72
G1-5	2	1	2	–	–	–	–	–	–	126	0.00	0.01	2	0.01	0.09
G2-5	2	125	9	–	–	–	–	–	–	12911	0.60	0.63	10	0.01	0.11
G3-5	3	1	3	–	–	–	–	–	–	2288	0.08	0.09	3	0.03	1.65
G4-5	3	625	11	–	–	–	–	–	–	68531	5.51	5.68	15	0.03	0.91
G5-5	4	1	3	–	–	–	–	–	–	3888	0.17	0.19	4	0.15	83.12
G6-5	4	3125	13	–	–	–	–	–	–	400627	55.00	56.14	22	0.15	7.61
G7-5	5	1	6	–	–	–	–	–	–	116030	7.53	7.97	4	0.99	450.63
G8-5	5	15625	15	–	–	–	–	–	–	2370575	546.68	558.51	32	0.99	59.72

TABLE 3.7: Results for the Gossip domain, where M means the planner ran out of memory, and – means we did not run it because it would clearly exceed the memory limit.

For the experiments, we run all approaches with 3 agents. Then, given that the performance of the action-based (relative) approach dominates our other approaches, we only run this encoding with the number of agents greater than 3.

[8]Downloadable from `https://github.com/FaustineMaffre/GossipProblem-PDDL-generator`

From the Table 3.7, problems G2, G4, G6 and G8 are four types of test cases that address classical gossip problem goals. Since the aim in the classical gossip problem is to have each agent know about others' knowledge, the size of the goal is $|Agt|^{|depth|}$. The problem types G1, G3, G5 and G7 are for comparison to show how *depth* affects single-goal problems.

For the results, the state-based approach is limited by the planner. The F-STRIPS planner we use handles function variables as integers. Therefore, for the problem with length larger than three, the possible state I_s's maximum value is $2^{27} - 1$. Because the F-STRIPS planner we use grounded actions, this results in most of the problems exceeding the maximum memory allocation on our Linux machine. These are indicated by M. Both action-based approaches are able to handle gossip problems with larger depth than the state-based approach. However, updating only relative knowledge prunes a large amount of knowledge formulae that are not going to be checked, reducing total execution time. Compared to Cooper et al. [125]'s approach, our approach has similar performance on the problems with full goals, and performs slightly better on the problem with single goals, as it will not generate irrelevant epistemic relations.

3.6.5 Discussion

Overall, the experiment results show that our solution outperforms Muise et al. [87]'s encoding solution (the state-of-the-art). As it can be seen from the results in both the Corridor and the Grapevine domains, the number of agents and the depth of epistemic relations do not increase the computation time as rapidly as the PDKB planner.

In terms of expressiveness, our PWP approach can handle a variety of complex epistemic relations, such as nested knowledge, distributed knowledge, and common knowledge, and epistemic logic reasoning with continuous domains. This can be found in the scenarios of the BBL and SN domains. In addition, even for the problem with epistemic relations embedded in the state, such as the Gossip domain, our model also shows that it can handle various problems regardless of the limitations from the planner itself.

The results show that the computation 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 during the searching process. Moreover, the amount of nodes involved in the search is affected by standard factors in search, such as the algorithm used by the planner and the difficulty of the problem itself (reflected by the plan length and the branching factor of the problem).

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 and supports the claim that customisable perspective functions are valuable.

3.7 Conclusion & Discussion

In this chapter, we revisited the agent's perspective model outlined by Hu [4], referred to as the PWP model, and introduced two novel semantics. We examined the soundness and completeness of the entire semantics and anchored the set of formulae ($\mathcal{NF}^+$) that is complete with the ternary semantics. Furthermore, we demonstrated how the PWP model can represent established epistemic logic models, indicating that it is sufficiently expressive to address problems solvable by those models. Additionally, we provided a detailed explanation of the complete planning process implementation, including the use of an external function for delegating epistemic reasoning to an external solver, as well as the comparative expressiveness of our approach. Finally, we conducted experiments on the same benchmarks used in Hu [4] and explored a challenging domain, Gossip, which could not be modeled by Hu's method.

However, there are still a few aspects that limit the significance of this work. The first is our PWP model reasons about knowledge only, which means the PWP logic is an S5 axiomatic system (discussed in Section 2.2.3). Extending the PWP model to handle belief could significantly increase its expressiveness while being challenging. Because the success of our PWP model is dependent on the property $f_i(s) \subseteq s$ for perspective functions, which implies knowledge cannot be false. The second approach (in Section 3.6.4.2) on the Gossip domain also provided us some intuition in terms of implementing the above intuition, which is unnatural in the normal (traditional) planning communities.

Secondly, the current implementation is inefficient in the number of external function calls. All epistemic relations in one action's preconditions or in the goals are evaluated separately, even if they are directly or closely related. Using goal conditions $S_a\varphi \wedge S_a\psi \wedge S_aS_b\varphi$ as an example, in the current implementation, there are 4 perspectives being generated to evaluate this goal, which are $f_a(s)$ for $S_a\varphi$, $f_a(s)$ for $S_a\psi$, and $f_b(f_a(s))$ ($f_a(s)$ being generated first) for $S_aS_b\varphi$. A more efficient way is to generate all relevant local perspectives one. Using the example, to evaluate $S_a\varphi \wedge S_a\psi \wedge S_aS_b\varphi$, only $f_a(s)$ and $f_b(f_a(s))$ are needed.

At last, although the current PWP model is valid (details can be found on the proofs of soundness and completeness), its application in planning contains risks to the reader who is not familiar with epistemic planning. Specifically, as described in the action example of the benchmarks (Section 3.6.1.1), the truth values of the epistemic formulae in the goal conditions are updated by the actions that could change agents' epistemic states. The assumption is that agents can only gain new knowledge while they cannot forget what they knew. However, this implementation could result in that the goal conditions can be met even if there are two contradicting epistemic formulae in the goal conditions, such as $K_ax=1$ and $K_ax=2$. That is, each of these goal conditions is achieved (set to be true) in different states. Although any modeler with a good understanding of epistemic planning would not do that, it is still a risk for the new beginner in this field. Thus, we need to extend the current PWP framework to make sure the epistemic formulae in the goals are consistent.

Chapter 4

Planning with Multi-Agent Belief using Justified Perspectives

A causal connection between earlier belief (or knowledge) of p and later belief (knowledge) of p is certainly a necessary ingredient in memory.

—Goldman

This chapter presents a model, built on the PWP model, for **KD45** belief over such plans, with the ability to solve model-free problems. Instead of keeping track of all possible beliefs (like Kripke structures and DEL-based existing semantic approaches, or updating the knowledge base in syntactic approaches), we use a lazy-evaluation approach that searches through previous states of the plan to re-construct what has been seen, and by whom, to evaluate nested belief formulae. Our results show that we can efficiently solve existing benchmarks in epistemic and doxastic planning, even with a simple prototype planner.

4.1 Introduction and Motivation

In epistemic planning problems, agents need to reason about the ontic world and the epistemic world. There is extensive research on epistemic logic reasoning and epistemic planning, each with its own strengths and limitations. As introduced in Section 2.3, most epistemic planning approaches either explicitly maintain the Kripke structures using DEL, or explicitly maintain the knowledge/belief database by using conditional effects to update and revise it. Both approaches suffer from an exponential growth w.r.t. the length of the action sequence for the former and the depth of the epistemic formula used in the latter one (as discussed in Section 2.3.3).

In the previous chapter (Chapter 3), we formalised the lazy-evaluation approach, PWP, which was originally proposed by Hu [4]. The PWP model (Definition 3.4) uses the perspective function to define which variables each agent 'sees' in each state, and from this, a multi-agent epistemic logic (knowledge) is built using the "what you get is what you see" paradigm [1, 2, 140, 141]. By doing so, epistemic reasoning can be performed without generating and reasoning over all epistemic relations. Section 3.6 shows PWP is able to handle more expressive problems than standard PDDL-based epistemic planners and avoids the costly pre-compilation [87, 88, 125] and maintenance of Kripke models [6, 106, 119]. In addition, their agent's perspective model only depends on the state variables' valuation, so it can be applied to model-free planners as long as they expose their current state, such as when planning with simulators [63].

The weakness of PWP is that it can plan only with knowledge, but not belief. By following the discussion (in Section 2.2) of the difference between knowledge and belief, in knowledge logic (axiomatic system **S5**), $K_i\varphi \rightarrow \varphi$ is an axiom, while in belief logics (axiomatic system **KD45**), it is not. Thus, approaches such as PWP cannot model problems in which agents can have incorrect beliefs.

In the following parts of this chapter, we extend the PWP approach to model *justified belief*. We call this Planning with Multi-agent Belief using Justified Perspectives (JP). The intuition is that when people reason about something they cannot see, they generate *justified* belief by retrieving information from their 'memory' that supports that belief [35]. In our model, this information comes from the states they have observed in the past. So, an agent believes something if they saw it in the past, and has no evidence to suggest it no longer holds. This includes nested beliefs about other agents' beliefs. We illustrate this idea with the following state sequence in NIB domain (Example 1.2) as an example, which intuitively follows the *Sally-Anne* Task (Example 1.1).

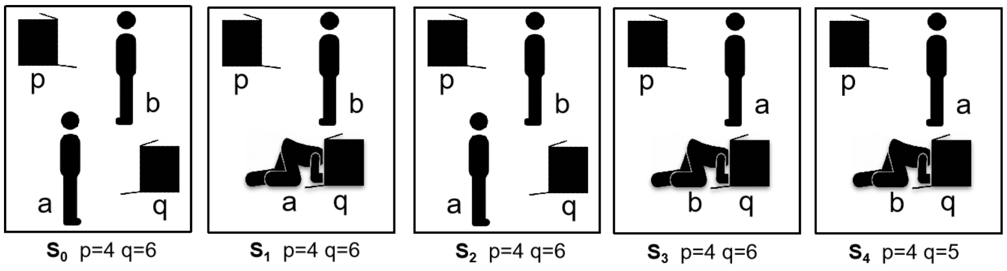


FIGURE 4.1: Plan 4.2 to solve Example 4.1.

Example 4.1. *Following the same example setup as in Example 1.2, the task becomes a false belief task, such that:*

1. *Number q is 5 ($q=5$);*
2. *Agent b believes that q is 5 ($B_b(q=5)$);*
3. *Agent a believes that q is 6 ($B_a(q=6)$);*
4. *Agent b believes that agent a believes that q is 6 ($B_bB_a(q=6)$).*

Recall that in the NIB domain, in Example 1.2, the agents are not allowed to peek into the same box at the same time. Without this requirement, a valid plan would be:

Plan 4.1. "**(peek a q)**", "**(peek b q)**", "**(return a)**", "**(decrement q)**"

Following the above plan (Plan 4.1), belief formula $B_bB_a(q=6)$ is straightforward because agent b observed agent a peeking at q after the action "**peek b q**″ — where both a and b peeked at q and saw each other doing so — while a wasn't peeking when q decreased by 1.

A more challenging plan, as shown in Figure 4.1, to reason about (given the premises that agents cannot both peek into the same box) is:

Plan 4.2. "**(peek a q)**", "**(return a)**", "**(peek b q)**", "**(decrement q)**"

In this plan, agent a and b no longer peek into the box containing q at the same time, which means we do not have $K_bK_a(q=6)$ in state s_2. All desired epistemic formulae are met for the same reason as in Plan 4.1, except item 4. For item 4, similarly, agent b recalls that the last time it saw agent a peeking to see q was s_1, when $peeking_{aq}$ is *true*. However, agent b holds no knowledge or belief on the value of q from s_1, which means b

cannot generate the justified belief in s_1. Fortunately, agent b gains belief of q from s_3. So, agent b still can justify its belief $B_b B_a(q=6)$:

1. agent b recalls that the last time agent a peeked inside of the box containing q is s_1; and
2. after b saw a peek, the next time b saw the value of the $(q=6)$ is at state s_3.

In a model-free setting, reasoning about this is particularly challenging. Agents do not have access to the action model, so they cannot reason about what other agents have seen. Instead, they can only partially observe states and re-construct belief by observing states and who else observes each state.

4.2 Related Work and Background

Most of the background and related work is covered in Chapter 2. In here, we only cover the works that are closely relevant to this section.

4.2.1 Background on Knowledge and Belief

Recall that the relation between knowledge and belief is not clearly defined (in Section 2.2.6). Some authors state that knowledge is truthful belief [142–144], while others claim that knowledge is truthful *justified* belief [145–148]. Bjorndahl and Özgün [149] define the topology for knowledge and belief based on the different types of justification, while Grossi and van der Hoek [148] states that belief is generated, endorsed, or justified by external arguments. However, there are no definitions of nested belief based on agents' previous observations.

Others use possible worlds to define both knowledge and belief [3, 150], which can be found in Section 2.2.3 and Section 2.2.5. The idea in these logics is that the agents have a possibility relation $\mathcal{K}_i$ that models whether the agent i can distinguish between two states. Both the knowledge formula $K_i\varphi$ and belief formula $B_i\varphi$ are defined as that φ holds in all the worlds that agent i considers possible. The difference is that the possibility relation $\mathcal{K}_i$ in modeling knowledge needs to be *reflective* and *symmetric*[1],

[1]Reflective means for all $s \in \mathcal{S}$, (s, s) must be in $\mathcal{K}$, while symmetric means if there is a possibility relation (s, t) in $\mathcal{K}_i$, (t, s) must also be in $\mathcal{K}_i$.

while it is *serial* in modeling beliefs. Thus, such approaches have to maintain one Kripke structure but two types of accessibility relations (one type for knowledge and another type for belief) to model both knowledge and beliefs. The constraints on those accessibility relations must hold to ensure a subset of axioms in the KB axiomatic system (Definition 2.7) hold; e.g. if an agent knows something then it believes it. In this thesis, knowledge is based on what an agent currently 'sees', while belief is based on what it currently sees and has seen in the past.

The theoretical foundation [3] for knowledge is the **S5** axiomatic system (Definition 2.2), while for belief is the **KD45** axiomatic system (Definition 2.2a). The difference between these two sets of axioms is that: **S5** includes the axiom $K_i\varphi \rightarrow \varphi$ (Axiom T), which states that an agent's knowledge must be the truth (reflexivity); while **KD45** does not have this axiom, so relations generated between possible worlds are derived from the agent's imperfect information of the world (could be false). Axiom D, replacing **T** in **KD45**, captures $\neg K_i false$, which is preserved by the serial possibility relation.

4.2.2 Related Work

Most DEL approaches do not support false-belief because they are built on **S5** logics. False-belief is challenging to model in DEL because it removes the 'correct' possibility relation between worlds, which results in the agent's belief state becoming isolated [116]. So, in order to allow agents to recover from false-belief, it requires special sensing or announcing actions [6, 116, 151]. In addition, it is costly to maintain all agents' possible worlds.

As for the non-DEL-based approach, Muise et al. [88] define a proper epistemic knowledge base (PEKB) that contains all epistemic formulae as literals. They convert an epistemic planning problem into a classical planning problem using the precondition and conditional effects in actions to update and revise the knowledge and belief literals following some modal axiom, such as those of $KD45_n$. The advantage of their approach is that the model can be solved by any existing classical planner that supports conditional effects. The limitations are: it cannot handle disjunctive belief; the depth of belief is bounded; and the number of literals grows exponentially on the depth of epistemic formulae, so the pre-compilation step has exponential time complexity.

4.2.3 KB Axioms

As claimed by Gochet and Gribomont [81], the epistemic logic systems that handle knowledge and belief (KB systems) at the same time could contain the unwanted axiom ($B_iK_i\varphi \rightarrow K_i$ in Theorem 2.8). In order to avoid that, each epistemic logic system needs to drop one of the three axioms from the KB axiomatic system (Definition 2.7): **KB1** ($K_i\varphi \rightarrow B_i\varphi$), **D** ($\neg B_i false$) and **5** ($\neg B_i\varphi \rightarrow B_i\neg B_i\varphi$). However, every one of those three axioms is intuitive: Axiom **KB1** means if agent i knows φ, then i must believe it; Axiom **D** (often in the form $B_i\varphi \rightarrow \neg B_i\neg\varphi$) means agent i cannot believe in a contradiction; and, Axiom **5** means if agent i does not believe φ, then i should believe he/she does not believe φ.

Fortunately, the epistemic logic system we proposed in this chapter (the JP model) follows the intuition of the PWP model, which is "what you see is what you know" ($S_i\varphi \wedge \varphi \leftrightarrow K_i\varphi$). In addition, it is also intuited by Goldman [35]'s idea that for the part you do not see, you believe in what you have seen, unless you see evidence suggesting otherwise. Combining those two intuitions together, we can form two (intuitive but might not be accurate) intuitions of the relation between knowledge and belief: 1) Knowledge is transient (only makes sense for the current state); and, 2) Belief is past knowledge.

In addition, the JP model follows the KB axiomatic system introduced in Section 2.7, except Axiom **KB2** as a KB logic system only needs one of the bridge axioms to hold (**KB1** or **KB2**). The unwanted axiom in Theorem 2.8 ($B_iK_i\varphi \rightarrow K_i\varphi$), arising from combining Axioms **KB1**, **D**, and **5**, is not unwanted in the JP model, considering its strict definitions of knowledge and justified belief. The reason why the axiom in Theorem 2.8 is the concern that agents believe they know something because they actually know it. Intuitively, in our logic, $B_iK_i\varphi$ only holds when $K_i\varphi$ holds, which means that believing knowledge does not cause new knowledge. That is, agents know something is the premise of them believing they know it. A formalized proof is provided in Section 4.3.3.4.

4.3 Justified Perspective (JP) Model

In this section, we introduce the Justified Perspective (JP) Model. The related background, including the epistemic logic of knowledge and belief, and epistemic planning, are introduced in Chapter 2.

High-level speaking, we add two belief operators, B_i and H_i compared to the language (Definition 3.3) in the PWP model. The belief operator B_i captures the intuition that we believe something if we have seen it before, and we have seen no contradicting evidence since. While the H_i operator captures whether we have a (committed) belief of something, no matter whether we believe it to be true or false. Here "H" can be read as "has a belief about".

The term, *justified perspective*, represents the combination of agents' perspectives (what they observe) of the world and their past perspectives (what they observed) of the world. The intuition of the justified perspective model, as mentioned in Section 4.1, is that agents reason about the unseen from their "memory". Thus, instead of only using the current state for reasoning, the JP model uses the **state sequence**. Similar to the PWP approach, this section introduces the signature, language, and model of justified perspectives. We then present two semantics: one complete semantics for soundness and completeness, and a ternary semantics for computational efficiency. As a state could be repeatedly appearing in the sequence (see Plan 4.2), theoretically, the sequence space could be infinite. Thus, we use $\vec{\mathcal{S}}^n$ to represent the sequence space with the length of n. A state in the sequence $\vec{s}$ at timestamp t (beginning at 0) is denoted by s_t or $\vec{s}[t]$.

4.3.1 Signature and Language

The signature $\Sigma = (Agt, V, D, \mathcal{R})$ of our model is exactly the same as the signature defined in the PWP model (in Definition 3.1), as well as the definition of the state (Definition 3.2). We also denote the state space and complete state space as $\mathcal{S}$ and $\mathcal{S}_c$, respectively. As previously noted, the JP model uses state sequences for reasoning rather than individual states. A sequence is indicated by a lower-case letter with "$\vec{\ }$" as its header, where $\vec{\mathcal{S}}$ and $\vec{\mathcal{S}_c}$ symbolize the sequence space and the complete-state sequence space, respectively.

In addition, the state override function $\langle\ \ \rangle$ (Definition 3.10) is also used in the JP model. Moreover, since the JP model works with state sequences, we define the extensions of state override function usage as follows:

Definition 4.1 (State Override Function with Sequences)**.** Given a state sequence $\vec{s_1}$ (with the length of $n+1$) and a state sequence *seq* (with the same length) and a state override function $\langle\ \ \rangle$ from Definition 3.10, the definition of overriding $\vec{s_1}$ with $\vec{s}$ is defined as:

$$\vec{s_1}\langle\vec{s}\rangle = \left[\vec{s_1}[0]\langle\vec{s}[0]\rangle, \ldots, \vec{s_1}[n]\langle\vec{s}[n]\rangle\right]$$

That is, we allow the override function to override a sequence of states by another sequence of states, given that those two sequences have the same length.

Definition 4.2 (Language)**.** The language $\mathcal{L}_{KB}(\Sigma)$ is defined by the grammar:

$$\varphi ::= r(V_r) \mid \neg\varphi \mid \varphi \wedge \varphi \mid S_i v \mid S_i\varphi \mid K_i\varphi \mid H_i\varphi \mid B_i\varphi,$$

where $r(V_r)$ represents a predicate symbol applied to terms V_r, $V_r \subseteq V$, and $r \in \mathcal{R}$.

Both B_i and H_i are belief operators. $B_i\varphi$ means that agent i believes that proposition φ is true, while $H_i\varphi$ represents i *has* a belief about φ (i.e. it means $B_i\varphi \vee B_i\neg\varphi$). Semantically, operator H_i is to operator B_i what operator S_i is to K_i. However, intuitively, agents' knowledge is generated from their observation, while this relation is different between B and H. The details are explained in semantics' definition.

4.3.2 Model Instances

In this section, we define the instances of the JP model, which need to be defined by the modeller when they use our approach to reason about the knowledge and belief formulae in our language $\mathcal{L}_{KB}(\Sigma)$.

Definition 4.3 (JP Model)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, an instance of the justified perspective model M is defined as:

$$M = (Agt, V, D, \pi, O_1, \ldots, O_k),$$

in which Agt, V, D are from the given signature, π is the evaluation function. The detailed definition is given as follows:

- The domain of variable $v \in V$ is D_v, which is a set of all possible values of v (from the definition of the signature). In here, a "None" value represented by symbol $\perp$ is included ($D_v := D_v \cup \{\perp\}$), which represents that the value of a variable is not part of a particular agents' observation. A state with all variables assigned with $\perp$, denoted as $s_\perp$ ($s_\perp = \{v = \perp \mid v \in V\}$). Thus, a special sequence is a sequence with all states as $s_\perp$, denoted as $\vec{s_\perp}$.
- The interpretation function $\pi : \mathcal{S} \times \mathcal{R} \to \{true, false\}$ determines whether the atomic term $r(V_r)$ is true in s. π is undefined if any of its arguments t_i is a variable $v \in V$ that is not assigned a value in a given state s, i.e. $v \notin s \vee v \neq \perp$.
- Functions $O_1, \dots, O_k$ are inherited from PWP model defined in Definition 3.4, except we rename the perspective function f_i into observation function O_i, as f_i is our notation for the justified perspective function (Definition 4.6). In addition, since the JP model evaluates epistemic relations based on state sequence, here we allow the observation function to take input of a sequence as well. That is, $O_i(\vec{s}) = [O_i(\vec{s}[0]), \dots, O_i(\vec{s}[n])]$ for a sequence $\vec{s}$ with length of $n + 1$.

Since a state is a set (of variable assignments), the set operations are also applicable on the state, such as set union operator "$\cup$" and set minus operator "$\setminus$". However, since we have introduced a special value (assignment) for representing none value, some specific rules of set relations and operators need to be clarified.

Lemma 4.4. *For any variable $v \in V$, we have $v = \perp \in \{v = e\}$ and $v \in \{v = e\}$ for any $e \in D_v$.*

The special none value is acted as a place holder for reasoning about nested epistemic relations. The state could also be defined as a set of variable identifiers union with a set of assignments, such as $\{v, v \to e\}$. In this way, $v = \perp$ could be represented by v only without a valid assignment. Thus, it is straightforward for the above Lemma to hold. But for simplicity, we model the state as a set of assignments only.

It is worthy to note that the observation functions in our Model (Definition 4.3), which is from the PWP approach (Definition 3.4), also follow the above lemma. Specifically, from Lemma 4.4, we also have $\{v = \perp\} \subset \{v = e\}$, where $e \in D_v \setminus \{\perp\}$. Therefore, the subset relation in Property 3 in Definition 3.4 also follows this, which we used later in proofs for Theorem 4.9, 4.7 and 4.11.

4.3.2.1 Justified Perspectives

In this section, we propose two functions (a *retrieval function* in Definition 4.5 and a *justified perspective function* in Definition 4.6) which are critical to generating agents' justified perspective with arbitrary nesting.

First, we define a retrieval function R to retrieve a variable's value from the latest timestamp (state) that the agent had an 'eye' on this variable. From this, we will define the perspective function f_i to construct the agent's justified perspective, and reason about the agent's justified belief following the intuition discussed in Section 4.1.

Definition 4.5 (Retrieval function)**.** Given a sequence of states $\vec{s}$, a timestamp ts and a variable v, the retrieval function, $R : \vec{\mathcal{S}} \times \mathbb{Z} \times V \rightarrow D$, is defined as:

$$R(\vec{s}, ts, v) = \begin{cases} \vec{s}[\max(\mathrm{LT})](v) & \text{if LT} \neq \{\} \\ \vec{s}[\min(\mathrm{RT})](v) & \text{else if RT} \neq \{\} \\ \perp & \text{otherwise} \end{cases}$$

where:

$$\mathrm{LT} = \{j \mid v \in \vec{s}[j] \wedge \vec{s}[j](v) \neq \perp \wedge j \leq ts\},$$

$$\mathrm{RT} = \{j \mid v \in \vec{s}[j] \wedge \vec{s}[j](v) \neq \perp \wedge ts < j \leq |\vec{s}|\}.$$

Here, $\vec{s}$ represents the sequence of states of a plan from a particular perspective, which could be an agent's perspective or the global perspective. The sets LT and RT denote the sets of timestamps in which variable v is observed (i.e., defined and not equal to $\perp$) to the left and right of timestamp ts, respectively. Specifically, LT includes all timestamps less than or equal to ts, and RT includes all timestamps strictly greater than ts, assuming timestamps are indexed as a sequential list of natural numbers. Intuitively, LT contains all timestamps before and on ts that variable v is seen, while RT are all timestamps after ts that variable v is seen. Recall that $\perp$ is a special value 'None'. If this function returns $\perp$, it means that the agent has never seen a not-'None' value of variable v.

The function R plays a crucial role. If we see an agent i seeing variable v, we know that agent i learns the value of v. However, what value should we believe that i believes? The function R determines this. If the value of variable v exists at time ts, then this is in 'our' perspective, $\vec{s}$, and we see the variable at the same time as i, so R returns the value of v in state s_{ts}. This is the straightforward case.

However, if we do not see variable v at time ts, what value should we assign to agent i's belief? The retrieval function R searches the timestamps before ts to find the most recent reference to v. Intuitively, if we see that agent i sees v at ts, but we do not see the value of v itself at time ts, then we believe that agent i believes the value is the same as the last time we saw v. For example, if we peek at the coin in the box and see it is a tail, and then we observe agent i peeking at the coin, it implies $B_a coin = tail$ should hold, because *tail* is the most recent observation of the coin.

If there is no value of v before ts, the function R retrieves the value by searching forward (the timestamps after ts). Intuitively, if we believe that agent i sees v at ts, but we have not seen variable v previously, then we assign i's belief about v the next time we see v after ts.

This is what we see in Plan 4.2 – agent b forms a belief about agent a based on agent b's observation after agent a's observation. If there is no value found about v within $\vec{s}$, then R function returns $\bot$, as the variable has not been seen from $\vec{s}$.

Other design decisions could be made for R: searching forward first, then backwards; finding the value closest to ts; or 'forgetting' the value of a variable after a certain number of timestamps. Ultimately, there is no 'correct' design here and no design can handle all possible cases, but we believe our choice above is intuitive and justified.

We can now give the definition of a perspective justified function f_i for agent i. Intuitively, a justified perspective function models an agent's perspective over the sequence of states in a plan; specifically, an agent's belief about each variable in each state from a given state sequence, which can either be the sequence of global states or another agent's justified perspectives.

Definition 4.6 (Justified Perspective Function)**.** Given the input state sequence $\vec{s} = [s_0, \ldots, s_n]$, a Justified Perspective (JP) function for agent i, $f_i : \vec{\mathcal{S}_c} \rightarrow \vec{\mathcal{S}_c}$, is defined as follows:

$$f_i([s_0, \dots, s_n]) = [s'_0, \dots, s'_n]$$

where for all $t \in [0, n]$ and all $v \in V$:

$$\begin{aligned}
lt_v &= \max(\{j \mid v \in O_i(s_j) \wedge j \leq t\} \cup \{-1\}), && (1)\\
e &= R([s_0, \dots, s_t], lt_v, v), && (2)\\
s''_t &= \{v{=}e \mid s_t(v) = e \vee v \notin O_i(s_t\langle\{v{=}e\}\rangle)\}, && (3)\\
s'_t &= s_\perp\langle s''_t\rangle. && (4)
\end{aligned}$$

This definition is not so straightforward, so let's give it a high-level intuition. The justified perspective function requires input of a current perspective (assume it is our perspective). To reason what an agent believes, a justified perspective is constructed as a sequence of states with the same length as our perspective. In each timestamp, for each variable, we: 1) recall the last time ("lt_v") the agent saw the variable v, which is -1 if the agent has not seen this previously; 2) retrieve the value of the variable from 'our' memory using R (Definition 4.5); 3) form the state that the agent believes by removing values that an agent has seen previously, but which are inconsistent with its current observation; and 4) consider that the agents believe the value of all missing variables is $\perp$ from the generated state (for future reasoning). Now, we explain the above steps in detail.

First, recall that the sequence $\vec{s} = [s_0, \dots, s_n]$ can be either the global state sequence or the perspective of another agent. Any state from the global state sequence is a complete state, while Line (4) in Definition 4.6 ensures any state from any perspective of any agent is also a complete state — $s_\perp$ is always a complete state (Definition 4.3). By doing so, although we cannot reason about the other agent's belief of the missing variables for now, it keeps the information for us to reason about it in the future. The outcome of overriding a complete state with another state (could be partial or complete) is a complete state. Thus, both input and output for the JP function are a sequence of complete states.

Intuitively, an agent's belief should be consistent with the agent's observation in the current state: if an agent sees something, it must believe it. Thus, we propose Theorem 4.7

Theorem 4.7. *For any timestamp t in a given state sequence $\vec{s}$, $O_i(\vec{s}[t]) \subset f_i(\vec{s})[t]$.*

Proof. This proof is straightforward. For any variable $v \in V$ such as $v \in O_i(\vec{s}[t])$, we have $lt_v = t$ due to Line (1) in Definition 4.6. Then, if $\vec{s}[t](v) \neq \perp$, we have $R([s_0, \ldots, s_t], t, v) = \vec{s}[t](v)$, which means $f_i(\vec{s})[t](v) = e = O_i(\vec{s}[t])(v)$. If $\vec{s}[t](v) = \perp$, based on Theorem 4.9, $O_i(\vec{s}[t])(v) = f_i(\vec{s})[t](v) = \perp$. Therefore, $O_i(\vec{s}[t]) \subset f_i(\vec{s})[t]$. $\square$

The timestamp lt_v is the last timestamp that agent i sees variable v by state s_t, according to the current perspective, which is -1 if agent i has not seen v at all. This tells us the last time that agent i was seen observing variable v in the current perspective. This evidence is used to justify belief [35].

However, if the current perspective represents an agent's perspective, agent j, rather than a global perspective, then agent j may not have seen variable v at time lt_v—it may have merely observed agent i seeing v, without seeing (knowing) v itself; e.g. the two agents peek at the coin in the box at different times. We use $R([s_0, \ldots, s_t], lt_v, v)$ (Definition 4.5) to find what value agent j will believe variable v was in state s_t. That is, the most recent value before lt_v or (if not found) the closest after lt_v, as defined by R. Therefore, the value $R([s_0, \ldots, s_t], lt_v, v)$ is the value of v that agent j 'believes' agent i saw (Line 2).

Line (3) in this definition forms a justified (complete/partial) state of the agent i at timestamp t. The assignment of variable v is in this justified state if: the retrieved value of v ($R([s_0, \ldots, s_t], lt_v, v) = e$) is the same as the input perspective ($s_t(v) = e$); or ($s_t(v) \neq e$) and the agent i cannot prove the value changed. The latter condition requires some further explanation.

There are two possible scenarios when $s_t(v) \neq e$: either $v \in O_i(s_t)$ or $v \notin O_i(s_t)$. For the first scenarios, $v \in O_i(s_t)$, along with the premise ($s_t(v) \neq e$), it is intuitive that if $s_t(v)$ is a non-none value, then agent j should believe that agent i believes (is seeing) v equals to $s_t(v)$, which is what j believes v is. Thus, we propose Theorem 4.8 to prove, in the first scenarios, $s_t(v) = \perp$. Then, even if the retrieval function R returns a non-none value, agent j is still not able to believe the agent i believes v is this value. Therefore, we propose Theorem 4.9 and prove that even if e from Line 3 is not $\perp$, j is still not able to believe others believe $v = e$ due to the same reason j cannot believe $v = e$. That is, $v \notin s''_t$.

Theorem 4.8. *For any timestamp t in a given perspective $\vec{s}$, if $v \in O_i(\vec{s}[t])$ and $R([s_0, \dots, s_t], t, v) \neq s[t](v)$, then we have $s[t](v) = \perp$.*

Proof. Assume $s[t](v)$ equal to some not-'None' value $e \in (D_v \setminus \{\perp\})$.

Since $v \in O_i(\vec{s}[t]) \wedge \vec{s}[t](v) \neq \perp$, it must be that LT from Definition 4.5 is not an empty set (contains t at least), which means $max(\text{LT}) = t$. Thus, referring to the first condition, we have $R([s_0, \dots, s_t], t, v) = s[t](v)$. This is not consistent with the given condition ($R([s_0, \dots, s_t], t, v) \neq s[t](v)$), which means our assumption does not hold. Therefore, we prove $s[t](v) = \perp$ by contradiction. □

Theorem 4.9. *For any timestamp in a given justified perspective $\vec{s}$, if $v \in O_i(\vec{s}[t])$ and $\vec{s}[t](v) = \perp$, then we have $f_i(\vec{s})[t](v) = \perp$.*

Proof. First, the given premise ensures $\vec{s}[t](v) = \perp$, which indicates that the given sequence is from another agent's perspective (global sequence would not have any variable with value of $\perp$). Let this agent be j and the state sequence for the JP function to get $\vec{s}$ as $\vec{s_k}$. That is, $f_j(\vec{s_k}) = \vec{s}$.

Then, $s[t](v) \equiv f_j(\vec{s_k})[t](v) = \perp$ has two possibilities: either j has never seen the value of v since the beginning of the input sequence $\vec{s_k}$ ($R([\vec{s_k}[0], \dots, \vec{s_k}[t]], t, v) = \perp$); or j has seen the value of v, but j also sees v has been changed according to Line (3) in Definition 4.6.

For the first case, it is intuitive. For all timestamps $t' \in \{0, \dots, t\}$, we have either $v \notin O_j(\vec{s_k}[t'])$ or $v \in O_j(\vec{s_k}[t']) \wedge \vec{s_k}[t'](v) = \perp$, which ensures for all t', $f_j(\vec{s_k})[t](v) \equiv \vec{s}[t](v) = \perp$. Thus, $R([\vec{s}[0], \dots, \vec{s}[t]], t, v) = \perp = \vec{s}[t](v)$, which means the first case will not hold for the given premise.

As for the second case, j has seen some value e ($R([\vec{s_k}[0], \dots, \vec{s_k}[t]], t, v) = e$) of v before, but $\vec{s}[t](v) = \perp$ due to $e \neq \vec{s_k}[t](v) \wedge v \in O_j(\vec{s_k}[t]\langle\{v = e\}\rangle)$. Thus, there exists some value e of v that is not none, which effectively means the retrieval function will get a not none value of v. Assume the value returned by the retrieval function is $e \in D_v \setminus \{\perp\}$, we have $e \neq \vec{s}[t](v)$, which means the left condition for Line (3) in Definition 4.6 is not met. Then, based on $\vec{s}[t] \subsetneq \vec{s}[t]\langle\{v{=}e\}\rangle$ (set relation) and $v \in O_i(\vec{s}[t])$ (the premise), we have $v \in O_i(\vec{s}[t]) \subsetneq O_i(\vec{s}[t]\langle\{v{=}e\}\rangle)$ according to Property (3) of the observation function (in Definition 3.4). Thus, the right condition of Line (3) in Definition 4.6 is also not met.

That is, the assignment of v is not in s''_t from Line (3), and a none assignment $(v{=}\bot)$ is added according to Line (4). □

Therefore, the latter condition $(s_t(v) \neq e)$ can only be triggered by the second scenario $(v \notin O_i(s_t))$. Based on our intuition in Section 4.1, agents believe v's value stays unchanged unless they see it otherwise [35]. Agents saw the value changed either by direct observation, or by indirect inference $(v \notin O_i(s_t))$. Direct observation is trivial, agents saw the value of the variable changed, while indirect inference is trickier. To explain it, let us introduce another example as follows:

Example 4.2. *Consider a corridor with 3 rooms, r_1, r_2, and r_3. Three agents a, b, and c are all located in r_1. They can only observe the room that they are in. Let a plan be agent b moving to r_2.*

The global sequence $\vec{s_g}$, agent a's observations of the global sequence $O_a(\vec{s_g})$ and a's justified perspective $f_a(\vec{s_g})$ are as follows:

- $\vec{s_g}$ is $[\{loc_a{=}r_1, loc_b{=}r_1, loc_c{=}r_1\}, \{loc_a{=}r_1, loc_b{=}r_2, loc_c{=}r_1\}]$;
- $O_a(\vec{s_g}) = [\{loc_a{=}r_1, loc_b{=}r_1, loc_c{=}r_1\}, \{loc_a{=}r_1, loc_b{=}r_2, loc_c{=}r_1\}]$;
- $f_a(\vec{s_g}) = [\{loc_a{=}r_1, loc_b{=}r_1, loc_c{=}r_1\}, \{loc_a{=}r_1, loc_b{=}\bot, loc_c{=}r_1\}]$.

Intuitively, the justified perspective of a should be the same as the global state for s_0. Since in s_1, since a is in r_1, we have $loc_a, loc_c \in O_a(s_1)$, which means $lt_{loc_a} = lt_{loc_c} = 1$ and $R(\vec{s_g}, 1, loc_a) = R(\vec{s_g}, 1, loc_c) = r_1$. As for loc_b, we have $lt_{loc_b} = 0$, which results in $e = R(\vec{s_g}, 0, loc_b) = r_1$. However, we have $s_1(loc_b) = r_2$, which means $s_1(loc_b) = r_1$ does not hold. Although it is perfectly fine for the agent to hold false beliefs in epistemic planning, agent a should be able to reason that this belief is false (inconsistent with a's own observation), which is the indirect inference that we design to capture.

In order to capture indirect inference, firstly, agents assume the value e of v is unchanged $(e = R([s_0, \ldots, s_t], lt_v, v))$, which means the state s_t is overridden by the assignment $v{=}e$. In this case, agent a assumes $loc_b = r_1$. Formally, we have $O_a(s_1\langle\{loc_b{=}r_1\}\rangle) = O_a(\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}) = \{loc_a = r_1, loc_b = r_1, loc_c = r_1\}$. Then, if $v \notin O_i(s_t\langle\{v = e\}\rangle)$, which means agents are not able to see $v = e$ even if v's value is still e, then, it is reasonable to assume v's value is still e. While, if $v \in O_i(s_t\langle\{v{=}e\}\rangle)$, which means agents will see $v = e$ if v's value is still e, it is not reasonable to assume the value of v is still e (because the premise is that $v \notin O_i(s_t)$). In this case, if $loc_b = r_1$, agent a should still be able to see loc_b. Thus, it is not reasonable to assume loc_b's value is still r_1. Therefore, there is not going to be any assignment of v in s''_t (Line 3), while $s'_t(v) = \perp$ according to Line (4). In this case, the justified perspective of agent a is $f_a(\vec{s_g}) = [\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}, \{loc_a = r_1, loc_b = \perp, loc_c = r_1\}]$.

Overall, a JP function $f_i(\vec{s})$ forms a justified perspective of agent i under the input justified perspective $\vec{s}$. Thus, we can nest justified perspective functions arbitrarily to form nested beliefs.

In addition, we also proposed the following theorems for nested justified perspective function. Theorem 4.10 ensures if an agent believes something, then they must believe themselves to see that belief, while Theorem 4.11 represents that if an agent believes something, they believe that they believe it.

Theorem 4.10. *For any timestamp t in any state sequence $\vec{s}$, $O_i(\vec{s}[t]) \subseteq O_i(f(\vec{s})[t])$.*

Proof. According to Theorem 4.7, we have $O_i(\vec{s}[t]) \subseteq f_i(\vec{s})[t]$. Based on the monotonicity of the observation function (If $s \subseteq s'$, then $O_i(s) \subseteq O_i(s')$), we have $O_i(O_i(\vec{s}[t])) \subseteq O_i(f_i(\vec{s})[t])$. Given the idempotency of the observation function ($O_i(s) = O_i(O_i(s))$), ($O_i(O_i(\vec{s}[t])) = O_i(\vec{s}[t])$), we have $O_i(\vec{s}[t]) \subseteq O_i(f(\vec{s})[t])$. □

Theorem 4.11. *For any justified perspective $\vec{s}$, we have $f_i(\vec{s}) = f_i(f_i(\vec{s}))$.*

Proof. The base case is a sequence with one state $[s_0]$. We have $f_i([s_0]) = [s_\perp \langle O_i(s_0) \rangle]$ based on applying the JP function in Definition 4.6. According to Theorem 4.7 and the monotonicity of the observation function (Property 3), we have: $O_i(s_0) \subset f_i([s_0])[0]$ and $O_i(O_i(s_0)) \subset O_i(f_i([s_0])[0])$; $f_i([s_0])[0] \subset s_0$ and $O_i(f_i([s_0])[0]) \subset O_i(s_0)$, which are effectively: $O_i(s_0) \subset O_i(f_i([s_0])[0]) \wedge O_i(f_i([s_0])[0]) \subset O_i(s_0) \equiv O_i(s_0) = O_i(f_i([s_0])[0])$. Thus, $f_i([s_0]) = f_i(f_i([s_0]))$.

For a sequence of more than one element, any state ($f_i(\vec{s})[t]$) at timestamp t in agent i's justified perspective depends on the input state sequence and agent's observation before t, which are $[\vec{s}[0], \dots, \vec{s}[t]]$ and $[O_i(\vec{s}[t]), \dots, O_i(\vec{s}[t])]$ respectively.

Any variable $v \in V$ can be classified into one of the following conditions:

1. $v \in O_i(f_i(\vec{s})[t])$

2. $v \notin O_i(f_i(\vec{s})[t])$

For Condition (1), according to Theorem 4.7, $O_i(f_i(\vec{s})[t]) \subset f_i(f_i(\vec{s}))[t]$. Since $v \in O_i(f_i(\vec{s})[t])$, following the same proof for Theorem 4.7, $f_i(\vec{s})[t](v) = f_i(f_i(\vec{s}))[t](v)$.

For Condition (2), $v \notin O_i(f_i(\vec{s})[t])$ indicates $v \notin O_i(\vec{s}[t])$ (Theorem 4.10). The value of both $f_i(\vec{s})[t](v)$ and $f_i(f_i(\vec{s}))[t](v)$ depends on the values from previous timestamp $t-1$ (lt_v is smaller than t in the process of generating both justified perspectives ($f_i(\vec{s})$ and $f_i(f_i(\vec{s}))$) according to Definition 4.6). Then, whether $f_i(\vec{s})[t-1](v)$ and $f_i(f_i(\vec{s}))[t-1](v)$ are the same can be reasoned recursively until it reaches the base case.

Overall, $f_i(\vec{s}) = f_i(f_i(\vec{s}))$ holds. □

4.3.3 Semantics

Now, we give two different **KD45** semantics: complete semantics and ternary semantics, which extend our respective **S5** semantics for the PWP model from Chapter 3. The complete semantics have an exponential worst-case time complexity, while the ternary semantics have a polynomial time complexity (for a given plan length) and have the same properties of incompleteness as the PWP semantics.

4.3.3.1 Complete Semantics

The complete semantics inherits the definitions of Items (a) - (e) in Definition 3.11, but with three minor changes: (1) the frame is a pair $M, \vec{s}$ instead of M, s, i.e. it requires a sequence of states instead of a single state s; (2) the S_i operator uses O_i instead of f_i (our function O_i is equivalent to PWP's f_i perspective function, while our justified perspective function f_i generalises for belief); and (3) the evaluation of atomic propositions is based on the final state of the sequence $\vec{s}$.

Similarly as PWP, our complete semantics is to reconstruct the possible worlds and evaluate epistemic logic formula accordingly. However, simply overriding the none value variables in the agent's justified perspectives would result in inconsistent perspectives, which is mainly caused by Line 3 in Definition 4.6. Thus, we propose a possible sequence function to generate the set of all possible sequences for agent i given a sequence $C(f_i(\vec{s}), i)$ as follows:

Definition 4.12 (Possible Sequence Function)**.** Given a state sequence $\vec{s}$ for agent i (could be either justified perspective of i, or a sequence of observations of i) with length of $n+1$, all possible sequences that agree with $\vec{s}$ can be generated by the possible sequence function, $C : \vec{\mathcal{S}} \times Agt \rightarrow \mathcal{P}(\vec{\mathcal{S}_c})$, can be defined as:

$$C(\vec{s}, i) = \{\big[w_0, \dots, w_n\big] \ \mid w_0 \in W_0, \dots w_n \in W_n\}$$

where for all $t \in [0, n]$:

$$\begin{aligned} W_t &= \{w' \mid w' \in W'_t, \forall v \in O_i(w') \rightarrow v \in O_i(\vec{s}[t])\} \\ W'_t &= \{s_c\langle \vec{s}[t] \setminus s_\perp\rangle \ \mid s_c \in \mathcal{S}_c\} \end{aligned}$$

Intuitively, this function identifies possible worlds W'_t for each state in the agent's justified perspective $\vec{s}[t]$ by adding possible values to those variables that are equal to $\perp$. However, some of the newly added values might be inconsistent with the agent's justified perspective function due to the indirect inference mentioned in Definition 4.6. This is triggered by the agent i cannot see the variable v ($v \notin O_i(\vec{s}[t])$), but the agent i sees v once the state is "filled up" with possible value e of v. That is, if the value of v is e originally, then the agent i should be able to see it. Thus, agent i must not believe the value of v could possibly be e.

Using Example 4.2, the sequence of agent a's observation $O_a(\vec{s_g})$ is $[\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}, \{loc_a = r_1, loc_c = r_1\}]$ and a's justified perspective $f_a(\vec{s_g})$ is $[\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}, \{loc_a = r_1, loc_b = \perp, loc_c = r_1\}]$. The possible sequences that are consistent with both can be generated by the sample process as follows: $W'_0 = W_0 = \{\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}\}$ as the agent observes everything in the timestamp 0; W'_1 contains three possible states, as loc_b has three not-'None' values ($D_{loc_b} = \{r_1, r_2, r_3, \perp\}$), while W_1 only contains two possible states due to $loc_b = r_1$ not being consistent with a's justified perspective. This is because variable loc_b is in the possible world $\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}$ from W'_1, but it is not in neither $O_a(f_a(\vec{s_g})[1])$ nor $O_a(O_a(\vec{s_g})[1])$, which are both $\{loc_a = r_1, loc_c = r_1\}$. Thus, $C(f_a(\vec{s_g}), a) = C(O_a(\vec{s_g}), a)$, which only contains two possible sequences: $[\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}, \{loc_a = r_1, loc_b = r_2, loc_c = r_1\}]$ and $[\{loc_a = r_1, loc_b = r_1, loc_c = r_1\}, \{loc_a = r_1, loc_b = r_3, loc_c = r_1\}]$.

Extending the intuition above, adding possible values to variables that contain no value should not change agent i's "observation", because of indirect inference (discussed in Definition 4.6), which is if some value e makes i sees v, while originally i cannot see v, then i would consider e is not possible based on the fact i cannot see v. In here, the word "observation" only refers to the variable symbols in agents' observations, not the value. That is, agent i observed a variable $v \in O_i(\vec{s}[t])$ while knowing or believing it's none due to lack of belief of this variable from its parent's perspective ($\vec{s}[t](v) = \perp$). In another word, the newly added possible value does not make agent i see any new variables, which is ensured by Definition 4.12. In this case, by filling the possible values, agents could update their observation (from $O_i(\vec{s}[t])(v) = \perp$), but only by the value of variables, not the variables themselves. The formalization of this is proposed as the following lemma.

Lemma 4.13. *Given a state sequence $\vec{s}$ for agent i (could be i's observation sequence or justified perspective) with the length of n, for any $\vec{g}$ in $C(\vec{s}, i)$, we have $\forall t \in \{0, \dots, |\vec{s}|\}, V_t = V'_t$, where $V_t = \{v \mid \forall v \in O_i(\vec{s}[t])\}$ and $V'_t = \{v \mid \forall v \in O_i(\vec{g}[t])\}$.*

From its definition (Definition 4.12), we have any sequence generated by a possible sequence function is consistent with the agent's justified perspective. By consistent, we mean $\forall \vec{g} \in C(\vec{s}, i), \forall t \in \{0, \dots, |\vec{s}|\}, \vec{s}[t] \subseteq \vec{g}[t]$. This is straightforward as the possible sequence $\vec{g}$ is just adding a consistent possible value to $\vec{s}$.

Then, does agent i hold the consistent observations and beliefs under the possible world $\vec{g} \in C(\vec{s}, i)$ (i believes possible) need some discussion. Thus, we propose the following two theorems.

Theorem 4.14. *Given agent i's observations of a state sequence $O_i(\vec{s})$ with the length of $n+1$, for any $\vec{g}$ in $C(O_i(\vec{s}), i)$, we have $\forall t \in \{0, \dots, n\}, O_i(\vec{s})[t] \subseteq O_i(\vec{g})[t]$.*

Proof. The proof for this is trivial. According to Definition 4.12, the possible sequences are generated from W_t, which is adding consistent possible values for variables v such that $f(\vec{s})[t](v) = \perp$. With Lemma 4.13, we have $O_i(\vec{s}[t]) \subseteq O_i(\vec{g}[t])$. □

Intuitively, one should think the above theorem should use "=" ($\forall t \in \{0, \dots, n\}, O_i(\vec{s})[t] = O_i(\vec{g})[t]$). The reason for using "$\subseteq$" has been mentioned in the paragraph above the theorem. The agent could update their observation of the variables equal to $\perp$ with their possible values. Those variables with value none $\perp$ in the agent's observation indicate that the agent sees those variables, but does not "know" the value of them due to their values being missing in the parent perspectives.

Theorem 4.15. *Given agent i's perspective of a state sequence $f_i(\vec{s})$ with the length of $n+1$, for any $\vec{g}$ in $C(f_i(\vec{s}), i)$, we have $\forall t \in \{0, \dots, n\}, f_i(\vec{s})[t] \subseteq f_i(\vec{g})[t]$.*

Proof. We prove this by induction. For the base case (timestamp 0), we have $f_i(\vec{s})[0] = s_\perp\langle O_i(\vec{s}[0])\rangle$ and $f_i(\vec{g})[0] = s_\perp\langle O_i(\vec{g}[0])\rangle$. Since $O_i(\vec{s}[0]) \subseteq O_i(\vec{g}[0])$, we have $f_i(\vec{s})[0] \subseteq f_i(\vec{g})[0]$.

For any timestamp $t \in \{0, \dots, |\vec{s}|\}$, the justified perspective of $\vec{g}$ for agent i is generated by Definition 4.6, in which the lt_v for all variables are the same as in $f_i(\vec{s})[t]$ due to Lemma 4.13. To clarify this proof, let Condition 1, 2, and 3 be $lt_v = t$, $\vec{s}[t](v) \neq \perp$, and $f_i(\vec{s})[t](v) \neq \perp$ in this proof. Then, if the last timestamp i sees v is the current timestamp t, v has a not-none assignment and this assignment is consistent with i's observation (all conditions hold), we have $f_i(\vec{s})[t](v) = f_i(\vec{g})[t](v)$; if the current value of v from $\vec{s}[t]$ is not consistent with i's observation (Condition 1 and 2 hold, while 3 does not), we have $f_i(\vec{g})[t](v) = \vec{g}(v)$ and $f_i(\vec{s})[t](v) = \perp$; if v is none from the sequence (Condition 2 does not hold, while others hold), we have the same conclusion as the previous one; if v is not observed by i in the current timestamp, then both values are determined by their values in the previous timestamp ($t-1$), which are $f_i(\vec{s})[t](v) = f_i(\vec{s})[t-1](v)$ and $f_i(\vec{g})[t](v) = f_i(\vec{g})[t-1](v)$, based on Definition 4.6.

Then, recursively run this proof from t to $t-1$ all the way until the base case, we have $\forall t \in \{0, \dots, |\vec{s}|\}, f_i(\vec{s})[t] \subseteq f_i(\vec{g})[t]$. □

The intuition behind this theorem is that, in the possible worlds (sequences), generated from agents' justified perspectives, their original beliefs (in their justified perspectives) must hold. That is, they cannot consider a world (sequence) possible, if what they believe is not held in this world (sequence). In addition, we can push the above theorem a little bit further. If the agents consider a world (sequence) $\vec{g}$ to be possible based on their justified perspectives, then, in this sequence ($\vec{g}$), the agents should believe this sequence ($\vec{g}$) is possible. Thus, we proposed Theorem 4.16.

Theorem 4.16. *For any possible sequence $\vec{g} \in C(f_i(\vec{s}), i)$ that i considers possible given a state sequence $\vec{s}$, we have $\vec{g} \in C(f_i(\vec{g}), i)$.*

Proof. This proof can be done by considering two conditions: (1) $\exists t \in \{0, \dots, |\vec{s}|\}, f_i(\vec{s})[t] \neq f_i(\vec{g})[t]$ and (2) vice versa.

The Condition (2) is trivial, as $f_i(\vec{s}) = f_i(\vec{g})$, we have $C(f_i(\vec{s}), i) = C(f_i(\vec{g}), i)$, which means $\vec{g}$ must be in $C(f_i(\vec{g}), i)$. The Condition (1) can be proved by induction.

For the timestamp 0, this is trivial. Since $\vec{g}[0]$ is consistent with $O_i(\vec{s}[0])$ (Lemma 4.13), we have $f_i(\vec{g})[0] = O_i(\vec{g}[0]) \cup s_\perp$. Therefore, $\vec{g}[0]$ is a valid state in W_0 when generating $C(f_i(\vec{g}), i)$ as $O(f_i(\vec{g})[0]) = O_i(O_i(\vec{g}[0]))$. This is because: $O_i(\vec{g}[0]) \subseteq f_i(\vec{g})[0]$ indicates $O_i(O_i(\vec{g}[0])) \subseteq O_i(f_i(\vec{g})[0])$; and, $f_i(\vec{g})[0] \subseteq \vec{g}[0]$ indicates $O_i(f_i(\vec{g})[0]) \subseteq O_i(\vec{g}[0])$.

Similarly as proof for Theorem 4.15, for any timestamp $t \in \{0, \dots, |\vec{s}|\}$, the justified perspective of $\vec{g}$ for agent i are generated by Definition 4.6, in which the lt_v for all variables are the same as in $f_i(\vec{s})[t]$ due to Lemma 4.13. Different from that proof, the condition $\vec{g}[t](v) = \perp$ must not hold, as the $\vec{g}$ is a possible world (sequence), which does not contain none value. Thus, the conditions required discussion are: (1) whether the most recent timestamp agent i sees v is t ($lt_v = t$); and, (2) whether agent i believes a not-none value of v ($f_i(\vec{g})[t](v) \neq \perp$). If the first condition holds (agent i sees v at t), the second one must hold (the value of v that agent i believes at t must not be none). This is because the state has been filled with consistent possible values for the none variables, which makes the condition for indirect inferences not exist. For this proof, if the first condition holds, we have $f_i(\vec{g})[t](v) = \vec{g}[t](v)$. This indicates the value of v at timestamp t will not be overridden by possible values when generating $C(f_i(\vec{g}), i)$. If the first condition does not hold, then the value of $f_i(\vec{g})[t](v)$ is equal to $f_i(\vec{g})[t-1](v)$ based on Definition 4.6.

Then, recursively run this proof from t to $t-1$ to the base case, we have $\vec{g} \in C(f_i(\vec{g}), i)$. □

After constructing the possible worlds (sequences) for agent's justified perspectives, we can now provide a complete semantics for our model.

Definition 4.17 (Complete Semantics)**.** The complete semantics for the model M with a given state sequence $\vec{s}$ (with $n+1$ states) can be defined as:

(a)	$(M, \vec{s}) \vDash r(V_r)$	iff	$\pi(\vec{s}[n], r(V_r)) = \mathit{true}$
(b)	$(M, \vec{s}) \vDash \phi \land \psi$	iff	$(M, \vec{s}) \vDash \phi$ and $(M, \vec{s}) \vDash \psi$
(c)	$(M, \vec{s}) \vDash \neg\varphi$	iff	$(M, \vec{s}) \nvDash \varphi$
(d)	$(M, \vec{s}) \vDash S_i v$	iff	$v \in O_i(\vec{s}[n])$
(e)	$(M, \vec{s}) \vDash S_i\varphi$	iff	$\forall \vec{g} \in C(O_i(\vec{s}), i), (M, \vec{g}) \vDash \varphi$ or $\forall \vec{g} \in C(O_i(\vec{s}), i), (M, \vec{g}) \vDash \neg\varphi$
(f)	$(M, \vec{s}) \vDash K_i\varphi$	iff	$(M, \vec{s}) \vDash \varphi \land S_i\varphi$
(g)	$(M, \vec{s}) \vDash H_i\varphi$	iff	$(M, \vec{s}) \vDash B_i\varphi$, or $(M, \vec{s}) \vDash B_i\neg\varphi$
(h)	$(M, \vec{s}) \vDash B_i\varphi$	iff	$\forall \vec{g} \in C(f_i(\vec{s}), i), (M, \vec{g}) \vDash \varphi$

where C is the possible sequence function defined in Definition 4.12.

Any formula in our language $\mathcal{L}(\Sigma)$ is always a relation in some perspectives, which eventually are evaluated by the last state in that perspective. Therefore, Items (e), (f), (g) and (h) are evaluated in the format of (a). Items (b) and (c) are straightforward for logic operator $\land$ and $\neg$ in the language $\mathcal{L}(\Sigma)$.

Items (d), (e) and (f) are the same as in PWP (see Definition 3.13), except the formula is evaluated on a state sequence instead of a single state. While the definition of the new belief operator, Item (h), requires some discussion, Item (g) is straightforwardly dependent on it.

At a high level, the definition of $B_i\varphi$ aims to capture is that agent i believes φ if in its past (including present) it knew φ: that is, $K_i\varphi$ was true. However, this does not capture situations where $\varphi \land S_i\varphi$, such as $S_i\varphi$ and φ are observed as true in different past states, or even φ contains references to variables observed in different states.

For example, consider the proposition $B_a(x + y \geq 0)$, $D_x = D_y = \{-1, 1\}$ and agent a observes $x = 1$ in state s_0, then observes $y = 1$ in state s_1, while not observing x in state s_1 at all. The complete state space for is $\mathcal{S}_c = \{\{x{=}1, y{=}1\}, \{x{=}1, y{=}{-1}\}, \{x{=}{-1}, y{=}1\}, \{x{=}{-1}, y{=}{-1}\}\}$.

It is not the case that $M, s_0 \vDash S_a(x + y \geq 0)$ or $M, s_1 \vDash S_a(x + y \geq 0)$ because agent a does not see the value of y in state s_0 or the value of x in state s_1. However, it seems valid to state that $M, s_1 \vDash B_a(x + y \geq 0)$ because it can remember $x = 1$ from state s_0, and has no evidence to suggest x has changed. Based on the Item (h), $f_a(\vec{s})$ is needed to evaluate the proposition $B_a(x + y \geq 0)$.

In the last state s_1, the perspective function identifies the most recent timestamps in which x and y are seen by agent a, which are 0 and 1 respectively. Then, the retrieval function R retrieves the value of x and y, which are $x = 1$ and $y = 1$. So, the last state in agent a's justified perspective $f_a(\vec{s})$ at s_1 is $\{x{=}1, y{=}1\}$. Then, in the previous state s_0 (initial state), the lt_x and lt_y (following Definition 4.6) identified by the perspective function are 0 and -1. So that, R retrieves x's value is 1, and a's justified perspective at timestamp 0 ($f_a(\vec{s})$ at s_0) is $\{x{=}1, y{=}\perp\}$. That is, $f_a(\vec{s}) = [\{x{=}1, y{=}\perp\}, \{x{=}1, y{=}1\}]$

Then, according to Definition 4.12, for each complete state $g \in \mathcal{S}_c$, applying the function override $g\langle\rangle$ on each state from a's justified perspective with none assignment removed $(f_a(\vec{s})[t] \setminus s_\perp)$, we have $W'_0 = W_0 = \big\{\{x{=}1, y{=}-1\}, \{x{=}1, y{=}1\}\big\}$ and $W'_1 = W_1 = \big\{\{x{=}1, y{=}1\}\big\}$. That is, two possible sequences are formed as $C(f_a(\vec{s}), a) = \{\overrightarrow{sa_1}, \overrightarrow{sa_2}\}$, where: $\overrightarrow{sa_1} = [\{x{=}1, y{=}-1\}, \{x{=}1, y{=}1\}]$ and $\overrightarrow{sa_2} = [\{x{=}1, y{=}1\}, \{x{=}1, y{=}1\}]$.

Then, we have $M, \vec{s} \vDash B_a(x + y \geq 0)$ is equivalent to $M, \overrightarrow{sa_1} \vDash (x + y \geq 0) \land M, \overrightarrow{sa_2} \vDash (x + y \geq 0)$. Then, based on Item (a) in semantics, both formulae hold, which means $M, \vec{s} \vDash B_a(x + y \geq 0)$ holds.

4.3.3.2 Ternary Semantics

Now, we show how to implement our model using ternary logic semantics, based on the ternary semantics from the PWP model in Section 3.3. This semantics offers a polynomial time complexity logic, compared to the complete semantics, which is exponential in the number of states in the problem. It sacrifices completeness for efficiency. The ternary values for propositions are: 0 denotes false, 1 denotes true, and $\frac{1}{2}$ means the truth value is unknown (unable to be proved).

Definition 4.18 (Ternary Semantics)**.** The ternary semantics for the model M with a given state sequence $\vec{s}$ (with n states) can be defined as:

$$
\begin{array}{llcl}
\text{(a)} & T[\vec{s}, r(V_r)] & = & 1 \text{ if } \pi(\vec{s}[n], r(V_r)) = true; \\
 & & & 0 \text{ else if } \pi(\vec{s}[n], r(V_r)) = false; \\
 & & & \frac{1}{2} \text{ otherwise} \\
\text{(b)} & T[\vec{s}, \phi \land \psi] & = & \min(T[\vec{s}, \phi], T[\vec{s}, \psi]) \\
\text{(c)} & T[\vec{s}, \neg\varphi] & = & 1 - T[\vec{s}, \varphi] \\
\text{(d)} & T[\vec{s}, S_i v] & = & \frac{1}{2} \text{ if } v \notin \vec{s}[n] \text{ or } i \notin \vec{s}[n] \\
 & & & 0 \text{ else if } v \notin O_i(\vec{s}[n]) \\
 & & & 1 \text{ otherwise} \\
\text{(e)} & T[\vec{s}, S_i \varphi] & = & \frac{1}{2} \text{ if } T[\vec{s}, \varphi] = \frac{1}{2} \text{ or } i \notin \vec{s}[n]; \\
 & & & 0 \text{ else if } T[O_i(\vec{s}), \varphi] = T[O_i(\vec{s}), \neg\varphi] = \frac{1}{2}; \\
 & & & 1 \text{ otherwise} \\
\text{(f)} & T[\vec{s}, K_i \varphi] & = & T[\vec{s}, \varphi \land S_i \varphi] \\
\text{(g)} & T[\vec{s}, H_i \varphi] & = & \frac{1}{2} \text{ if } T[\vec{s}, \varphi] = \frac{1}{2} \\
 & & & 0 \text{ else if } T[\vec{s}, B_i \varphi] = T[\vec{s}, B_i \neg\varphi] = \frac{1}{2}; \\
 & & & 1 \text{ otherwise} \\
\text{(h)} & T[\vec{s}, B_i \varphi] & = & T[f_i(\vec{s_\perp}\langle\vec{s}\rangle), \varphi]
\end{array}
$$

It is worthy to mention that, compared to the complete semantics, Item (h) in the ternary semantics requires the input sequence $\vec{s}$ to be filled by a non-state sequence $\vec{s_\perp}$. This ensures the input of the JP function is a complete state sequence, considering Item (e) could result in a partial state sequence becoming the input of Item (h). While this would not happen in the complete semantics, as the observations are filled in with possible worlds (Item (e) in Definition 4.17).

This ternary semantics gives $\frac{1}{2}$ value when the targeting state or sequence does not have sufficient information to evaluate. This could cause some of the formulae (non-logical separable formula in Definition 3.17) to be evaluated as 'unknown' ($\frac{1}{2}$). While we argue that the agents might not have a pre-knowledge of all V and D (all possible worlds) from the model, it is reasonable for them to not know the unobserved part of the world. For example, using our motivated problem (Plan 4.2 in Example 4.1), the state sequence $\vec{s}$

would be [2]:

- $\vec{s}[0] = \{(false, false, false, false), p{=}4, q{=}6\}$,
- $\vec{s}[1] = \{(false, \mathbf{true}, false, false), p{=}4, q{=}6\}$,
- $\vec{s}[2] = \{(false, false, false, false), p{=}4, q{=}6\}$,
- $\vec{s}[3] = \{(false, false, false, \mathbf{true}), p{=}4, q{=}6\}$,
- $\vec{s}[4] = \{(false, false, false, \mathbf{true}), p{=}4, q{=}5\}$

Let sequence $\vec{s'}$ only contain s_0 and s_1, which is the state sequence at timestamp 1. The seeing relation S_bS_aq, which is $T[\vec{s'}, S_bS_aq]$ in ternary semantics, is evaluated as $\frac{1}{2}$.

Agent b's observation is generated by the observation function O_b, which is the same as f_i^{NIB} defined in Section 3.1.1:

$$O_b(\vec{s'}) = [\{(false, false, false, false)\}, \{(false, true, false, false)\}]$$

From agent b's observation, we can generate agent a's observation as:

$$O_a(O_b(\vec{s'})) = [\{(false, false, false, false)\}, \{(false, true, false, false)\}]$$

$T[O_b(\vec{s'}), S_aq] = \frac{1}{2}$ because $q \notin O_b(\vec{s'}[1])$. Therefore, $T[\vec{s'}, S_bS_acoin] = 0$. Intuitively speaking, agent b can only see agent a peeking into the box and b cannot see that number q is in the box. However, if this seeing relation is in the complete semantics $(M, \vec{s'} \models S_bS_aq)$, the evaluation is true.

This is because, in the complete semantics, $M, \vec{s'} \models S_bS_aq$ is equivalent to $\forall \vec{g} \in C(O_b(\vec{s'}), b)$, $M, \vec{g} \models S_aq$ by applying item (e), where all worlds (sequences) agent b considers possible are:

$$C(O_b(\vec{s'}), b) = \left\{ \begin{bmatrix} \{(false, false, false, false), p{=}n_1, q{=}n_2\}, \\ \{(false, true, false, false), p{=}n_3, q{=}n_4\} \end{bmatrix} \middle| \begin{array}{r} n_1, n_2, n_3, \\ n_4 \in \{0, \ldots, 99\} \end{array} \right\}$$

[2]In the following example, we denote the value of all *peeking* variables as a tuple of boolean value, with the order as "$peeking_{ap}$,$peeking_{aq}$,$peeking_{bp}$,$peeking_{bp}$" for readability. For example, the initial state would be $\{(false, false, false, false), p{=}4, q{=}6\}$.

Then, according to item (d), the set of all agent a's observations for all sequences in $C(O_b(\vec{s'}), b)$ is generated as the following set:

$$\left\{ \begin{bmatrix} \{(false, false, false, false)\}, \\ \{(false, true, false, false), q{=}n_4\} \end{bmatrix} \middle| \; n_4 \in \{0, \ldots, 99\} \right\}$$

Since q is in the last state of all sequences in the above set, we have $S_a q$ holds for all the sequences we generated for agent b. Thus, $M, \vec{s'} \vDash S_b S_a q$ holds.

As for a belief relation $B_b S_a q$ with the same sequence, $T[\vec{s'}, B_b S_a q] = 1$. This is because belief is evaluated based on the justified perspective of agent b, according to item (h) in Definition 4.18:

$$f_b(\vec{s'}) = \begin{bmatrix} \{(false, false, false, false), p{=}\bot, q{=}\bot\}, \\ \{(false, true, false, false), p{=}\bot, q{=}\bot\} \end{bmatrix}$$

Then, under b's justified perspective, agent a's observations are:

$$O_a(f_b(\vec{s'})) = \begin{bmatrix} \{(false, false, false, false)\}, \\ \{(false, true, false, false), q{=}\bot\} \end{bmatrix}$$

Since q is in $O_a(f_b(\vec{s'}))[1]$, we have $T[f_b(\vec{s'}), S_a q] = 1$. Thus, $T[\vec{s'}, B_b S_a q] = 1$.

4.3.3.3 Complexity

The time complexity for the complete semantics and the ternary semantics is similar to the PWP approach. The only difference is the time complexity for the new justified perspective function.

To evaluate $M, \vec{s} \vDash \varphi$, the worst-case scenario is that φ is a belief formula with the depth of d. Then, the justified perspective function complexity is in $\Theta(d \cdot |V|^2 \cdot |\vec{s}|^3)$, which is for each variable, getting the lt_v, getting $R(\vec{s}, lt_v, v)$, for s in $\vec{s}$ and for each level of nesting from φ. In addition, if the history of all corresponding justified perspectives is well-stored, this can be done in linear time in terms of $|\vec{s}|$. That is, if all corresponding justified perspectives (including all corresponding observations) from timestamp 0 to $t{-}1$ are stored and can be visited in constant time, then, to generate justified perspectives for timestamp t, besides generating $O_i(\vec{s}[n])$, only need to retrieve $R(\vec{s}, lt_v, v)$ (lt_v is from observations), which is linear in terms of $|\vec{s}|$.

For the complete semantics, in $\vec{s}$, each state could have $|V| \times |D|$ possibilities. Thus, the number of possible sequences is $|V \times D|^{|\vec{s}|}$. So, the complexity of the query in the complete semantics is in $\Theta(d \cdot |V|^2 \cdot |\vec{s}|^3 \cdot |V \cdot D|^{|\vec{s}|})$, which is exponential on the input size. While, in the ternary semantics (Definition 4.18), the complexity of epistemic formula evaluation is in the same complexity class as generating the corresponding justified perspectives, assuming Item (a) is in $\Theta(1)$.

4.3.3.4 The Axiomatic System for the JP model

As noted in Section 4.2, there is no underlying definition for our justified belief. So, there is no underlying model to which we can prove soundness or completeness. We can only show our model is sound with respect to **KD45** logic as follows.

Theorem 4.19. *The following axioms hold, making this a **KD45** logic:*

K (Distribution):	$B_i\varphi \land B_i(\varphi \to \psi) \to B_i\psi$
D (Consistency):	$B_i\varphi \to \neg B_i\neg\varphi$
4 (Positive Introspection):	$B_i\varphi \to B_iB_i\varphi$
5 (Negative Introspection):	$\neg B_i\varphi \to B_i\neg B_i\varphi$

Proof. Based on the definition of B_i, $M, \vec{s} \models B_i\varphi$ is equivalent to for all $\vec{g} \in C(f_i(\vec{s}), i)$, such that $M, \vec{g} \models \varphi$. From this, Axiom K is: $M, \vec{g} \models \varphi$ and $M, \vec{g} \models (\varphi \to \psi)$ imply $M, \vec{g} \models \psi$, which holds trivially.

For Axiom D, $M, \vec{s} \models B_i\varphi$ is equivalent to $\forall \vec{g} \in C(f_i(\vec{s}), i),\ M, \vec{g} \models \varphi$. By induction, we have $\nexists \vec{g} \in C(f_i(\vec{s}), i),\ M, \vec{g} \not\models \varphi$, which means $M, \vec{s} \not\models \neg B_i\varphi$. Thus, Axiom D holds.

For Axiom 4, $M, \vec{s} \models B_i\varphi$ is equivalent to for all $\vec{g} \in C(f_i(\vec{s}), i)$, such that $M, \vec{g} \models \varphi$. Based on Theorem 4.15, since $f_i(\vec{s})$ is consistent with any $f_i(\vec{g})$, we have $C(f_i(\vec{g})) \subseteq C(f_i(\vec{s}))$. Because $\vec{g} \in C(f_i(\vec{s}), i), M, \vec{g} \models \varphi$, we have $\vec{g} \in C(f_i(\vec{g})), i), M, \vec{g} \models \varphi$. That is, Axiom 4 holds.

For Axiom 5, $M, \vec{s} \models \neg B_i\varphi$ is $M, \vec{s} \not\models B_i\varphi$, which is effectively $\exists \vec{g} \in C(f_i(\vec{s}), i),\ M, \vec{g} \not\models \varphi$.

This can be proved by considering two conditions: (1) $\exists\vec{g} \in C(f_i(\vec{s}), i),\ M, \vec{g} \not\models \varphi$ and $\nexists\vec{g} \in C(f_i(\vec{s}), i),\ M, \vec{g} \models \varphi$; (2) $\exists\vec{g} \in C(f_i(\vec{s}), i),\ M, \vec{g} \not\models \varphi$ and $\exists\vec{g'} \in C(f_i(\vec{s}), i),\ M, \vec{g'} \models \varphi$.

The first one is effectively $\forall\vec{g} \in C(f_i(\vec{s}), i),\ M, \vec{g} \not\models \varphi$. According to Theorem 4.16, for all $\vec{g} \in C(f_i(\vec{s}), i)$, there exists a $\vec{p}$ (effectively $\vec{p} = \vec{g}$ as $\vec{g} \in C(f_i(\vec{g}), i)$) such that $M, \vec{p} \not\models \varphi$.

The second one requires some discussion. It only happens when $\vec{g} \neq f_i(\vec{s})$ and $\vec{g'} \neq f_i(\vec{s})$. Let the length of the sequence $|\vec{s}|$ be n. This is caused either by the possible values filled in for the variables that are not in $O_i(\vec{s}[n])$ or the variables such that $O_i(\vec{s}[n])(v) = \perp$.

The first condition is straightforward. The variables not in $O_i(\vec{s}[n])$ are also not in $O_i(\vec{s}[n])$ (Lemma 4.13), which means the possible values that generated $\vec{g}$ is also valid when generating all possible sequence for $C(f_i(\vec{g'}), i)$, which make this Axiom holds. While, the second condition is not possible in complete semantics. The input sequence for Item (g) and (h) are state sequence with complete not-none value assignments based on Definitions 4.17 and 4.12. Thus, the observation of any variables v for any timestamp t could not be none. Overall, Axiom 5 holds for all possible conditions. □

Recall in Section 2.2.6, Voorbraak [93] proposed another bridge axiom (Theorem 2.8) between knowledge and belief ($B_iK_i\varphi \rightarrow K_i\varphi$), which is claimed to be the unwanted axiom by Gochet and Gribomont [81]. As reasoned at the end of Section 4.2, this axiom becomes valid in our definition of knowledge and belief (justified belief). Here, we formalised it by proposing the following theorem to show this unwanted axiom holds for both the complete semantics and the ternary semantics.

Theorem 4.20. *Given a sequence of states $\vec{s}$ with a length of $n+1$, we have $O_i(f_i(\vec{s})[n]) = O_i(\vec{s}[n])$.*

The above theorem is tricky to prove since $f_i(\vec{s})[n]$ does not guarantee to be a subset of $\vec{s}[n]$ (we cannot use the monotonicity of the observation function directly).

Proof. We prove this by contradiction.

Firstly, according to Theorem 4.10, we have $O_i(\vec{s}[n]) \subseteq O_i(f_i(\vec{s})[n])$. This means all assignments in $O_i(\vec{s}[n])$ are also in $O_i(f_i(\vec{s})[n])$.

Then, assuming there exists an assignment $(v = e)$, such that $v = e \in O_i(f_i(\vec{s})[n])$ and $v = e \notin O_i(\vec{s}[n])$.

According to the definition of the JP function (Definition 4.6), this assignment $(v = e)$ must be in one of the following conditions:

1. $v \in O_i(\vec{s}[n])$
2. $v \notin O_i(\vec{s}[n])$

For Condition (1), since $v \in O_i(\vec{s}[n])$, we have either $O_i(\vec{s}[n])(v) = f_i(\vec{s}[n])(v)$ or $O_i(\vec{s}[n])(v) \neq f_i(\vec{s}[n])(v)$. The former cannot be the case as $f_i(\vec{s}[n])(v) = O_i(f_i(\vec{s})[n])(v)$ ($v = e \in O_i(f_i(\vec{s})[n])$ from our assumption), which results in $O_i(f_i(\vec{s})[n])(v) = O_i(\vec{s}[n])(v)$, which violates our assumption ($v = e \notin O_i(\vec{s}[n])$). The latter ($O_i(\vec{s}[n])(v) \neq f_i(\vec{s}[n])(v)$) results in $\vec{s}[n](v) = \perp$ (Theorem 4.8), which means $O_i(\vec{s}[n])(v) = \perp$. Then, according to Theorem 4.9 (we have both $v \in O_i(\vec{s}[n])$ and $\vec{s}[n](v) = \perp$), we have $f_i(\vec{s}[n])(v) = \perp = O_i(\vec{s}[n])(v)$, which violates our assumption ($v = e \notin O_i(\vec{s}[n])$).

For Condition (2), when $v \notin O_i(\vec{s}[n])$, Line 3 in Definition 4.6 ensures the agent i has a belief of v but still cannot see v in i's justified perspectives. Thus, v is also not in $O_i(f_i(\vec{s})[n])$, which violates our assumption ($v = e \in O_i(f_i(\vec{s})[n])$).

Therefore, we have considered all conditions for the above assumption and all of them are proved to contain contractions. Thus, the assumption does not hold, which proves Theorem 2.8. $\square$

Now, with the above theorem, we are able to show that the unwanted axiom ($B_i K_i \varphi \to K_i \varphi$) in Theorem 2.8 holds. Using the ternary semantics as an example. For any given sequence $\vec{s}$ with its length as $n+1$, $T[\vec{s}, B_i K_i \varphi] = 1$ equals $T[f_i(\vec{s}), S_i \varphi \wedge \varphi] = 1$, which is $T[O_i(f_i(\vec{s})), \varphi] = 1$ and $T[f_i(\vec{s}), \varphi] = 1$. Based on Theorem 4.20, we have $O_i(f_i(\vec{s}))[n] = O_i(\vec{s})[n]$, which means $T[O_i(f_i(\vec{s})), \varphi] = 1$ indicates $T[O_i(\vec{s}), \varphi] = 1$. In addition, since the observation function is contractive ($O_i(s) \subseteq s$), $\pi(O_i(\vec{s})[n], \varphi) = 1$ (it is not $\frac{1}{2}$ means all variables needed to evaluate φ are in $O_i(\vec{s})[n]$) indicates $\pi(\vec{s}[n], \varphi) = 1$. With both $T[O_i(\vec{s}), \varphi] = 1$ and $T[\vec{s}, \varphi] = 1$ (from $\pi(\vec{s}[n], \varphi) = 1$), we have $T[\vec{s}, K_i \varphi] = 1$.

Then, to sum up, the axiomatic system that the JP model follows is all KB axioms (except Axiom **KB2**) in Definition 2.7. The

Quoted text: "

Definition 4.21 (Axioms for KB in the JP model)**.** Presented are 9 axioms:

K: (Knowledge)	$\big(K_i\varphi \land K_i(\varphi \rightarrow \psi)\big) \rightarrow K_i\psi$
T: (Knowledge)	$K_i\varphi \rightarrow \varphi$, $\varphi \rightarrow \neg K_i\neg\varphi$
4: (Knowledge)	$K_i\varphi \rightarrow K_iK_i\varphi$
5: (Knowledge)	$\neg K_i\varphi \rightarrow K_i\neg K_i\varphi$
K: (Belief)	$\big(B_i\varphi \land B_i(\varphi \rightarrow \psi)\big) \rightarrow B_i\psi$
D: (Belief)	$\neg B_i\ false$
4: (Belief)	$B_i\varphi \rightarrow B_iB_i\varphi$
5: (Belief)	$\neg B_i\varphi \rightarrow B_i\neg B_i\varphi$
KB1:	$K_i\varphi \rightarrow B_i\varphi$

"

Since the JP model is an extension of the PWP model, axioms for knowledge (**KT45**, which is also known as **S5**) are proved to hold in Section 3.2 (for the complete semantics) and Section 3.3. While axioms for belief (**KD45**) are proved in Theorem 4.19. The bridge axiom $KB1$ holds trivially due to Theorem 4.7. As we argue above, having all of Axiom **KB1**, **Axiom D**, and Axiom **5** (causing the unwanted axiom) is not an issue in the JP model. This is because our definitions of the knowledge and (justified) belief are more strict. If an agent justifiably believes that they know something, then they actually know it. Otherwise (if they do not hold that knowledge), they should not justifiably believe they hold that knowledge. In other words, the only way $B_iK_i\varphi$ can be true is if $K_i\varphi$ holds as well. That is, although the unwanted axiom in Theorem 2.8 holds in our model, it does not represent a 'causal' relation between believing knowledge and knowledge.

4.4 Implementation

In this section, we show how we model the epistemic planning problem, including the encoding for epistemic logic formulae, and efficient pruning techniques that can enhance the performance of the search algorithm.

4.4.1 Problem Formalization

Since the JP model works with state sequences, the truth values of agents' knowledge or belief formulae do not depend only on the current state, but depend on the whole state sequence. Thus, firstly, we have to define non-Markovian in planning. Then, we can extend the formulation to incorporate the external function in F-STRIPS, similar to how it is in the PWP approach (Section 3.5). At last, we provide a formal encoding to represent any epistemic planning instance with a language that is extended from Planning Domain Definition Language (PDDL).

4.4.1.1 Non-Markovain Fully Observable Deterministic Domain

There is no existing definition of classical planning with a non-Markovian setting, so we have to define it on our own. We now introduce the basic semantics model of a Non-Markovian Fully Observable Deterministic Domain (NMFODD). An NMFODD is an extension of the classical planning problem introduced in Section 2.1.2.1, which is inspired by the definition of Non-Markovian Fully Observable Non-deterministic Domain (NMFOND) [47]. The action function, transition function, and goal function depend on the entire track of history of the states.

An NMFODD problem is represented by a tuple:

$$P = (\mathcal{S}, s_0, A, a^{\rightarrow}, t^{\rightarrow}, g^{\rightarrow}),$$

where:

- $\mathcal{S}$ is the state space (the set of all possible states), and $\vec{\mathcal{S}}$ denotes the sequence space;
- s_0 is the initial state ($s_0 \in \mathcal{S}$);
- A is the set of all actions;
- $a^{\rightarrow}$ is the *action function*, which maps the current state sequence to a set of available actions: $a^{\rightarrow} : \vec{\mathcal{S}} \rightarrow \{A\}$;
- $t^{\rightarrow}$ is the *transition function*, which takes the current sequence and an action and returns the next state: $t^{\rightarrow} : \vec{\mathcal{S}} \times A \rightarrow \mathcal{S}$;

- $g^{\rightarrow}$ is the *goal function*, which determines whether the goal is achieved given the current sequence: $g^{\rightarrow} : \vec{\mathcal{S}} \rightarrow \{true, false\}$.

4.4.1.2 NMFODD problems in F-STRIPS

Similarly to the PWP approach, in order to model an epistemic planning problem using the JP model defined in the previous section, we have to introduce an external function in NMFODD. Here, we provide an extended version of F-STRIPS (in Section 2.1.3.3), namely Non-Markovian Functional STRIPS (NM-F-STRIPS). Any problem instance in NMFODD can be represented with NM-F-STRIPS as a tuple:

$$P = (\mathcal{L}_F, \mathcal{O}_F^{\rightarrow}, \mathcal{I}_F, \mathcal{G}_F^{\rightarrow}),$$

where: $\mathcal{L}_F$ and $\mathcal{I}_F$ are the same as in F-STRIPS; $\mathcal{O}_F^{\rightarrow}$ represents all transitions and $\mathcal{G}_F^{\rightarrow}$ represents all goal propositions. Both $\mathcal{O}_F^{\rightarrow}$ and $\mathcal{G}_F^{\rightarrow}$ include non-Markovian propositions (could be in both precondition and effect for the operators). Those non-Markovian propositions are evaluated with the input of the whole state sequence in the search path.

4.4.1.3 Epistemic Planning problem in NM-F-STRIPS

Now, we can provide a formal formulation for modeling any Epistemic Planning instance using the JP model in NM-F-STRIPS, namely EP-NM-F-STRIPS.

Given any instance in EP-NM-F-STRIPS, let the signature of this instance be $\Sigma = (Agt, V, D, \mathcal{R})$, the language as $\mathcal{L}_{KB}(\Sigma)$, and the JP model for this instance as $M = (Agt, V, D, \pi, O_1, \ldots, O_k)$, this instance can be represented by a tuple:

$$P = (Agt, V, D, E, \mathcal{O}_F^{\rightarrow}, \mathcal{I}_F, \mathcal{G}_F^{\rightarrow}, f^{\rightarrow}),$$

where:

- Agent set Agt and variable V are from signature Σ (also the same as in model M);

- All variable's domain D is from model M (same D in Σ with a special 'None' value $\bot$), where $V \times D$ forms the state spaces $\mathcal{S}_c \subsetneq \mathcal{S}$, sequence space $\vec{\mathcal{S}}$ and complete sequence space $\vec{\mathcal{S}_c}$;
- E is the set of all involved epistemic formulae ($\epsilon \subset \mathcal{L}_{KB}(\Sigma)$);
- $\mathcal{O}_F^{\rightarrow}$ is the set of all operators $o^{\rightarrow}$ ($o^{\rightarrow} : \vec{\mathcal{S}_c} \rightarrow \mathcal{S}_c$);
- $\mathcal{I}_F$ is the initial state ($\mathcal{I}_F \in \mathcal{S}_c$);
- $\mathcal{G}_F^{\rightarrow}$ is the set of goal conditions, which could contain variable assignments (ontic goal conditions) and epistemic formulae (epistemic goal conditions from E);
- $f^{\rightarrow}$ is the external function that implements M and the ternary semantics (Definition 4.18) to evaluate the epistemic formula based on the state sequence from the current search path ($f^{\rightarrow} : \vec{\mathcal{S}} \times E \rightarrow \{true, false\}$).

An epistemic formula $\epsilon \in E$ can be in the goal conditions ($\epsilon \in \mathcal{G}_F^{\rightarrow}$) and preconditions of an operator $o^{\rightarrow} \in \mathcal{O}_F^{\rightarrow}$ ($ep \in Pre(o^{\rightarrow})$).

4.4.1.4 Functional PDDL Encoding

Now, we propose a PDDL-like language encoding to describe any epistemic planning instance $P = (Agt, V, D, E, \mathcal{O}_F^{\rightarrow}, \mathcal{I}_F, \mathcal{G}_F^{\rightarrow}, f^{\rightarrow})$. The base of this language used is PDDL2.1 [52], which allows the usage of functions, while an external function (from F-STRIPS) is added. Thus, we name this language Functional Planning Domain Definition Language (F-PDDL).

The signature of V is defined as functions. An example of functions in the NIB domain (Example 4.1) is given as follows:

```
(:functions
    (peeking ?i - agent ?n - number)
    (value ?n - number)
)
```

Code Example 4.1: F-PDDL Functions Example

The domain D for each variable is defined as *ranges*:

```
(:ranges
    (value integer [0,99])
    (peeking enumerate ['t','f'])
)
```

Code Example 4.2: F-PDDL Ranges Example

The set of all actions ($\mathcal{O}_{\vec{F}}$) is represented by action schema in PDDL. An example (ontic) action "**return**" is given as follows:

```
(:action return
    :parameters (?i - agent ?n - number)
    :precondition (and
        (= (peeking ?i ?n) 't'))
    )
    :effect (and
        (assign (peeking ?i ?n) 'f')
    )
)
```

Code Example 4.3: F-PDDL Action Example: "**return**"

Although the given example is only related to ontic state, the action schema allows epistemic formulae in the precondition and effect. More examples can be found in Section 4.5.

An example goal conditions for the same coin example can be represented as follows:

```
(:goal
    (and
        (= (value q) 5)
        (= (@ep ("+ b [b]") (= (value q) 5)) ep.true)
        (= (@ep ("+ b [a]") (= (value q) 6)) ep.true)
        (= (@ep ("+ b [b] + b [a]") (= (value q) 6)) ep.true)
    )
)
```

Code Example 4.4: F-PDDL Goal Example using External Function @*ep*

The above goal conditions contain one ontic goal condition ($q = 5$) and three epistemic goal conditions ($B_b q = 5$, $B_a q = 6$ and $B_b B_a q = 6$). The external function is represented by `(@ep ("<query>") (<`φ`>))`, where `"<query>"` is all epistemic operators (from S, K, H and B in language $\mathcal{L}_{KB}(\Sigma)$) in the epistemic formula, where the φ is the $r(V_r)$, $+$ and $-$ represent affirmation and negation.

Besides embedding a normal epistemic formula, epistemic planning would benefit more from a more abstract representation. Compared to STRIPS language, PDDL provides more flexibility using action schemas instead of specifying propositions in operators. Specifically, in order to do so in PDDL, action effects can be modeled in the declarative format in terms of updating the state variables based on the previous state (Markovian). However, the actions when modeling problems in EP-NM-STRIPS could be non-Markovian. Some action's effects might be updating the value of a variable to what the acting agent believes (non-Markovian effects). Thus, we have to use another type of external function ($@jp$) to retrieve the value from the agent's justified perspective. An example code for the same goal conditions as in Code Example 4.4 is provided as follows.

```
(:goal
    (and
        (= (value q) 5)
        (= (@jp ("b [b]") (value q)) 5)
        (= (@jp ("b [a]") (value q)) 6)
        (= (@jp ("b [b] b [a]") (value q)) 6)
    )
)
```

Code Example 4.5: F-PDDL Goal Example using External Function $@jp$

This goal set checks goal conditions for a sequence $\vec{s}$ with a length of $n+1$ by:

- $\vec{s}[n](q) = 5$;
- $f_b(\vec{s})[n](q) = 5$;
- $f_a(\vec{s})[n](q) = 6$;
- $f_a(f_b(\vec{s}))[n](q) = 6$.

In the above examples, the usage of the external function $@jp$ is similar to $@ep$, while a more intuitive example can be found in Section 4.5.1.3.

4.4.2 Planner

As there is no existing planner that can directly solve an EP-NM-F-STRIPS instance, we have to create one. One potential solution is to adapt a planner that can solve F-STRIPS instances by: 1), fitting the input of the external function with the current search path; 2), introducing the epistemic formula in language; 3), implementing the JP model, including ternary semantics (with a history of all generated justified perspectives for efficiency). However, all the classical search algorithms would not work on solving an EP-NM-F-STRIPS instance because even the simplest pruning procedure, *duplication elimination* (pruning visited state), would not work due to the Non-Markovian assumption. Therefore, we have to develop our own planner and algorithms. To ensure completeness, the basic search algorithm used is Breadth First Search (BFS) without duplication elimination. Although duplicate elimination is impossible for solving a general NM-F-STRIPS instance, as the searched path will never be repeated, there are some optimizations we can use to solve EP-NM-F-STRIPS, as it is a subset of general NM-F-STRIPS.

4.4.2.1 Duplication Elimination

Duplication elimination is important to our planner, not only because of the efficiency it provides, but also because it allows the planner to prove a problem instance is unsolvable. Since the JP model uses the sequence of the states, the search cannot use the current state itself for duplication elimination. As the agent's knowledge and belief are formed from the state sequence through justified perspectives, a dictionary of perspectives is used as the identification for this sequence.

We now define the dictionary keys of any epistemic formula as its corresponding perspective functions in a high-order function format.

Definition 4.22 (High-Order Epistemic Function)**.** Given an agent set (size of k) from the signature $Agt \in \Sigma$, let i and j be any agents in Agt, O_i be any observation function in $O_0, \ldots, O_k$, f_i be any justified perspective function in $f_0, \ldots, f_k$, and *empty* as a special function:

1. $O_i \cdot O_j(\vec{s}) = O_j(O_i(\vec{s}))$
2. $f_i \cdot O_j(\vec{s}) = O_j(f_i(\vec{s}))$
3. $f_i \cdot f_j(\vec{s}) = f_j(f_i(\vec{s}))$
4. $O_i \cdot f_j(\vec{s}) = f_j(\vec{s_\perp}\langle O_i(\vec{s})\rangle)$
5. $empty(\vec{s}) = \vec{s}$

This definition is straightforward. Since both the observation function O and the justified perspective function f take a state sequence as input, they can compact (nest) freely, except the input of the justified perspective function needs to be a complete state sequence. Thus, when compacting a justified perspective function on an observation function, we need to fill in the result sequence of the observation function (could be a partial state sequence) with 'None' value $\perp$ by using the state override function (in Definition 4.1). The special function *empty* is introduced as the global perspective, as well as the base case when generating all perspective keys in the following definition.

Definition 4.23 (Perspective Keys PK)**.** Given any formula φ, its perspective keys , $PK(\varphi)$, is:

$PK(\varphi) = \{pk(\varphi)\}$, where:

$$pk(\varphi) = \begin{cases} O_i \cdot pk(\psi) & \varphi \text{ is in the format of } S_i\psi \\ pk(\psi), O_i \cdot pk(\psi) & \varphi \text{ is in the format of } K_i\psi \\ f_i \cdot pk(\psi) & \varphi \text{ is in the format of } B_i\psi \text{ or } H_i\psi \\ empty & \text{otherwise} \end{cases}$$

Thus, any ontic formula will have the key as an empty string, which indicates the global perspectives. The keys of any epistemic formula, using $B_iS_j\varphi$ as an example, will perspective function names of all its corresponding epistemic operators, $f_i \cdot O_j \cdot empty$ [3]. While the keys of a formula $B_iK_j\varphi$ are $\{f_i \cdot O_j, f_i\}$. Since the keys of any epistemic formula are a high-order function of its corresponding perspectives, we can use keys to generate its perspectives.

Definition 4.24 (Duplication Pruning Set Function)**.** Given an EP-NM-F-STRIPS problem instance as $P = (Agt, V, D, E, \mathcal{O}_F^{\rightarrow}, \mathcal{I}_F, \mathcal{G}_F^{\rightarrow}, f^{\rightarrow})$ and the current expanding sequence as $\vec{s}$, let $E' \subseteq E$ be $E' = \mathcal{G}_F^{\rightarrow} \cup PRE(\mathcal{O}_F^{\rightarrow})$, where $PRE(\mathcal{O}_F^{\rightarrow}) = \bigcup_{o^{\rightarrow} \in \mathcal{O}_F^{\rightarrow}} pre(o^{\rightarrow})$ then we can define the Duplication Pruning Set (DPS) function ($DPS : PK \rightarrow \mathcal{P}(PK \times S)$) as follows:

$$DPS(\vec{s}) = \{key \rightarrow key(\vec{s})[n] \mid key \in \bigcup_{\epsilon \in E'} PK(\epsilon)\}$$

Thus, the duplication pruning set for any given sequence $\vec{s}$ is a dictionary that contains keys and the last state of corresponding perspectives based on these keys.

[3]The high-order function name $f_i \cdot O_j \cdot empty$ is effectively the same as $f_i \cdot O_j$ (Definition 4.22). Thus, for simplicity, we omit the compaction of $\cdot empty$ in the following content.

Theorem 4.25. *For any two given sequences $\vec{s_1}$ and $\vec{s_2}$ with length of n_1+1 and n_2+1 respectively, if $DPS(\vec{s_1}) = DPS(\vec{s_2})$, there does not exist a formula $\varphi \in \mathcal{L}_{KB}(\Sigma)$ such that $T[\vec{s_1}, \varphi] \neq T[\vec{s_2}, \varphi]$.*

Proof. Proof by contradiction.

Assuming $DPS(\vec{s_1}) = DPS(\vec{s_2})$, there exists a formula $\varphi \in \mathcal{L}_{KB}(\Sigma)$ such that, $T[\vec{s_1}, \varphi] \neq T[\vec{s_2}, \varphi]$. We prove this inductively on the structure of φ based on the cases in the ternary semantics (Definition 4.18).

Firstly, let us discuss the base cases. For any epistemic formula φ in language $\mathcal{L}_{KB}(\Sigma)$, after applying our ternary semantics for any amount of time, it would eventually come in the format of $r(V_r)$ or $S_i v$, which matches Case (a) or Case (d) in the ternary semantics.

Case (a): This is straightforwardly evaluated based on the last state of the sequence, where the sequence is generated with some key $pk(\varphi) \in PK(\varphi)$. If $T[pk(\varphi)(\vec{s_1}), r(V_r)] \neq T[pk(\varphi)(\vec{s_2}), r(V_r)]$, then we have $\pi(pk(\varphi)(\vec{s_1})[n_1], r(V_r)) \neq \pi(pk(\varphi)(\vec{s_2})[n_2], r(V_r))$. So that, we also have $pk(\varphi)(\vec{s_1})[n_1] \neq pk(\varphi)(\vec{s_2})[n_2]$. Due to $pk(\varphi)(\vec{s_1})[n_1] = DPS(\vec{s_1})[pk(\varphi)]$ and $pk(\varphi)(\vec{s_2})[n_2] = DPS(\vec{s_2})[pk(\varphi)]$, we have $DPS(\vec{s_1}) \neq DPS(\vec{s_2})$, which is contradictory to our assumption.

Case (d): In this case, we have two perspective keys to evaluate, $pk(\varphi) \in PK(\varphi)$ and $pk(\varphi)' \in PK(\varphi)$, where $pk(\varphi)' = pk(\varphi) \cdot O_i^{-1}$. Firstly, if $T[pk(\varphi)'(\vec{s_1}), S_i v]$ or $T[pk(\varphi)'(\vec{s_2}), S_i v]$ is $\frac{1}{2}$ and the other is not (assuming $T[pk(\varphi)'(\vec{s_1}), S_i v] = \frac{1}{2}$), then, we have either $i \notin pk(\varphi)'(\vec{s_1})[n_1]$ or $v \notin pk(\varphi)'(\vec{s_1})[n_1]$. Since $T[pk(\varphi)'(\vec{s_2}), S_i v] \neq \frac{1}{2}$, both i and v are in $pk(\varphi)'(\vec{s_2})[n_2]$. Thus, we have $pk(\varphi)'(\vec{s_1})[n_1] \neq pk(\varphi)'(\vec{s_2})[n_2]$. Due to $pk(\varphi)'(\vec{s_1})[n_1] = DPS(\vec{s_1})[pk(\varphi)']$ and $pk(\varphi)'(\vec{s_2})[n_2] = DPS(\vec{s_2})[pk(\varphi)']$, we have $DPS(\vec{s_1}) \neq DPS(\vec{s_2})$, which is contradictory to our assumption. Secondly, if both $T[pk(\varphi)'(\vec{s_1}), S_i v]$ and $T[pk(\varphi)'(\vec{s_2}), S_i v]$ do not equal to $\frac{1}{2}$, we have exact one of them is 1 and another is 0 (assuming $T[pk(\varphi)'(\vec{s_1}), S_i v] = 1$). Since $T[pk(\varphi)'(\vec{s_1}), S_i v] = 1$ and $T[pk(\varphi)'(\vec{s_2}), S_i v] = 0$, we have $v \in pk(\varphi)(\vec{s_1})[n_1]$ and $v \notin pk(\varphi)(\vec{s_2})[n_2]$. Thus, we have $pk(\varphi)(\vec{s_1})[n_1] \neq pk(\varphi)(\vec{s_2})[n_2]$ Due to $pk(\varphi)(\vec{s_1})[n_1] = DPS(\vec{s_1})[pk(\varphi)]$ and $pk(\varphi)(\vec{s_2})[n_2] = DPS(\vec{s_2})[pk(\varphi)]$, we have $DPS(\vec{s_1}) \neq DPS(\vec{s_2})$, which is contradictory to our assumption.

Case (b) and **Case (c)**: Conjunction and negation are trivial, we just apply the semantics and it will match one of the other cases.

Case (e) and **Case (g)**: Those can be proved with the same reasoning process as Case (d).

Case (f) and **Case (h)**: Those can be proved by applying the semantics, Case (f) becomes two formulae and Case (h) becomes one formula, which can be recursively matched with any of the cases in this proof. Eventually, they will match the base cases. □

4.4.2.2 Pruning by "Have No Belief" (HNB)

The justified perspectives of some epistemic planning domains can be reasoned with direct observation, while others require indirect inference. To differentiate those two types, a definition for the domains that require indirect inference is presented as follows:

Definition 4.26. Let all Epistemic Planning Domains be EPD, domains that require indirect inference EPD_{II} can be defined as:

$$EPD_{II} = \left\{ M \;\middle|\; \begin{array}{l} \forall M = (Agt, V, D, \pi, O_1, \ldots, O_k) \in EPD, \\ \text{where: } \exists s_c \in \mathcal{S}_c, \\ \quad \exists v \in V, \\ \quad \exists i \in Agt, \\ \quad \text{if } v \notin O_i(s_c), \\ \qquad \exists e \in D_v \text{ such that } v \in O_i\left(s_c\langle\{v{=}e\}\rangle\right) \end{array} \right\}$$

In addition, the set of epistemic planning domains that only requires direct observation EPD_{DO} can be represented by $EPD_{DO} = EPD \setminus EPD_{II}$.

The nature of any problem instance in EPD_{DO} ensures that once the "have seen" relation $H_i\varphi$ becomes true, it will never become false again. In other words, any epistemic goal relation that contains "Have No Belief" (HNB) must stay true for all time. Once it becomes false, it will never be true again. Thus, pruning by HNB is another optimization that can be done.

The definition of the HNB goal formula is abstract. An epistemic formula that can be considered as an HNB formula, if and only if, there is an odd number of negations that appear before the last belief operator H, such as $\neg H_i\varphi$, or $\neg B_i H_j \varphi$.

4.4.2.3 Search Algorithm

As the search space for any NM-F-STRIPS state space is unbounded, any depth-based search algorithm, such as Depth First Search (DFS) would not be complete. Thus, we choose BFS as our baseline, namely `bfs`. In addition, in order to prove unsolvability for the unsolvable instances, we embedded duplication elimination (Section 4.4.2.1) into `bfs` as `bfsdc`. Besides, pruning by HNB is also included in the search `bfsdcu`.

There is no existing efficient heuristic function for solving problems in NMFODD as far as we know. The most common heuristic function, *Delete Relaxation Heuristic* [72], is not suitable to use, as it affects the justified perspective of agents. In the delete relaxed problem, when an assignment is never false, it cannot form the desired false-belief.

Since relaxation is the principle to design a good heuristic function [40], the most efficient and admissible general heuristic function we proposed is *Precondition and Delete Relaxation* (PDR) heuristic function (h_{PDR}). The idea is to remove the precondition list and delete list for all actions in an EP-NM-F-STRIPS problem instance, which becomes $P_{\neg Pre\&\neg Del} = (Agt, V, D, E, \mathcal{O}^{\rightarrow}_{F_{\neg Pre\&\neg Del}}, \mathcal{I}_F, \mathcal{G}^{\rightarrow}_F, f^{\rightarrow})$. The summation of all actions' costs in an optimal plan to solve $P_{\neg Pre\&\neg Del}$ is the value of h_{PR}.

As Bonet and Geffner mentioned, calculating such a heuristic (h_{PDR}) is as hard as solving the problem itself. Thus, we proposed Goal-Counting heuristic h_{GC} to approximate h_{PSR} as follows:

Definition 4.27 (Goal-Counting Heuristic)**.** Let $P = (Agt, V, D, E, \mathcal{O}^{\rightarrow}_F, \mathcal{I}_F, \mathcal{G}^{\rightarrow}_F, f^{\rightarrow})$ be the planning instance, $G_{epistemic} \subseteq \mathcal{G}^{\rightarrow}_F$ be the set of epistemic goal conditions and $G_{ontic} \subseteq \mathcal{G}^{\rightarrow}_F$ be the set of ontic goal conditions. The value of the goal counting heuristic for a given sequence $\vec{s}$ can be defined as:

$$h_{GC}(\vec{s}) = \Big|\{\epsilon | \epsilon \in G_{epistemic},\ \neg f^{\rightarrow}(\vec{s}, \epsilon)\}\Big| + \Big|\{v | v = e \in G_{ontic},\ \vec{s}[n](v) \neq e\}\Big|,$$

where the length of the sequence $\vec{s}$ is $n + 1$.

Thus, we used h_{GC} with two standard heuristic search algorithms, A* and Greedy Best First Search, namely `astar` and `greedy`, which both are extensions from `bfsdcu`.

4.5 Experiments

To demonstrate the effectiveness and expressiveness of our model, we run experiments on both the existing benchmark problems for comparison with another state-of-the-art approach and newly proposed benchmarks that are difficult to solve by other approaches. In addition, we also design and run large-scale experiments on synthetic versions of problems to explore how the features, such as the number of agents, the depth of epistemic formulae, the number of epistemic formulae in the goal condition and the length of the solution, affect the performance of each proposed search algorithm (`bfs`, `bfsdc`, `bfsdcu`, `astar` and `greedy`) of our planner as well as the performance of epistemic formulae reasoning with the JP model in external functions ($f^{\rightarrow}$).

4.5.1 Benchmark Experiments

In this section, we experiment on the common benchmark epistemic planning problems from the PDKB approach [88], as well as some trickier domains, which are either too complex or impractical to be modeled by other approaches. The benchmarks we used from PDKB approaches include *Selective Communication in Grid* (SCs), *Corridor* and *Grapevine*. The domain *Thief* is omitted, since finding the plan itself in it cannot reflect the full ability of epistemic reasoning. Since there are two roles in this domain, guard and thief, and their objectives are conflictive, the plan that meets anyone's objective would make no sense to the other under the centralized planning setting (as discussed in Assumption 8). The implementation of the benchmarks from the PDKB planner is used as a comparison.

Besides, as one of the benchmarks in many works [117, 140], we also include the *Coin* domain. Unfortunately, we did not find an attempt for the coin domain of the PDKB approach. In addition, BBL is included besides the benchmarks. Similarly, as mentioned in Section 3.6.2, two-dimensional line of sight evaluation is not suitable to be implemented in PDKB (propositional).

The viability experiments are performed on a Linux machine with 8 CPUs (Intel Core i7-10510U CPU 1.80 GHz $\times$ 8) and 16 gigabytes (GB) memory. However, since the planner does not have any parallel implementation, the full power of the machine is not reached to eliminate outlier performance. The timeout is set to be 600 seconds, and the memory limit is 8GB for both approaches. The search algorithm used is `greedy` using goal counting as a heuristic function (mentioned in Section 4.4.2.3). The source code, benchmarks, and experiment results can be found online at:

`https://github.com/guanghuhappysf128/bpwp`

Throughout this section, we will use the following notation:

- Agt: the set of agents in the problem instance;
- d: the maximum depth of any nested epistemic formulae in the problem instance;
- $\mathcal{G}$: the set of goal conditions in the problem instance;
- $\vec{p}$: the computed sequential plan; and,
- $|Gen|$ and $|Exp|$: the number of nodes generated and expanded during the search.

4.5.1.1 Selective Communication (SC)

The domains SCs were initially defined by Alshehri et al. [124] (inspired by Wu et al. [132]) where agents' task is cooperatively searching for survivors. They defined 5 different scenarios — the first 4 are with different capabilities of moving and communication, while the last one alters those agents' ability of perception. Specifically, the 5 scenarios are:

1 *Non-epistemic Goal* (**!epgoal**) scenario: It is a simple scenario where the goal is non-epistemic, requiring agents to search through the entire grid.

2 *Epistemic Goal* (**epgoal**) scenario: In addition to searching the entire grid, the goal also requires all agents to believe the locations of all survivors. During this process, the communication between agents are on a public channel that is visible to everyone.

3 *Broadcast Communication* (**board**) scenario: In this scenario, a commander, an agent who remains stationary, is designated. The communication is the same as **epgoal**. The goal is to have the entire grid searched and only the commander to believe the locations of all survivors.

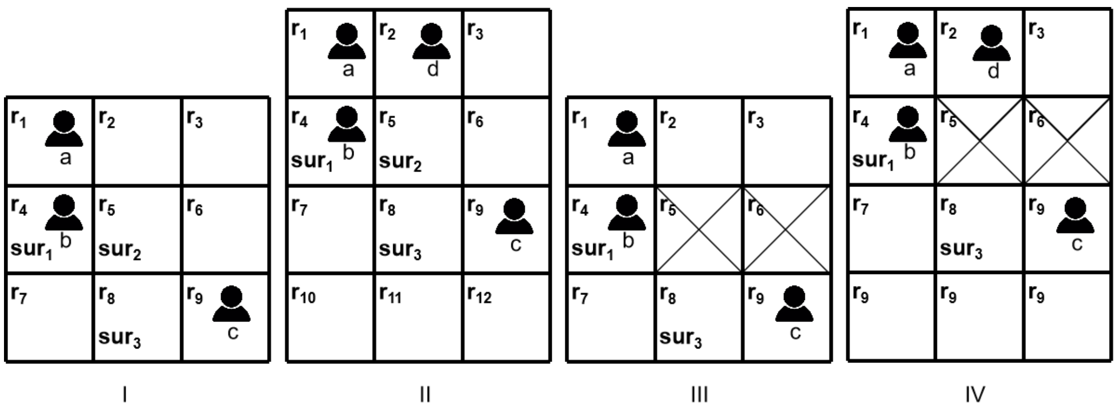


Figure 4.2: Example layouts and initial states for Grid domain.

4 *Non-Broadcast Communication* (**!board**) scenario: All is the same as **board**, except only one of the agent can communicate to the commander.

5 *Blocked Cells* (**blocked**) scenario: The communication is the same as **epgoal**, while blocked cells are introduced. The agent has to sense the next cell and believe it is unblocked before moving into it. Agents can communicate not only the location of survivors but also whether a cell is blocked or not to improve efficiency of overall plan (avoid every agent sense a room once, etc). In addition to the epistemic goals that every agent believes the locations of all survivors, specific ending location for each agent is design to verify the plan efficiency improvements on communication of blocked cells.

They implement the above 5 scenarios on 2 different domains: *Grid* and *Block World for Team* (BW4T). Since the rule of communication, which is the rule of seeing for the observation functions O_i in our model, is the same, we only select one domain to implement (Grid). In the Grid domain, there are k agents and 3 survivors located in a $x \times y$ grid. They tested with 3 or 4 agents, in a 3×3 or a 3×4 grid respectively (4 instances per scenario) as shown in Figure 4.2.

The actions they modelled are: **move**, **observe** and **communicate**. Agents' beliefs are generated by either observing a cell or "hearing" from others' belief by **communicate**. Specifically, those beliefs are generated and updated by the action effects. While, in our model, the belief is generated by state sequence. For example, agents' belief of the location of a survivor is generated either by: agents having been in the same room as that survivor; or, agents having heard the location of that survivor from communication with others. By doing so, we can delegate the epistemic reasoning to the external function $f^{\rightarrow}$ instead of specifying the epistemic effects in action effects in any planning language. Therefore, the different communication rules can be modeled as one single domain file instead of different domain files.

Specifically, in the Grid domain, let Sur be the set of survivors, we have $V = \{(loc\ i), (movable\ i), (sharable\ i), (receivable\ i), (loc\ j), (shared\ j) | i \in Agt, j \in Sur\}$. The observation function for each scenario can be represented by Table 4.1. For simplicity, we only include visibility of the survivors' location here. Agents can "see" the location of the survivor j if: 1) the agent is in the same location as j (it is intuitive for the agent to see survivors without a "sense" action if they are in the same location); and, 2) the location of j is shared by others and agent i can receive this message, which indicates it is either in scenario 2 or 5 (where everyone can receive messages), or i is a commander (the only one that can receive).

Scenarios	$(loc\ i)$	$(receivable\ i)$	$(loc\ j)$	$(shared\ j)$	$(loc\ j) \in O_i(\vec{s}[n])$
All	x	-	x	-	$True$
2,3,4,5	-	True	-	True	$True$

TABLE 4.1: Clarification for seeing relation in difference scenario.

Due to the seeing rules for the grid problem, we have to make two changes (in both methods for consistency) to the original goal conditions in PDKB, in which it is unsolvable by our modeling. Firstly, we have to remove $\neg B_b(\text{loc sur}_1) = r_4$, because Agent b locates in r_4 initially, it does not make sense for agent b not seeing sur_1 (it makes sense in PDKB as they require a sense action to explicitly update the belief $\neg B_b(\text{loc sur}_1) = r_4$). Secondly, we have to remove the goal conditions that require Agent c and d to be at r_3 in the **blocked** problem instances. The reason is that in the goal conditions, PDKB instances also require $\neg B_c(\text{loc sur}_1) = r_4$ and $\neg B_d(\text{loc } sur_1) = r_4$. In **blocked** scenario (Scenario III and IV in Figure 4.2), it is impossible in the JP model to have agents passing r_4 (to reach r_3) without having $\neg B_i(\text{loc sur}_1) = r_4$ no longer hold. For consistency in comparing results, we also updated goal conditions in the PDKB instance accordingly before running their solver.

The result is shown in Table 4.2.The JP approach has a better performance (shorter solving time, as bold font in the table) compared to PDKB in most of the cases, except for those trickier ones (larger amount of nodes generation in PDKB). This happened

Problem		Our approach					PDKB				
$\|Agt\|$	$\|\mathcal{G}_F^{\vec{}}\|$	$\|Gen\|$	$\|Exp\|$	JP	Total	$\|\vec{p}\|$	$\|Gen\|$	$\|Exp\|$	Pre	Total	$\|\vec{p}\|$
!epgoal:											
3	9	56	6	0.05	**0.10**	6	161	27	0.83	0.94	15
3	12	92	9	0.11	**0.20**	9	287	37	1.18	1.33	21
4	9	53	5	0.03	**0.08**	5	193	26	1.13	1.27	14
4	12	84	8	0.13	**0.22**	8	341	45	2.18	2.36	20
epgoal:											
3	18	74	10	0.95	1.05	9	2.7K	1.4K	0.69	**0.81**	42
3	21	358	48	11.58	12.30	16	4.8K	2.5K	1.24	**1.43**	53
4	21	507	49	13.82	14.79	13	1.0M	73.1K	1.13	**9.25**	44
	24	506	46	23.49	24.86	17	10.2K	4.8K	2.00	**2.42**	43
broad:											
3	12	39	7	0.12	**0.16**	7	155	25	6.95	7.69	19
4	12	49	6	0.12	**0.17**	6	155	25	0.81	0.93	19
3	15	61	10	0.32	**0.41**	10	243	38	1.75	1.99	25
4	15	65	8	0.32	**0.40**	8	374	54	2.84	3.10	25
!broad:											
3	12	47	8	0.12	**0.16**	8	300	58	8.83	9.43	23
4	12	59	7	0.15	**0.20**	7	300	58	9.54	10.14	23
3	18	68	12	0.76	**0.84**	12	502	70	34.95	37.90	30
4	18	88	11	0.95	**1.07**	11	502	70	38.88	40.59	30
blocked:											
3	11	619	32	5.60	6.27	14	796	230	3.06	**3.35**	46
3	14	1.4K	64	28.18	30.31	20	1.0K	279	6.04	**6.54**	59
3	12	619	32	8.36	9.11	14	1.2K	397	4.97	**5.42**	49
4	16	2.2K	75	90.85	95.07	20	2.8K	1.2K	9.24	**10.13**	68

Table 4.2: Experimental results for Grid domain, where JP, Pre and Total represent time (in seconds) took by external function (JP function) calls, pre-compilation step in PDKB and total instance solving time.

due to the search algorithm and heuristic function that we used could be optimized, as well as some implementation details.

4.5.1.2 Corridor

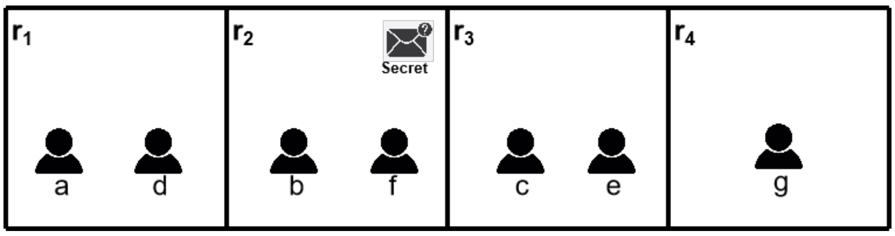


Figure 4.3: The layout and example initial state for the Corridor domain.

The corridor domain was originally presented by Kominis and Geffner, which is also used in the experiments for the PWP approach in this thesis (in Section 3.6.1). Several agents located in four rooms of a corridor try to learn a secret (Figure 4.3). One of the agents (a) has the ability to *move* between rooms, *sense* the secret, *shout* and *shout_lie*. The action *shout* announces the true value of the secret as long as agent a knows the secret (by performing *sense* action before), while the action *shout_lie* announces the false value of the secret. For both actions, agents in the same room or adjacent rooms learn the shouted value of the secret. The objective is to find a plan for agent a that makes some agents believe the secret while some other agents believe the secret is false.

The setup of the experiments in PDKB [88] is in terms of agents (3, 5 and 7) and epistemic formulae depth (1, 3 and 5). The initial states for each instance are the same as the 7-agent one, except for having fewer agents, which is shown in Figure 4.3. Muise et al. use the same goal conditions for every instance to show how the number of agents and epistemic depth affect the performance. We follow their domain and experiment design. The state space of the corridor domain can be modelled by $V = \{loc_i, sensed, sct, shared\_sct, loc_{shared\_sct} \mid i \in Agt\}$ and $D_{loc_j} = \{1, 2, 3, 4\}$ (where $j \in Agt \cup \{sct\}$ and others are boolean variables). The goal conditions are $B_b \neg sct$ and $B_c sct$.

An example implementation can be found in Appendix C.2.

Problem		Our approach					PDKB				
$\|Agt\|$	d	$\|Gen\|$	$\|Exp\|$	JP	Total	$\|\vec{p}\|$	$\|Gen\|$	$\|Exp\|$	Pre	Total	$\|\vec{p}\|$
3	1	32	9	<u>0.01</u>	**0.01**	5	35	16	0.05	0.08	8
5	1	32	9	<u>0.01</u>	**0.02**	5	37	16	0.06	0.10	8
7	1	32	9	<u>0.01</u>	**0.03**	5	38	16	0.07	0.11	8
3	3	32	9	<u>0.03</u>	**0.04**	5	35	16	0.57	0.62	8
5	3	32	9	<u>0.04</u>	**0.05**	5	37	16	3.66	3.86	8
7	3	32	9	<u>0.03</u>	**0.04**	5	38	16	11.95	12.74	8
3	5	32	9	<u>0.03</u>	**0.05**	5	35	16	42.48	45.64	8
5	5	32	9	<u>0.05</u>	**0.07**	5	-	-	-	TO	-
7	5	32	9	<u>0.05</u>	**0.06**	5	-	-	-	TO	-

TABLE 4.3: Experimental results for Corridor domain.

The results can be found in Table 4.3. Since the goal conditions and related initial states are the same, the number of nodes generated and expanded is constant across all problems, while only the number of agents and the depth of epistemic formulae affect the execution time. We can see that those two features have a great impact on the pre-compilation time in the PDKB approach, but not on our justified perspective evaluation. Following the PWP approach (Section 3.5), the JP approach also uses lazy evaluation. The epistemic formulae are evaluated only when the search node is being generated rather than generating all epistemic formulae at the pre-compilation step (what PDKB does).

In addition to efficiency, the expressiveness of two models is another interesting aspect to compare. In PDKB implementation, [88] assume the agents know the location of all other agents and the agents know where the secret has been shared as well as the shared value of the secret (they just do not believe the value unless it was shared directly to them). Therefore, we implemented the observation function following their assumption. That is, agents see every variable all the time, except for sct, which has only been seen by agent i if and only if Agent a shared or lied in the same or adjacent location as i.

We believe the first assumption is not necessary but reasonable, while the second assumption is not reasonable. For example, one of the effects in the PDKB corridor domain file for the action agent a lying in r_4 is $B_b B_c \neg sct$, which we believe is not reasonable. The effect of the action should only be viewed by those in r_4 of r_3, as r_2 is not adjacent to r_4 (as shown in Figure 4.3). Therefore, we proposed a new observation function that, in addition to sct, determining whether agents see both loc_{shared_sct} and $shared\_sct$ also depends on whether their relative distance is smaller or equal to 1. The experimental results are not presented here as it is exactly the same (Table 4.3), due to the current goal conditions not being affected by this change. However, we are able to reason about some nesting propositions, such as $\neg H_b B_c sct$ (effectively, $\neg B_b B_c sct \wedge \neg B_b \neg B_c sct$), while the PDKB approach cannot. Both versions of the observation function implementation can be found in Appendix C.2.

4.5.1.3 Grapevine

Grapevine, proposed by Muise et al. [87], is a similar problem to Corridor. The seeing rule becomes that the secret will be known to everyone in the same room. With only

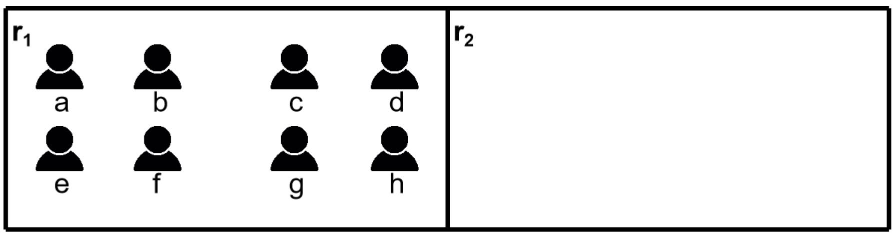


FIGURE 4.4: The layout and example initial state for the Grapevine domain.

two rooms available, the scenario makes sharing secrets while hiding from others more difficult. The basic setup is for each agent to have their own secrets (a propositional variable). Initially, all agents are located in room r_1 (as shown in Figure 4.4). Agents can move between those two rooms and share or lie about their own secret. In addition, they can also share about others' secrets based on what they believe.

For the observation function, firstly, agents see each other's location. This is because there are only two rooms, agents are either in the same room, or in the other room. The visibility of the secrets is related to sharing and lying actions. In order to model sharing and lying, we need three variables for each agent's secret. Using agent a's secret as as an example, those three variables are: `(truth_value ?s)` (denoted as tas), `(lying_value ?s)` (denoted as las) and `(shared_value ?s)` (denoted as sas), representing the secret's truth value, lying value, and currently sharing value (could be shared by any agent). tas and las are visible solely to their owner agent, agent a, while the agents that perceive sas is determined by the room in which the secret as is being shared.

The action agents share or lie about their own secret would have the effect:

- `(assign (shared_value ?s) (truth_value ?s))`, or
- `(assign (shared_value ?s) (lying_value ?s))` respectively.

While the action `sharing_others_secret` needs some discussion. Intuitively, agents could only share the value of others' secret of the value that they believe. In existing approaches, including PDKB and PWP, they have to specify the value of what agents believe in the modeling language. Specifically, they model this action as the agent can choose to share others' secret as truth value or share others' secret as false value. That is, they have to enumerate all possible values of the secret for this action. Although the secret in grapevine is only boolean, both approaches miss the opportunity to model variables with larger D_v or even continuous D_v.

However, with the capability of introducing external function $@jp$, we are able to model `sharing_others_secret` as one PDDL action:

```
1  (:action sharing_others_secret
2      :parameters (?a - agent, ?s - secret)
3      :precondition (
4          (= (own ?a ?s) 0)
5          (= (sharing) 0)
6          (!= (@jp ("b [?a]") (shared_value ?s)) jp.none)
7      )
8      :effect (
9          (assign (shared_loc ?s) (agent_loc ?a))
10         (assign
11             (shared_value ?s)
12             (@jp ("b [?a]") (shared_value ?s)))
13         (assign (sharing) 1)
14     )
15 )
```

Code Example 4.6: F-PDDL example action in Grapevine domain using external function @jp.

The preconditions require that the agent does not own the secret (Line 4), there is no other secret being shared at the moment (Line 5) and the agent has a not-'None' belief in the secret (Line 6). The effects represent that the secret is shared in the same location as the agent (Line 9), the secret is being shared (Line 13) and the shared value of the secret, `(shared_value ?s)`, is being shared as what the agent believes (Line 10-Line 12). This representation models the agent's belief as part of the language without specifying its value, which allows the JP model to have the potential to model continuous domains.

The experimental results can be found in Table 4.4. As it shows in the table, PDKB took a costly pre-compilation step when d becomes 2 (around 9 seconds when $|Agt|$ is 4 and 200 seconds when $|Agt|$ is 8). In other words, their solving times are highly dependent on d and $|Agt|$. Although our approach could solve these problems within a short period of time (a few seconds) independent of those features, it failed to solve the final two problems because of the large number of nodes generated or expanded. This is due to both the complexity of the problem (branching factors) and the lack of effective heuristic functions.

Problem			Our approach						PDKB				
$\lvert Agt\rvert$	$\lvert\mathcal{G}_{\vec{F}}\rvert$	d	$\lvert Gen\rvert$	$\lvert Exp\rvert$	$\lvert$JP$\rvert$	JP	Total	$\lvert\vec{p}\rvert$	$\lvert Gen\rvert$	$\lvert Exp\rvert$	Pre	Total	$\lvert\vec{p}\rvert$
4	2	1	93	11	4.2	0.08	**0.10**	4	80	15	0.48	0.56	4
4	4	1	167	22	7.0	0.17	**0.20**	9	108	14	0.47	0.56	7
4	8	1	387	48	13.3	1.37	1.44	18	137	14	0.48	**0.57**	12
4	2	2	78	10	4.1	0.07	**0.08**	4	57	13	9.16	11.22	5
4	4	2	90.5K	12.4K	10.2	-	MO	-	88	11	9.24	**11.28**	7
4	8	2	4.1K	548	21.7	50.89	52.72	24	1.2K	330	9.18	**11.75**	23
8	2	1	189	15	4.3	0.26	**0.30**	4	112	16	3.74	4.69	4
8	4	1	343	30	7.1	0.86	**0.93**	9	5.9K	194	3.75	5.68	11
8	8	1	807	60	13.5	7.28	7.48	18	970	260	3.72	**4.67**	16
8	2	2	158	14	4.1	0.29	**0.33**	4	169	31	199.60	481.85	5
8	4	2	26.2K	2.3K	7.4	-	MO	-	355	34	202.56	**483.75**	9
8	8	2	7.2K	524	17.9	-	MO	-	561	51	200.78	**496.39**	15

Table 4.4: Experimental results for Grapevine domain, where |JP| is the average length of the justified perspectives for all external function calls.

For branching factors, considering 8 agents examples, the agents are able to move between rooms (8 available actions), share or lie about their own secrets (16 available actions), and share others' secrets (56 available actions in the worst case). Since any agent can share others' secrets once they have belief about them, there is no direct pruning that can be done regarding the branching factors.

As for the heuristic function, using the 5th instance as an example, the goal conditions are: $B_bB_c\neg\psi$, $B_c\psi$, $B_dB_c\psi$ and $B_b\psi$, where ψ is `(= (shared as) True)`. The goal counting heuristic for all successors of the root nodes is 4 except: 1) it becomes 3 when agent a lies about as; and, 2) it becomes 1 when agent a shares the truth of as. Since the search algorithm is greedy, it expands the latter condition first. A valid solution would be to move d away and let a lie about as, however, the goal counting heuristic for that node would be 2. This is because once a lies with b and c in the room, both B_cas and B_bas become false. Having the heuristic value for the correct node as 2 means, with the greedy search algorithm, this node will only be expanded when all the nodes with heuristic smaller than 2 have been expanded, which caused our planner to run out of memory.

As for the last two instances with both 8 agents and $d = 2$, our planner failed to find a solution for the similar reason as above. The number of nodes generated and expanded is much higher for instances with $|\mathcal{G}_{\vec{F}}| = 4$ than $|\mathcal{G}_{\vec{F}}| = 8$, while the latter one has much longer state sequences when evaluating epistemic formulae. This is expected due to the duplication elimination procedure mentioned in Section 4.4.2.1. The size for duplication elimination depends on $\mathcal{G}_{\vec{F}} \cup PRE(\mathcal{O}_{\vec{F}})$ according to Definition 4.24. Specifically, in these two instances, the latter one contains more goal conditions, which results in the latter one having a larger DPS key size. That is, the state sequence would be less likely to be eliminated due to the larger DPS key size, resulting in the average length of the search path being much longer.

4.5.1.4 Coin

The Coin domain is based on the problem described in Example 4.1. It was originally inspired by the coin example in Baral et al. [117] and the false-belief task by Bolander [140]. Even though there is no attempt from the PDKB approach on this domain, since the scale of this domain is small (all instances can be solved within 0.01 seconds), we still include this because it is our motivational example, and the results show the expressiveness of our approach.

The coin domain contains two agents, a and b, and a coin c could be either *head* or *tail*. The coin is in the box and not visible to all agents unless they are peeking into the box. The actions that agents can take are either "**peek**" into the box or "**return**" to their standing position. In addition, there is a secret hidden agent who can "**flip**" the coin without either a or b noticing, unless they are peeking into the box. The task is to form some false beliefs between agents as it is shown in Figure 4.5.

We used three variables to model it: `(peeking a)`, `(peeking b)` and `(face c)`, where `c` could be `Head` or `Tail`, and `peeking` is a binary relation. Agents are able to see whether the others are peeking into the box all the time, while the value of c is only visible to the agent that is currently peeking.

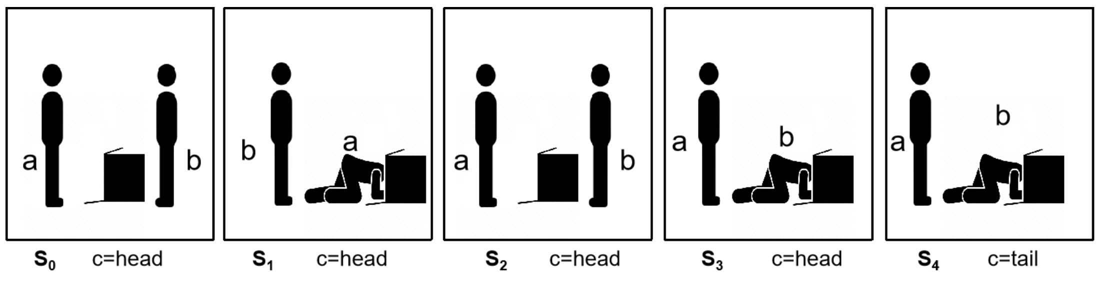


FIGURE 4.5: A solution for the 5th instance in Coin domain.

$\|Agt\|$	d	$\|\vec{p}\|$	$\|Gen\|$	$\|Exp\|$	JP	Total	$\mathcal{G}_{\vec{F}}$
2	1	1	4	1	0.00	0.00	$B_a c{=}head$
2	1	2	6	2	0.00	0.00	$B_a c{=}tail$
2	1	3	11	4	0.00	0.00	$B_a c{=}head \wedge B_b c{=}head$
2	1	4	11	4	0.00	0.00	$B_a c{=}head \wedge B_b c{=}tail$
2	2	4	36	14	0.02	0.03	$c{=}tail \wedge B_b c{=}tail \wedge B_a c{=}head \wedge B_b B_a c{=}head$
2	2	8	37	14	0.01	0.03	$B_b B_a c{=}head \wedge B_a B_b c{=}tail$

TABLE 4.5: Experimental results for Coin domain.

The results are shown in Table 4.5. Most of the results are trivial, while the last two could use some explanation.

The 5th is the same as our motivating example (Example 4.1), which contains 4 goal conditions. The plan returned is the same as Plan 4.2. The first three goal conditions are straightforward, but not the last one ($B_b B_a c = tail$). In order to evaluate this epistemic formula, we need to extract justified perspective for b first. The initial state is {`(peeking a)`=$False$, `(peeking b)`=$False$, `(face c)=head`} [4].

As shown in Figure 4.5, the global sequence of the plan is:

$$\vec{s} = \texttt{[F-F-Head,T-F-Head,F-F-Head,F-T-Head,F-T-Tail]}.$$

Agent b's observations of the global sequence are:

$$O_b(\vec{s}) = \texttt{[F-F-\_, T-F-\_, F-F-\_, F-T-Head, F-T-Tail]},$$

[4]For similarity, in this section, we only use value to represent state. Therefore, the given initial state will be represented as $F - F - Head$

where '_' represents the variable not in b's observation. Then, according to Definition 4.6, we have agent b's justified perspective as:

$$f_b(\vec{s}) = \texttt{[F-F-}\bot\texttt{,T-F-}\bot\texttt{,F-F-}\bot\texttt{,F-T-Head,F-F-Tail]}.$$

Agent a's observations in agent b's perspective are:

$$O_a(f_b(\vec{s})) = \texttt{[F-F-\_, T-F-}\bot\texttt{, F-F-\_, F-T-\_, F-T-\_]}.$$

Then, applying f_a on $f_b(\vec{s})$ for timestamp 0, we have $\texttt{F-F-}\bot$ since $lt_c = -1$. For timestamp 1 and 2, although $lt_c = 1$ in both of them, the retrieved value of c is still $\bot$, because, for both cases, the value of c in the input sequence for retrieval function R are all $\bot$ ($[f_b(\vec{s})[0], f_b(\vec{s})[1]]$ for timestamp 1 and $[f_b(\vec{s})[0], \ldots, f_b(\vec{s})[2]]$ for timestamp 2).

While, for timestamp 3 and 4, we still have $lt_c = 1$ (as 1 is the last timestamp b sees a sees coin). Then, due to LT $= \{\}$ for both (agent b has not seen c before or at timestamp $lt_c = 1$) and RT $= \{3\}$ for timestamp 3 and RT $= \{3, 4\}$ (agent b has seen c twice after timestamp $lt_c = 1$) for timestamp 4 in Definition 4.5, both $R([[f_b(\vec{s})[0], \ldots, f_b(\vec{s})[3]]], 1, c)$ and $R([[f_b(\vec{s})[0], \ldots, f_b(\vec{s})[4]]], 1, c)$ equal the value of c in the timestamp 3 of the input sequence, which is $Head$. Therefore, agent a's justified perspective under agent b's belief (b's justified perspective) is:

$$f_a(f_b(\vec{s})) = \texttt{[F-F-}\bot\texttt{,T-F-}\bot\texttt{,F-F-}\bot\texttt{,F-T-Head,F-F-Head]}$$

Therefore, the last goal condition for the 5th example is achieved.

In the last instance, the plan returned by our planner is "**peek(a)**", "**return(a)**", "**peek(b)**", "**return(b)**", "**peek(a)**", "**flip(c)**","**return(a)**", "**peek(b)**". However, an optimal plan for that instance would be "**peek(b)**", "**return(b)**", "**flip(c)**" and "**peek(a)**". Intuitively, b believes a sees what b saw, and a believes b saw what a sees. Both goal conditions are fulfilled at the last timestamp. However, due to the greedy search algorithm with goal counting heuristic, the planner seeks to achieve any of the goal conditions as soon as possible.

The first three actions in the first plan result in $B_bB_ac = Head$ (although $B_aB_bc = Head$). The following plan makes agent a believe b believes c is $Tail$, while $B_bB_ac =$

$\|Agt\|$	d	$\|\vec{p}\|$	$\|Gen\|$	$\|Exp\|$	JP	Total	$\mathcal{G}_{\vec{F}}$
2	1	3	37	9	0.01	0.02	$B_b v = True$
2	1	3	41	10	0.03	0.04	$B_a v = True \land B_b v = True$
2	2	7	149	37	0.36	0.40	$B_b B_a v = True \land B_a B_b v = True$
2	2	5	325	81	0.38	0.46	$B_b B_a v = True$
2	3	5	325	81	0.45	0.53	$B_b B_a B_b v = True$
2	4	5	381	95	0.78	0.89	$B_a B_b B_a B_b v = True$

Table 4.6: Experimental results for BBL domain.

Head stays the same due to the design decision (checking past first) of the retrieval function R (in Definition 4.5).

4.5.1.5 Big Brother Logic (BBL)

BBL [2] contains stationary cameras that can turn and observe a certain angular range in a 2-dimensional plane. For example, camera a and camera b (agents) are located in positions $(3, 3)$ and $(2, 2)$ respectively, while a propositional object `v` with value $True$ is located in position $(1, 1)$. Cameras have two actions: *clockwise-turn* and *anticlockwise-turn*. For simplicity, we set the angle of turning to be enumerated from the set $\{0°, \pm45°, \pm90°, \pm135°, 180°\}$ and the turning angle to 45°, but as the external functions are implemented in Python, we can replace this with floating point numbers to model continuous directions. We use the same problems as in the PWP approach (Section 3.6.2.3), but with modified goals to support belief instead of knowledge.

The observation function is the same as the PWP approach: $j \in O_i(s)$ iff

$$\begin{gathered}\left(|\arctan(\tfrac{|s(y_i)-s(y_j)|}{s(x_i)-s(x_j)}) - s(dir_i)| \leq \tfrac{s(ang_i)}{2}\right)\\ \lor \\ \left(|\arctan(\tfrac{|s(y_i)-s(y_j)|}{s(x_i)-s(x_j)}) - s(dir_i)| \geq \tfrac{360°-s(ang_i)}{2}\right)\end{gathered} \tag{4.1}$$

where (x_i, y_i) is the location of the agent i, while (x_j, y_j) is the location of the target (could be an agent or an object).

An implementation (including both domain and problem file, as well as the observation function) can be found in Appendix C.1.

Results are shown in Table 4.6. Initially, camera a faces $-135°$, while camera b faces $90°$. Since $v \in O_a(s_0)$, the plan for second instances is the same as the first one, which is only to take three anti-clockwise turns. The optimal plan for the rest of the instances should be to turn clockwise for 5 times, as they require b sees loc_a (indicating b sees a sees loc_b). However, due to the nature of greedy with goal counting heuristic, for the 3rd instance, turning anti-clockwise for 7 times will achieve one of the goals first ($B_b v = True$), resulting in the goal counting heuristic becoming 1 instead of 2.

4.5.2 Large-Scale Experiments

In addition to the benchmark experiments, we perform a large-scale experiment in order to examine the search algorithms, including the proposed two pruning methods, and the complexity of the ternary semantics.

The search algorithms we explore are `bfs`, `bfsdc`, `bfsdcu`, `astar`, and `greedy` (defined in Section 4.4.2.3). The number of problem instances solved (including proven unsolvable) is chosen as a direct comparison between search algorithms. In addition, for the solvable instances by all algorithms, the efficiency can be compared by the number of nodes that have been expanded. On the other hand, for the proved unsolvable instances by all algorithms (except `bfs`, which cannot prove unsolvability), the number of nodes that have been expanded to prove the problem instance is unsolvable can also show the algorithms' efficiency.

The domain we selected for large scale experiments is BBL, since it is: firstly, not too trivial to be solved in 0.1 seconds, such as the Coin or Corridor domain; secondly, not too complex (easily becoming unsolvable due to the large branching factors), including the Grapevine domain; lastly, not containing too many ontic goal conditions (which would result in the ontic planning parts affecting the results), such as SC domains.

The whole experiment was conducted on three Nectar Research Cloud VMs with 64 VCPUs and 256GB RAM. The Nectar Research Cloud is a collaborative Australian research platform supported by the NCRIS-funded Australian Research Data Commons (ARDC). The experiments are performed in parallel within a Docker environment with the 8GB memory limitation and 600 seconds time limitation.

4.5.2.1 Experiment Design

We vary the following parameters in these experiments:

- The maximum depth of epistemic formulae from 1 to 4 in increments of 1;
- The number of agents between 2 to 4 in increments of 1;
- The number of goal propositions from 1 to 4, in increments of 1.

For this, we have 48 parameter combinations. For each setting, we ran up to 500 random instances. The random factors are the epistemic goal formulae selected and the initial state. The number of randomly selected instances is smaller than 500 for the first setting, as there are only 384 instances for this setting (2-agents, maximum goal depth is 1 and 1 goal proposition). For simplicity, we set the randomness for the initial by randomizing the initial direction for all agents, which means the location of the agents is the same as in Section 4.5.1.5. The locations of the newly added agents c and d are (0,0) and (0,1) respectively. Since there are 8 possible directions, the number of possible initial states is 64 for 2-agents. The number of possible goal epistemic formulae with max depth of 1 is 6, which are $B_a\psi$, $\neg H_a\psi$, $\neg B_a\psi$, $B_b\psi$, $\neg H_b\psi$ and $\neg B_b\psi$, where ψ is `(= (value v) True)`. Therefore, the number of possible instances for the first setting is $64 \times 8 = 384$. Overall, there are 23884 instances run for each algorithm.

For each instance in this experiment, we record the solvability of the instance, the search time, the average external function call time, the number of nodes generated, the number of nodes expanded, the max length of the search path, the average length of the search path (which is the same as the average length of the input for the justified perspective function). For readability, we merged the results of different maximum goal epistemic formulae settings together, which means the result will have 12 subplots ($2-4$ agents and $1-4$ epistemic goals size).

Solvability is an important metric in planning. The status includes *solvable*, *proven unsolvable*, *time out* and *memory out*. A problem instance is solvable means the search algorithm has found a solution in time, while a problem instance is "proven unsolvable" means the search algorithm has successfully finished and failed to find a solution. Both "time out" and "memory out" mean the search algorithm is unable to prove the solvability of the problem instance.

4.5.2.2 Results

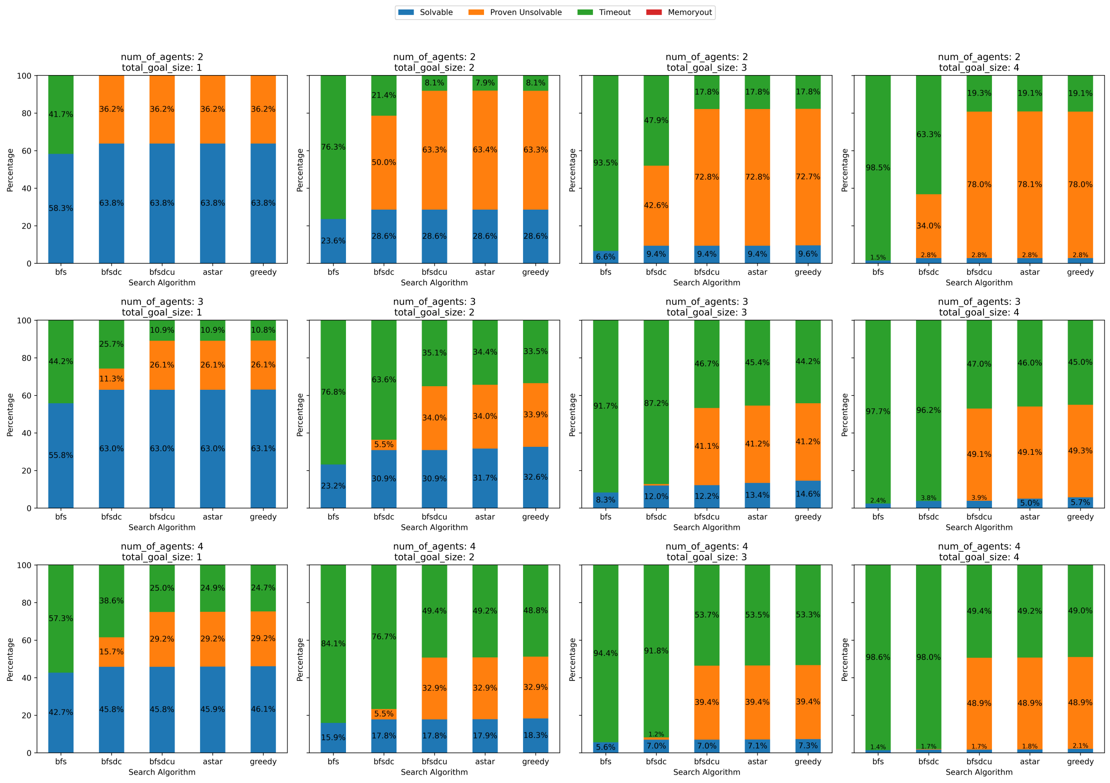


FIGURE 4.6: Solvability (in percentage) for all search algorithms of all instances.

The solvability results are summarized in Figure 4.6. As the nature of the plain BFS search (without duplication elimination), `bfs` cannot prove unsolvability for any instance. Because BBL belongs to EPD_{DO} (defined in Section 4.4.2.2), pruning by HNB is another optimization that can be used in any of the search algorithms. As can be seen from the results, with the HNB pruning (`bfsdcu`) can prove unsolvability for a lot more instances compared to `bfsdc`, except when the problem instances are too trivial (2 agents and 1 goal condition). The performances for `bfsdcu`, `astar` and `greedy` are quite similar in terms of proving unsolvability, because they all use the same duplication elimination and HNB pruning. This is because to prove unsolvability, the search algorithm needs to expand all nodes (except those that can be safely pruned), which might be slightly affected by the node expansion orders, but it is highly dependent on the problem's reachable state space itself. Therefore, some further analysis is needed.

We use the number of nodes expanded as the indicator to show the performance of the search, rather than execution time, which can be affected by other jobs running on the machine. Since the instances that are solved by the initial state would have the same

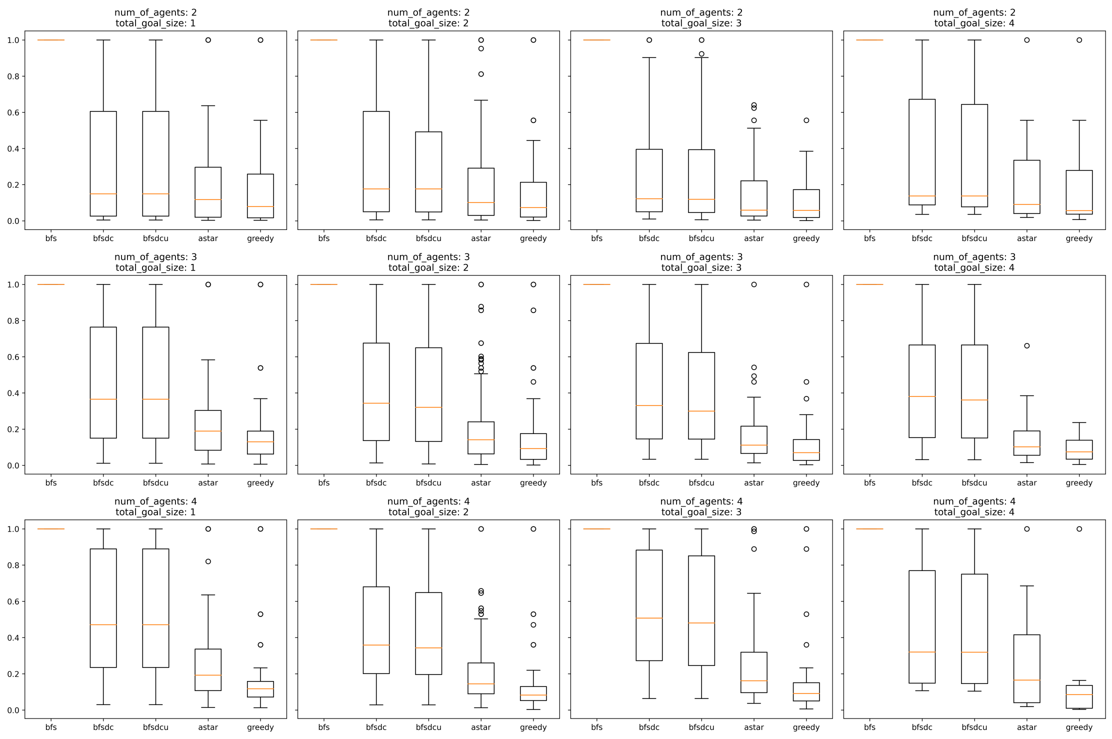


FIGURE 4.7: The ratio of nodes expanded between `bfs` and other search algorithms for the `bfs`-solvable instances.

number of nodes expanded, which is 1, we filtered out those instances when generating results.

The first result (as shown in Figure 4.7) is from all instances that are solvable by `bfs`, in which the ratio is generated by dividing the number of nodes expanded with the search algorithm by the number of nodes expanded with `bfs` (our baseline in this case). The performance of `bfsdc` and `bfsdcu` is similar as the solvable instances that contain HNB formulae are not common, which is evaluated later in this section. With the problem instances becoming more complex, `astar` has better performance than those two blind search algorithms, while `greedy` has better performance than `astar`. It is worth mentioning that the results are generated with all solvable instances, except one outlier (`problem_bbl_a2_g4_d3_34074_init_a2_00062.pddl`) to increase readability. That problem instance contains 4 goal conditions, 3 of which become true with one plan while the other one needs a completely different plan to make it true. As mentioned in Section 4.5.1, `greedy` has poor performance that is stuck in the local optimum, which results in it expanding 4 times more nodes than the baseline. Despite this raw (1 out of 23884) outlier, `greedy` is the most efficient search algorithm on the solvable cases.

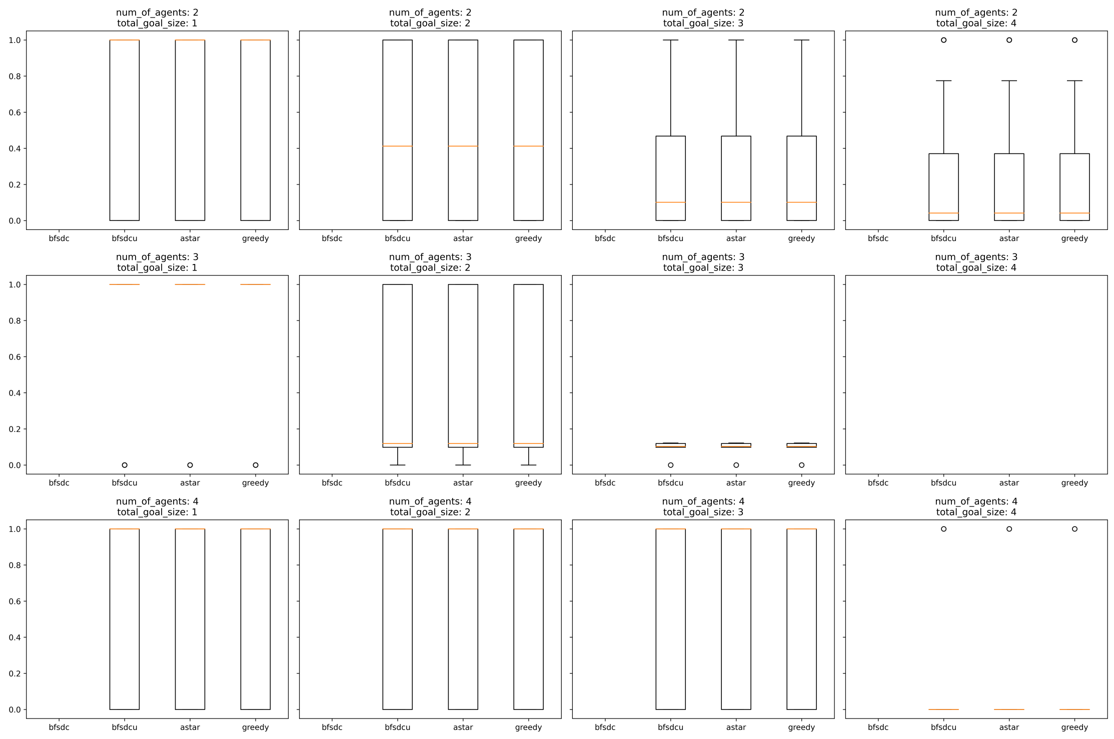


FIGURE 4.8: The ratio of nodes expanded between `bfsdc` and other search algorithms (excepts `bfs` as it cannot prove unsolvability) for the `bfsdc`-proven-unsolvable instances.

As for unsolvability, as shown in Figure 4.8, `greedy` has the same performance as `bfsdc` for some subplots in all instances that can be proved unsolvable by `bfsdc`. For some subplots, the numbers of instances that can be proven unsolvable by `bfsdc` are too few (less than 10%, which is less than 50 instances). This can be shown from Figure 4.6. When the number of agents is greater than 2, and the goal size is larger than 1, the percentage of the problem instance that is proven to be unsolvable by `bfsdc` is less than 10%. Thus, those results are not reliable. In addition, when the goal size is 1, the heuristic function goal counting has minimum effectiveness on improving the search efficiency ($h_{GC}(\vec{s}) = 1$ for any sequence before generated goal state). Thus, the performances between `bfsdc` and others are almost the same.

Therefore, only the results in three subplots (with the number of agents and goal size pairs as: `2-2`, `2-3` and `2-4`) are reliable. From those results, the number of nodes expanded to prove the instance is unsolvable for the other three search algorithms (`bfsdcu`, `astar` and `greedy`) is much fewer compared to `bfsdc`.

The HNB pruning cannot be evaluated directly from any of the above figures. We have shown above that with HNB pruning, `bfsdcu` can prove unsolvability for a lot more instances compared to `bfsdc` (in Figure 4.6). However, does this make the search

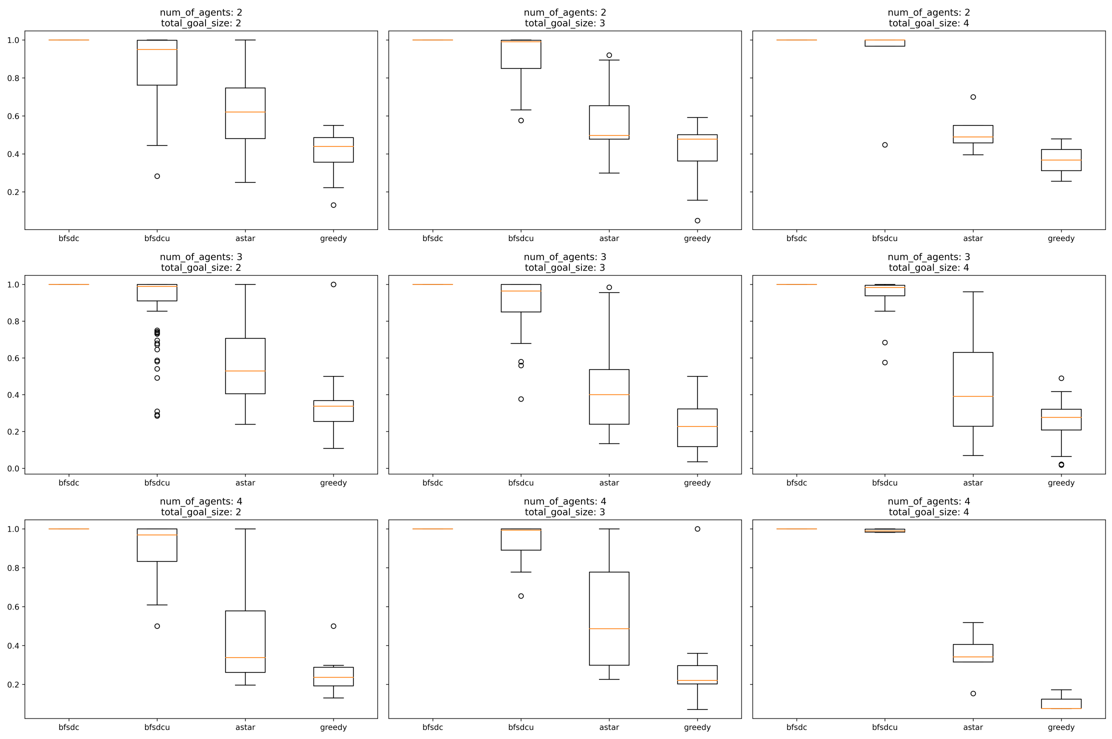


FIGURE 4.9: The ratio of nodes expanded between `bfsdc` and other search algorithms that used HNB pruning for the `bfsdc`-solvable and HNB involving instances.

algorithm more efficient (expanding fewer nodes) for the solvable cases remain to be evaluated. The ratio of the number of nodes expanded compared to `bfs` (as shown in Figure 4.7) cannot reflect this, because many randomly generated goal conditions do not contain HNB epistemic formulae. Moreover, even in those instances containing HNB, many instances are solved without involving HNB pruning. This happens either because the instance is too trivial, or the HNB relation(s) in the goal condition are too complex to be negated. Therefore, we show the results in Figure 4.9 by filtering out those instances.

This result shows that with HNB pruning, the search algorithm expands fewer nodes in most of the cases when the problem is solvable. Therefore, the HNB pruning is efficient no matter whether the problem is solvable or not. But it has a greater impact on proving unsolvability.

At last, the efficiency of the justified perspective function is shown in Figure 4.10. The instances run by `bfs` are excluded, as they have the same performance for all unsolvable cases and, with some noise from the server running, they show some strange straight vertical lines (same average depth but slightly different average time) in the result. The results show that, in solving each problem instance, no matter which search algorithm is used, the average time used to reason about epistemic logic in our implementation is linear in terms of the search path length. This verifies our claim in Section 4.3.3.3.

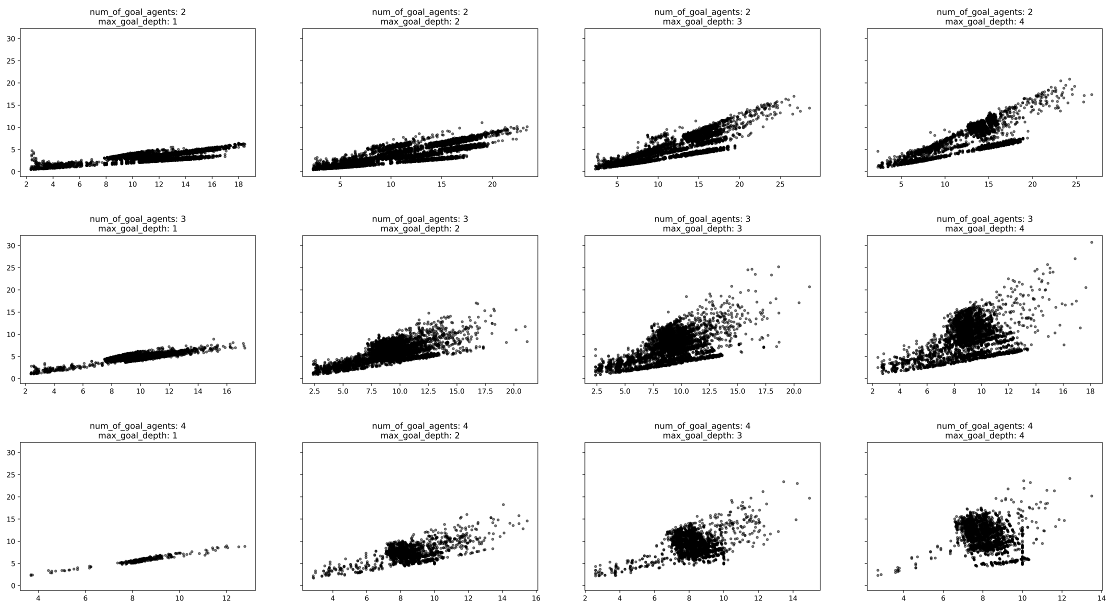


FIGURE 4.10: The average length of the search path and average external function calling time for all instances, where the x-axis represents the length and y-axis represents the time (in milliseconds).

Overall, though massive experiments in the BBL domain, we have shown both optimizations — Duplication Elimination and HNB pruning — improve efficiency. We also believe this conclusion is general, since there is no reason why the results on the BBL domain cannot be generalized to other epistemic planning domains. As for the search algorithm, we believe in most of the cases, `greedy` (with goal-counting as heuristic) would have better performance, except when the goal conditions could make it stuck in the local optimal.

4.6 Conclusion and Discussion

In this chapter, we extend the **S5** Logic of PWP to the JP model in order to handle beliefs. The JP model reasons about the knowledge and belief relation by constructing a corresponding justified perspective, built on the intuition that humans reason about the unseen from their past observations unless they see (saw) evidence to suggest otherwise. We give the definition of the retrieval function and justified perspective function. Along with the observation function from the PWP model, agents' justified perspective can be constructed with arbitrary nesting.

Then, similarly as in the PWP approach (Chapter 3), we provide two forms of the semantics: the complete semantics (Definition 4.17) and the ternary semantics (Definition 4.18). Although it is not possible to prove the soundness and correctness of our justified belief model, we prove that the new logic satisfies the principles of belief described by the axioms of the logic **KD45**. In addition, we show the complexity of the ternary semantics can be reasoned in polynomial time.

Besides, we formalised the epistemic planning problems that our model can handle as EP-NM-F-STRIPS and provide a F-PDDL encoding to represent those instances. Moreover, we develop the JP model as an (action) model-free (planning) tool and embed it as the external functions in the F-PDDL encoding. By integrating classical planning search algorithms, we are able to provide a planning tool to solve epistemic planning with knowledge and beliefs.

At last, we show the expressiveness and efficiency of our planning tool through experiments on standard benchmarks. In addition, we perform a large-scale experiment to compare the efficiency of different search algorithms, as well as to verify our claim about the complexity of the ternary semantics.

As for the concerns raised about the PWP model (mentioned in Section 3.7), by reasoning over all epistemic relations (including knowledge and beliefs) from agents' justified perspective, the JP model: 1) handles knowledge and belief; 2) improves the efficiency of the external function (using DPS set); and, 3) handles inconsistency in the problem description for the modeler. The first concern is handled as one of the main contributions of the work in this chapter. The second and third are solved by evaluating all epistemic queries described in F-PDDL once when the node is generated (for goal conditions) and expanded (for preconditions) instead of once for each query.

The only limitation compared to the JP model with the PWP model is for the group epistemic relations. The PWP model is able to handle single-(nested-)knowledge and group-(nested)-knowledge, while the JP model handles single-(nested)-belief. As for the group belief, the approach used in the PWP approach would not work. For example, the distributed knowledge in the PWP model uses set union to merge agents' perspectives.

While in the JP model, since agents' justified perspectives can have false values (leading to justified false-beliefs), merging their justified perspectives would cause conflicts. Therefore, how to model agents' justified group beliefs becomes the next problem to solve.

Chapter 5

Planning with Group Belief using Group Justified Perspectives

In this chapter, we extend the single-agent (nested) justified beliefs model to the Group Justified Perspective (GJP) model, to handle group justified beliefs. We follow the same intuition as the JP model that when people reason about something they cannot see, they generate justified beliefs by retrieving the information they have seen in the past, unless they have seen evidence to suggest otherwise [35]. By defining group justified perspective functions, we can reason about uniform belief, distributed belief, and common belief, even mixed with individual and group knowledge operators. Our finding is that, different from the above intuition, when it comes to group beliefs, agents do not have to form a group knowledge to generate a group belief. For example, a common belief could be formed even though its corresponding common knowledge has not been formed beforehand.

5.1 Introduction and Motivation

As introduced in the previous chapter (Chapter 4), the JP model is efficient and expressive in handling single-agent (nested) belief. Following the intuition of 'belief is past knowledge', they are able to construct the justified perspective for agents based on their past observations. However, applying this intuition naïvely to group belief is neither complete nor consistent. It is possible that some agents in a group see value changes that affect their own knowledge and belief while the group's belief stays the same. In addition, it is possible to form a common belief about a proposition even if there was no prior common knowledge about this previously. For example, consider agent a looking in a box and seeing a coin with heads, and then agent b looking into the box a minute after agent a and seeing it is heads. At no point did they see the coin at the same time, so they cannot form common knowledge that the coin is heads (it may have changed in the minute in between). However, they can form a common belief that it is heads because they each saw heads and have no evidence to suggest the value has changed.

We illustrate this idea with our motivating domain, NIB (Example 1.2), following the same initial state as in Example 2.1.

Example 5.1. *Recall that in Example 1.2, there are two agents, a and b, two numbers, p and q in separate boxes. The agents have to peek into the box to see the value of the number in it, and each box can only be peeked at by one agent at a time.*

Now, let's consider the following two interesting and challenging group belief tasks:

1. *a and b have a common belief that $q > 4$ and:*
   - *agent a and b believes that they have a common belief about q's value;*
   - *while q's values in their believed common belief are different.*
2. *a and b have a common belief that $q > 4$ and:*
   - *agent a believes that they have a common belief about q's value;*
   - *while b believes they don't have a common belief about q's value;*
   - *and b believes a believes that they have a common belief about q.*

For the first task, the condition is that both a and b believe they have a common belief on q's value while q's values in their believed common belief are different. This indicates the common beliefs they have are false beliefs. Even though each of them holds a false belief of their common belief, the common belief (about a formula that holds in both agents' believed common beliefs) can still be formed. As shown in Figure 5.1, a valid plan to achieve Task 1 above would be:

Plan 5.1. "**(peek a q)**", "**(return a)**", "**(decrement q)**", "**(peek b q)**"

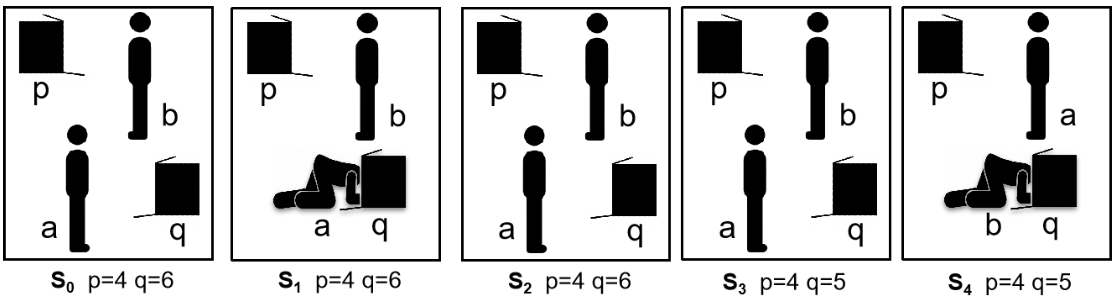


FIGURE 5.1: An example plan to solve Task 1 in Example 5.1.

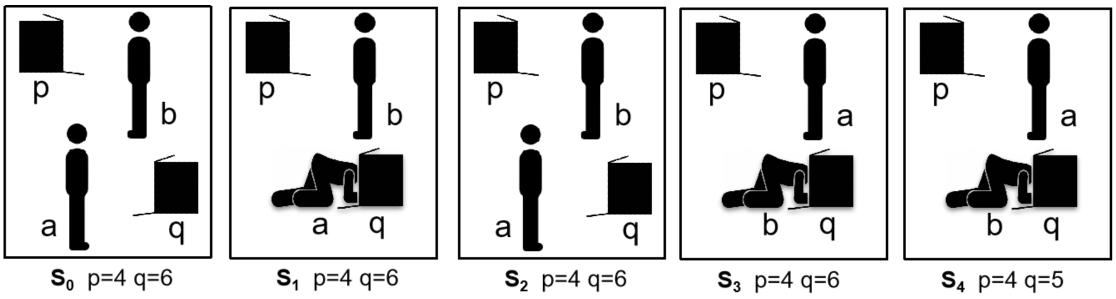


FIGURE 5.2: An example plan to solve Task 2 in Example 5.1.

In the plan, agents a and b do not peek into the box containing q at the same time. So, at no point neither the statement "agent a knows that agent b knows $q > 4$" ($K_aK_bq > 4$) nor $K_bK_aq > 4$ holds. Further, the common knowledge $CK_{\{a,b\}}q > 4$ does not hold.

However, we assert that the common belief $CB_{\{a,b\}}q > 4$ should hold if agents have memory. Since agent a sees $q = 6$ at s_1 and agent b sees $q = 5$ at s_4, both $B_aq = 6$ and $B_bq = 5$ hold, which implies both $B_aq > 4$ and $B_bq > 4$. In addition, since agent a sees agent b peeking into the box at s_4 and $B_aq = 6$, $B_aB_bq = 6$ should hold. Similarly, $B_bB_aq = 5$ should hold. Therefore, we have both $B_aB_bq > 4$ and $B_bB_aq > 4$. Given that a and b both saw that each other peeked in the box, and saw that each saw that each peeked into the box, etc, both a and b believe each other believes $q > 4$ with infinite depth. From the definition by Fagin et al. [3], this constitutes common belief.

In addition, in the view of the agent a, $q = 6$ is a common belief among a and b, since agent a saw $q = 6$ at s_1 and saw agent b see q at s_3. For the similar reasoning, agent b believes $CB_{\{a,b\}}q = 5$. Thus, Task 1 is achieved.

As for Task 2, as shown in Figure 5.2, a valid plan is as follows:

Plan 5.2. "**(peek a q)**", "**(return a)**", "**(peek b q)**", "**(decrement q)**"

The common belief $CB_{\{a,b\}}q > 4$ holds for the similar reason as above. Agent a holds the same false belief ($B_a CB_{\{a,b\}}q = 6$), while agent b saw $q = 6$ immediately after a saw the number and saw $q = 5$ in the last state, which indicates agent b no longer believes there is a common belief among a and b on the value of q.

In the following parts of this chapter, we propose group perspective functions to reason about uniform belief, distributed belief, and common belief (as in the GJP model). We discuss an implementation that extends an existing epistemic planning tool, and report experiments on key domains in epistemic planning. Our results show that we can efficiently [1] and expressively solve interesting problems with group belief, even with a basic blind search algorithm.

5.2 Background

Most of the background and related works are introduced in Chapter 2. Thus, we only include those that are relevant to this work and have not been mentioned earlier.

Recall that semantically speaking, if $K_i\varphi$ (agent i knows φ is true) holds (Axiom T), then φ holds; while if $B_i\varphi$ (agent i believes φ), it is not necessarily the case that φ holds. In short: agents can have incorrect beliefs, but not incorrect knowledge. For group beliefs, there are mainly three types: uniform beliefs, also known as shared beliefs; distributed beliefs; and common beliefs.

Uniform belief, denoted $EB_G\varphi$, is straightforward — it means that everyone in group G believes proposition φ. There are a number of approaches to model uniform belief [152, 153].

Distributed belief, denoted $DB_G\varphi$, combines the beliefs of all agents in group G. It is, effectively, the pooled beliefs of group G if the agents were to "communicate" everything they believe to each other. Any model has to consider the pooled beliefs from each agent and the pooled beliefs from the group that are not held by any of its individual agents, but are held by the group. For example, if agent a believes $x = 1$ (and nothing else) and agent b believes $y = 1$ (and nothing else), distributively, the group $\{a, b\}$ believes that $x = y$, even though no individual agent believes this. Distributed belief is challenging

[1] Note: we do not have any existing approach to compare to.

because agents can have conflicting beliefs: if agent a believes $x = 1$ and agent b believes $x = 2$, what should the distributed belief be?

There are two main approaches to model distributed belief: (1) belief merging [154–156]; and (2) merging the agents' epistemic accessibility relations [90, 142, 157–162]. Typically, merging conflicting beliefs is solved using some form of ordering over agents or propositions, meaning that some agents (propositions) receive priority over others. In this chapter, we give two definitions of distributed belief: one that accepts inconsistent distributed belief; and one in which conflicting beliefs are removed entirely, leading to a modal operator that obeys the axiom of consistency (axiom D). In what is the most closely related work to ours, Herzig et al. [163] combine the two approaches of belief merging and the merging of accessibility relations to define a logic for modeling explicit and implicit distributed beliefs. Explicit distributed belief is obtained from each agent's individual belief base; while implicit belief is derived from the group's collective belief base. In addition, they also introduce customized belief combination operators to model consistent distributed beliefs.

Common belief, denoted $CB_G\varphi$, is defined as: all agents in G believe φ, all agents in G believe that all agents in G believe φ, all agents in G believe ..., up to an infinite depth of nesting. The existing work [164–168] reasons for belief on belief bases or possible worlds. The PWP approach (in Chapter 3) forms the common knowledge of a group by finding the fixed point (see Definition 3.7) intersection of all perspectives from all agents in the group, showing that this fixed point always exists within a finite bound. However, this approach cannot handle justified beliefs.

5.3 Group Justified Perspective (GJP) Model

In this section, we formally propose our group justified perspective (GJP) model by adding group operations for uniform belief, distributed belief, and common belief to the JP model, inheriting the existing group modal operators from the PWP approach.

5.3.1 Preliminaries

Firstly, recall the signature of both PWP and JP models as follows: A *signature* Σ is described by the tuple $\Sigma = (Agt, V, D, \mathcal{R})$, with Agt being a finite set of agent identifiers (of size k), V as a finite set of variables (of size m) such that $Agt \subseteq V$ and $m \leq k$, implying agent identifiers serve as variables. Furthermore, D denotes the set of all domains, where each D_{v_i} corresponds to a possibly infinite domain of constant symbols for each variable $v_i \in V$. Lastly, $\mathcal{R}$ denotes a finite collection of predicate symbols. Domains can be discrete or continuous.

In addition, we follow the same definitions and notations used in the JP model from Section 4.3, including but not limited to, the definition of state (a set of variable assignments), the state space ($\mathcal{S}$), the complete-state space ($\mathcal{S}_c$), the sequence space ($\vec{\mathcal{S}}$), the complete sequence space ($\vec{\mathcal{S}_c}$), and the override function for both state (Definition 3.10) and sequence (Definition 4.1).

Then, the language of the GJP model can be defined by the following grammar:

Definition 5.1 (Language)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, the language $\mathcal{L}_{GKB}(\Sigma)$ is defined by the grammar:

$$\begin{aligned}
\varphi &::= r(V_r) \mid \neg\varphi \mid \varphi \wedge \varphi \mid S_i v \mid S_i\varphi \mid K_i\varphi,\\
\varphi &::= ES_G\varphi \mid DS_G\varphi \mid CS_G\varphi \mid EK_G\varphi \mid DK_G\varphi \mid CK_G\varphi,\\
\varphi &::= B_i\varphi \mid EB_G\varphi \mid DB_G\varphi \mid CB_G\varphi,
\end{aligned}$$

where $r \in \mathbb{R}$, $V_r \subseteq V$ are the terms of r, $r(V_r)$ are predicates and $\mathcal{R}$ is the set of all predicates; $i \in Agt$ is any agent and $G \subseteq Agt$ is a group of agents.

The group seeing operators, ES, DS and CS, and knowledge operators, EK, DK and CK are from the PWP model (in Definition 3.6), while the B_i operator is from the JP model (in Definition 4.2). In this section, we add the operators $EB_G\varphi$, $DB_G\varphi$ and $CB_G\varphi$ to represent that agents from group G jointly, distributedly and commonly believe φ respectively.

The definition of a model instance in GJP is the same as defined in Definition 4.3. We copied it as follows for easier referencing:

Quoted text: "

Definition 5.2 (JP Model)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, an instance of the justified perspective model M is defined as:

$$M = (Agt, V, D, \pi, O_1, \dots, O_k),$$

in which Agt, V, D are from the given signature, π is the evaluation function. The detailed definition is given as follows:

- The domain of variable $v \in V$ is D_v, which is a set of all possible values of v (from the definition of the signature). In here, a "None" value represented by symbol $\perp$ is included ($D_v := D_v \cup \{\perp\}$), which represents that the value of a variable is not part of a particular agents' observation. A special complete state is that a state with all variables assigned with $\perp$, which denoted as $s_\perp$ ($s_\perp = \{v = \perp \mid v \in V\}$). Thus, a special sequence is a sequence with all state as $s_\perp$, which denoted as $\vec{s_\perp}$.
- The interpretation function $\pi : \mathcal{S} \times \mathcal{R} \to \{true, false\}$ that determines whether the atomic term $r(V_r)$ is true in s. π is undefined if any of its arguments t_i is a variable $v \in V$ that is not assigned a value in a local state s, i.e. $v \notin s \vee v \neq \perp$.
- Functions $O_1, \dots, O_k$ are inherited from PWP model defined in Definition 3.4. In addition, $O_i(\vec{s}) = [O_i(\vec{s}[0]), \dots, O_i(\vec{s}[n])]$ for a sequence $\vec{s}$ with length of $n + 1$.

"

Similar, since a state is a set (of variable assignments), the set operations are also applicable on the state, such as set union operator "$\cup$" and set minus operator "$\backslash$". In addition, we also follow Lemma 4.4 in the JP model, in which: for any variable $v \in V$, we have $v = \perp \in \{v = e\}$ and $v \in \{v = e\}$ for any $e \in D_v$.

As for the functions, the retrieval function R and the justified perspective function f

follow their original definitions (Definition 4.5 and Definition 4.6) in JP as well. Additionally, the function C for possible sequences (Definition 4.12) is included.

The last part is the semantics. Similarly as in the PWP model and the JP model, we provide both complete semantics and ternary semantics. Since we are only introducing the group belief operators in this chapter, the semantics for other operators are inherited from their original definition in the corresponding model. Specifically, for individual knowledge (including seeing) and belief operators, item (a) to item (h) from the JP model is inherited (in Definition 4.17 for the complete semantics and in Definition 4.18 for the ternary semantics). In addition, for the group seeing and knowledge operators, item (i) to item (p) are adopted from item (g) to item (n) in Definition 3.12 for the complete semantics and in Definition 3.22 for the ternary semantics by converting the input from a state to a sequence. Therefore, the item index for the group belief operators starts from character 'q'. The full definitions (including all epistemic operators that our model can handle) of each semantics are provided at the end of this thesis (In Section 6.1).

5.3.2 Semantics for Group Belief

In this section, we define group justified perspective functions for uniform belief, distributed belief, and common belief, and add ternary semantics for them.

5.3.2.1 Uniform Belief

Uniform Belief is straightforward. Since a uniform belief of φ is that everyone in the group believes φ, the uniform justified perspective function is just a set union of everyone's individual justified perspectives.

Definition 5.3. (Uniform Justified Perspectives)

$$ef_G(\vec{s}) = \bigcup_{i \in G} \{f_i(\vec{s})\}$$

Definition 5.4 (Ternary Semantics for Uniform Belief)**.** Omitting the model M for readability, uniform belief EB_G for group G is defined:

(q) $T[\vec{s}, EB_G\varphi] \quad = \quad \min(\{T[\vec{g}, \varphi] \mid \vec{g} \in ef_G(s_\perp\langle\vec{s}\rangle)\})$

The ternary value of $T[\vec{s}, EB_G\varphi]$ depends on the agent that holds the most conservative beliefs of φ.

While, for the complete semantics, whether all agents in the group uniformly believe φ depends on all possible sequences that any agent believes possible.

Definition 5.5 (Complete Semantics for Uniform Belief)**.** Uniform belief EB_G for group G is defined:

$$\text{(q)} \quad M, \vec{s} \models EB_G\varphi \quad \text{iff} \quad \forall i \in G, \forall \vec{w} \in C(f_i(\vec{s}), i), M, \vec{w} \models \varphi$$

That is, for each agent i in the group G, the set $C(f_i(\vec{s}), i)$ contains all possible sequences that agent i believes possible. If and only if φ holds for all $\vec{w} \in C(f_i(\vec{s}), i)$, we have $M, \vec{s} \models EB_G\varphi$.

5.3.2.2 Distributed Belief

Distributed Belief is more challenging than distributed knowledge. The Knowledge Axiom $T : K_i\varphi \Rightarrow \varphi$, which states that knowledge must be true, does not hold for belief. This means that agents can hold incorrect beliefs. If we simply take the distributed union of the perspectives for all agents $i \in G$, as it is done in PWP, we could obtain conflicting beliefs, so the implicit distributed belief would be inconsistent. To ensure consistency, we form the group distributed justified perspective instead of just uniting each agent's justified perspective. Intuitively, agents follow their own observations and "listen" to agents that have seen variables more recently. The distributed perspective function df is defined as follows.

Definition 5.6 (Distributed Justified Perspectives)**.** The distributed justified perspective function for a group of agents G is defined as follows:

$$df_G([s_0, \ldots, s_n]) = [s'_0, \ldots, s'_n]$$

where for all $t \in [0, n]$ and all $v \in \text{dom}(s_t)$:

$$\begin{aligned}
lt_v &= \max(\{j \mid v \in \textstyle\bigcup_{i\in G} O_i(s_j) \wedge j \leq t\} \cup \{-1\}), & (1)\\
e &= R([s_0, \dots, s_t], lt_v, v), & (2)\\
s''_t &= \{v = e \mid s_t(v) = e \vee v \notin \textstyle\bigcup_{i\in G} O_i(s_t\langle\{v = e\}\rangle)\}, & (3)\\
s'_t &= s_\perp\langle s''_t\rangle. & (4)
\end{aligned}$$

In this definition, the group distributed justified perspective follows everyone's observation and uses the retrieval function R (in Definition 4.5) to identify the value of the variables which are or were not seen by any agent from the group. Intuitively, given any agent i in the group, the value from i's observation in timestamp t, $O_i(s_t)$, which leads to knowledge, must be true (Axiom T) in s_t. While, the value of an unseen variable is determined by anyone in the group that saw it last. To be specific, the last timestamp the group sees v, lt_v, is determined by the group observation (formed by union), and then, value e is retrieved by identifying the closest value that is consistent with it. Line (3) ensures the "group memory" is consistent with the group observation, while Line (4) ensures the group justified perspective is a sequence of complete states. So, this definition mimics the definition of the JP function from Definition 4.6, except that the variable's value in a state s'_t are taken by the agent(s) that have the most recent view of it.

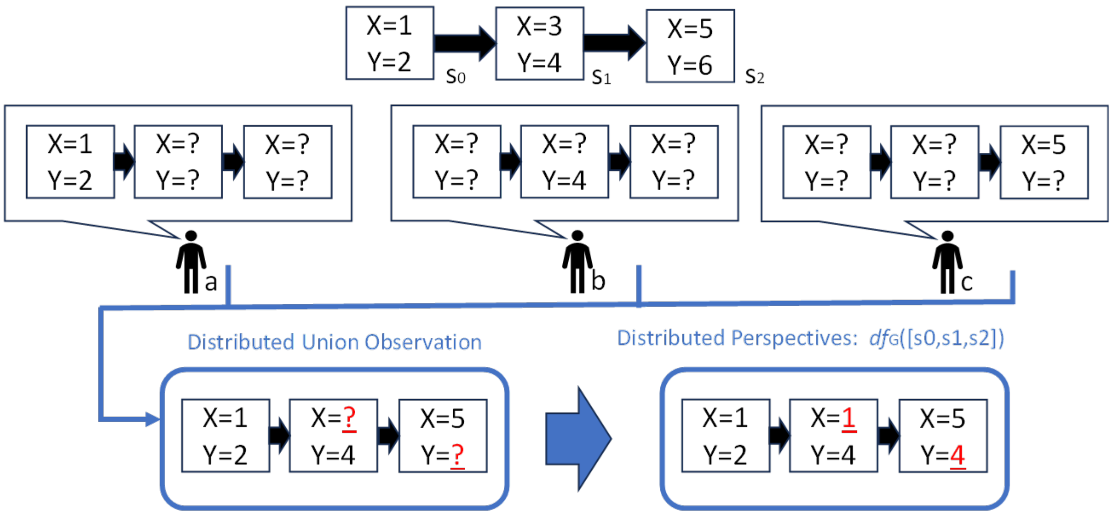

X=1 Y=2
s0
X=3 Y=4
s1
X=5 Y=6
s2
X=1 Y=2 | X=? Y=? | X=? Y=?
X=? Y=? | X=? Y=4 | X=? Y=?
X=? Y=? | X=? Y=? | X=5 Y=?
a
b
c
Distributed Union Observation
X=1 Y=2 | X=? Y=4 | X=5 Y=?
Distributed Perspectives: dfG([s0,s1,s2])
X=1 Y=2 | X=1 Y=4 | X=5 Y=4

FIGURE 5.3: State sequence $\vec{s}$ and $df_G(\vec{s})$ in Example 5.2.

Example 5.2. *Let the set of variables be* $V = \{x, y\}$*, domains be* $D_x = D_y = \{1, \dots, 6\}$*,* *and a state*[2] *sequence be* $\vec{s} = [s_0, s_1, s_2]$*, where:* $s_0 =$ 1-2*,* $s_1 =$ 3-4 *and* $s_2 =$ 5-6*. Assume a sees x and y in* s_0*, while b sees y in* s_1 *and c sees x in* s_2*. So,* $O_a(\vec{s})$*=[*1-2*,*τ-τ*,*τ-τ*],* $O_b(\vec{s})$*=[*τ-τ*,*τ-4*,*τ-τ*] and* $O_c(\vec{s})$*=[*τ-τ*,*τ-τ*,*5-τ*]. This is visualised in Figure 5.3.*

So, we can see from Example 5.2 that forming distributed belief is about finding the observation from each agent and deducing the value of group unseen variables, following the same intuition as JP Model (In Section 4.1). Missing values in group observations (noted as "?") are retrieved from the "group memory" (previous group observation), equating to retrieval from the agent who last observed this value according to the timestamp. Thus, $df_G(\vec{s}) = [1-2, 1-4, 5-4]$.

Definition 5.7 (Ternary Semantics for Distributed Belief)**.** The distributed ternary semantics are defined using function T, omitting the model M for readability:

$$\text{(r)} \quad T[\vec{s}, DB_G\varphi] \quad = \quad T[df_G(s_\perp\langle\vec{s}\rangle), \varphi]$$

This semantics guarantees that the group distributed justified belief is consistent. That is done by only merging agents' observations into the group distributed observation, which is consistent with the global state, and deducing the value of unseen variables from it. This definition is particularly nice as many existing definitions of distributed belief require us to define preference relations over e.g. agents or states, to resolve conflicts; see e.g. [169]. In our definition, the preference relation is implicit – it prefers more recent observations over older observations.

In order to give a definition for the complete semantics, identifying the possible sequences that are distributedly believed by the group is necessary. Therefore, we propose the following definition (adapted from the possible sequence function in Definition 4.12):

Definition 5.8 (Distributed Possible Sequence Function)**.** Given a state sequence $\vec{s}$ for agent i (could be either justified perspective of i, or a sequence of observations of i) with length of $n+1$, all possible sequences that agree with $\vec{s}$ can be generated by the possible sequence function, $C_d : \vec{\mathcal{S}} \times Agt \rightarrow \mathcal{P}(\vec{\mathcal{S}_c})$, can be defined as:

$$C_d(\vec{s}, G) = \{[w_0, \ldots, w_n] \;\mid w_0 \in W_0, \ldots w_n \in W_n\}$$

[2]We use the shorthand m-n to represent the state $[x = m, y = n]$ and τ to represent the 'value' of an unseen variable.

where for all $t \in [0, n]$:

$$\begin{aligned} W_t &= \{w' \mid w' \in W'_t, \forall v \in \textstyle\bigcup_{i \in G} O_i(w') \to v \in \bigcup_{i \in G} O_i(\vec{s}[t])\} \\ W'_t &= \{s_c\langle \vec{s}[t] \setminus s_\perp \rangle \ \mid s_c \in \mathcal{S}_c\} \end{aligned}$$

This function is the same as the possible sequence function, which generates all possible worlds in W'_t and filters out worlds containing inconsistent observations in W_t. The only difference is the observation used to ensure consistency becomes the group distributed observations $(\bigcup_{i \in G} O_i(\vec{s}[t]))$.

Then, the complete semantics for the distributed belief can be defined as:

Definition 5.9 (Complete Semantics for Distributed Belief)**.** Distributed belief DB_G for group G is defined:

(r) $M, \vec{s} \vDash DB_G\varphi$ iff $\forall \vec{w} \in C_d(df_G(\vec{s}), G), M, \vec{w} \vDash \varphi$

An intuitive example (Example 5.3) is given, adapted from Example 4.2, which is used to demonstrate that the past value is removed by indirect inferences in the JP function.

Example 5.3. *Consider a corridor with 3 rooms, r_1, r_2, and r_3. Three agents a and b are all located in r_1, while agent c is located in r_3. They can only observe the room that they are in. Let a plan be agent b moving to r_2.*

The global sequence $\vec{s_g}$, each agent's observations $O_i(\vec{s_g})$ and justified perspective $f_i(\vec{s_g})$ are as follows:

- $\vec{s_g}$ is $[\{loc_a = r_1, loc_b = r_1, loc_c = r_3\}, \{loc_a = r_1, loc_b = r_2, loc_c = r_3\}]$;
- $O_a(\vec{s_g}) = [\{loc_a = r_1, loc_b = r_1\}, \{loc_a = r_1\}]$;
- $f_a(\vec{s_g}) = [\{loc_a = r_1, loc_b = r_1, loc_c = \perp\}, \{loc_a = r_1, loc_b = \perp, loc_c = \perp\}]$;
- $O_b(\vec{s_g}) = [\{loc_a = r_1, loc_b = r_1\}, \{loc_b = r_2\}]$;
- $f_b(\vec{s_g}) = [\{loc_a = r_1, loc_b = r_1, loc_c = \perp\}, \{loc_a = r_1, loc_b = r_2, loc_c = \perp\}]$;
- $O_c(\vec{s_g}) = [\{loc_c = r_3\}, \{loc_c = r_3\}]$;
- $f_c(\vec{s_g}) = [\{loc_a = \perp, loc_b = \perp, loc_c = r_3\}, \{loc_a = \perp, loc_b = \perp, loc_c = r_3\}]$.

The reasoning about each agent's justified perspective can be found after Example 4.2. While for the distributed justified perspectives, $df_{\{a,c\}}$ needs some discussion.

Since the locations of a and c are seen by themselves all the time, only the value of b needs to be constructed. According to Definition 5.6, lt_{loc_b} is the latest timestamp that variable loc_b is seen by any agent from the group $\{a, c\}$, which is 0 since no one sees b in timestamp 1 and a saw b in timestamp 0. Then, following Line 2 in Definition 5.6, $R(\vec{s_g}, 0, loc_b)$ returns r_1, which is the value of loc_b when a and b were both in r_1. However, based on Line 3, if loc_b is r_1, then loc_b should be in the distributed observation of a and c, which indirectly infers that loc_b should not be r_1. Therefore, we have:

$$df_{\{a,c\}}(\vec{s_g}) = [\{loc_a{=}r_1, loc_b{=}r_1, loc_c{=}r_3\}, \{loc_a{=}r_1, loc_b{=}\bot, loc_c{=}r_3\}]$$

It seems there is no distributed belief in group $\{a, c\}$ about b's location. However, following the complete semantics, we have $M, \vec{s} \vDash DB_{\{a,c\}}(loc_b{=}r_2)$ since $C_b(df_{\{a,c\}}(\vec{s_g}), \{a, c\})$ is $\{[\{loc_a{=}r_1, loc_b{=}r_1, loc_c{=}r_3\}, \{loc_a{=}r_1, loc_b{=}r_2, loc_c{=}r_3\}]\}$. Specifically, there are three possible states in W_1' from $C_b(df_{\{a,c\}}(\vec{s_g}), \{a, c\})$, which are:

- $w_1' = \{loc_a{=}r_1, loc_b{=}r_1, loc_c{=}r_3\}$
- $w_2' = \{loc_a{=}r_1, loc_b{=}r_2, loc_c{=}r_3\}$
- $w_3' = \{loc_a{=}r_1, loc_b{=}r_3, loc_c{=}r_3\}$

While state w_1' and w_3' are removed when generating W_1 from $C_b(df_{\{a,c\}}(\vec{s_g}), \{a, c\})$ because if $loc_b{=}r_1$ or $loc_b{=}r_3$, then the group of a and c should distributively see loc_b in timestamp 1.

5.3.2.3 Common Belief

Common belief is the infinite nesting of beliefs. Our definition avoids having to calculate the infinite regression by calculating the fixed point of the group's perspectives.

Definition 5.10 (Common Justified Perspectives)**.** Given a set of perspectives (that is, a set of sequences of states) $\vec{S}$, the common justified perspective is defined as:

$$cf_G(\vec{s}) = \begin{cases} \bigcup_{\vec{s} \in \vec{S}} ef_G(\vec{s}) & \text{if } \bigcup_{\vec{s} \in \vec{S}} ef_G(\vec{s}) = \vec{s} \\ cf_G(\bigcup_{\vec{s} \in \vec{S}} ef_G(\vec{s})) & \text{otherwise.} \end{cases}$$

The function applies a set union on the uniform perspectives of the group for each input perspective. Then, the common perspective function repeatedly calls itself by using the output of one iteration as the input of the next iteration, until the input set and output set are the same, which means a convergence of the common perspectives. Semantically speaking, each iteration adds one level deeper nested perspectives of everyone's uniform belief for evaluation on whether everyone in the group believes.

The common justified perspectives function cf_G contains the fixed point of all agents' perspectives, their perspectives about others' perspectives, and so on to infinite depth. Although the depth is infinite, the definition of cf_G converges in finite iterations:

Theorem 5.11. *Given a state sequence of length n, the iterations needed for $cf_G(\vec{S})$ to converge are bounded above $2^{|V| \times n}$.*

Proof. Since for each variable in the last state of a justified perspective $\vec{w}$, its value is either visible (same as its in the last state of the global perspective), or not visible (same as its in the second-last state from $\vec{w}$), the number of possible states in each index of a justified perspective is $2^{|V|}$. So, the number of possible perspectives given a global state sequence $\vec{s}$ with a length of n is $2^{|V| \times n}$. In calculating cf_G, either the base case holds (that is, combining the perspective of the group for all $\vec{s} \in \vec{S}$ does not change the common perspective), so it terminates and adds no new perspectives; or the recursive step holds. In this case, the input of the cf function is a set that contains perspectives from each agent in the format of $\vec{S} = \{f_i(\vec{s}), f_i(\vec{s}'), \cdots \mid \forall i \in G\}$. Then, we apply f_j for each agent j in the group G on each perspective from $\vec{S}$ as $\vec{S}' = \bigcup_{\vec{s} \in \vec{S}} ef_G(\vec{s})$. For each $f_i(\vec{s})$ from $\vec{S}$, we have $f_j(f_i(\vec{s}))$ in $\vec{S}'$ for each agent j in the group G. With Theorem 4.11, we have $f_j(f_i(\vec{s})) = f_i(\vec{s})$ when $i = j$. Therefore, we have $\vec{S} \subseteq \vec{S}'$. At worst, we add one new sequence each iteration, meaning that $cf_G(\vec{S})$ converges by at most $2^{|V| \times n}$ iterations. □

Although in the worst-case scenario, the maximum number of iterations is $|cf_G(\{\vec{s}\})|$, practically, in our experiments, we find that it converges after a few iterations (see Section 5.4).

Definition 5.12 (Ternary Semantics for Common Belief)**.** The group ternary semantics are defined using the function T, omitting the model M for readability:

$$\text{(s)} \quad T[\vec{s}, CB_G\varphi] \quad = \quad \min(\{T[\vec{g}, \varphi] \mid \vec{g} \in cf_G(\{s_\perp\langle\vec{s}\rangle\})\})$$

Similar to the complete semantics for distributed belief, a new possible sequence function is needed to identify all sequences that are commonly believed to be possible by the group.

Definition 5.13 (Common Possible Sequence Function)**.** Given a state sequence $\vec{s}$ for agent i (could be either justified perspective of i, or a sequence of observations of i) with length of $n+1$, all possible sequences that agree with $\vec{s}$ can be generated by the possible sequence function, $C_c : \vec{\mathcal{S}} \times Agt \rightarrow \mathcal{P}(\vec{\mathcal{S}_c})$, can be defined as:

$$C_c(\vec{s}, G) = \{\big[w_0, \ldots, w_n\big] \ \mid w_0 \in W_0, \ldots w_n \in W_n\}$$

where for all $t \in [0, n]$:

$$\begin{aligned} W_t \quad &= \{w' \mid w' \in W'_t, \forall v \in cO(G, w') \rightarrow v \in cO(G, \vec{s}[t])\} \\ W'_t \quad &= \{s_c\langle\vec{s}[t] \setminus s_\perp\rangle \ \mid s_c \in \mathcal{S}_c\} \\ &cO(G, s) \text{ is defined in Definition 3.7} \end{aligned}$$

Compared to the distributed possible sequence function C_d, the common possible sequence function only removes a possible value when everyone in the group commonly sees it in w' but not in $\vec{s}[t]$, which means everyone did not commonly see it before filling in the missing variables with their possible values.

Then, the complete semantics for the distributed belief can be defined as:

Definition 5.14 (Complete Semantics for Common Belief)**.** Common belief CB_G for group G is defined:

$$\text{(s)} \quad (M, \vec{s}) \models CB_G\varphi \quad \text{iff} \quad \forall \vec{c} \in cf_G(\{\vec{s}\}), \forall \vec{w} \in C_c(\vec{c}, G), (M, \vec{w}) \models \varphi$$

An example for group justified perspective functions is provided using the same problem in Example 5.1 [3] as follows:

Example 5.4. *For example with Plan 5.1, let $G = \{a, b\}$ We have $ef_G(\vec{s})$ is $\{[\tau, 6, 6, 6, 6], [\tau, \tau, \tau, \tau, 5]\}$. Then, since the current $ef_G(\vec{s})$ is not equal to $\{\vec{s}\}$, we nestedly apply ef_G to generate $cf_G(\{\vec{s}\})$. From Theorem 4.11, we have $f_a(f_a(\vec{s})) = f_a(\vec{s})$ and $f_b(f_b(\vec{s})) = f_b(\vec{s})$. $f_b(f_a(\vec{s}))$ is $[\tau, \tau, \tau, \tau, 6]$ and $f_a(f_b(\vec{s}))$ is $[\tau, \tau, \tau, \tau, 5]$. So that, the current cf_G is a set that contains the following perspectives: 1: $[\tau, 6, 6, 6, 6]$; 2: $[\tau, \tau, \tau, \tau, 5]$; 3: $[\tau, \tau, \tau, \tau, 6]$; 4: $[\tau, \tau, \tau, \tau, 5]$.*

Since this is also not equal to $ef_G(\vec{s})$, we again apply ef_G on each perspective. Item (1) and (2) result in the same set, as the previous step, while both f_a and f_b on item (3) result in item (3) itself and both f_a and f_b on item (4) result in item (4) itself. Now, we have that $cf_G(\{\vec{s}\})$ has converged.

For ternary semantics, following Example 5.4, we have $ef_G(\vec{s}) = \{[\tau, 6, 6, 6, 6], [\tau, \tau, \tau, \tau, 5]\}$. The ternary representation $T[\vec{s}, EB_G n < 7]$ is evaluated as the minimum value in $\{T[[\tau, 6, 6, 6, 6], n < 7], T[[\tau, \tau, \tau, \tau, 5], n < 7]\}$, which is $\{1, 1\}$ due to $\{\pi(6, n < 7), \pi(5, n < 7)\}$.

For the $T[\vec{s}, CB_G n < 7]$, the (converged) common group justified perspectives are $\{[\tau, 6, 6, 6, 6], [\tau, \tau, \tau, \tau, 5], [\tau, \tau, \tau, \tau, 6], [\tau, \tau, \tau, \tau, 5]\}$, which is evaluated as $\{1, 1, 1, 1\}$. Therefore, $T[\vec{s}, CB_G n < 7]$ is 1.

To sum up, this section presents semantics for group joint, distributed, and common belief. Given that joint belief evaluates each agent's justified perspective, while distributed belief synthesizes a justified perspective from a group's collective observations, the time complexity of these is polynomial, specifically scaled by the number of agents, analogous to the JP model. Theorem 5.11 shows that the worst-case time complexity for common belief is exponential, factoring in the iterations to identify the fix-point set (common justified perspectives) of individual justified perspectives.

[3] For simplicity, we only show the value of number q instead of the agent's local state. The variables `(peeking a p)`, `(peeking a q)`, `(peeking b p)` and `(peeking b q)` are visible to all agents at all times.

5.4 Experiments

Since there are no planning benchmarks for group belief, we select three domains (Number, Grapevine, and Big Brother Logic) from the previous chapter and add several challenging problem instances (7 for each domain) that use group belief, including instances with inconsistent or nested beliefs.

5.4.1 Implementation

The source code of the planner, the domain, the problem, and external function files, as well as experimental results, are downloadable from:

`https://github.com/guanghuhappysf128/bpwp/tree/aamas`.

We use the F-PDDL encoding (in Section 4.4.1.4). To demonstrate the efficiency of our model instead of the particular search algorithms, we use the BrFS (Breadth-First Search) search algorithm with duplicate removal, which is `bfsdc` in Section 4.4.2.3. The experiments are run on a Linux machine (Ubuntu 20.04) with 8 CPUs (Intel i7-10510U 1.80GHz) and 16GB RAM. The external functions, implemented in Python, evaluate the belief formulae (either in action preconditions or goals) as search nodes are generated. We implement the group justified perspective model and its corresponding ternary semantics. All results are shown in Table 5.1.

5.4.2 Number

Number is an adapted domain from the coin domain in Section 4.5.1.4. The problem settings, as described in Example 5.1, are of agents taking turns to peek into boxes containing one changeable number each. For simplicity, we only use one number q in the experiments. We created 7 instances to evaluate our group's justified perspective operators and ran these through our planner. As can be seen from the column Max and Avg, the number of iterations in cf_G that it takes to find a fixed-point for a common perspective set is around $2-3$, which is nearly the same as checking one of the nested uniform perspectives. The planner was able to solve the problems with low time costs even for a complex problem such as $N4$.

ID	Exp	Gen	Common		External		Total	$\lvert p\rvert$	Goals
			Max	Avg	$\lvert calls\rvert$	AT(ms)	T(s)		
N0	39	140	0	0	207	0.085	0.048	4	$EB_G q < 6$
N1	7	25	0	0	33	0.095	0.006	2	$DB_G q < 6$
N2	39	140	4	2.199	207	0.255	0.075	4	$CB_G q < 6$
N3	120	435	4	2.461	668	0.414	0.354	6	$EB_G q < 6 \wedge \neg CB_G q < 6$
N4	347	1273	5	2.716	2041	0.798	1.906	8	$EB_G EB_G q < 6 \wedge \neg CB_G q < 6$
N5	31	111	3	2.134	161	0.307	0.068	4	$\neg EB_G n{=}5 \wedge \neg EB_G n{=}6 \wedge CB_G q < 6$
N6	50	177	3	1.649	257	0.301	0.107	4	$B_a CB_G n{=}6 \wedge B_b CB_G n{=}5$
G0	5	35	4	3.028	41	3.17	0.149	1	$CB_G sct_a{=}t$
G1	66	450	4	3.195	691	5.94	4.542	4	$EB_G sct_a{=}t \wedge CB_G sct_a{=}t \rightarrow \frac{1}{2}$
G2	240	1828	5	3.496	3282	9.984	35.764	6	$EB_G EB_G sct_a{=}t \wedge CB_G sct_a{=}t \rightarrow \frac{1}{2}$
G3	103	913	4	3.018	1138	4.882	6.14	3	$B_b CB_G sct_a{=}f \wedge CB_{\{a,c,d\}} sct_a{=}t$
G4	328	2959	4	2.664	3792	10.717	42.776	4	$CB_{\{b,c\}} CB_G sct_a{=}f \wedge CB_{\{a,d\}} sct_a{=}t$
G5	66	450	4	3.195	691	6.469	4.89	4	$DB_G EB_G sct_a{=}t \wedge CB_G sct_a{=}t \rightarrow \frac{1}{2}$
G6	70	455	0	0	734	1.326	1.396	4	$DB_G EB_G sct_a{=}t \wedge B_a EB_G sct_a{=}t \rightarrow \frac{1}{2}$
BBL0	2	8	4	3.111	11	9.387	0.107	1	$CB_G o_2{=}2$
BBL1	5	19	4	3.143	26	10.617	0.279	2	$EB_G o_2{=}2 \wedge CB_G o_2{=}2 \rightarrow \frac{1}{2}$
BBL2	177	708	4	3.492	1011	21.724	22.072	5	$CB_G o_1{=}1$
BBL3	189	756	4	3.485	1067	42.853	45.841	5	$CB_G(o_1{=}1 \wedge o_2{=}2)$
BBL4	1595	6380	0	0	10295	0.486	6.05	9	$EB_G(o_1{=}1 \wedge o_2{=}2 \wedge o_3{=}3)$
BBL5	1595	6380	0	0	10295	0.39	5.007	9	$EB_G o_1 < o_2$
BBL6	2	8	0	0	11	0.663	0.009	1	$DB_G o_1 < o_2$

TABLE 5.1: Result for three domains (N0-N6, G0-G6, BBL0-BBL6 are instances for Number, Grapevine and BBL respectively). G represents the group of agents – $\{a, b\}$ for Number and BBL; and, $\{a, b, c, d\}$ for Grapevine. 'Exp" and "Gen" are the number of nodes expanded and generated during search, "Max" and "Avg" under "Common" as the maximum and average level of nesting required to compute cf_G, $|calls|$ and "AT(ms)" under "External" as the number and average time of external function calls, and $|p|$ as plan length. Since we implement the ternary semantics, we denote the ternary evaluation result $T[\vec{s}, \varphi]$ equal to 0, $\frac{1}{2}$ and 1 as $\neg\varphi$, $\varphi \rightarrow \frac{1}{2}$ and φ respectively.

Item (1) would be solved by just one agent peeking into the box and the number decreased by 1. Items (0) and (2) are solved by decrementing the number first and all agents take turns to peek, as both of the agents a and b uniformly and commonly believe $n = 5$. Items (N3) and (N4) are more challenging and deserve some discussion. Item (N3) means both a and b believe q is smaller than 6, while they do not commonly believe it. The returned plan is: "**(peek a)**", "**(return a)**", "**(peek b)**", "**(decrement**", "**(return b)**", "**(peek a)**". Intuitively, a sees q first, and then b sees q changed. When a sees q again, as the design decision made in the JP function (Definition 4.6), a will assume when b sees q, q is the value a believes (which is 6).

Item (N4) means both a and b believe that both of them believe q is smaller than 6, while they still do not commonly believe it. The returned plan is: "**(peek a)**", "**(return a)**", "**(peek b)**", "**(subtraction)**", "**(return b)**", "**(peek a)**", "**(return a)**", "**(peek b)**". It is simpler to show the perspectives of each agent [4]:

[4]Here, we also omit $peeking_a$ and $peeking_b$ for readability.

$$\begin{aligned}
[f_a(\vec{s})] &= [\tau, 6, 6, 6, 6, 6, 5, 5, 5] \\
[f_b(\vec{s})] &= [\tau, \tau, 6, 6, 6, 6, 5, 5, 5] \\
[f_b(f_a(\vec{s}))] &= [\tau, \tau, \tau, 6, 6, 6, 6, 6, 5] \\
[f_a(f_b(\vec{s}))] &= [\tau, \tau, \tau, 6, 6, 6, 5, 5, 5] \\
[f_a(f_b(f_a(\vec{s})))] &= [\tau, \tau, \tau, 6, 6, 6, 6, 6, 6]
\end{aligned}$$

In the last state in the perspective, a sees q , where the q is $f_b(f_a(\vec{s}))[6] = 6$. So that we have $EB_G EB_G q < 6 \wedge \neg CB_G q < 6$. Item (5) and item (6) are the same as Task 2 and Task 1 in our motivating example (Example 5.1) respectively.

5.4.3 Grapevine

Grapevine is a benchmark domain in epistemic planning [88]. In two adjacent rooms, 4 agents, in the same room, each have their own secret. All agents can move between the two rooms and *share* or *lie* about a secret sct_i, if either the secret is their own secret or they have heard the secret from someone. That is, they need to believe the secret ($B_i sct_i$) before they can share or lie about it. Initially, all agents a, b, c, d are located in the left room, and they only know about their own secrets.

We created 7 instances inspired from the original non-group instances [88], which include some formulae with arbitrary nesting, such as DB nested with EB. As outlined in the table, the average number of iterations of cf_G used to find the fixed point of common perspectives is around $2.5 - 3.5$.

As for the specific group beliefs in the goal conditions, item (0) is trivial. To form a common belief in the group about sct_a, agent a just needs to share sct_a when everyone is in the same room.

Items (G1) and (G2) are more interesting. Item (G1) represents that everyone in a group believes sct_a but there is no common belief on sct_a, while item (G2) represents that everyone in a group believes that everyone believes sct_a, but still no common belief is formed. The plan to solve (G1) is still intuitive: "**(move-right b)**", "**(sharing a** sct_a**)**", "**(move-right a)**" and "**(sharing a** sct_a**)**". One of the agents moves to the other room first, then a shares sct_a, and moves to the other room to share with that agent individually. The plan for (G2) is the same as (G1) but with 2 extra actions: "**(move-left b)**" and "**(sharing b,**sct_a**)**". Intuitively, after everyone knows sct_a, agent

b returns to the original room and tells others that it believes the sct_a as well by sharing sct_a.

Items (3) and (4) form different common beliefs. Item (3) is formed by agent a lying about sct_a while b is in the room and shares the actual secret value after b leaves. While item (4) is formed by agent a doing the same process for both agent b and c.

Different from the plan of items (1) and (5), item (6) contains $B_a EB_G sct_a \rightarrow \frac{1}{2}$. So that a's secret is shared by another agent to b in another room rather than a.

5.4.4 Big Brother Logic (BBL)

BBL [2] is a domain that stationary cameras can turn and observe with a certain angular range in a two-dimensional plane. The cameras do not have any volume to block others' line of sight. For simplicity, we limit the camera's angle of turning to a set $\{0°, \pm 45°, \pm 90°, \pm 135°, 180°\}$ and the field is in a 5×5 grid. Initially, camera a and b are located at $(3, 3)$ and $(1, 1)$ with directions of $-135°$ and $90°$, while o_1, o_2, and o_3 are located at $(0, 0)$, $(2, 2)$, and $(3, 3)$, with values 1, 2, and 3 respectively. We have run 7 instances adapted from the origin group knowledge instance in the PWP approach (in Section 3.6.2). The average number of iterations of cf_G needed to find a fixed-point common perspective is slightly higher than $3.0 - 3.5$; and again, the main cost is evaluating epistemic formulae, but the search algorithm also contributes to computation time.

Items (BBL5) and (BBL6) are worthy of discussion. For item (BBL5), both agents need to turn around to see both o_1 and o_3. While for item (BBL6), only agent b turns to see o_3. By "pulling" agent a's and b's justified perspective together, we have both o_1 and o_3 in $df_G(\vec{s})[n]$.

5.5 Conclusion

In this chapter, we extend the JP model in Chapter 4 to handle group beliefs, namely the GJP model; provide definitions for *ef* (Definition 5.3), *df* (Definition 5.6), and *cf* (Definition 5.10) to construct the group's uniform justified perspectives, the group's distributed justified perspective, and the group's common justified perspectives; define ternary (in Definition 5.4, Definition 5.7, and Definition 5.12) and complete (in Definition 5.5, Definition 5.9, and Definition 5.14) semantics for all group belief operators (EB, DB, and CB); implement its ternary semantics as a model-free planning tool; and demonstrate its expressiveness and efficiency on new domains. The results show that our approach can effectively handle multi-agent epistemic planning problems with group beliefs and do so efficiently, even with a simple prototype F-PDDL planner implementing a Breadth First Search with duplication elimination.

Chapter 6

Conclusion and Future Directions

In this chapter, we first summarize our work, followed by stating its contributions. Then, we discuss the limitations of our work and propose future directions.

6.1 Summary

This thesis has explored the challenges and advancements in epistemic planning, specifically addressing issues of scalability, generalizability, expressiveness, and efficiency through novel planning approaches based on Justified Perspectives (JP). The epistemic relations this work handles include agents' individual or group (nested) knowledge and beliefs, Given a signature Σ of the problem, those epistemic relations can be formally represented by the following language:

Definition 6.1 (Language $\mathcal{L}_{GKB}(\Sigma)$)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, the language $\mathcal{L}_{GKB}(\Sigma)$ is defined by the grammar:

$$\begin{aligned}
\varphi &::= r(V_r) \mid \neg\varphi \mid \varphi \land \varphi \mid S_i v \mid S_i\varphi \mid K_i\varphi \\
\varphi &::= ES_G\varphi \mid DS_G\varphi \mid CS_G\varphi \mid EK_G\varphi \mid DK_G\varphi \mid CK_G\varphi \\
\varphi &::= B_i\varphi \mid EB_G\varphi \mid DB_G\varphi \mid CB_G\varphi
\end{aligned}$$

where $r \in \mathbb{R}$, $V_r \subseteq V$ are the terms of r, $r(V_r)$ are predicates and $\mathcal{R}$ is the set of all predicates; $i \in Agt$ is any agent and $G \subseteq Agt$ is a group of agents.

Then, following our definitions of the state (Definition 3.2) and notation of the state space ($\mathcal{S}$), the complete state space ($\mathcal{S}_c$), the sequence space ($\vec{\mathcal{S}}$), and the complete sequence space ($\vec{\mathcal{S}_c}$), we can define an instance of our model as:

Definition 6.2 (Model)**.** Given a signature $\Sigma = (Agt, V, D, \mathcal{R})$, an instance of our model M is defined as:

$$M = (Agt, V, D, \pi, O_1, \dots, O_k),$$

in which Agt, V, D are from the given signature, π is the evaluation function. The detailed definition is given as follows:

- The domain of variable $v \in V$ is D_v, which is a set of all possible values of v (from the definition of the signature). In here, a "None" value represented by symbol $\perp$ is included ($D_v := D_v \cup \{\perp\}$), which represents that the value of a variable is not part of a particular agents' observation. A state with all variables assigned with $\perp$, denoted as $s_\perp$ ($s_\perp = \{v = \perp \mid v \in V\}$). Thus, a special sequence is a sequence with all states as $s_\perp$, denoted as $\vec{s_\perp}$.
- The interpretation function $\pi : \mathcal{S} \times \mathcal{R} \to \{true, false\}$ determines whether the atomic term $r(V_r)$ is true in s. π is undefined if any of its arguments t_i is a variable $v \in V$ that is not assigned a value in a given state s, i.e. $v \notin s \vee v \neq \perp$.
- Functions $O_1, \dots, O_k$ are inherited from PWP model defined in Definition 3.4. In addition, $O_i(\vec{s}) = [O_i(\vec{s}[0]), \dots, O_i(\vec{s}[n])]$ for a sequence $\vec{s}$ with length of $n+1$.

Then, with the definitions of the sequence override function $\langle \quad \rangle$ (Definition 4.1), the retrieval function R (Definition 4.5), the justified perspective function f_i (Definition 4.6), the common observation function cO (Definition 3.7), the uniform justified perspective function ef (Definition 5.3), the distributed justified perspective function df (Definition 5.6), and the common justified perspective function cf (Definition 5.10), we can provide a ternary semantics for our model with language $\mathcal{L}_{GKB}(\Sigma)$ as follows:

Definition 6.3 (Ternary Semantics)**.** Give a signature $\Sigma = (Agt, V, D, \mathcal{R})$ and an instance of our model $M = (Agt, V, D, \pi, O_1, \dots, O_k)$ with the current state sequence as $\vec{s}$ with $n+1$ states, the ternary semantics for language $\mathcal{L}_{GKB}(\Sigma)$ can be defined as (where the model M is omitted for readability):

$$
\begin{array}{lll}
\text{(a)} & T[\vec{s}, r(V_r)] = & 1 \text{ if } \pi(\vec{s}[n], r(V_r)) = true; \\
 & & 0 \text{ else if } \pi(\vec{s}[n], r(V_r)) = false; \\
 & & \frac{1}{2} \text{ otherwise} \\
\text{(b)} & T[\vec{s}, \phi \wedge \psi] = & \min(T[\vec{s}, \phi], T[\vec{s}, \psi]) \\
\text{(c)} & T[\vec{s}, \neg\varphi] = & 1 - T[\vec{s}, \varphi] \\
\text{(d)} & T[\vec{s}, S_i v] = & \frac{1}{2} \text{ if } v \notin \vec{s}[n] \text{ or } i \notin \vec{s}[n] \\
 & & 0 \text{ else if } v \notin O_i(\vec{s}[n]) \\
 & & 1 \text{ otherwise} \\
\text{(e)} & T[\vec{s}, S_i\varphi] = & \frac{1}{2} \text{ if } T[\vec{s}, \varphi] = \frac{1}{2} \text{ or } i \notin \vec{s}[n]; \\
 & & 0 \text{ else if } T[O_i(\vec{s}), \varphi] = T[O_i(\vec{s}), \neg\varphi] = \frac{1}{2}; \\
 & & 1 \text{ otherwise} \\
\text{(f)} & T[\vec{s}, K_i\varphi] = & T[\vec{s}, \varphi \wedge S_i\varphi] \\
\text{(g)} & T[\vec{s}, H_i\varphi] = & \frac{1}{2} \text{ if } T[\vec{s}, \varphi] = \frac{1}{2} \\
 & & 0 \text{ else if } T[\vec{s}, B_i\varphi] = T[\vec{s}, B_i\neg\varphi] = \frac{1}{2}; \\
 & & 1 \text{ otherwise} \\
\text{(h)} & T[\vec{s}, B_i\varphi] = & T[f_i(\vec{s_\perp}\langle\vec{s}\rangle), \varphi] \\
\text{(i)} & T[\vec{s}, ES_G\alpha] = & min(\{T[\vec{s}, S_i\alpha] \mid i \in G\}), \\
\text{(j)} & T[\vec{s}, EK_G\varphi] = & T[\vec{s}, \varphi \wedge ES_i\varphi], \\
\text{(k)} & T[\vec{s}, DS_G v] = & \frac{1}{2} \text{ if } v \notin \vec{s}[n] \text{ or } \forall i \in G, i \notin \vec{s}[n]; \\
 & & 0 \text{ if } v \notin \bigcup_{i \in G} O_i(\vec{s}[n]); \\
 & & 1 \text{ otherwise,} \\
\text{(l)} & T[\vec{s}, DS_G\varphi] = & \frac{1}{2} \text{ if } T[\vec{s}, \varphi] = T[\vec{s}, \neg\varphi] = \frac{1}{2} \text{ or } \forall i \in G, i \notin \vec{s}[n]; \\
 & & 0 \text{ if } T[\bigcup_{i \in G} O_i(\vec{s}), \varphi] = T[\bigcup_{i \in G} O_i(\vec{s}), \neg\varphi] = \frac{1}{2}; \\
 & & 1 \text{ otherwise,} \\
\text{(m)} & T[\vec{s}, DK_G\varphi] = & T[\vec{s}, \varphi \wedge DS_G\varphi], \\
\text{(n)} & T[\vec{s}, CS_G v] = & \frac{1}{2} \text{ if } v \notin \vec{s}[n] \text{ or } \exists i \in G, i \notin \vec{s}[n]; \\
 & & 0 \text{ if } v \notin cO(G, \vec{s}[n]); \\
 & & 1 \text{ otherwise,} \\
\text{(o)} & T[\vec{s}, CS_G\varphi] = & \frac{1}{2} \text{ if } T[\vec{s}, \varphi] = T[\vec{s}, \neg\varphi] = \frac{1}{2} \text{ or } \exists i \in G, i \notin \vec{s}[n]; \\
 & & 0 \text{ if } T[cO(G, \vec{s}), \varphi] = T[cO(G, \vec{s}), \neg\varphi] = \frac{1}{2};
\end{array}
$$

1 otherwise,

(p) $T[\vec{s}, CK_G\varphi] = T[\vec{s}, \varphi \wedge CS_G\varphi]$,

(q) $T[\vec{s}, EB_G\varphi] = \min(\{T[\vec{g}, \varphi] \mid \vec{g} \in ef_G(s_\perp\langle\vec{s}\rangle)\})$

(r) $T[\vec{s}, DB_G\varphi] = T[df_G(s_\perp\langle\vec{s}\rangle), \varphi]$

(s) $T[\vec{s}, CB_G\varphi] = \min(\{T[\vec{g}, \varphi] \mid \vec{g} \in cf_G(\{s_\perp\langle\vec{s}\rangle\})\})$

where: $\bigcup_{i\in G} O_i(\vec{s})$ is $[\bigcup_{i\in G} O_i(\vec{s}[0]), \ldots, \bigcup_{i\in G} O_i(\vec{s}[0])]$,
$cO(G, \vec{s})$ is $[cO(G, \vec{s}[0]), \ldots, cO(G, \vec{s}[n])]$.

Then, with the definitions of the possible sequence function C (Definition 4.12), the distributed possible sequence function C_d (Definition 5.8), and the common possible sequence function C_c (Definition 5.13), we can define the complete semantics as follows:

Definition 6.4 (Complete Semantics)**.** Give a signature $\Sigma = (Agt, V, D, \mathcal{R})$ and an instance of our model $M = (Agt, V, D, \pi, O_1, \ldots, O_k)$ with the current state sequence as $\vec{s}$ with $n+1$ states, the complete semantics for language $\mathcal{L}_{GKB}(\Sigma)$ can be defined as:

(a) $(M, \vec{s}) \models r(V_r)$ iff $\pi(\vec{s}[n], r(V_r)) = true$

(b) $(M, \vec{s}) \models \phi \wedge \psi$ iff $(M, \vec{s}) \models \phi$ and $(M, \vec{s}) \models \psi$

(c) $(M, \vec{s}) \models \neg\varphi$ iff $(M, \vec{s}) \not\models \varphi$

(d) $(M, \vec{s}) \models S_i v$ iff $v \in O_i(\vec{s}[n])$

(e) $(M, \vec{s}) \models S_i\varphi$ iff $\forall\vec{g} \in C(O_i(\vec{s}), i), (M, \vec{g}) \models \varphi$ or
$\forall\vec{g} \in C(O_i(\vec{s}), i), (M, \vec{g}) \models \neg\varphi$

(f) $(M, \vec{s}) \models K_i\varphi$ iff $(M, \vec{s}) \models \varphi \wedge S_i\varphi$

(g) $(M, \vec{s}) \models H_i\varphi$ iff $(M, \vec{s}) \models B_i\varphi$, or $(M, \vec{s}) \models B_i\neg\varphi$

(h) $(M, \vec{s}) \models B_i\varphi$ iff $\forall\vec{g} \in C(f_i(\vec{s}), i), (M, \vec{g}) \models \varphi$

(i) $(M, \vec{s}) \models ES_G\alpha$ iff for all $i \in G$, $(M, \vec{s}) \models S_i\alpha$

(j) $(M, \vec{s}) \models EK_G\varphi$ iff $(M, \vec{s}) \models (\varphi \wedge ES_G\varphi)$

(k) $(M, \vec{s}) \models DS_G v$ iff $v \in \bigcup_{i\in G} O_i(\vec{s}[n])$ or $|D_v| = 1$

(l) $(M, \vec{s}) \models DS_G\varphi$ iff $\forall\vec{w} \in C_d(\bigcup_{i\in G} O_i(\vec{s}), G), M, \vec{w} \models \varphi$ or
$\forall\vec{w} \in C_d(\bigcup_{i\in G} O_i(\vec{s}), G), (M, \vec{w}) \models \neg\varphi$

(m) $(M, \vec{s}) \models DK_G\varphi$ iff $(M, \vec{s}) \models (\varphi \wedge DS_G\varphi)$

(n) $(M, \vec{s}) \models CS_G v$ iff $v \in cO(G, \vec{s}[n])$ or $|D_v| = 1$

(o) $(M, \vec{s}) \models CS_G \varphi$ iff $\forall \vec{w} \in C_c(cO(G, \vec{s})), (M, \vec{w}) \models \varphi$ or $\forall \vec{w} \in C_c(cO(G, \vec{s})), (M, \vec{w}) \models \neg\varphi$

(p) $(M, \vec{s}) \models CK_G \varphi$ iff $(M, \vec{s}) \models (\varphi \wedge CS_G \varphi)$

(q) $(M, \vec{s}) \models EB_G \varphi$ iff $\forall i \in G, \forall \vec{w} \in C(f_i(\vec{s}), i), (M, \vec{w}) \models \varphi$

(r) $(M, \vec{s}) \models DB_G \varphi$ iff $\forall \vec{w} \in C_d(df_G(\vec{s}), G), (M, \vec{w}) \models \varphi$

(s) $(M, \vec{s}) \models CB_G \varphi$ iff $\forall \vec{c} \in cf_G(\{\vec{s}\}), \forall \vec{w} \in C_c(\vec{c}, G), (M, \vec{w}) \models \varphi$

where: $\bigcup_{i \in G} O_i(\vec{s})$ is $[\bigcup_{i \in G} O_i(\vec{s}[0]), \ldots, \bigcup_{i \in G} O_i(\vec{s}[0])]$,
$cO(G, \vec{s})$ is $[cO(G, \vec{s}[0]), \ldots, cO(G, \vec{s}[n])]$.

Moreover, we show the soundness and completeness of our logic, as well as the axiomatic system it follows, by proposing and proving several theorems. In addition, we formalized the epistemic planning problems that can be solved by our model as EP-NM-F-STRIPS (in Section 4.4.1.3) and show its encoding with a PDDL-like language, F-PDDL (in Section 4.4.1.4). As the problem becomes non-Markovian, we implement several search algorithms (in Section 4.4.2.3) that also work for a non-Markovian setting, with duplication elimination (in Section 4.4.2.1) for EP-NM-F-STRIPS and ‘Have Not Seen‘ pruning (in Section 4.4.2.2) for a subset of EP-NM-F-STRIPS (EPD_{DO}).

Last but not least, to demonstrate the efficiency and expressiveness of our model, we perform experiments on benchmark domains, including SCs (in Section 4.5.1.1), Coin (in Section 4.5.1.4), Corridor (in Section 4.5.1.2), and Grapevine (in Section 4.5.1.3), as well as some trickier domains, which are either too complex or impractical to be modeled by other approaches, including BBL (in Section 4.5.1.5). Furthermore, we perform comprehensive empirical experiments in Section 4.5.2 to compare the implemented search algorithms and examine their performance (efficiency) on a large scale.

6.2 Contribution

This thesis answered the research question in Section 1.2, including the sub-research questions in Section 2.5.1. The model we proposed is an efficient, expressive, generalizable, and scalable framework for solving epistemic planning problems, unlike other existing approaches. The efficiency is shown by our experimental results (in Section 3.6, Section 4.5 and Section 5.4), while the scalability is ensured as our model reasons about epistemic relations in polynomial time (in Section 4.3.3.3). The usage of the external function with PDDL-like encoding (F-PDDL), which can be implemented by any programming language, ensures the expressiveness of our approach. As mentioned in Section 3.5.2, in order to model problem instances in a domain, the modeler only has to define the observation function (the seeing rule of this domain). This means the user does not need to have a comprehensive understanding of epistemic logic to use our framework, which ensures its generalizability.

Our primary contributions are summarized as follows:

- **Extension of Agent's Perspective Model**: We developed multiple semantic formats for it (named the new model as PWP model), balancing efficiency and completeness in knowledge reasoning.
- **Justified Perspective (JP) Model**: We introduced a framework that integrates justified beliefs, allowing agents to reason over unseen information using current and past observations.
- **Formalization of Epistemic Logic Encoding**: A robust encoding schema was proposed to model epistemic planning problems (EP-NM-F-STRIPS) in a structured planning language (F-PDDL).
- **Development of an Efficient Planner**: We implemented a F-STRIPS planner with state sequence as its external function input, and provided various search algorithms, improving computational feasibility while preserving epistemic expressiveness.
- **Expansion to Multi-Agent Group Beliefs**: We extended JP to group epistemics, handling distributed belief and common belief efficiently.
- **Comprehensive Empirical Evaluation**: Our experiments demonstrated the effectiveness of our approach over existing epistemic planning methods in terms of scalability and expressiveness.

Together, these contributions push the boundaries of planning-based epistemic reasoning in multi-agent systems and provide a foundation for applying these models in real-world AI applications.

6.3 Limitation

Despite the progress made, this research has certain limitations, which are mainly caused by the planning assumptions.

Although our epistemic reasoning process is shown to be polynomial, the lack of an efficient heuristic function is the bottleneck that limits the performance of our planner. This was caused by the epistemic planning problem becoming non-Markovian, which makes most of the state-of-the-art heuristic functions inapplicable, such as delete relaxation [72].

In addition, as classical planning is the basic fragment of planning-based techniques, the assumptions it follows limit its potential for real-world application. For example, the static assumption (Assumption 5) means the state only changes by the acting agent. This assumption does not make sense in many applications such as robotics, in which the environment in the real world could also change the state. For example, gravity could change the position of a falling object.

Another limitation is that the epistemic planning is claimed to be useful in a multi-agent cooperation setting or a human-agent interaction setting. However, most of the research in epistemic planning is based on centralised planning, which is not suitable for either setting.

6.4 Future Direction

This thesis provides a foundation for future work in epistemic planning, with several promising directions for further exploration.

Firstly, an efficient heuristic function for non-Markovian problems (or just EP-NM-F-STRIPS problems) would significantly improve the performance of our planner. We believe the key to finding such a heuristic function lies in two aspects, the agents' justified perspectives and the width-based search [80]. The agents' justified perspectives are used to evaluate epistemic formulae in the problem, which are used for duplication elimination. It acts like extracted features of the state sequence. In addition, width-based search algorithms can have decent performance without using any information from the action model [63], which means it only uses information from the current state. Our potential idea is to use the last states of all justified perspectives as the state representation to implement the width-based search algorithms.

Secondly, relaxing the assumptions of epistemic planning is another choice to improve its potential for real-world applications. Using the static assumptions as an example, currently, in the JP model, agents assume the unseen variables stay unchanged until they see evidence to suggest otherwise. This intuition does not make sense when reasoning about dynamic variables, such as the position of a falling object. Agents should assume the changing pattern of the variable, $h = \frac{1}{2}gt^2$ for a falling object, and predict its height based on the last observation and its changing pattern when they no longer see it. There is an on-going work [170] (supervised by me) to address this assumption. By integrating a model of evolution for state variables, the new *Predictive Justified Perspective* (PJP) model is able to handle the changing variables through a value updated mechanism based on existing observations. Although the newly generated perspectives (what agents believe) are not necessarily subsets of originated observations (what agents see), the semantics of the new model is proven to be still in **KD45**. In addition, by changing this value updated mechanism (predictive model) into: a noise handling mechanism, such as Kalman Filter [171]; or an outlier rejection mechanism, such as *Random Sample Consensus* (RANSAC) [172], the updated JP model is able to handle the noisy observations (relaxing Assumption 3), which is more suitable for a real-world application.

Last but not least, most of the existing work on epistemic planning follows a centralised setting, which is not suitable for its application. Agents' ability to reason about epistemic logic is essential when it's in a decentralised multi-agent setting to interact with other agents preferably without a predefined communication protocol. Thus, investigating decentralised epistemic planning is essential for its multi-agent applications. Besides, whether the epistemic planning framework can be used in a human-agent interaction setting is also important. Research in this setting is limited to a few instances [173, 174] However, none of those works contain comprehensive human experiments to verify the viability of using epistemic planning to model human belief inferences. Furthermore, the recent explosion of research on Large Language Models (LLMs) allows the epistemic planning model to potentially assess LLMs' reasoning capabilities [175, 176].

Appendix A

Byzantine Generals Domain

The form of the common knowledge between two agents a and b does not only rely on the intersection on both agents' perspectives, but also relies on the intersection on both agents' nested perspectives over that intersection, etc. Until we reach a fixed point, which one intersection l of agents' nested perspectives is the same as the intersection on agents' perspectives over l. A common example is provided as follows:

The classic example is the Byzantine Generals:

Example A.1. *There are 2 generals who cannot directly communicate and must decide on when to attack their common enemy. Each general will attack only if the 2 generals have common knowledge of the time of the attack, but such (infinitely-nested) common knowledge cannot be attained by sending a messenger back and forth k times between the generals, since on the last trip the messenger could fail to arrive.*

Let a and b be two generals, p_a and p_b be messages they want to send to each other. For simplicity, let's set the maximum nested depth is 4. The initial state is:

$\{p_a,\ K_aK_aK_ap_a,\ p_b,\ K_bK_bK_bp_b\}$.

By sending the messenger from a to b, the current state now becomes:

$\{p_a,\ K_aK_aK_ap_a,\ p_b,\ K_bK_bK_bp_b,\ K_bK_bK_bp_a,\ K_bK_bK_ap_a,\ K_bK_aK_ap_a\}$.

After that, let b send messenger back, the state becomes:

$\{p_a,\ K_aK_aK_ap_a,\ K_aK_aK_ap_b,\ K_aK_aK_bp_b,\ K_aK_bK_bp_b,\ K_aK_bK_ap_a,\ K_aK_bK_bp_a,$
$K_aK_aK_bp_a,\ p_b, K_bK_bK_bp_b,\ K_bK_bK_bp_a,\ K_bK_bK_ap_a,\ K_bK_aK_ap_a\}.$

Then, let's apply the perspective functions on the current state:

$O_a(s) = \{K_aK_ap_a, K_aK_ap_b, K_aK_bp_b, K_bK_bp_b, K_bK_ap_a, K_bK_bp_a, K_aK_bp_a, \}$

$O_b(s) = \{K_bK_bp_b, K_bK_bp_a, K_bK_ap_a, K_aK_ap_a\}.$

If we query common knowledge, we must evaluate the intersection between $O_a(s)$ and $O_b(s)$, denoting as s', which is:

$s' = \{K_bK_bp_b, K_bK_bp_a, K_bK_ap_a, K_aK_ap_a\}.$

But for common knowledge, we need to apply perspective functions until we reach termination. Applying another layer of perspective function on s':

$O_a(s') = \{K_ap_a\}$ and $O_b(s') = \{K_bp_b, K_bp_a, K_ap_a\}$

Their intersection s'' is $\{K_ap_a\}$.

Since $s' \neq s''$, we must apply another layer of perspective function, and then we will get their intersection becomes an empty set, which is their common knowledge. As the fixed point is an empty set, there is no common knowledge between a and b.

Overall, their nth perspective function intersection would be the sender's local perspective over $k-n$th messenger sending. Their perspectives are never the same between the time $k-n$th and $k-n-1$th, and it terminates as empty. Thus, they will never achieve common knowledge by sending messenger back and forth.

Appendix B

PDDL Examples

Here we give the example PDDL files representing classical planning problems mentioned in the main body of this thesis.

B.1 Example NIB problem using PDDL 1.0

The NIB problem can be found in Example 2.1. Although it is in the main body of this thesis, for easier reference, we copy the code example for the domain file as follows:

```
(define (domain NIB)

    (:requirements :strips :typing :negative-preconditions)

    (:types
        agent num value
    )

    (:predicates
        (peeking ?a - agent ?n - num)
        (standing ?a - agent)
        (free ?n - num)
        (value ?n - num ?v - value)
        (increasing ?v1 ?v2 - value)
        (decreasing ?v1 ?v2 - value)
    )

    (:action peek
        :parameters (?a - agent ?n - num)
        :precondition (and
            (standing ?a)
            (free ?n)
        )
        :effect (and
            (peeking ?a ?n)
            (not (free ?n))
```

```
                (not (standing ?a))
            )
        )

        (:action return
            :parameters (?a - agent ?n - num)
            :precondition (and
                (peeking ?a ?n)
            )
            :effect (and
                (free ?n)
                (standing ?a)
                (not (peeking ?a ?n))
            )
        )

        (:action increment
            :parameters (?n - num ?v1 ?v2 - value)
            :precondition (and
                (value ?n ?v1)
                (increasing ?v1 ?v2)
            )
            :effect (and
                (value ?n ?v2)
                (not (value ?n ?v1))
            )
        )

        (:action decrement
            :parameters (?n - num ?v1 ?v2 - value)
            :precondition (and
                (value ?n ?v1)
                (decreasing ?v1 ?v2)
            )
            :effect (and
                (value ?n ?v2)
                (not (value ?n ?v1))
            )
        )
)
```

Code Example B.1: PDDL1.0 Domain: NIB

```
(define (problem NIB_example)
    (:domain NIB)
    (:objects
        a b - agent
        p q - num
        v0 v1 v2 v3 v4 v5 v6 v7 v8 v9
        v10 v11 v12 v13 v14 v15 v16 v17 v18 v19
        v20 v21 v22 v23 v24 v25 v26 v27 v28 v29
        v30 v31 v32 v33 v34 v35 v36 v37 v38 v39
        v40 v41 v42 v43 v44 v45 v46 v47 v48 v49
        v50 v51 v52 v53 v54 v55 v56 v57 v58 v59
        v60 v61 v62 v63 v64 v65 v66 v67 v68 v69
        v70 v71 v72 v73 v74 v75 v76 v77 v78 v79
        v80 v81 v82 v83 v84 v85 v86 v87 v88 v89
        v90 v91 v92 v93 v94 v95 v96 v97 v98 v99 - value
    )

    (:init
        (standing a)
        (standing b)

        (free p)
        (free q)

        (value p v4)
        (value q v6)

        (increasing v0 v1)(decreasing v1 v0)
        (increasing v1 v2)(decreasing v2 v1)
        (increasing v2 v3)(decreasing v3 v2)
        (increasing v3 v4)(decreasing v4 v3)
        (increasing v4 v5)(decreasing v5 v4)
        (increasing v5 v6)(decreasing v6 v5)
        (increasing v6 v7)(decreasing v7 v6)
        (increasing v7 v8)(decreasing v8 v7)
        (increasing v8 v9)(decreasing v9 v8)
        (increasing v9 v10)(decreasing v10 v9)
        (increasing v10 v11)(decreasing v11 v10)
        (increasing v11 v12)(decreasing v12 v11)
        (increasing v12 v13)(decreasing v13 v12)
        (increasing v13 v14)(decreasing v14 v13)
        (increasing v14 v15)(decreasing v15 v14)
        (increasing v15 v16)(decreasing v16 v15)
        (increasing v16 v17)(decreasing v17 v16)
        (increasing v17 v18)(decreasing v18 v17)
        (increasing v18 v19)(decreasing v19 v18)
        (increasing v19 v20)(decreasing v20 v19)
        (increasing v20 v21)(decreasing v21 v20)
        (increasing v21 v22)(decreasing v22 v21)
        (increasing v22 v23)(decreasing v23 v22)
        (increasing v23 v24)(decreasing v24 v23)
        (increasing v24 v25)(decreasing v25 v24)
        (increasing v25 v26)(decreasing v26 v25)
        (increasing v26 v27)(decreasing v27 v26)
        (increasing v27 v28)(decreasing v28 v27)
        (increasing v28 v29)(decreasing v29 v28)
        (increasing v29 v30)(decreasing v30 v29)
        (increasing v30 v31)(decreasing v31 v30)
        (increasing v31 v32)(decreasing v32 v31)
        (increasing v32 v33)(decreasing v33 v32)
        (increasing v33 v34)(decreasing v34 v33)
        (increasing v34 v35)(decreasing v35 v34)
        (increasing v35 v36)(decreasing v36 v35)
        (increasing v36 v37)(decreasing v37 v36)
        (increasing v37 v38)(decreasing v38 v37)
        (increasing v38 v39)(decreasing v39 v38)
        (increasing v39 v40)(decreasing v40 v39)
        (increasing v40 v41)(decreasing v41 v40)
        (increasing v41 v42)(decreasing v42 v41)
        (increasing v42 v43)(decreasing v43 v42)
        (increasing v43 v44)(decreasing v44 v43)
        (increasing v44 v45)(decreasing v45 v44)
        (increasing v45 v46)(decreasing v46 v45)
        (increasing v46 v47)(decreasing v47 v46)
        (increasing v47 v48)(decreasing v48 v47)
        (increasing v48 v49)(decreasing v49 v48)
        (increasing v49 v50)(decreasing v50 v49)
        (increasing v50 v51)(decreasing v51 v50)
        (increasing v51 v52)(decreasing v52 v51)
        (increasing v52 v53)(decreasing v53 v52)
        (increasing v53 v54)(decreasing v54 v53)
        (increasing v54 v55)(decreasing v55 v54)
        (increasing v55 v56)(decreasing v56 v55)
        (increasing v56 v57)(decreasing v57 v56)
        (increasing v57 v58)(decreasing v58 v57)
        (increasing v58 v59)(decreasing v59 v58)
        (increasing v59 v60)(decreasing v60 v59)
        (increasing v60 v61)(decreasing v61 v60)
        (increasing v61 v62)(decreasing v62 v61)
        (increasing v62 v63)(decreasing v63 v62)
        (increasing v63 v64)(decreasing v64 v63)
        (increasing v64 v65)(decreasing v65 v64)
        (increasing v65 v66)(decreasing v66 v65)
        (increasing v66 v67)(decreasing v67 v66)
        (increasing v67 v68)(decreasing v68 v67)
        (increasing v68 v69)(decreasing v69 v68)
        (increasing v69 v70)(decreasing v70 v69)
        (increasing v70 v71)(decreasing v71 v70)
        (increasing v71 v72)(decreasing v72 v71)
        (increasing v72 v73)(decreasing v73 v72)
        (increasing v73 v74)(decreasing v74 v73)
        (increasing v74 v75)(decreasing v75 v74)
        (increasing v75 v76)(decreasing v76 v75)
        (increasing v76 v77)(decreasing v77 v76)
        (increasing v77 v78)(decreasing v78 v77)
        (increasing v78 v79)(decreasing v79 v78)
        (increasing v79 v80)(decreasing v80 v79)
        (increasing v80 v81)(decreasing v81 v80)
        (increasing v81 v82)(decreasing v82 v81)
        (increasing v82 v83)(decreasing v83 v82)
        (increasing v83 v84)(decreasing v84 v83)
        (increasing v84 v85)(decreasing v85 v84)
        (increasing v85 v86)(decreasing v86 v85)
        (increasing v86 v87)(decreasing v87 v86)
        (increasing v87 v88)(decreasing v88 v87)
        (increasing v88 v89)(decreasing v89 v88)
        (increasing v89 v90)(decreasing v90 v89)
        (increasing v90 v91)(decreasing v91 v90)
        (increasing v91 v92)(decreasing v92 v91)
        (increasing v92 v93)(decreasing v93 v92)
        (increasing v93 v94)(decreasing v94 v93)
        (increasing v94 v95)(decreasing v95 v94)
        (increasing v95 v96)(decreasing v96 v95)
        (increasing v96 v97)(decreasing v97 v96)
        (increasing v97 v98)(decreasing v98 v97)
        (increasing v98 v99)(decreasing v99 v98)


    )

    (:goal (and
            (not (free p))
            (not (free q))
        )
    )
)
```

Code Example B.2: PDDL1.0 Problem: NIB

Appendix C

F-PDDL Examples

Here we give the example PDDL files as well as the external function codes.

C.1 Example BBL problem using F-PDDL

The BBL problem can be found in Section 4.5.1.5. The domain file and example problem file (using $B_b B_a v = True \wedge B_a B_b v = True$ as the goal) are given as follows:

```
(define
    (domain bbl)

    (:types ;todo: enumerate types and their hierarchy here, e.g.
   car truck bus - vehicle
        locatable
        turnable askable - locatable
    )



    (:functions
        (dir ?a - turnable)
        (x ?a - locatable)
        (y ?a - locatable)
        (v ?a - askable)
    )

    ;define actions here
    (:action turn_clockwise
        :parameters (?i - turnable)
        :precondition (
        )
        :effect (
            ; increase sth by 1
            (increase (dir ?i) 1)
        )
    )

    (:action turn_anti_clockwise
        :parameters (?i - turnable)
        :precondition (
        )
        :effect (
            ; increase sth by 1
            (decrease (dir ?i) 1)
        )
    )

)
```

Code Example C.1: F-PDDL Domain: BBL

```
(define
    (problem bbl04)
    (:domain bbl)

    (:agents
        a b - turnable
    )

    (:objects
        p - askable
    )

    (:init
        (assign (dir a) 'sw')
        (assign (dir b) 'n')
        (assign (x a) 3)
        (assign (x b) 2)
        (assign (x p) 1)
        (assign (y a) 3)
        (assign (y b) 2)
        (assign (y p) 1)
        (assign (v p) 't')
    )

    ; the @ represent this is an epistemic evaluation
    ;
    (:goal
        (and
            (= (@ep ("+ b [b] + b [a]") (= (v p) 't')) ep.true)
            (= (@ep ("+ b [a] + b [b]") (= (v p) 't')) ep.true)
        )
    )

    ; D, domain of variables, in order to differentiate from the
   domain, we use range as key word
    (:ranges
        (dir enumerate ['w','nw','n','ne','e','se','s','sw'])
        (x integer [0,4])
        (y integer [0,4])
        (v enumerate ['t','f'])
    )

    (:rules

    )
)
```

Code Example C.2: F-PDDL Problem: BBL

The observation function for BBL (Equation 4.1) is implemented in Python as follows:

```python
def checkVisibility(self,state,agent_index,var_name,entities:
    typing.Dict[str,Entity],
                    functions:typing.Dict[str,Function],
                    function_schemas:typing.Dict[str,
    FunctionSchema]):
     if not agent_index in entities.keys():
         raise ValueError(f"agent_index [{agent_index}] not found
    in entities")
     if not entities[agent_index].enetity_type == EntityType.AGENT:
         raise ValueError(f"agent_index [{agent_index}] is not an
    agent")
     if var_name not in functions.keys():
         raise ValueError(f"var_name [{var_name}] not found in
    functions")

     function = functions[var_name]
     function_schemas_name = function.function_schema_name
     target_list = function.entity_index_list

     # for the bbl domain, all visibility function should be the
    same
     # based on whether the agents physically see the objects/
    agents or not
     # and all functions in bbl domain have only one entity
     if len(target_list) != 1:
         raise ValueError("all function in bbl should have only one
     entity",var_name)

     target_index = target_list[0]
     try:
         #extract necessary variables from state
         # logger.debug(f"loading variables from state")
         target_x = state[f"x {target_index}"]
         target_y = state[f"y {target_index}"]
         agent_x = state[f"x {agent_index}"]
         agent_y = state[f"y {agent_index}"]
         agent_dir = dir_dict[state[f"dir {agent_index}"]]

         # extract necessary common constants from given domain
         # logger.debug(f"necessary common constants from given
    domain")
         agent_angle = common_constants[f"angle {agent_index}"]

         # agent is able to see anything in the same location
         if target_x == agent_x and target_y == agent_y:
             return True

         # generate two vector
         v1 = np.array((target_y - agent_y,target_x - agent_x))
         v1 = v1 / np.linalg.norm(v1)
         radians = math.radians(agent_dir)
         v2 = np.array((math.cos(radians),math.sin(radians)))
         # logger.debug(f'v1 {v1}, v2 {v2}')
         cos_ = v1.dot(v2)
```

```
        d_radians = math.acos(cos_)
        d_degrees = math.degrees(d_radians)
        # logger.debug(f'delta angle degree is {round(d_degrees,3)}')

        if d_degrees <= agent_angle/2.0 and d_degrees >= -agent_angle/2.0:
            inside = True
        else:
            inside = False
        # logger.debug(f'visibility is {inside}')
        return inside
    except KeyError as e:
        self.logger.debug(e)
        self.logger.debug("state: %s",state)
        return False
    except TypeError as e:
        self.logger.debug(e)
        self.logger.debug("state: %s",state)
        return False
```

Code Example C.3: F-PDDL observation functions: BBL

C.2 Example Corridor problem using F-PDDL

The Corridor problem can be found in Section 4.5.1.2. The domain file and example problem file are given as follows:

```
(define
    (domain corridor)

    (:types
        secret agent

    )

    (:functions
        (agent_loc ?a - agent)
        (movable ?a - agent)
        (secret_loc ?s - secret)
        (sensed ?s - secret)

        (secret_lying_value ?s - secret)
        (secret_truth_value ?s - secret)
        (shared_value ?s - secret)
        (shared_loc ?s - secret)
    )

    ;define actions here
    (:action move_right
        :parameters (?a - agent)
        :precondition (
            (= (movable ?a) 1)
            ; (= (sharing) 0)
        )
        :effect (
            (increase (agent_loc ?a) 1)
        )
    )

    (:action move_left
        :parameters (?a - agent)
        :precondition (
             (= (movable ?a) 1)
        )
        :effect (
            (decrease (agent_loc ?a) 1)
        )
    )

    (:action sense
        :parameters (?a - agent, ?s - secret)
        :precondition (
            ; (= (sharing) 1)
            (= (movable ?a) 1)
            (= (agent_loc ?a) (secret_loc ?s))

        )
        :effect (
            (assign (sensed ?s) 1)
        )
    )


    (:action shout_truth
        :parameters (?a - agent, ?s - secret)
        :precondition (
            (= (movable ?a) 1)
            (= (sensed ?s) 1)
        )
        :effect (
            (assign (shared_loc ?s) (agent_loc ?a))
            (assign (shared_value ?s) (secret_truth_value ?s))
        )
    )

    (:action shout_lie
        :parameters (?a - agent, ?s - secret)
        :precondition (
            (= (movable ?a) 1)
            (= (sensed ?s) 1)
        )
        :effect (
            (assign (shared_loc ?s) (agent_loc ?a))
            (assign (shared_value ?s) (secret_lying_value ?s))
        )
    )
)
```

Code Example C.4: F-PDDL Domain: Corridor

```
(define
    (problem corridor09)
    (:domain corridor)

    (:agents
        a b c d e f g - agent
    )

    (:objects
        s - secret
    )

    (:init
    ; valid locations are 1,2,3,4
    ; secret is at location -1 if not shared (avoid adjcent as well)
        (assign (agent_loc a) 1)
        (assign (agent_loc b) 2)
        (assign (agent_loc c) 3)
        (assign (agent_loc d) 1)
        (assign (agent_loc e) 3)
        (assign (agent_loc f) 2)
        (assign (agent_loc g) 4)

        (assign (movable a) 1)
        (assign (movable b) 0)
        (assign (movable c) 0)
        (assign (movable d) 0)
        (assign (movable e) 0)
        (assign (movable f) 0)
        (assign (movable g) 0)

        (assign (secret_loc s) 2)
        (assign (sensed s) 0)

        (assign (secret_truth_value s) 't')
        (assign (secret_lying_value s) 'f')

        (assign (shared_value s) 'f')
        (assign (shared_loc s) -1)
    )


    (:goal (and
            (= (@ep ("+ b [b] + b [b] + b [b] + b [b] + b [b]") (= (shared_value s) 'f')) ep.true)
            (= (@ep ("+ b [c] + b [c] + b [c] + b [c] + b [c]") (= (shared_value s) 't')) ep.true)
        )
    )

    (:ranges
        (agent_loc integer [1,4])
        (secret_loc integer [1,4])
        (movable integer [0,1])
        (sensed integer [0,1])
        (shared_loc integer [-1,4])
        (secret_truth_value enumerate ['t','f'])
        (secret_lying_value enumerate ['t','f'])
        (shared_value enumerate ['t','f'])
    )

    (:rules

    )
)
```

Code Example C.5: F-PDDL Problem: Corridor

We also provide two versions of the observation function for the corridor domain here. One (Code Example C.6) is consistent with the PDKB approach; the other one (Code Example C.7) is the one we believe is more meaningful.

```
def checkVisibility(self,state,agent_index,var_name,entities:
    typing.Dict[str,Entity],
                        functions:typing.Dict[str,Function],
                        function_schemas:typing.Dict[str,
    FunctionSchema]):
     if not agent_index in entities.keys():
         raise ValueError(f"agent_index [{agent_index}] not found
    in entities")
     if not entities[agent_index].enetity_type == EntityType.AGENT:
         raise ValueError(f"agent_index [{agent_index}] is not an
    agent")
     if var_name not in functions.keys():
         raise ValueError(f"var_name [{var_name}] not found in
    functions")

     function = functions[var_name]
     function_schemas_name = function.function_schema_name
     target_list = function.entity_index_list


     if 'agent_loc'  == function_schemas_name:
         # it means this is agent's location
         # based on the assumption from PDKB
         # agent can see all the location all the time
         return True
     elif 'movable'  == function_schemas_name:
         return True
     elif 'secret_loc'  == function_schemas_name:
         return True
     elif 'sensed' == function_schemas_name:
         return True
     elif 'secret_truth_value'  == function_schemas_name \
         or 'secret_lying_value' == function_schemas_name:
         # it depends on whether agent has sensed the value if it
    is agent a (movable)
         movable = state[f'movable {agent_index}']
         if movable == 1:
             sensed = state[f'sensed {target_list[0]}']
             return sensed == 1
         else:
             return False
     elif 'shared_loc'  == function_schemas_name:
         # it means this is a location variable
         # agent will know the secret is been shared if they are in
     the same room where the secret shared
         # if state[var_name] == 0:
         #     return True
         # return abs(int(state[var_name])-int(state[f'agent_loc {
    agent_index}']))<=1

         ### this is true becasue the assumption from PDKB
         return True

     elif 'shared_value'  == function_schemas_name:
         # it depends on whether agent sees that secret been shared
         shared_loc = state[f'shared_loc {target_list[0]}']
         agent_loc = state[f'agent_loc {agent_index}']
         return abs(int(shared_loc)-int(agent_loc))<=1
     else:
         raise ValueError(f"function_schemas_name [{function_schemas_name}] not found")
```

Code Example C.6: F-PDDL observation functions: Corridor (PDKB version)

```
 def checkVisibility(self,state,agent_index,var_name,entities:typing.Dict[str,Entity],
                     functions:typing.Dict[str,Function],
                     function_schemas:typing.Dict[str,FunctionSchema]):
     if not agent_index in entities.keys():
         raise ValueError(f"agent_index [{agent_index}] not found in entities")
     if not entities[agent_index].enetity_type == EntityType.AGENT:
         raise ValueError(f"agent_index [{agent_index}] is not an agent")
     if var_name not in functions.keys():
         raise ValueError(f"var_name [{var_name}] not found in functions")

     function = functions[var_name]
     function_schemas_name = function.function_schema_name
     target_list = function.entity_index_list


     if 'agent_loc'  == function_schemas_name:
         # it means this is agent's location
         # based on the assumption from PDKB
         # agent can see all the location all the time

         # this changes the assumption of all location are known.
         if state[f'agent_loc {agent_index}'] == state[f'agent_loc {target_list[0]}']:
             return True
         else:
             return False
     elif 'movable'  == function_schemas_name:
         return True
     elif 'secret_loc'  == function_schemas_name:
         return True
     elif 'sensed' == function_schemas_name:
         return True
     elif 'secret_truth_value'  == function_schemas_name \
         or 'secret_lying_value' == function_schemas_name:
         # it depends on whether agent has sensed the value if it is agent a (movable)
         movable = state[f'movable {agent_index}']
         if movable == 1:
             sensed = state[f'sensed {target_list[0]}']
             return sensed == 1
         else:
             return False
     elif 'shared_loc' == function_schemas_name:
         # it means this is a location variable
         # agent will know the secret is been shared if they are in the same room where the secret shared
         # if state[var_name] == 0:
         #     return True
         # return abs(int(state[var_name])-int(state[f'agent_loc {agent_index}']))<=1

         ### this is true becasue the assumption from PDKB
         # return True

         # change the agent know secret been sharing all the time.
         if state[var_name] == -1:
             return True
         shared_loc = state[f'shared_loc {target_list[0]}'] if type(state[f'shared_loc {target_list[0]}']) == int else -1
         agent_loc = state[f'agent_loc {agent_index}'] if type(state[f'agent_loc {agent_index}']) == int else -3
         return abs(int(shared_loc)-int(agent_loc))<=1

     elif 'shared_value' == function_schemas_name:
         # it depends on whether agent sees that secret been shared
         shared_loc = state[f'shared_loc {target_list[0]}'] if type(state[f'shared_loc {target_list[0]}']) == int else -1
         agent_loc = state[f'agent_loc {agent_index}'] if type(state[f'agent_loc {agent_index}']) == int else -3
         return abs(int(shared_loc)-int(agent_loc))<=1
     else:
         raise ValueError(f"function_schemas_name [{function_schemas_name}] not found")
```

Code Example C.7: F-PDDL observation functions: Corridor (JP version)

Bibliography

[1] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier. 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)*, pages 193–201, 2016. doi: 10.3233/978-1-61499-672-9-193. URL https://doi.org/10.3233/978-1-61499-672-9-193.

[2] Olivier Gasquet, Valentin Goranko, and François Schwarzentruber. 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*, pages 325–332, 2014. URL http://dl.acm.org/citation.cfm?id=2615786.

[3] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 2003. ISBN 0262562006.

[4] Guang Hu. What you get is what you see: Decomposing epistemic planning using functional strips. Masters research thesis, University of Melbourne, 2020. URL http://hdl.handle.net/11343/235775.

[5] S. C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938. ISSN 00224812. URL http://www.jstor.org/stable/2267778.

[6] Tiep Le, Francesco Fabiano, Tran Cao Son, and Enrico Pontelli. EFP and PG-EFP: epistemic forward search planners in multi-agent domains. In *Proceedings of the 28th ICAPS*, pages 161–170, 2018. URL https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17733.

[7] Xiao Huang, Biqing Fang, Hai Wan, and Yongmei Liu. 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*, pages 1093–1101, 2017. doi: 10.24963/ijcai.2017/152. URL https://doi.org/10.24963/ijcai.2017/152.

[8] Uta Frith and Christopher D. Frith. Development and neurophysiology of mentalizing. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):459–473, March 2003. ISSN 0962-8436. doi: 10.1098/rstb.2002.1218. URL https://royalsocietypublishing.org/doi/10.1098/rstb.2002.1218.

[9] Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950. doi: 10.1093/MIND/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

[10] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. ISBN 9780134610993. URL http://aima.cs.berkeley.edu/.

[11] Ray Kurzweil. *The Singularity is Near*, pages 393–406. Palgrave Macmillan UK, London, 2014. ISBN 978-1-137-34908-8. doi: 10.1057/9781137349088_26. URL https://doi.org/10.1057/9781137349088_26.

[12] Cynthia Breazeal. Toward sociable robots. *Robotics and Autonomous Systems*, 42(3):167–175, 2003. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(02)00373-1. URL https://www.sciencedirect.com/science/article/pii/S0921889002003731. Socially Interactive Robots.

[13] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

[14] Simon Baron-Cohen, Alan M Leslie, and Uta Frith. Does the autistic child have a "theory of mind"? *Cognition*, 21(1):37–46, 1985.

[15] Neil C. Rabinowitz, Frank Perbet, H. Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew M. Botvinick. Machine theory of mind. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference*

on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of *Proceedings of Machine Learning Research*, pages 4215–4224. PMLR, 2018. URL http://proceedings.mlr.press/v80/rabinowitz18a.html.

[16] Frank S. He, Yang Liu, Alexander G. Schwing, and Jian Peng. Learning to play in a day: Faster deep reinforcement learning by optimality tightening. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rJ8Je4clg.

[17] Yufan Wu, Yinghui He, Yilin Jia, Rada Mihalcea, Yulong Chen, and Naihao Deng. Hi-tom: A benchmark for evaluating higher-order theory of mind reasoning in large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10691–10706. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.717. URL https://doi.org/10.18653/v1/2023.findings-emnlp.717.

[18] Jayanta Sadhu, Ayan Antik Khan, Noshin Nawal, Sanju Basak, Abhik Bhattacharjee, and Rifat Shahriyar. Multi-tom: Evaluating multilingual theory of mind capabilities in large language models. *CoRR*, abs/2411.15999, 2024. doi: 10.48550/ARXIV.2411.15999. URL https://doi.org/10.48550/arXiv.2411.15999.

[19] Michal Kosinski. Evaluating large language models in theory of mind tasks. *Proceedings of the National Academy of Sciences*, 121(45), October 2024. ISSN 1091-6490. doi: 10.1073/pnas.2405460121. URL http://dx.doi.org/10.1073/pnas.2405460121.

[20] Melanie Mitchell. Why AI is harder than we think. *CoRR*, abs/2104.12871, 2021. URL https://arxiv.org/abs/2104.12871.

[21] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009. ISSN 0010-0277. doi: https://doi.org/10.1016/j.cognition.2009.07.005. URL https://www.sciencedirect.com/science/article/pii/S0010027709001607. Reinforcement learning and higher cognition.

[22] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In Laura A. Carlson, Christoph Hölscher, and Thomas F. Shipley, editors, *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011*. cognitivesciencesociety.org, 2011. URL https://mindmodeling.org/cogsci2011/papers/0583/index.html.

[23] Julian Jara-Ettinger, Laura Schulz, and Joshua B. Tenenbaum. The naïve utility calculus: Joint inferences about the costs and rewards of actions. In David C. Noelle, Rick Dale, Anne S. Warlaumont, Jeff Yoshimi, Teenie Matlock, Carolyn D. Jennings, and Paul P. Maglio, editors, *Proceedings of the 37th Annual Meeting of the Cognitive Science Society, CogSci 2015, Pasadena, California, USA, July 22-25, 2015*. cognitivesciencesociety.org, 2015. URL https://mindmodeling.org/cogsci2015/papers/0174/index.html.

[24] Yang Wu, Chris L. Baker, Joshua B. Tenenbaum, and Laura Schulz. Joint inferences of belief and desire from facial expressions. In Paul Bello, Marcello Guarini, Marjorie McShane, and Brian Scassellati, editors, *Proceedings of the 36th Annual Meeting of the Cognitive Science Society, CogSci 2014, Quebec City, Canada, July 23-26, 2014*. cognitivesciencesociety.org, 2014. URL https://escholarship.org/uc/item/4tm729mw.

[25] Chris Baker, Julian Jara-Ettinger, Rebecca Saxe, and Joshua B. Tenenbaum. Rational quantitative attribution of beliefs, desires, and percepts in human mentalizing. *Nature Human Behavior*, 1, 03/2017 2017. doi: doi:10.1038/s41562-017-0064. URL http://www.nature.com/articles/s41562-017-0064.

[26] Yang Wu, Chris L. Baker, Joshua B. Tenenbaum, and Laura Schulz. Rational inference of beliefs and desires from emotional expressions. *Cogn. Sci.*, 42(3):850–884, 2018. doi: 10.1111/COGS.12548. URL https://doi.org/10.1111/cogs.12548.

[27] Lance Ying, Tan Zhi-Xuan, Lionel Wong, Vikash Mansinghka, and Joshua B. Tenenbaum. Grounding language about belief in a bayesian theory-of-mind. *CoRR*, abs/2402.10416, 2024. doi: 10.48550/ARXIV.2402.10416. URL https://doi.org/10.48550/arXiv.2402.10416.

[28] Lance Ying, Tan Zhi-Xuan, Lionel Wong, Vikash Mansinghka, and Joshua B. Tenenbaum. Understanding epistemic language with a bayesian theory of mind. *CoRR*, abs/2408.12022, 2024. doi: 10.48550/ARXIV.2408.12022. URL https://doi.org/10.48550/arXiv.2408.12022.

[29] Tan Zhi-Xuan, Jordyn L. Mann, Tom Silver, Josh Tenenbaum, and Vikash Mansinghka. Online bayesian goal inference for boundedly rational planning agents. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/df3aebc649f9e3b674eeb790a4da224e-Abstract.html.

[30] Robert C. Moore. Reasoning about knowledge and action. In Raj Reddy, editor, *Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, MA, USA, August 22-25, 1977*, pages 223–227. William Kaufmann, 1977. URL http://ijcai.org/Proceedings/77-1/Papers/033.pdf.

[31] Kurt Konolige. *A first-order formalization of knowledge and action for a multiagent planning system.* SRI International, Artificial Intelligence Center, 1980.

[32] Douglas E. Appelt. A planner for reasoning about knowledge and action. In Robert Balzer, editor, *Proceedings of the 1st Annual National Conference on Artificial Intelligence, Stanford University, CA, USA, August 18-21, 1980*, pages 131–133. AAAI Press/MIT Press, 1980. URL http://www.aaai.org/Library/AAAI/1980/aaai80-038.php.

[33] Leora Morgenstern. Knowledge preconditions for actions and plans. In John P. McDermott, editor, *Proceedings of the 10th International Joint Conference on Artificial Intelligence. Milan, Italy, August 23-28, 1987*, pages 867–874. Morgan Kaufmann, 1987. URL http://ijcai.org/Proceedings/87-2/Papers/064.pdf.

[34] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single and multi-agent systems. *JANCL*, 21(1):9–34, 2011. doi: 10.3166/jancl.21.9-34. URL https://doi.org/10.3166/jancl.21.9-34.

[35] Alvin I. Goldman. What is justified belief? In George Pappas, editor, *Justification and Knowledge: New Studies in Epistemology*, pages 1–25. D. Reidel, 1979.

[36] Plato. *Meno*. Hackett Publishing, 1980. Originally published ca. 380 BCE.

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

[38] Allen Newell and Herbert Alexander Simon. Gps, a program that simulates human thought. *conference on Learning Automata*, 1961.

[39] Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971. doi: 10.1016/0004-3702(71)90010-5. URL https://doi.org/10.1016/0004-3702(71)90010-5.

[40] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. Elsevier, 2004. ISBN 978-1-55860-856-6.

[41] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. *An Introduction to the Planning Domain Definition Language*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2019. ISBN 978-3-031-00456-8. doi: 10.2200/S00900ED2V01Y201902AIM042. URL https://doi.org/10.2200/S00900ED2V01Y201902AIM042.

[42] Stuart J. Russell and Peter Norvig. *Artificial intelligence - a modern approach: the intelligent agent book*. Prentice Hall series in artificial intelligence. Prentice Hall, 1995. ISBN 978-0-13-103805-9. URL https://www.worldcat.org/oclc/31288015.

[43] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artif. Intell.*, 69(1-2):165–204, 1994. doi: 10.1016/0004-3702(94)90081-7. URL https://doi.org/10.1016/0004-3702(94)90081-7.

[44] Drew V. McDermott. A temporal logic for reasoning about processes and plans. *Cogn. Sci.*, 6(2):101–155, 1982. doi: 10.1207/S15516709COG0602_1. URL https://doi.org/10.1207/s15516709cog0602_1.

[45] Craig Boutilier, Thomas L. Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *J. Artif. Intell. Res.*, 11:1–94, 1999. doi: 10.1613/JAIR.575. URL https://doi.org/10.1613/jair.575.

[46] Thomas L. Dean and Michael P. Wellman. *Planning and control.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991. ISBN 1558602097.

[47] Ronen I. Brafman and Giuseppe De Giacomo. Planning for ltlf /ldlf goals in non-markovian fully observable nondeterministic domains. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1602–1608. ijcai.org, 2019. doi: 10.24963/IJCAI.2019/222. URL https://doi.org/10.24963/ijcai.2019/222.

[48] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. Pddl-the planning domain definition language. *Technical Report CVC TR-98-003/DCS TR-1165*, 1998. URL https://api.semanticscholar.org/CorpusID:59656859.

[49] Derek Long, Henry A. Kautz, Bart Selman, Blai Bonet, Hector Geffner, Jana Koehler, Michael Brenner, Jörg Hoffmann, Frank Rittinger, Corin R. Anderson, Daniel S. Weld, David E. Smith, and Maria Fox. The AIPS-98 planning competition. *AI Mag.*, 21(2):13–33, 2000. doi: 10.1609/AIMAG.V21I2.1505. URL https://doi.org/10.1609/aimag.v21i2.1505.

[50] Fahiem Bacchus. The AIPS '00 planning competition. *AI Mag.*, 22(3):47–56, 2001. URL https://ojs.aaai.org/index.php/aimagazine/article/view/1571.

[51] Maria Fox and Derek Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.*, 20:61–124, 2003. doi: 10.1613/jair.1129. URL https://doi.org/10.1613/jair.1129.

[52] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.*, 27:235–297, 2006. doi: 10.1613/JAIR.2044. URL https://doi.org/10.1613/jair.2044.

[53] Stefan Edelkamp and Jörg Hoffmann. Pddl2. 2: The language for the classical part of the 4th international planning competition. Technical report, Technical Report 195, University of Freiburg, 2004.

[54] Alfonso E. Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5):619–668, 2009. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2008.10.012. URL https://www.sciencedirect.com/science/article/pii/S0004370208001847. Advances in Automated Plan Generation.

[55] Mihn Do Malte Helmert and Ioannis Refanidis. Changes in pddl 3.1, 2008. URL https://ipc08.icaps-conference.org/deterministic/PddlExtension.html. Accessed: 2025-01-02.

[56] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013. doi: 10.1613/JAIR.3912. URL https://doi.org/10.1613/jair.3912.

[57] Héctor Geffner. *Functional Strips: A More Flexible Language for Planning and Problem Solving*, pages 187–209. Springer US, Boston, MA, 2000. ISBN 978-1-4615-1567-8. doi: 10.1007/978-1-4615-1567-8_9. URL https://doi.org/10.1007/978-1-4615-1567-8_9.

[58] Peter Jonsson and Christer Bäckström. Tractable planning with state variables by exploiting structural restrictions. In Barbara Hayes-Roth and Richard E. Korf, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 2*, pages 998–1003. AAAI Press / The MIT Press, 1994. URL http://www.aaai.org/Library/AAAI/1994/aaai94-153.php.

[59] Ken Currie and Austin Tate. O-plan: The open planning architecture. *Artif. Intell.*, 52(1):49–86, 1991. doi: 10.1016/0004-3702(91)90024-E. URL https://doi.org/10.1016/0004-3702(91)90024-E.

[60] Jana Koehler. Planning under resource constraints. In Henri Prade, editor, *13th European Conference on Artificial Intelligence, Brighton, UK, August 23-28 1998, Proceedings.*, pages 489–493. John Wiley and Sons, 1998.

[61] Philippe Laborie and Malik Ghallab. Planning with sharable resource constraints. In *Proceedings of the Fourteenth International Joint Conference on Artificial*

Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes, pages 1643–1651. Morgan Kaufmann, 1995. URL http://ijcai.org/Proceedings/95-2/Papers/081.pdf.

[62] Edwin P. D. Pednault. ADL: exploring the middle ground between STRIPS and the situation calculus. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89). Toronto, Canada, May 15-18 1989*, pages 324–332. Morgan Kaufmann, 1989.

[63] Guillem Francès, Miquel Ramírez, Nir Lipovetzky, and Hector Geffner. 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*, pages 4294–4301, 2017. doi: 10.24963/ijcai.2017/600. URL https://doi.org/10.24963/ijcai.2017/600.

[64] Nils J. Nilsson. *Problem-solving methods in artificial intelligence*. McGraw-Hill computer science series. McGraw-Hill, 1971. ISBN 0070465738. URL https://www.worldcat.org/oclc/185451766.

[65] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. doi: 10.1007/BF01386390. URL https://doi.org/10.1007/BF01386390.

[66] Edward A Feigenbaum, Avron Barr, and Paul R Cohen. Chapter ii - search. In Avron Barr and Edward A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, pages 19–139. Butterworth-Heinemann, 1981. ISBN 978-0-86576-089-9. doi: https://doi.org/10.1016/B978-0-86576-089-9.50007-7. URL https://www.sciencedirect.com/science/article/pii/B9780865760899500077.

[67] Blai Bonet and Hector Geffner. Planning as heuristic search: New results. In Susanne Biundo and Maria Fox, editors, *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, pages 360–372. Springer, 1999. doi: 10.1007/10720246_28. URL https://doi.org/10.1007/10720246_28.

[68] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4 (2):100–107, 1968. doi: 10.1109/TSSC.1968.300136. URL https://doi.org/10.1109/TSSC.1968.300136.

[69] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artif. Intell.*, 1(3): 193–204, 1970. doi: 10.1016/0004-3702(70)90007-X. URL https://doi.org/10.1016/0004-3702(70)90007-X.

[70] Malte Helmert and Carmel Domshlak. 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*, 2009. URL http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/735.

[71] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.*, 39:127–177, 2010. doi: 10.1613/jair.2972. URL https://doi.org/10.1613/jair.2972.

[72] Blai Bonet and Hector Geffner. Planning as heuristic search. *Artif. Intell.*, 129 (1-2):5–33, 2001. doi: 10.1016/S0004-3702(01)00108-4. URL https://doi.org/10.1016/S0004-3702(01)00108-4.

[73] Christian Dornhege, Patrick Eyerich, Thomas Keller, Sebastian Trüg, Michael Brenner, and Bernhard Nebel. Semantic attachments for domain-independent planning systems. In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI, 2009. URL http://aaai.org/ocs/index.php/ICAPS/ICAPS09/paper/view/754.

[74] Christian Dornhege, Marc Gissler, Matthias Teschner, and Bernhard Nebel. Integrating symbolic and geometric planning for mobile manipulation. In *2009 IEEE International Workshop on Safety, Security & Rescue Robotics (SSRR 2009)*, pages 1–6, 2009. doi: 10.1109/SSRR.2009.5424160.

[75] Andre Gaschler, Ronald P. A. Petrick, Oussama Khatib, and Alois C. Knoll. Kaboum: Knowledge-level action and bounding geometry motion planner. *J.*

Artif. Intell. Res., 61:323–362, 2018. doi: 10.1613/jair.5560. URL https://doi.org/10.1613/jair.5560.

[76] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Unifying perception, estimation and action for mobile manipulation via belief space planning. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pages 2952–2959. IEEE, 2012. doi: 10.1109/ICRA.2012.6225237. URL https://doi.org/10.1109/ICRA.2012.6225237.

[77] Josef Bajada, Maria Fox, and Derek Long. Temporal planning with semantic attachment of non-linear monotonic continuous behaviours. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1523–1529. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/218.

[78] Peter Gregory, Derek Long, Maria Fox, and J. Christopher Beck. Planning modulo theories: Extending the planning paradigm. In Lee McCluskey, Brian Charles Williams, José Reinaldo Silva, and Blai Bonet, editors, *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI, 2012. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4693.

[79] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(T). *J. ACM*, 53(6):937–977, 2006. doi: 10.1145/1217856.1217859. URL https://doi.org/10.1145/1217856.1217859.

[80] Nir Lipovetzky and Hector Geffner. 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*, pages 540–545, 2012. doi: 10.3233/978-1-61499-098-7-540. URL https://doi.org/10.3233/978-1-61499-098-7-540.

[81] Paul Gochet and E. Pascal Gribomont. Epistemic logic. In Dov M. Gabbay and John Woods, editors, *Logic and the Modalities in the Twentieth Century*,

volume 7 of *Handbook of the History of Logic*, pages 99–195. Elsevier, 2006. doi: 10.1016/S1874-5857(06)80028-2. URL https://doi.org/10.1016/S1874-5857(06)80028-2.

[82] Plato. *Theaetetus*. Hackett Publishing, 1892. Oxford University Press, American branch.

[83] AJ Ayer. The problem of knowledge, 1956.

[84] Roderick Chisholm. Perceiving: A philosophical study, 1957.

[85] Edmund L. Gettier. Is justified true belief knowledge? *Analysis*, 23(6):121–123, 1963. doi: 10.1093/analys/23.6.121.

[86] Giacomo Bonanno. A simple modal logic for belief revision. *Synthese*, 147(2): 193–228, 2005. ISSN 00397857, 15730964. URL http://www.jstor.org/stable/20118657.

[87] Christian J. Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 3327–3334. AAAI Press, 2015. doi: 10.1609/AAAI.V29I1.9665. URL https://doi.org/10.1609/aaai.v29i1.9665.

[88] Christian Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. Efficient multi-agent epistemic planning: Teaching planners about nested belief. *Artif. Intell.*, 302:103605, 2022. doi: 10.1016/j.artint.2021.103605. URL https://doi.org/10.1016/j.artint.2021.103605.

[89] Kaarlo Jaakko Juhani Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca, NY, USA, 1962.

[90] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, 1992. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(92)90049-4. URL https://www.sciencedirect.com/science/article/pii/0004370292900494.

[91] Wiebe van der Hoek. Systems for knowledge and beliefs. In Jan van Eijck, editor, *Logics in AI, European Workshop, JELIA '90, Amsterdam, The Netherlands, September 10-14, 1990, Proceedings*, volume 478 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 1990. doi: 10.1007/BFB0018447. URL https://doi.org/10.1007/BFb0018447.

[92] Sarit Kraus and Daniel Lehmann. Knowledge, belief and time. *Theor. Comput. Sci.*, 58:155–174, 1988. doi: 10.1016/0304-3975(88)90024-2. URL https://doi.org/10.1016/0304-3975(88)90024-2.

[93] F. Voorbraak. *As Far as I Know: Epistemic Logic and Uncertainty*. Quaestiones infinitae. Department of Philosophy, Utrecht University, 1993. ISBN 9789039302590. URL https://books.google.com.au/books?id=2kno0AAACAAJ.

[94] Fred I. Dretske. *Knowledge and the Flow of Information*. MIT Press, Stanford, CA, 1981.

[95] Ronald Fagin and Joseph Y. Halpern. Belief, awareness, and limited reasoning: Preliminary report. In Aravind K. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985*, pages 491–501. Morgan Kaufmann, 1985. URL http://ijcai.org/Proceedings/85-1/Papers/095.pdf.

[96] Jacques Dubucs. On logical omniscience. *Logique et Analyse*, 34(133/134):41–55, 1991. ISSN 00245836, 22955836. URL http://www.jstor.org/stable/44085040.

[97] Wolfgang Lenzen. Recent work in epistemic logic. *Acta Philosophica Fennica*, 30: 1–219, 1978.

[98] Timothy Williamson. *Knowledge and its Limits*. Oxford University Press, 10 2002. ISBN 9780199256563. doi: 10.1093/019925656X.001.0001. URL https://doi.org/10.1093/019925656X.001.0001.

[99] Joseph Y. Halpern. Should knowledge entail belief? *J. Philos. Log.*, 25(5):483–494, 1996. doi: 10.1007/BF00257382. URL https://doi.org/10.1007/BF00257382.

[100] Jon Barwise. Scenes and other situations. *The Journal of Philosophy*, 78(7):369–397, 1981. ISSN 0022362X, 19398549. URL http://www.jstor.org/stable/2026481.

[101] Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '98, page 43–56, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605630.

[102] Jan Plaza. Logics of public announcements. In *Proceedings 4th international symposium on methodologies for intelligent systems*, pages 201–216, 1989.

[103] Jelle Gerbrandy. Dynamic epistemic logic. *Logic, Philosophy and Linguistics (LP)*, 1997.

[104] Hans Van Ditmarsch and Barteld Kooi. Semantic results for ontic and epistemic change. *Logic and the foundations of game and decision theory (LOFT 7)*, 3: 87–117, 2008.

[105] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Springer Publishing Company, Incorporated, 1st edition, 2007. ISBN 1402058381.

[106] Thomas Bolander. A gentle introduction to epistemic planning: The DEL approach. In Sujata Ghosh and R. Ramanujam, editors, *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017*, volume 243 of *EPTCS*, pages 1–22, 2017. doi: 10.4204/EPTCS.243.1. URL https://doi.org/10.4204/EPTCS.243.1.

[107] Vaishak Belle, Thomas Bolander, Andreas Herzig, and Bernhard Nebel. Epistemic planning: Perspectives on the special issue. *Artif. Intell.*, 316:103842, 2023. doi: 10.1016/J.ARTINT.2022.103842. URL https://doi.org/10.1016/j.artint.2022.103842.

[108] Benedikt Löwe, Eric Pacuit, and Andreas Witzel. DEL planning and some tractable cases. In Hans van Ditmarsch, Jérôme Lang, and Shier Ju, editors, *Logic, Rationality, and Interaction - Third International Workshop, LORI 2011, Guangzhou, China, October 10-13, 2011. Proceedings*, volume 6953 of *Lecture Notes in Computer Science*, pages 179–192. Springer, 2011. doi: 10.1007/

978-3-642-24130-7_13. URL https://doi.org/10.1007/978-3-642-24130-7_13.

[109] Pere Pardo and Mehrnoosh Sadrzadeh. Planning in the logics of communication and change. In Wiebe van der Hoek, Lin Padgham, Vincent Conitzer, and Michael Winikoff, editors, *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, pages 1231–1232. IFAAMAS, 2012. URL http://dl.acm.org/citation.cfm?id=2343937.

[110] Guillaume Aucher. Del-sequents for regression and epistemic planning. *J. Appl. Non Class. Logics*, 22(4):337–367, 2012. doi: 10.1080/11663081.2012.736703. URL https://doi.org/10.1080/11663081.2012.736703.

[111] Thomas Bolander, Martin Holm Jensen, and François Schwarzentruber. Complexity results in epistemic planning. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2791–2797. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/395.

[112] Mikkel Birkegaard Andersen, Thomas Bolander, and Martin Holm Jensen. Conditional epistemic planning. In Luis Fariñas del Cerro, Andreas Herzig, and Jérôme Mengin, editors, *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Toulouse, France, September 26-28, 2012. Proceedings*, volume 7519 of *Lecture Notes in Computer Science*, pages 94–106. Springer, 2012. doi: 10.1007/978-3-642-33353-8_8. URL https://doi.org/10.1007/978-3-642-33353-8_8.

[113] Guillaume Aucher and Thomas Bolander. Undecidability in epistemic planning. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 27–33. IJCAI/AAAI, 2013. URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6903.

[114] Tristan Charrier, Bastien Maubert, and François Schwarzentruber. On the impact of modal depth in epistemic planning. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1030–1036. IJCAI/AAAI Press, 2016. URL http://www.ijcai.org/Abstract/16/150.

[115] Filippos Kominis and Hector Geffner. Beliefs in multiagent planning: From one agent to many. In Ronen I. Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015*, pages 147–155. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10617.

[116] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. An action language for multi-agent domains. *Artif. Intell.*, 302:103601, 2022. doi: 10.1016/J.ARTINT.2021.103601. URL https://doi.org/10.1016/j.artint.2021.103601.

[117] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. An action language for multi-agent domains: Foundations. *CoRR*, abs/1511.01960, 2015. URL http://arxiv.org/abs/1511.01960.

[118] Filippos Kominis and Hector Geffner. 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.*, pages 186–194, 2017. URL https://aaai.org/ocs/index.php/ICAPS/ICAPS17/paper/view/15748.

[119] Francesco Fabiano, Alessandro Burigana, Agostino Dovier, and Enrico Pontelli. EFP 2.0: A multi-agent epistemic solver with multiple e-state representations. In J. Christopher Beck, Olivier Buffet, Jörg Hoffmann, Erez Karpas, and Shirin Sohrabi, editors, *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pages 101–109. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/ICAPS/article/view/6650.

[120] Francesco Fabiano, Biplav Srivastava, Jonathan Lenchner, Lior Horesh, Francesca Rossi, and Marianna Bergamaschi Ganapini. E-PDDL: A standardized way of defining epistemic planning problems. *CoRR*, abs/2107.08739, 2021. URL https://arxiv.org/abs/2107.08739.

[121] Hector J. Levesque. What is planning in the presence of sensing? In William J. Clancey and Daniel S. Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 2*, pages 1139–1146. AAAI Press / The MIT Press, 1996. URL http://www.aaai.org/Library/AAAI/1996/aaai96-169.php.

[122] Ronald P. A. Petrick and Fahiem Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, pages 212–222. AAAI, 2002. URL http://www.aaai.org/Library/AIPS/2002/aips02-022.php.

[123] Tim Miller, Paolo Felli, Christian J. Muise, Adrian R. Pearce, and Liz Sonenberg. 'Knowing whether' in proper epistemic knowledge bases. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1044–1050, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12291.

[124] Abeer Alshehri, Tim Miller, and Liz Sonenberg. Modeling communication of collaborative multiagent system under epistemic planning. *Int. J. Intell. Syst.*, 36(10): 5959–5980, 2021. doi: 10.1002/INT.22536. URL https://doi.org/10.1002/int.22536.

[125] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier. The epistemic gossip problem. *Discrete Mathematics*, 342(3):654–663, 2019. doi: 10.1016/j.disc.2018.10.041. URL https://doi.org/10.1016/j.disc.2018.10.041.

[126] Martin C. Cooper, Andreas Herzig, Frédéric Maris, Elise Perrotin, and Julien Vianey. Lightweight parallel multi-agent epistemic planning. In Diego Calvanese, Esra Erdem, and Michael Thielscher, editors, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pages 274–283, 2020. doi: 10.24963/kr.2020/28. URL https://doi.org/10.24963/kr.2020/28.

[127] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, Elise Perrotin, and Pierre Régnier. A lightweight epistemic logic and its application to planning. *Artif. Intell.*, 298:103437, 2021. doi: 10.1016/j.artint.2020.103437. URL https://doi.org/10.1016/j.artint.2020.103437.

[128] Ronald P. A. Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, pages 613–622. AAAI Press, 2004. URL http://www.aaai.org/Library/KR/2004/kr04-064.php.

[129] Hai Wan, Rui Yang, Liangda Fang, Yongmei Liu, and Huada Xu. 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*, pages 3257–3263, 2015. URL http://ijcai.org/Abstract/15/459.

[130] Jie Fan, Yanjing Wang, and Hans van Ditmarsch. Contingency and knowing whether. *Rew. Symb. Logic*, 8(1):75–107, 2015. doi: 10.1017/S1755020314000343. URL https://doi.org/10.1017/S1755020314000343.

[131] Andreas Herzig and Faustine Maffre. 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*, pages 249–263, 2015. doi: 10.1007/978-3-319-33509-4_20. URL https://doi.org/10.1007/978-3-319-33509-4_20.

[132] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, 175(2):487–511, 2011. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2010.09.008. URL https://www.sciencedirect.com/science/article/pii/S0004370210001578.

[133] James Hales, Tim French, and Rowan Davies. Refinement quantified logics of knowledge and belief for multiple agents. *Advances in Modal Logic*, 9:317–338, 2012.

[134] Miquel Ramírez, Michael Papasimeon, Nir Lipovetzky, Lyndon Benke, Tim Miller, Adrian R. Pearce, Enrico Scala, and Mohammad Zamani. Integrated hybrid planning and programmed control for real time UAV maneuvering. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 1318–1326. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL http://dl.acm.org/citation.cfm?id=3237896.

[135] Aristotle Aristotle and Aristotle. *Metaphysics*, volume 1. Harvard University Press Cambridge, MA, 1933.

[136] Brenda S. Baker and Robert E. Shostak. Gossips and telephones. *Discrete Mathematics*, 2(3):191–193, 1972. doi: 10.1016/0012-365X(72)90001-5. URL https://doi.org/10.1016/0012-365X(72)90001-5.

[137] Hector J. Levesque. A completeness result for reasoning with incomplete first-order knowledge bases. In Anthony G. Cohn, Lenhart K. Schubert, and Stuart C. Shapiro, editors, *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998*, pages 14–23. Morgan Kaufmann, 1998.

[138] Gerhard Lakemeyer and Yves Lespérance. Efficient reasoning in multiagent epistemic logics. 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*, pages 498–503, 2012. doi: 10.3233/978-1-61499-098-7-498. URL https://doi.org/10.3233/978-1-61499-098-7-498.

[139] Alvin I. Goldman. A causal theory of knowing. *Journal of Philosophy*, 64(12): 357–372, 1967. doi: 10.2307/2024268.

[140] Thomas Bolander. Seeing is believing: Formalising false-belief tasks in dynamic epistemic logic. In Andreas Herzig and Emiliano Lorini, editors, *Proceedings of the European Conference on Social Intelligence (ECSI-2014), Barcelona, Spain, November 3-5, 2014*, volume 1283 of *CEUR Workshop Proceedings*, pages 87–107. CEUR-WS.org, 2014. URL https://ceur-ws.org/Vol-1283/paper_14.pdf.

[141] Andreas Herzig, Emiliano Lorini, and Faustine Maffre. A poor man's epistemic logic based on propositional assignment and higher-order observation. In Wiebe van der Hoek, Wesley H. Holliday, and Wen-Fang Wang, editors, *Logic, Rationality, and Interaction - 5th International Workshop, LORI 2015 Taipei, Taiwan, October 28-31, 2015, Proceedings*, volume 9394 of *Lecture Notes in Computer Science*, pages 156–168. Springer, 2015. doi: 10.1007/978-3-662-48561-3_13. URL https://doi.org/10.1007/978-3-662-48561-3_13.

[142] Wiebe van der Hoek, Bernd van Linder, and John-Jules Meyer. Group knowledge is not always distributed (neither is it always implicit). *Mathematical Social Sciences*, 38(2):215–240, 1999. ISSN 0165-4896. doi: https://doi.org/10.1016/S0165-4896(99)00013-X. URL https://www.sciencedirect.com/science/article/pii/S016548969900013X.

[143] Nir Friedman and Joseph Y. Halpern. A knowledge-based framework for belief change, part II: revision and update. In *Proceedings of the 4th KR*, pages 190–201, 1994.

[144] Nir Friedman and Joseph Y. Halpern. A knowledge-based framework for belief change, part I: foundations. In *Proceedings of the 5th TARK*, pages 44–64, 1994.

[145] Richard B. Scherl. A situation-calculus model of knowledge and belief based on thinking about justifications. In *Proceedings of the 20th NMR, Part of the FLoC 2022*, volume 3197, pages 104–114, 2022. URL http://ceur-ws.org/Vol-3197/paper10.pdf.

[146] Sergei N. Artëmov. The logic of justification. *Rev. Symb. Log.*, 1(4):477–513, 2008. doi: 10.1017/S1755020308090060. URL https://doi.org/10.1017/S1755020308090060.

[147] Tuan-Fang Fan and Churn-Jung Liau. Doxastic reasoning with multi-source justifications based on second order propositional modal logic. In *Proceedings of the 16th AAMAS*, pages 1529–1531. ACM, 2017. URL http://dl.acm.org/citation.cfm?id=3091351.

[148] Davide Grossi and Wiebe van der Hoek. Justified beliefs by justified arguments. In *KR 2014*, 2014. URL http://www.aaai.org/ocs/index.php/KR/KR14/paper/view/7997.

[149] Adam Bjorndahl and Aybüke Özgün. Logic and topology for knowledge, knowability, and belief. *Rev. Symb. Log.*, 13(4):748–775, 2020. doi: 10.1017/S1755020319000509. URL https://doi.org/10.1017/S1755020319000509.

[150] Robert Stalnaker. On logics of knowledge and belief. *Philosophical Studies*, 128 (1):169–199, 2006.

[151] Kai Li and Jan van Eijck. Public announcements, public lies and recoveries. *J. Log. Lang. Inf.*, 31(3):423–450, 2022. doi: 10.1007/s10849-022-09351-4. URL https://doi.org/10.1007/s10849-022-09351-4.

[152] Petar Iliev. *On the relative succinctness of some modal logics.* PhD thesis, University of Liverpool, UK, 2013. URL http://repository.liv.ac.uk/14481/.

[153] Tim French, Wiebe van der Hoek, Petar Iliev, and Barteld P. Kooi. On the succinctness of some modal logics. *Artif. Intell.*, 197:56–85, 2013. doi: 10.1016/j.artint.2013.02.003. URL https://doi.org/10.1016/j.artint.2013.02.003.

[154] Sébastien Konieczny. On the difference between merging knowledge bases and combining them. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR 2000, Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, Breckenridge, Colorado, USA, April 11-15, 2000*, pages 135–144. Morgan Kaufmann, 2000.

[155] Sébastien Konieczny and Ramón Pino Pérez. Merging information under constraints: A logical framework. *J. Log. Comput.*, 12(5):773–808, 2002. doi: 10.1093/logcom/12.5.773. URL https://doi.org/10.1093/logcom/12.5.773.

[156] Patricia Everaere, Sébastien Konieczny, and Pierre Marquis. Belief merging versus judgment aggregation. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 999–1007. ACM, 2015. URL http://dl.acm.org/citation.cfm?id=2773279.

[157] Yì N. Wáng and Thomas Ågotnes. Public announcement logic with distributed knowledge: expressivity, completeness and complexity. *Synth.*, 190(Supplement-1):135–162, 2013. doi: 10.1007/s11229-012-0243-3. URL https://doi.org/10.1007/s11229-012-0243-3.

[158] Floris Roelofsen. Distributed knowledge. *J. Appl. Non Class. Logics*, 17(2): 255–273, 2007. doi: 10.3166/jancl.17.255-273. URL https://doi.org/10.3166/jancl.17.255-273.

[159] Thomas Ågotnes and Yì N. Wáng. Resolving distributed knowledge. *Artificial Intelligence*, 252:1–21, 2017. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2017.07.002. URL https://www.sciencedirect.com/science/article/pii/S0004370217300759.

[160] Anthia Solaki. Bounded multi-agent reasoning: Actualizing distributed knowledge. In Manuel A. Martins and Igor Sedlár, editors, *Dynamic Logic. New Trends and Applications - Third International Workshop, DaLí 2020, Prague, Czech Republic, October 9-10, 2020, Revised Selected Papers*, volume 12569 of *Lecture Notes in Computer Science*, pages 239–258. Springer, 2020. doi: 10.1007/978-3-030-65840-3_15. URL https://doi.org/10.1007/978-3-030-65840-3_15.

[161] Yanjun Li. Multi-agent conformant planning with distributed knowledge. In Sujata Ghosh and Thomas Icard, editors, *Logic, Rationality, and Interaction - 8th International Workshop, LORI 2021, Xi'ian, China, October 16-18, 2021, Proceedings*, volume 13039 of *Lecture Notes in Computer Science*, pages 128–140. Springer, 2021. doi: 10.1007/978-3-030-88708-7_10. URL https://doi.org/10.1007/978-3-030-88708-7_10.

[162] Éric Goubault, Roman Kniazev, Jérémy Ledent, and Sergio Rajsbaum. Semi-simplicial set models for distributed knowledge. In *LICS*, pages 1–13, 2023. doi: 10.1109/LICS56636.2023.10175737. URL https://doi.org/10.1109/LICS56636.2023.10175737.

[163] Andreas Herzig, Emiliano Lorini, Elise Perrotin, Fabián Romero, and François Schwarzentruber. A logic of explicit and implicit distributed belief. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325, pages 753–760. IOS Press, 2020. doi: 10.3233/FAIA200163. URL https://doi.org/10.3233/FAIA200163.

[164] Georg Meggle. *Common Belief and Common Knowledge*, pages 251–258. Springer Netherlands, Dordrecht, 2003. ISBN 978-94-007-1046-7. doi: 10.1007/978-94-007-1046-7_16. URL https://doi.org/10.1007/978-94-007-1046-7_16.

[165] François Schwarzentruber. Seeing, knowledge and common knowledge. In *Logic, Rationality, and Interaction - Third International Workshop, (LORI) 2011, Guangzhou, China, October 10-13, 2011. Proceedings*, pages 258–271, 2011. doi: 10.1007/978-3-642-24130-7_19. URL https://doi.org/10.1007/978-3-642-24130-7_19.

[166] Giacomo Bonanno. On the logic of common belief. *Math. Log. Q.*, 42:305–311, 1996. doi: 10.1002/malq.19960420126. URL https://doi.org/10.1002/malq.19960420126.

[167] Aviad Heifetz. Iterative and fixed point common belief. *J. Philos. Log.*, 28(1):61–79, 1999. doi: 10.1023/A:1004357300525. URL https://doi.org/10.1023/A:1004357300525.

[168] Giacomo Bonanno and Klaus Nehring. Common belief with the logic of individual belief. *Math. Log. Q.*, 46(1):49–52, 2000. doi: 10.1002/(SICI)1521-3870(200001)46:1⟨49::AID-MALQ49⟩3.0.CO;2-R. URL https://doi.org/10.1002/(SICI)1521-3870(200001)46:1<49::AID-MALQ49>3.0.CO;2-R.

[169] Churn-Jung Liau. Belief, information acquisition, and trust in multi-agent systems—a modal logic formulation. *Artificial Intelligence*, 149(1):31–60, 2003.

[170] Weijia Li, Guang Hu, and Yangmengfei Xu. Beyond static assumptions: the predictive justified perspective model for epistemic planning. *CoRR*, abs/2412.07941, 2024. doi: 10.48550/ARXIV.2412.07941. URL https://doi.org/10.48550/arXiv.2412.07941.

[171] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[172] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In Martin A. Fischler and Oscar Firschein, editors,

Readings in Computer Vision, pages 726–740. Morgan Kaufmann, San Francisco (CA), 1987. ISBN 978-0-08-051581-6. doi: https://doi.org/10.1016/B978-0-08-051581-6.50070-2. URL https://www.sciencedirect.com/science/article/pii/B9780080515816500702.

[173] Thomas Bolander, Lasse Dissing, and Nicolai Herrmann. Del-based epistemic planning for human-robot collaboration: Theory and implementation. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 120–129, 2021. doi: 10.24963/KR.2021/12. URL https://doi.org/10.24963/kr.2021/12.

[174] Shashank Shekhar, Anthony Favier, and Rachid Alami. Human-aware epistemic task planning for human-robot collaboration. In *ICAPS 2024 Workshop on Human-Aware Explainable Planning*, 2024. URL https://openreview.net/forum?id=s2e2kTWvhI.

[175] Zhuang Chen, Jincenzi Wu, Jinfeng Zhou, Bosi Wen, Guanqun Bi, Gongyao Jiang, Yaru Cao, Mengting Hu, Yunghwei Lai, Zexuan Xiong, and Minlie Huang. Tombench: Benchmarking theory of mind in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15959–15983. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.847. URL https://doi.org/10.18653/v1/2024.acl-long.847.

[176] Sneheel Sarangi, Maha Elgarf, and Hanan Salam. Decompose-tom: Enhancing theory of mind reasoning in large language models through simulation and task decomposition. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 10228–10241. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.682/.