Journal of Artificial Intelligence Research 75 (2022) 489-539 Submitted 10/2021; published 10/2022

Planning with Perspectives – Decomposing Epistemic Planning using Functional STRIPS

Guang Hu GHU1@STUDENT.UNIMELB.EDU.AU
Tim Miller TMILLER@UNIMELB.EDU.AU
Nir Lipovetzky NIR.LIPOVETZKY@UNIMELB.EDU.AU
School of Computing and Information Systems
The University of Melbourne, Australia

Abstract

In this paper, we present a novel approach to epistemic planning called *planning with perspectives* (PWP) that is both more expressive and computationally more efficient than existing state-of-the-art epistemic planning tools. Epistemic planning — planning with knowledge and belief — is essential in many multi-agent and human-agent interaction domains. Most state-of-the-art epistemic planners solve epistemic planning problems by either compiling to propositional classical planning (for example, generating all possible knowledge atoms or compiling epistemic formulae to normal forms); or explicitly encoding Kripke-based semantics. However, these methods become computationally infeasible as problem sizes grow. In this paper, we decompose epistemic planning by delegating reasoning about epistemic formulae to an external solver. We do this by modelling the problem using *Functional STRIPS*, which is more expressive than standard STRIPS and supports the use of external, black-box functions within action models. Building on recent work that demonstrates the relationship between what an agent 'sees' and what it knows, we define the *perspective* of each agent using an external function, and build a solver for epistemic logic around this. Modellers can customise the perspective function of agents, allowing new epistemic logics to be defined without changing the planner. We ran evaluations on well-known epistemic planning benchmarks to compare an existing state-of-the-art planner, and on new scenarios that demonstrate the expressiveness of the PWP approach. The results show that our PWP planner scales significantly better than the state-of-the-art planner that we compared against, and can express problems more succinctly.

1. Introduction

In this paper, we present an approach to epistemic planning called *planning with perspectives* (PWP) that is radically different to existing work. PWP exploits recent advances in automated planning using the Functional STRIPS language (Geffner, 2000; Frances & Geffner, 2015). The key insight used in PWP is to delegate reasoning about epistemic formulae to external functions in Functional STRIPS. External functions are arbitrary functions implemented in a programming language that can be called during planning. We formalise a logic that allows the external reasoner to determine the knowledge, including nested knowledge, of all agents using just the state passed to it by the planner.

In many scenarios, autonomous agents need to plan about the *knowledge* or *beliefs* of other agents in the environment. This concept is known as *epistemic planning* (Bolander & Andersen, 2011), a research topic that brings together the knowledge reasoning and planning communities.

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

There are typically two frameworks in which epistemic planning is studied. The first is to use DEL. This line of research investigates the decidability and complexity of epistemic planning and studies what type of problems it can solve (Bolander, 2014; Bolander, Jensen, & Schwarzentruber, 2015; Bolander, 2017) The second is to extend existing planning languages and solvers to epistemic tasks (Muise, Belle, Felli, McIlraith, Miller, Pearce, & Sonenberg, 2015a; Muise, Miller, Felli, Pearce, & Sonenberg, 2015b; Kominis & Geffner, 2015, 2017; Wan, Yang, Fang, Liu, & Xu, 2015; Huang, Fang, Wan, & Liu, 2017; Le, Fabiano, Son, & Pontelli, 2018; Wan, Fang, & Liu, 2021). In this paper, we take the latter direction.

Epistemic planning is undecidable in the general case. Thus, one of the main challenges of epistemic planning concerns computational efficiency. The dominant approach in this area relies on compilations. These solutions pre-compile epistemic planning problems into classical planning problems, using off-the-shelf classical planners to find solutions (Muise et al., 2015a, 2015b; Kominis & Geffner, 2015, 2017); or pre-compile the epistemic formulae into specific normal forms for better performance during search (Huang et al., 2017; Wan et al., 2021). Such approaches have been shown to be fast at planning, but the compilation is computationally expensive. For example, from Muise et al.'s (2015a) results, the planner takes less than a second to solve all of their small benchmark problems, but for some problems, the compilation time is more than half a minute as the problem size increases.

To mitigate the complexity of epistemic planning, this paper presents two contributions. In Section 3, we propose a model of epistemic logic that extends recent insights in defining what an agent knows as a function of what it "sees" (Cooper, Herzig, Maffre, Maris, & Régnier, 2016; Gasquet, Goranko, & Schwarzentruber, 2014). Cooper et al. (2016) define *seeing relations* for agent i as a modal operator S_i that "sees" whether a proposition is true. Then they define knowledge K for agent i of a proposition p as $K_i p \leftrightarrow p \wedge S_i p$; that is, if p is true and agent i sees p, then it knows p. Thus, the seeing modal operator is equivalent to the 'knowing whether' operator in epistemic logic (Fan, Wang, & van Ditmarsch, 2015; Miller, Felli, Muise, Pearce, & Sonenberg, 2016). We generalise the notion of seeing relations to *perspective functions*, which are functions that determine which variables an agent sees in its environment. The domain of variables can be discrete or continuous, not just propositional. The intuition behind perspective functions is similar to seeing relations, however, we show that by changing the definition of perspective functions, we can establish new epistemic logics, such as Big Brother Logic (Gasquet et al., 2014), a logic about visibility and knowledge in two-dimensional Euclidean planes.

In Section 4, we show how to integrate perspective functions within functional STRIPS models as *external functions* (Francès, Ramírez, Lipovetzky, & Geffner, 2017). External functions are black-box functions implemented in any programming language (in our case, $C++$), that can be called within action models. Epistemic reasoning is delegated to external functions, where epistemic formulae are evaluated lazily, avoiding the exponential blow-up from epistemic formulae present in other compilation-based approaches. This delegation effectively decomposes epistemic reasoning from search, and allows us to implement our approach in any functional STRIPS planner that supports external functions. Further, the modeller can implement new perspective functions that are tied to specific domains, and our model will use those functions to evaluate desired epistemic relations, effectively defining new external solvers. We show that perspective functions can be generic and implement many different variants of epistemic logic, such as Kripke semantics (Fagin, Halpern, Moses, & Vardi, 2003), Muise et al. (2015a)'s finite-depth, propositional epistemic logic (Muise et al., 2015a) or observability relations (Le et al., 2018); however, our experience suggests that tailoring perspective functions to specific domains results in more understandable and elegant models that can be solved more efficiently.

In Section 5, we evaluate our PWP approach against a state-of-the-art epistemic planning approach (Muise et al., 2015a) and against Cooper et al. (2016)'s Gossip problem solver (Cooper et al., 2016). In our experiments, we use a width-based functional STRIPS planner (Francès et al., 2017) that is able to evaluate the truth value of epistemic fluents with external solvers, and solve a wide range of epistemic problems efficiently, including but not limited to, nested knowledge, distributed knowledge and common knowledge. We compare our approach to a state-of-the-art epistemic planner that relies on a compilation to classical planning (Muise et al., 2015a). We implement three benchmarks problems, *Corridor* (Kominis & Geffner, 2015), Gossip (Cooper et al., 2016), and *Grapevine* (Muise et al., 2015a) to compare the computational performance, and model two new domains, *Big Brother Logic* and *Social-media Network*, to examine the expressiveness of our model. The results show that, unlike in the compilation-based approaches, execution time on epistemic reasoning increases polynomially in the depth of nesting, rather than exponentially.

2. Background and Related Work

In this section, we present the necessary background and discuss related work in three main areas: (1) classical planning; (2) epistemic logic; and (3) epistemic planning.

2.1 Classical Planning

Planning is the model-based approach to action selection in 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 (Geffner & Bonet, 2013). Models vary depending on the assumptions imposed on the dynamics of the world, from classical planning models where all actions have deterministic instantaneous effects and the world is fully known, up to temporal and POMDP models, where actions have durations and belief distributions about the state of the world respectively. Models are described concisely through declarative languages such as STRIPS and PDDL (Fikes & Nilsson, 1971; Haslum, Lipovetzky, Magazzeni, & Muise, 2019), general enough to allow the encoding of different problems, while at the same time

revealing important structural information that allow planners to scale to large problems. In fact, most planners rely on exploiting the structure revealed in the action theory to guide the search of solutions, from the very first general problem solver (Simon & Newell, 1963) up to the latest computational approaches based on SAT, and heuristic search (Rintanen, 2012; Richter & Westphal, 2010; Lipovetzky & Geffner, 2017).

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 (Bellemare, Naddaf, Veness, & Bowling, 2013). Thus, an extension of STRIPS, *Functional STRIPS* has been proposed by Geffner (2000), which allows modelling simulators with external functions. Consequently, a new family of width-based planners (Lipovetzky & Geffner, 2012, 2014, 2017) have been proposed, broadening the scope of planning and scaling even when the planning model is described through black-box simulators, only requiring the exposure of the state variables, but not imposing any syntax restriction on the action theory (Francès et al., 2017). Importantly, the denotation of some symbols can be given procedurally as external functions, which can be implemented in programming languages such as C/C++.

In this paper we focus on a model of epistemic planning that extends the *classical planning model* as a tuple $\mathcal{S} = \langle S, s_0, S_G, Act, A, f, c\rangle$ where S is a set of states, $s_0 \in S$ is the initial state, $S_G \subseteq S$ is the set of goal states, Act is the set of actions, $A(s)$ is the subset actions applicable in state s, f is the transition function so that $f(a, s)$ represents the state s' that results from doing action a in the state s, and $c(a, s)$ is a cost function. The solution to a classical planning model $\mathcal{S}$, called a plan, is a sequence of actions that maps the initial state into a goal state, i.e., $\pi = \langle a_0, \ldots, a_n\rangle$ is a plan if there is a sequence of states $s_0, \ldots, s_{n+1}$ such that $a_i \in A(s_i)$, $s_{i+1} = f(a_i, s_i)$ for $i = 0, \ldots, n$ and $s_{n+1} \in S_G$. The cost of plan π is given by the sum of action costs $c(a_i, s_i)$ and a plan is optimal if there is no plan with smaller cost.

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

```
action  move(?x - location ?g - ghost)
  prec  (valid_loc ?x)
  effs  (assign (loc ?g) ?x)
        (when (has_pacman ?x) (not (has_pacman ?x)))
```

Figure 1: Example STRIPS action of a ghost move action

In addition to STRIPS, the *Planning Domain Definition Language* PDDL (Haslum et al., 2019) is commonly used to model planning problems. A standard PDDL model contains two files: a domain file and a problem file. The domain specifies the descriptions of propositions (predicates) and operators (actions). The action description covers the parameters, precondition and effects. The problem gives the objects, initial state and goal condition. The set of all propositions or operators can be grounded by mapping the

objects with the descriptions. As syntactic sugar, PDDL allows using conditional effect (*when-effect*) to avoid redundant descriptions for similar actions. An example can be found in Figure 1. Instead of having two descriptions of the actions when a ghost moves into a location with or without pacman, the modeller can model this with single action description by using conditional effect.

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

The standard classical planning languages and solvers do not support the use of procedures or external theories. The first theoretical research that solve this problem is from Geffner (2000)'s Functional STRIPS language (F-STRIPS), where the denotation of (non-fluent) function symbols can be given using external functions. In addition, Dornhege, Eyerich, Keller, Trüg, Brenner, and Nebel (2009b) 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 (Dornhege, Gissler, Teschner, & Nebel, 2009c; Gaschler, Petrick, Khatib, & Knoll, 2018; Kaelbling & Lozano-Pérez, 2012; Bajada, Fox, & Long, 2015). Planning Modulo Theories were introduced by Gregory, Long, Fox, and Beck (2012), an idea inspired by SAT Modulo Theories (Nieuwenhuis, Oliveras, & Tinelli, 2006), 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(R) have been proposed as a new width-based planning (Lipovetzky & Geffner, 2012). The BFWS(R) family of planners (Francès et al., 2017) have been shown to scale up even in the presence of functional symbols defined procedurally[1].

Any classical F-STRIPS (Francès et al., 2017) problem can be represented by a tuple $(V, D, \mathcal{O}, \mathcal{I}, \mathcal{G}, \mathbb{F})$, where V and D are variables (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 cannot be defined as a propositional classical planning task, such as those whose effects are too complex to be modelled by propositional fluents, or even those whose actions and effects have some unrevealed corresponding relations.

Consider the example in Figure 2, adapted from (Francès et al., 2017), which models an action to move `Pacman` between locations. External functions start with the `@` charac-

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

```
action  move(?x - location ?p - pacman)
  prec  (@validMove (loc ?p) ?x)
  effs  (assign (loc ?p) ?x)
        (forall (?g - ghost) (@move_ghost (loc ?g) ?x))
```

Figure 2: Example F-STRIPS action of a pacman move action

ter. In this example, `@validMove` is an external function that checks whether the move is valid, encoding Pacman's maze implicitly. `@move_ghost` encodes the deterministic movement strategy followed by all ghosts, a strategy that depends on Pacman's Manhattan distance. `@validMove` was used for convenience as a maze can be expressed easily with propositional logic, but encoding the ghost movements is not trivial (Francès et al., 2017).

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.

2.2 Epistemic Logic

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

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

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

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

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

Fagin et al. (2003) define the semantics of epistemic logic using Kripke structures, as standard in modal logic. A Kripke structure is a tuple $M = (W, \pi, R_1, \ldots, R_n)$ where:

- W is a non-empty set of all possible worlds;
- π is an interpretation function such that $\pi(w) : Prop \to \{true, false\}$ defines which propositions are true and false in world $w \in W$; and
- $R_1, \ldots, R_n$ represents the *accessibility relations* over worlds for each of the n agents in Agt.

Given a world 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 $true$. R_i for agent i is a binary relation over worlds. For any pair of worlds v and w, if $(w, v) \in R_i$, 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 $(w, v) \in R_i$. With this definition of Kripke structures, we can define the semantics of knowledge.

Given a world w, a proposition p, a propositional formula φ and a Kripke model M, the truth of two basic formulae are defined as follows:

$$(M, w) \vDash p \quad \text{iff} \quad \pi(w)(p) = true$$
$$(M, w) \vDash K_i\varphi \quad \text{iff} \quad (M, v) \vDash \varphi \text{ for all } v \text{ such that } (v, w) \in R_i$$

Standard propositional logic rules define conjunction and negation. $(M, w) \vDash K_i\varphi$ is defined by formula φ being true at all worlds v reachable from w via the accessibility relation R_i. This allows knowledge to be nested; for example, K_aK_bp represents that agent a knows that agent b knows p, which means p is true at all worlds reachable by applying accessibility relation R_a followed by R_b. To be specific, $(M, w) \vDash K_aK_b\varphi$ is true if and only if $(M, v) \vDash K_b\varphi$ for all v such that $(v, w) \in R_a$, which means $(M, v') \vDash \varphi$ for all v' such that $(v', v) \in R_b$, for all v such that $(v, w) \in R_a$. This idea generalises to an arbitrary level of nested knowledge.

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

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

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

$E_G\varphi$ represents that everyone in group G knows φ and $C_G\varphi$ represents that it is *commonly known* in group G that φ is true, which means that everyone knows φ, and everyone knows that everyone knows φ, *ad infinitum*. $D_G\varphi$ represents *distributed* knowledge, which means 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.

The semantics for these group operators are defined as follows:

$$(M, w) \vDash E_G\varphi \quad \text{iff} \quad (M, w) \vDash K_i\varphi \text{ for all } i \in G$$
$$(M, w) \vDash C_G\varphi \quad \text{iff} \quad (M, v) \vDash \varphi \text{ for all } v \text{ that are } G{-}reachable$$
$$(M, w) \vDash D_G\varphi \quad \text{iff} \quad (M, v) \vDash \varphi \text{ for all } v \text{ such that } (w, v) \in \bigcap_{i \in G} R_i$$

By definition, $(M, w) \vDash E_G\varphi$ holds if and only if φ is known by all agents in G. World v is $G{-}reachable$ from w if w can reach v within k steps of accessible relations, or for some k where $k \geq 1$ (Fagin et al., 2003). 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 the possible worlds that all agents from G agree 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.

2.2.1 Seeing and Knowledge

Recently Gasquet et al. (2014) noted the relationship between what an agent sees and what it knows. They define a more specific task of logically modeling and reasoning about cooperating tasks of vision-based agents, which they call *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 scenario, agents are stationary cameras in a Euclidean plane $\mathbb{R}^2$, and they assume that those cameras can see anything in their sight range, and they do not block others' sight. They extend Fagin et al. (2003)'s logic by noting that, at any point in time, what an agent knows, including nested knowledge, can be derived directly from what it can see in the current 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.

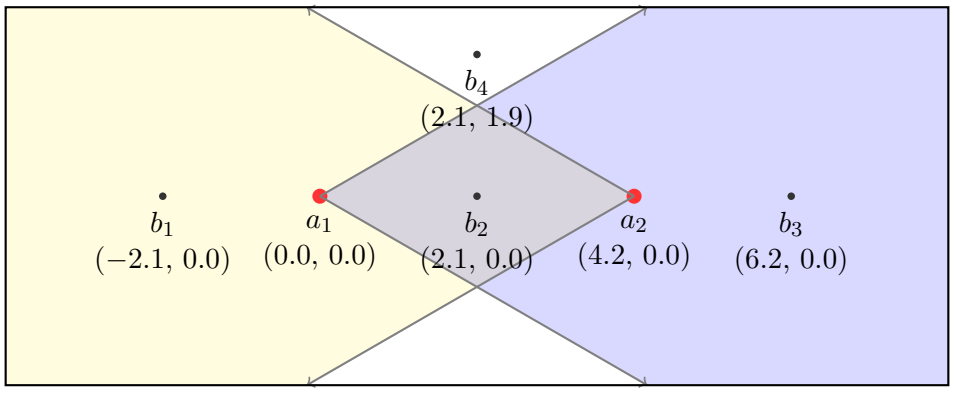

Figure 3: Example for Big Brother Logic

Figure 3 shows an example with two agents, a_1 and a_2, and model $((0.0, 0.0), 60°, D, R)$ and $((4.2, 0.0), 60°, D, R)$ respectively, along with four objects, b_1, b_2, b_3 and b_4. Based on the current 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. (2014) show the relationship between seeing and knowing. For example, $K_a(b \triangleright c)$ is defined as $a \triangleright b \wedge a \triangleright c \wedge b \triangleright c$.

Gasquet et al. (2014) also define a common knowledge operator, in a similar manner to that of Fagin et al. (2003)'s definition based on $G{-}reachable$ worlds. In Figure 3, 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 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 'a_2 can see b_2' ", etc. Thus, some common knowledge has been established.

Cooper et al. (2016) generalise this idea to seeing propositions, rather than just seeing other agents in a Euclidean plane. 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. They add an extra type of formulae α that describes formulae (propositions) that can be seen:

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

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 nestings of seeing relationships that see atomic propositions.

From this, they note the equivalences $K_i p \leftrightarrow p \wedge S_i p$ and $K_i \neg p \leftrightarrow \neg p \wedge S_i p$. To be specific, they disallow $S_i \neg p$. This tight correspondence between seeing and knowing is intuitive: an agent knows p is true if p is true and the agent can see the variable p. Such a relationship is the same as the one between knowing something is true and *knowing whether* something is true (Miller et al., 2016; Fan et al., 2015; Petrick & Bacchus, 2002).

Comparing these two bodies of work, Gasquet et al. (2014) 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 constrained only to vision in physical spaces. While in Cooper et al. (2016)'s world, the seeing operator applies to propositional variables, and thus visibility can be interpreted more abstractly; for example, "seeing" (hearing) a message over a telephone. This connection between seeing and knowing is similar to the idea of sensing actions in partially-observable planning (Dornhege et al., 2009c; Dornhege, Eyerich, Keller, Trüg, Brenner, & Nebel, 2009a; Gaschler et al., 2018; Kaelbling & Lozano-Pérez, 2012; Bajada et al., 2015; Le et al., 2018; Cooper, Herzig, Maris, Perrotin, & Vianey, 2020; Kominis & Geffner, 2015, 2017; Fabiano, Burigana, Dovier, & Pontelli, 2020), as seeing/sensing generates new knowledge. However, sensing actions are actions, whereas the idea of 'seeing' is a relation over properties of states. Further, our work looks at nested knowledge in the framework of epistemic planning, which is a generalisation of partially-observable planning (Bolander & Andersen, 2011).

This paper generalises seeing relations to perspective functions, which are domain-dependent functions defining what agents see in particular worlds. The result is more

flexible than seeing relations, and allows Big Brother Logic to be defined with a simple perspective function, as well as new perspective functions for new logics; for example, Big Brother Logic extended to three-dimensional planes, or visibility of messages on a social network.

2.3 Epistemic Planning

Bolander and Andersen (2011) introduce the concept of epistemic planning, for both single agent and multi-agent domains.

Their planning framework is defined in *dynamic epistemic logic* (DEL) (Van Ditmarsch, van Der Hoek, & Kooi, 2007), which has been shown to be undecidable in general, but decidable with (non-epistemic) propositional preconditions (Bolander et al., 2015). In addition, for DEL model checking, they also proved PSPACE-hardness of the plan verification problem. This formalism has been used to explore theoretical properties of epistemic planning; for example, Engesser, Bolander, Mattmüller, and Nebel (2017) used concepts of perspective shifting to reason about other agents' contributions to joint goals. Along with implicit coordination actions, their model can solve some problems elegantly without communication between agents.

Since epistemic planning is formalised in DEL, there has been substantial work on DEL-based planning. However, in this paper, our focus is on the design, implementation, and evaluation of planning tools, rather than on logic-based models of planning. Therefore, we focus on related work on planning tools and evaluation on benchmarks.

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

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

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

Kominis and Geffner (2015) adapt methods from partially-observable planning for representing beliefs of a single agent, and convert that method to handle multi-agent settings. Their idea is to maintain the problem's Kripke structure. By their definition (adapting STRIPS), an epistemic planning problem P is a tuple $\langle A, F, I, O, N, U, G\rangle$. In their model, A is the set of agent identifiers, I is a set of possible initial states rather than just one initial state in STRIPS. Then, instead of tracking the problem by only updating actual states, they combine the Kripke structure with the state at time step t and possible initial states by using beliefs, $B(t)$. A set of beliefs $B(t)$ contains a set of $B(s_i, t)$ for each possible initial state s_i. And in each $B(s, t)$, there are the actual state $v(s, t)$ and indistinguishable relations $r_i(s, t)$ between current state and possible initial state for each agent i. By doing so, they are able to construct the Kripke structure from the initial state for each agent.

To keep the Kripke structure during the search, they define three kinds of action sets: O, N and U. O represents all physical actions, which is the same O as in classical planning. The operator updates the actual current state s based on deterministic transition functions. 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 (1996). 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. They evaluate their model on the *Muddy Children*, *Sum* and *Word Rooms* (Kominis & Geffner, 2015) domains. As far as we can tell from their experiments, they keep the depth of the epistemic relation fixed at one and vary the number of agents or the number of rooms. Their results show that their model is able to solve all cases presented with different suitable planners. However, from their result, there is an exponential growth on the time consumption due to the increased scale of the problems. In addition, following their approach, the modeller must specify the **sense** action for each pair of (i, φ) and **update** action for each φ.

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

More recently, rather than compiling epistemic planning problems into classical planning, Huang et al. (2017) build a native multi-agent epistemic planner, and propose a general representation framework for multi-agent epistemic problems (Huang et al., 2017). They consider the multi-agent epistemic planning problem from a third person point of view. Based on a well-established concept of belief change algorithms (both revision and update algorithms), they design and implement a planner called MEPK to encode belief change as

the result of planning actions. They evaluate their approach with *Grapevine*, *Hexa Game* and *Gossip*, among others. From their results, it is clear that their approach can handle a variety of problems, and performance on some problems is better than other approaches. While this approach is different from Kominis and Geffner (2015) and Muise et al. (2015a), it still requires a compilation phase before planning to re-write epistemic formula into a specific normal form called *alternating cover disjunctive formulae* (ACDF) (Hales, French, & Davies, 2012). The ACDF formula is worst-case exponentially longer than the original formula. The results show that this step has a similar computational burden as either Kominis and Geffner (2015) or Muise et al. (2015a).

In addition to Kominis and Geffner (2015), Baral, Gelfond, Pontelli, and Son (2015) also explore "sensing" actions. They define an action language $m\mathcal{A}^*$ that encodes Kripke structures using three components: the actual world; the beliefs of each agent about the actual world; and the beliefs of each agent about the beliefs of the other agents. Compared to Kominis and Geffner's work, they provide a theoretical foundation for developing an epistemic planner using action language, while Kominis and Geffner convert their action language into classical PDDL.

Le et al. (2018) build two epistemic forward planners named EFP and PG-EFP. They define their planning problem based on $m\mathcal{A}^*$ as a tuple $(\mathcal{AG}, \mathcal{F}, A, O)$, where $\mathcal{AG}$ is the set of agent identifiers and $\mathcal{F}$ is the set of fluents. Action set A and observability statement set O compose the actions and effects. In their A, they specify preconditions and three possible effects: ontic, sensing and announcement, which works as follows: ontic effects change the state (actual world); sensing actions reveal truth value of some fluent f; and, announcement actions announce the truth value of some fluent f, which affects 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 those two planners based on their model with different search algorithms, BFS and heuristic search for EFP and PG-EFP 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. (2015a)'s and Huang et al. (2017)'s solutions on *Corridor*, *Collaboration-and-communication* (Kominis & Geffner, 2015), and *Assembly Line* (Huang et al., 2017). 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.

Overall, our PWP approach to planning differs from traditional epistemic planning approaches as we do not compile epistemic planning problems into classical planning problems (Muise et al., 2015a; Kominis & Geffner, 2015), which means our model does not require a costly pre-compilation step. Compared to more recent work (Huang et al., 2017; Le et al., 2018), our approach works on any F-STRIPS planner.

In addition, PWP is implemented using F-STRIPS (a state based language) rather than an action language. Further, our approach can have continuous domains, supports common and distributed knowledge, and has no limit on the depth of epistemic formula.

3. Epistemic Logic using Perspectives

In this section, we define the syntax and semantics of our *agent perspective model*, including distributed and common knowledge. We present a sound and complete semantics with an exponential time entailment operator, and a tractable version of this logic that is sound, and is complete for a wide-class of queries, including for queries able to be specified in planning languages like PDDL. The intuition behind the logic is from Big Brother Logic (Gasquet et al., 2014) and Cooper et al. (2016)'s seeing operators.

3.1 Language

Extending Cooper et al. (2016)'s idea of seeing propositional variables, our model is based on a model of functional STRIPS (F-STRIPS) (Geffner, 2000), which uses variables and domains, rather than standard propositions found in classical planning. We allow agents to see variables with discrete and continuous domains, and knowledge is derived from what variables they see.

Definition 3.1. A *signature* is a tuple $\Sigma = (Agt, V, D_{v_1}, \ldots D_{v_n}, R)$, in which Agt is a finite set of agent identifiers, V is a finite set of variables such that $Agt \subseteq V$ (agent identifiers can be used as variables), D_{v_i} is a possibly infinite domain of constant symbols, one for each variable $v_i \in V$, and R is a finite set of predicate symbols. Domains can be discrete or continuous. We define $D = \bigcup_{v \in V} D_v$.

The language $\mathcal{L}(\Sigma)$ is defined by the grammar:

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

in which $r \in R$, terms $t_1, \ldots, t_k \in V \cup D \cup Agt$, $i \in Agt$, and $v \in V$. A relation r is a k-ary propositional relation; $S_i\varphi$ is a visibility formula that means agent i sees the truth value of formula φ, $S_i v$ is a visibility formula that means that agent i sees the value of the variable v, and $K_i\varphi$ is a knowledge formula. 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. For readability, we use just $\mathcal{L}$ to represent $\mathcal{L}(\Sigma)$ for the remainder of this paper.

The important concept in this logic, adapted from Cooper et al. (2016) and (Gasquet et al., 2014), 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 knows 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. (2016).

Consider the example Big Brother Logic domain in Figure 3 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.

3.2 Semantics

In this section, we outline semantics for our logic: one sound and complete semantics that has exponential time complexity, and a sound but incomplete semantics with polynomial time complexity. The key part of the semantics is the use of states of the form $\{v_1 = e_1, \ldots, v_k = e_k\}$, rather than possible worlds found in Kripke semantics, and the use of *perspective functions* rather than Kripke relations. Perspective functions are of the form $f_i(s)$, where $i \in Agt$ and s is a state. The expression $f_i(s)$ simply specifies which variables valuation $v_i = e_k$ in s agent i is able to see. By enforcing particular constraints on the perspective functions, we get nice tractability results for the logic.

Definition 3.2. A model M is defined as $M = (Agt, V, D_{v_1}, \ldots, D_{v_k}, \pi, f_1, \ldots, f_n)$, in which Agt, V, D_{v_i} are as in Definition 3.1. A state $s : V \to D$ is a mapping from variables to values. A *global state* is a total function (a complete assignment for all variables in V), while a *local state* is a partial function (some variables may not be assigned). We use $s(v_i)$ to represent the value of v_i in s. If variable v_i is not in the domain of local state s, then $s(v_i) = null$. The set of all states (local and global) is denoted as $\mathcal{S}$, while the set of all global states is $\mathcal{S}_G \subsetneq \mathcal{S}$. The set of all models is denoted $\mathcal{M}$.

π is an interpretation function $\pi : \mathcal{S} \times R \to \{true, false\}$ that determines whether the atomic term $r(t_1, \ldots, t_n)$ is true in s. π is undefined if any of its argument t_i is a variable in V that is not also in $\mathrm{dom}(s)$.

Finally, $f_1, \ldots, f_n : \mathcal{S} \to \mathcal{S}$ 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.

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.

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 $f_i(s) \subseteq f_i(s')$

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.

Given this definition of a model, we first define the following semantics for our language, which we call the *naïve* semantics, where s is a local or global state:

$$\begin{array}{lll}
\text{(a)} & (M,s) \vDash r(t_1,\ldots,t_k) & \text{iff} \quad \pi(s, r(t_1,\ldots,t_k)) = \mathit{true} \\
\text{(b)} & (M,s) \vDash \phi \wedge \psi & \text{iff} \quad (M,s) \vDash \phi \text{ and } (M,s) \vDash \psi \\
\text{(c)} & (M,s) \vDash \neg\varphi & \text{iff} \quad (M,s) \nvDash \varphi \\
\text{(d)} & (M,s) \vDash S_i v & \text{iff} \quad v \in \mathrm{dom}(f_i(s)) \\
\text{(e)} & (M,s) \vDash S_i\varphi & \text{iff} \quad (M, f_i(s)) \vDash \varphi \text{ or } (M, f_i(s)) \vDash \neg\varphi
\end{array}$$

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.

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 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 proposition, then it cannot see the truth value of that proposition. However, this causes problems with formulae such as seeing tautologies or contradictions. For example, the formula $M, s \vDash S_i(v = e \vee \neg(v = e))$ will evaluate to false if v is not in the domain of $f_i(s)$. 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.

3.2.1 Non-naïve Semantics

To handle the issue of local states, we define the expression $g[s]$ to mean function override: $g[s](v) = s(v)$ when $v \in \mathrm{dom}(s)$ and $g(v)$ otherwise.

Using this, we extend the semantics for $S_i\varphi$ to handle non-visible variables. This new semantics states that whenever a variable v is not visible, we evaluate it against every possible global value it could hold. Formally:

$$\begin{array}{lll}
\text{(e*)} & (M,s) \vDash S_i\varphi & \text{iff} \quad \text{for all } g \in \mathcal{S}_G, (M, g[f_i(s)]) \vDash \varphi \\
& & \qquad \text{or for all } g \in \mathcal{S}_G, (M, g[f_i(s)]) \vDash \neg\varphi
\end{array}$$

The effect of evaluating φ (and $\neg\varphi$) under $g[f_i(s)]$ for every g 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 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 non-naïve 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 $\text{dom}(f_i(s))$, 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 non-naïve semantics is both sound and complete. It is straightforward to show this by simply defining Kripke structures corresponding to our models.

For each model $M = (Agt, D, D_{v1}, \ldots, D_{vk}, \pi, f_1, \ldots f_n)$, we can map to a corresponding Kripe structure $M' = (S, \pi, R_1, \ldots, R_n)$. First, we map states to worlds: each global state $g \in \mathcal{S}_G$ corresponds to a world in $\mathcal{W}$. Second, perspective functions are mapped to Kriple relations: given a perspective function $f_i(s)$, the corresponding Kripke relation R_i can be constructed by taking each global state g and its corresponding world w, and defining $(u, w) \in R_i$ for every $u \in \mathcal{W}$ such that u and w agree on all variables in $\text{dom}(f_i(s))$. Effectively, this means that for any variable $v \in f_i(g)$, all reachable worlds in $R_i(w)$ 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 $R_i(w)$ corresponds to the set of states $\{g[f_i(s)] \mid g \in \mathcal{S}_G\}$, 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 $R_i(w)$, and given the equivalence $S_i\varphi \leftrightarrow (K_i\varphi \vee K_i\neg\varphi)$, our non-naïve semantics are sound and complete.

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.

We take the concept from Levesque (1998) for reasoning about knowledge bases with incomplete information, in which propositions can be 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 (1998), we prove that the semantics are complete for a wide-class of formulae based on *logically separable formula*.

Following the notation by Levesque (1998), we define the semantics using a function $T \in (\mathcal{M} \times \mathcal{S}) \times \mathcal{L} \rightarrow \{0, 1, \frac{1}{2}\}$, which takes the knowledge base (a model and state pair (M, s) where s can be local or global) and a formula φ, and returns 1 for true, 0 for false, and $\frac{1}{2}$ for unknown. In order to systematically handling unknown seeing relation in a partial state, we request the agent identifier i be part of the state.

Function T is defined as follows, omitting the model M for readability:

$$
\begin{array}{lllll}
\text{(a)} & T[s, r(t_1, \ldots, t_k)] & = & 1 \text{ if } \pi(s, r(t_1, \ldots, t_k)) = true; \\
 & & & 0 \text{ if } \pi(s, r(t_1, \ldots, t_k)) = false; \\
 & & & \frac{1}{2} \text{ otherwise} \\
\text{(b)} & T[s, \phi \wedge \psi] & = & \min(T[s, \phi], T[s, \psi]) \\
\text{(c)} & T[s, \neg\varphi] & = & 1 - T[s, \varphi] \\
\text{(d)} & T[s, S_i v] & = & \frac{1}{2} \text{ if } i \notin \text{dom}(s) \text{ or } v \notin \text{dom}(s) \\
 & & & 0 \text{ if } v \notin \text{dom}(f_i(s)) \\
 & & & 1 \text{ otherwise} \\
\text{(e)} & T[s, S_i\varphi] & = & \frac{1}{2} \text{ if } T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2} \text{ or } i \notin \text{dom}(s); \\
 & & & 0 \text{ if } T[f_i(s), \varphi] = T[f_i(s), \neg\varphi] = \frac{1}{2}; \\
 & & & 1 \text{ otherwise}
\end{array}
$$

The definitions of (d) and (e) deserves 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. 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 $g \in \mathcal{S}_G$' in the non-naïve 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 hold. 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 non-naïve semantics; otherwise, it cannot be seen.

Definition 3.3. (Soundness and completeness, adapted from Levesque (1998).) Consider a function $h : (\mathcal{M} \times \mathcal{S}) \times \mathcal{L} \to \{0, 1, \frac{1}{2}\}$. 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 $(M, s) \vDash \phi$ and if $h[(M, s), \varphi] = 0$ then $(M, s) \vDash \neg\phi$;
- h is *complete* iff for every $M \in \mathcal{M}, s \in \mathcal{S}$, and $\varphi \in \mathcal{L}$, if $(M, s) \vDash \phi$ then $h[(M, s), \varphi] = 1$ and if $(M, s) \nvDash \phi$ then $h[(M, s), \varphi] = 0$.

Clearly the function T is incomplete compared to the non-naïve 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 3.1. Given a formula $\varphi \in \mathcal{L}$, then:

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

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 3.1. (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 $(M, s) \models \varphi$, and if $T[(M, s), \varphi] = 0$ then $(M, s) \not\models \varphi$.

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 $\models$ 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, $(M, s) \models \varphi$ and $(M, s) \models \psi$. Therefore, from the definition of $\models$, we have that $(M, s) \models \varphi \wedge \psi$.

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, $(M, s) \not\models \varphi$ or $(M, s) \not\models \psi$. Therefore, from the definition of $\models$, we have that $(M, s) \not\models \varphi \wedge \psi$. 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, $(M, s) \not\models \varphi$ and therefore from the definition of $\models$ we have that $(M, s) \models \neg\varphi$. 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 $\models$. 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 3.1, this implies that for all $g \in \mathcal{S}_G$, $T[g[f_i(s)], \varphi] \in \{0, 1\}$ or for all $g \in \mathcal{S}_G$, $T[g[f_i(s)], \neg\varphi] \in \{0, 1\}$. By induction, this means that for all $g \in \mathcal{S}_G$, either $(M, g[f_i(s)]) \models \varphi$ or $(M, g[f_i(s)]) \models \neg\varphi$. Therefore, from the definition of $\models$, we have that $(M, s) \models S_i\varphi$.

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 3.1, it must be that there exists $g \in \mathcal{S}_G$, such that $T[g[f_i(s)], \varphi] \in \{0, 1\}$ and there exists $g \in \mathcal{S}_G$, such that $T[g[f_i(s)], \neg\varphi] \in \{0, 1\}$. By induction, this means that there exists $g \in \mathcal{S}_G$, such that $(M, g[f_i(s)]) \models \varphi$ and there exists $g \in \mathcal{S}_G$, $(M, g[f_i(s)]) \models \neg\varphi$. Therefore, from the definition of $\models$, we have that $(M, s) \not\models S_i\varphi$. □

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

Definition 3.4. (Logical separability) Adapted from Levesque (1998), 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 3.5. (Normal form $\mathcal{NF}$, adapted from (Lakemeyer & Lespérance, 2012)) 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 gramma:

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

where the set $\{\psi, \psi'\}$ 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 3.2. (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 $g \in \mathcal{S}_G$,

$(M, g[f_i(s)]) \models \varphi$ or for all $g \in \mathcal{S}_G$, $(M, g[f_i(s)]) \models \neg\varphi$. By induction and $\varphi \in \mathcal{NF}$, we have that for all $g \in \mathcal{S}_G$, $T[g[f_i(s)], \varphi] = 1$ or for all $g \in \mathcal{S}_G$, $T[g[f_i(s)], \neg\varphi] = 1$. If one of these two expressions hold for all $g \in \mathcal{S}_G$, then they must also hold for all $g[s]$ 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 $(M, s) \nvDash S_i\varphi$. From the definition of $\models$, we have that there exists $g \in \mathcal{S}_G$, such that $(M, g[f_i(s)]) \models \varphi$ and there exists $g \in \mathcal{S}_G$, such that $(M, g[f_i(s)]) \models \neg\varphi$. By induction and $\varphi \in \mathcal{NF}$, we have that there exists $g \in \mathcal{S}_G$, such that $T[g[f_i(s)], \varphi] = 1$ and there exists $g \in \mathcal{S}_G$, such that $T[g[f_i(s)], \neg\varphi] = 1$. From Lemma 3.1 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 3.6. (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 3.3. (Soundness and completeness of T in global states). Let $M \in \mathcal{M}$ be a model, $g \in \mathcal{S}_G$ be global state, and $\varphi \in \mathcal{L}$ be a formula in $\mathcal{NF}^+$. Then, if $T[(M, g), \varphi] = 1$ then $(M, s) \models \varphi$, and if $T[(M, g), \varphi] = 0$ then $(M, s) \nvDash \varphi$; and if $(M, s) \models \varphi$ then $T[(M, s), \varphi] = 1$, and if $(M, s) \nvDash \varphi$ then $T[(M, s), \varphi] = 0$.

Proof. This is a small extension to the proofs of Theorems 3.1 and 3.2, 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 3.1 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 3.3, 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 an planning task is equivalent with the completeness and soundness of our ternary semantics.

3.4 Group Knowledge

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

3.4.1 Language

We extend our language with group operators:

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

in which G is a set (group) of agents, and α[2] is a variable v or formula φ.

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

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

$$\begin{array}{lcll} EK_G\varphi & \leftrightarrow & \varphi \wedge ES_G\varphi & \leftrightarrow \quad \bigwedge_{i\in G} K_i\varphi \\ DK_G\varphi & \leftrightarrow & \varphi \wedge DS_G\varphi & \\ CK_G\varphi & \leftrightarrow & \varphi \wedge CS_G\varphi. & \end{array}$$

As such, we define the semantics only for the remaining operators.

3.4.2 Non-naïve Semantics of Group Knowledge

Given a model M and state s, we extend our non-naïve semantics to include the group operators as follows:

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

(g) $(M,s) \vDash DS_G v$ iff $v \in dom(\bigcup_{i\in G} f_i(s))$

(h) $(M,s) \vDash DS_G\varphi$ iff for all $g \in \mathcal{S}_G$, $(M, g[s']) \vDash \varphi$
or for all $g \in \mathcal{S}_G$, $(M, g[s']) \vDash \neg\varphi$, where $s' = \bigcup_{i\in G} f_i(s)$

(i) $(M,s) \vDash CS_G v$ iff $v \in dom(cf(G,s))$

(j) $(M,s) \vDash CS_G\varphi$ iff for all $g \in \mathcal{S}_G$, $(M, g[s']) \vDash \varphi$
or for all $g \in \mathcal{S}_G$, $(M, g[s']) \vDash \neg\varphi$, where $s' = cf(G,s)$.

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

3.4.3 Ternary Semantics of Group Knowledge

From the non-naïve semantics, it is straightforward to extend function T of the ternary semantics to handle group knowledge, given agent identifier i is part of the state:

2. We use α for simplicity

$$
\begin{array}{lrcl}
\text{(f)} & T[s, ES_G\alpha] & = & \min(\{T[s, S_i\alpha] \mid i \in G\}) \\
\text{(g)} & T[s, DS_G v] & = & \frac{1}{2} \text{ if } v \not\subseteq \text{dom}(s) \text{ or } \forall i \in G,\ i \notin \text{dom}(s); \\
 & & & 0 \text{ if } v \notin \text{dom}(\bigcup_{i \in G} f_i(s)); \\
 & & & 1 \text{ otherwise} \\
\text{(h)} & T[s, DS_G\varphi] & = & \frac{1}{2} \text{ if } T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2} \text{ or } \forall i \in G,\ i \notin \text{dom}(s) \\
 & & & 0 \text{ if } T[s', \varphi] = T[s', \neg\varphi] = \frac{1}{2}, \text{ where } s' = \bigcup_{i \in G} f_i(s); \\
 & & & 1 \text{ otherwise} \\
\text{(i)} & T[s, CS_G v] & = & \frac{1}{2} \text{ if } v \not\subseteq \text{dom}(s) \text{ or } \exists i \in G,\ i \notin \text{dom}(s); \\
 & & & 0 \text{ if } v \notin \text{dom}(cf(G, s)); \\
 & & & 1 \text{ otherwise} \\
\text{(j)} & T[s, CS_G\varphi] & = & \frac{1}{2} \text{ if } T[s, \varphi] = T[s, \neg\varphi] = \frac{1}{2} \text{ or } \exists i \in G,\ i \notin \text{dom}(s); \\
 & & & 0 \text{ if } T[s', \varphi] = T[s', \neg\varphi] = \frac{1}{2}, \text{ where } s' = cf(G, s); \\
 & & & 1 \text{ otherwise}
\end{array}
$$

It is straightforward to see that the complexity of T with these new definitions is still $\Theta(n \times |\varphi|)$.

Theorem 3.4. (Soundness and completeness of T for group operators) Let $M \in \mathcal{M}$ be a model, $g \in \mathcal{S}_G$ be global state, and $\varphi \in \mathcal{L}$ be a formula in $\mathcal{NF}^+$. Then, if $T[(M, g), \varphi] = 1$ then $(M, s) \models \varphi$, and if $T[(M, g), \varphi] = 0$ then $(M, s) \not\models \varphi$; and if $(M, s) \models \varphi$ then $T[(M, s), \varphi] = 1$, and if $(M, s) \not\models \varphi$ 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 $i \in G$, $(M, s) \models S_i\alpha$. Therefore, from the definition $\models$, we have that $(M, s) \models ES_G\alpha$.

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 $(M, s) \not\models S_i\alpha$ for some $i \in G$. Therefore, from the definition of $\models$, we have that $(M, s) \not\models ES_G\alpha$.

Completeness, case (f): Assume $(M, s) \models ES_G\alpha$. From the definition of $\models$, this means that for all $i \in G$, $(M, s) \models S_i\alpha$. 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 $(M, s) \not\models ES_G\alpha$. From the definition of $\models$, this means that there exists an $i \in G$, such that $(M, s) \not\models S_i\alpha$. 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 3.1, 3.2, and 3.3. 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. □

Reasoning about common knowledge and common visibility is more complex than other modalities. Common knowledge amongst 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.

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 $\bigcap_{i \in G} f_i(s)$, the resulting state is either a proper subset of s or is s. 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 3.5. Function $cf(G, s)$ converges on 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 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.

3.5 Example Logics

Our logic is expressive enough to represent several well-known epistemic logics, including epistemic logic based on Kripke semantics, as we show in this section. For simplicity, we note the variable and its value as a tuple (v, e) in this session.

Example 3.1. (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 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 $\forall w \in \text{dom}(s)$, $r(t_1, \ldots, t_k))$ 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. We show an example of this for the Muddy Children problem in Section 4.3.

The reader may have noted that 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 instead use a state s to represent the set of *impossible* worlds. So, a global state g would contain all variables except those representing the current world w in Kripke semantics. Then, $f_i(s)$ returns the set of impossible variables according to agent i, and the evaluation function is defined as $\forall w \notin \text{dom}(s)$, $r(t_1, \ldots, t_k))$ holds (note the $\in$ replaced by $\notin$).

For example, given a state in our original model $s = \{x = 1\}$ and $dom(x) = \{1, 2, 3\}$. The inversed state for Kripke semantics s' would be $\{nx2, nx3\}$.Then, given agent a and agent b's perspectives are $\{nx2\}$ and $\{nx3\}$ respectively, their distribute knowledge would still be the union $\{nx2, nx3\}$.

Example 3.2. (Proper Epistemic Knowledge Bases (PEKBs) and Cooper et al. (2016)'s seeing logic)

PEKBs (Muise et al., 2015b; Lakemeyer & Lespérance, 2012) and Cooper et al. (2016)'s seeing logic are closely related, and our logic can represent both using the same representation as Cooper et al. (2016)'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. The domain is not relevant. The perspective function $f_i(s) = \{(\alpha, e) \mid (S_i\alpha, e) \in s\}$. The evaluation function $\pi(s)(p)$ is true if and only if $p \in \text{dom}(s)$.

For Cooper, Herzig, Maffre, Maris, Perrotin, and Régnier (2021)'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 as Cooper et al. (2021) by adding a variable $JS\alpha$ for each literal α. We then define $f_i(s) = \{(\alpha, e) \mid (S_i\alpha, e) \in s \vee JS\alpha \in s\}$.

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 $\{p, S_ip, S_jp, S_iS_ip, S_iS_jp, S_jS_ip, S_jS_jp, JSp\}$.

In their PEKB-based planner, Muise et al. (2015a) 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:

$$f_a(s) = \{v \mid a \triangleright v, v \in s\} \cup \{v_1, \ldots, v_n\}$$

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.

Example 3.3. (Big Brother Logic (BBL) (Gasquet et al., 2014)) 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:

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

where $i \triangleright v$ is defined as in Section 2.2.1 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:

$$\begin{array}{c} \left(|\arctan(\frac{|s(y_i)-s(y_j)|}{s(x_i)-s(x_j)}) - s(dir_i)| \leq \frac{s(ang_i)}{2} \right) \\ \vee \\ \left(|\arctan(\frac{|s(y_i)-s(y_j)|}{s(x_i)-s(x_j)}) - s(dir_i)| \geq \frac{360^\circ - s(ang_i)}{2} \right) \end{array} \quad (1)$$

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 predicate $v \in \{x_j, y_j, ang_j\}$ captures that the locations and angles 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 next 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 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 difficult for a reader to understand. However, implementing the above in C++ is straightforward to implement and straightforward for a reader.

4. Epistemic Planning with Perspectives

In this section, we define our model of *planning with perspectives* (PWP), encode it within a planner, and solve some well-known epistemic planning benchmarks. Two key aspects in planning are the planning language and solver (planner). We use F-STRIPS as the modelling language, and BFWS(R_0) (Francès et al., 2017) as the planner. Using F-STRIPS with external functions allows us to decompose the planning task from the epistemic logic reasoning.

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.

4.1 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.

As defined in Section 3, an epistemic logic model is defined as $M = (Agt, V, D_{v_1}, \ldots, D_{v_k}, \pi, f_1, \ldots, f_n)$. 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; $D = D_{v_1} \cup \ldots \cup D_{v_n}$ 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, conditions, 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* $?v_1$ ...$?v_n$ $?q$), 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 paper, omitting the variables. In the implementation, the modeller decides which variables are needed as the input to the external functions.

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 $\mathcal{G}$. These formulae are evaluated using the semantics in Section 3.

Defining preconditions and goals with desirable epistemic formulae is straightforward. For example, in Figure 3, 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 Example 3.2 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.

4.2 Perspective 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 3.2 that each agent i has a perspective function, $f_i : S \to S$, which takes a state and returns the local state from the perspective of agent i. In most problems we have modelled, 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 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 is evaluated, the planner calls the related epistemic logic external function. Instead of generating all truth values at the pre-compilation phase or storing entire knowledge structures, the planner evaluates epistemic queries lazily. In other words, the epistemic logic reasoning task is moved from the planner to the external func-

tions. The underlying planner has no concept of epistemic logic, and simply uses its search algorithm to find the goal.

Despite having general representations such as PEKBs (see Example 3.2), in our experience, using domain-specific perspective function results in shorter, more elegant models that are more straightforward to specify and verify. We give the external function for BBL domain here, and several examples in Section 5.

Example 4.1. External function for BBL domain

In Figure 3, 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 $f_{a_2}(l)$.

To be more specific about perspective functions, assume the global state s in the BBL example contains all variables for $\{a_1, a_2, b_1, b_2, b_3, b_4\}$[3], 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 1 in Example 3.3. By applying this perspective function on all variables in the given state (could be a local state when evaluate 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.

4.3 Expressiveness

The intuitive idea about perspective functions is based on what agents can see, as determined by the current state. The relation between $t = f_i(s)$ and s corresponds roughly to Kripke accessibility relations $(s, t) \in \mathcal{R}_i$. Perspective functions return one partial world that the agent i is certain about, rather than a set of worlds that i considers possible. The advantage is that applying a perspective function provides us with only one state, rather than multiple states in Kripke semantics, preventing the explosion in model size.

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 $\bigcap_{t \in W}(s, t) \in \mathcal{R}_i$ from Kripke semantics. This eliminates disjunctive knowledge about variables; the only uncertainty being that an agent does not

3. $a_1, a_2, b_1, b_2, b_3, b_4$ 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 $x\_a_1, y\_a_1, dir\_a_1, ang\_a_1$.

see a variable. For example, in the well-known *Muddy Children*[4] problem (Fagin et al., 2003), the knowledge is not only generated by what each child can see by the others' appearance, which is modelled straightforwardly using perspective functions, but also can be derived from the questions made by their parent 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 parent 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 parent for the kth time will result in removal of m_k from the state until all the children only have m_m in their local state. This is similar to how Kripke semantics are encoded in Example 3.1, but also show that being able to customise perspective functions to specific domains can be useful.

A comparison on expressiveness between our model and other approaches is given in Table 1.

	Nested	Depth	CK	DK	Continuous Domains	Disjunctive Knowledge
PWP Model	Y	U	Y	Y	Y	Y/N
Muise et al. (2015a)	Y	B	N	N	N	N
Kominis and Geffner (2015)	Y	B	N	N	N	Y
Huang et al. (2017)	Y	U	I[5]	N	N	Y
Le et al. (2018)	Y	U	I[5]	N	N	Y

Table 1: 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 (Le et al., 2018), or using a group of nested knowledge to approximate common knowledge (Huang et al., 2017). For depth, 'U' means no bound on the depth of queries, while 'B' means there is a fixed bound.

There are four major points of difference that we identify: (1) Our perspectives model can handle domains in which the depth of epistemic relations are unbounded. Each level of

4. There are n children, and m of them with mud on their forehead. They can all see each others' foreheads, but not their own. To help them find out whether themselves are muddy or not, their parent can help them by asking one question to them all for k times: do you know whether you are muddy or not?

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 Kominis and Geffner (2015), Le et al. (2018) keep 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. (2015a)'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 logical reasoning and problem description. 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 functional STRIPS planners that support continuous variables (Ramirez, Papasimeon, Lipovetzky, Benke, Miller, Pearce, Scala, & Zamani, 2018). (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 expressiveness, such as the support for continuous variables.

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

5. Experiments & Results

In this section, we evaluate our approach on several problems: *Corridor* (Kominis & Geffner, 2015), *Grapevine* (Muise et al., 2015a), *Big Brother Logic (BBL)* (Gasquet et al., 2014), *Social-media Network* (*SN*) and *Gossip* (Baker & Shostak, 1972). Corridor, Grapevine, and Gossip are well-known epistemic planning problems, which we use to compare 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, modelling information sharing over a digital social network platform. PWP has an advantage on 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`.

5.1 Benchmark Problems

In this section, we briefly describe the corridor and grapevine problems, which are benchmark problems that we use to compare against Muise et al. (2015a)'s epistemic planner, and is currently the state-of-the-art in epistemic planning.

Corridor The corridor problem was originally presented by Kominis and Geffner (2015). It models selective communication among agents. The basic setup is in a corridor of rooms, in which there are several agents. An agent is able to move around adjacent rooms, sense the secret in a room, and share the secret. The rule of communication is that when an agent shares the secret, all the agents in the same room or adjacent rooms 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 Grapevine, proposed by Muise et al. (2015a), is a similar problem to Corridor. With only two rooms available for agents, the scenario makes sharing secrets while hiding from others more difficult. The basic setup is each agent has their own secret, and they can share their secret among everyone in the same room. Since there are only two rooms, the secret is only shared within the room. The basic actions for agents are: moving between rooms and sharing their secrets. The emphasis of this domain is on sharing one's secret to 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.

5.1.1 Encoding

Both the Corridor and Grapevine domains are modeled similar 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 indicator 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 Grapevine domain '**{"query_info":{"id":"p1","query":"ck 1,2 sct_1:value:2"}}**' represents $CK_{1,2}$ sct_1=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 assigns the truth value to each query (goal) `?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 into this format.

5.1.2 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 are similar. The visibility of secrets for both domains depends on the location of the agent whose perspective is modelled. Therefore, both rules take the location of the speaking agent and the hearing agent, and returns 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:

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

5.1.3 Results

To evaluate computational performance of PWP, we compare to Muise et al. (2015a)'s PDKB planner. Their planner has been used to compare on Corridor and Grapevine domains against many others' solutions (Kominis & Geffner, 2015; Huang et al., 2017; Le et al., 2018). From their results and results from Huang et al. (2017) and Le et al. (2018), it is fair to say that PDKB is a state-of-the-art planner. 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. (2015a), the *FF* planner.

We ran the problems with both planners on a Linux machine with 8 CPUs (Intel Core i7-7700K CPU @ 4.20GHz × 8) and 16 gigabytes of memory. We measure the number of atoms (fluents) and the number of nodes generated during the search to compare the size of the same problem modelled 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 2 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

Parameters			PWP					PDKB			
						TIME(s)				TIME(s)	
$\|Agt\|$	d	$\|\mathcal{G}\|$	$\|Atom\|$	$\|Gen\|$	$\|Calls\|$	Calls	Total	$\|Atom\|$	$\|Gen\|$	Compile	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 2: Results for the Corridor and Grapevine Problems

number of goals, $|Atom|$ the number of atomic fluents, $|Gen|$ the number of generated nodes in the search, and $|Calls|$ the number of calls made to external functions. The symbol "–" represents there is no result within a 10-minute time limit. In the Grapevine tests, to eliminate any influence from the different length 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 the search nodes generated and the number of the 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 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 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 exceed even half a second, while for the PDKB approach, many do, some running for close to a minute.

5.2 Big Brother Logic

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

5.2.1 Encoding

Figure 4 visualises 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.

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 : $D(x)$=$D(y)$=$\{-20, \ldots, 20\}$, $D(dir)$=$\{-180, \ldots, 180\}$, and $D(q)$=$\{0, 1\}$, where $q \in Q$;
- $\mathcal{O}$: **move**(dx, dy) and **turn**(d), where $dx, dy \in \{-2, \ldots, 2\}$ and $d \in \{-45, \ldots, 45\}$;
- $\mathcal{I} = [x = 5,\ y = 5,\ dir = 45]$;
- $\mathcal{G} = \{q = 1\}$; and
- $\mathbb{F}$: (**@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.

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 $|V|$).

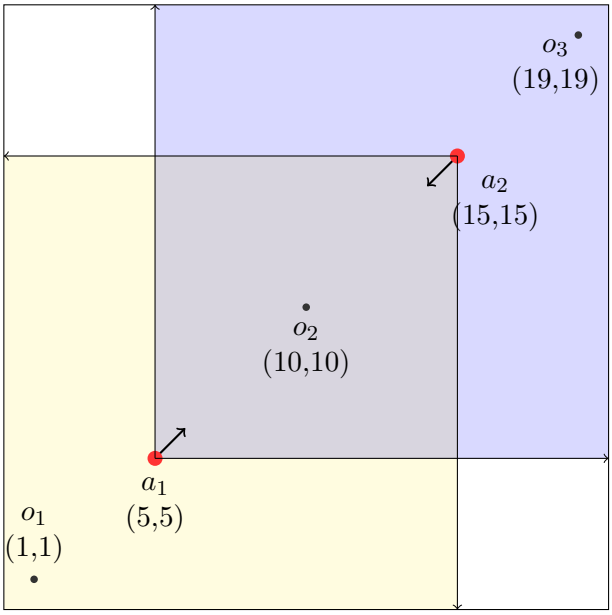

Figure 4: Example for Big Brother Logic setup

We need to check the knowledge queries in the actions (precondition, conditions), or goals. Both action **move**(dx, dy) and action **turn**(d) can change all of agents' perspectives, and therefore, can influence knowledge.

5.2.2 External Functions

Inputs to the external functions would be the query (in the format of our language described in Section 3) 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 Example 3.3, except that because the angle and position of all agents except a_1 are known, it can be simplified to just:

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

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.

5.2.3 Goal Conditions

As for the goal conditions, some queries q can be achieved for the problem in Figure 4 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: $K_{a_1} o_3{=}e_3 \wedge \neg K_{a_2} o_3{=}e_3$
2. Nested Knowledge query: $S_{a_1} S_{a_2} o_3 \wedge \neg K_{a_1} S_{a_2} o_3$
3. Group Knowledge query: $EK_{a_1,a_2} o_2{=}e_2 \wedge \neg EK_{a_1,a_2} o_3{=}e_3$
4. Distributed Knowledge query: $DK_{a_1,a_2} o_1{=}e_1 \wedge \neg DK_{a_1,a_2} o_1{=}e_3$
5. Common Knowledge query: $CK_{a_1,a_2} o_2{=}e_2 \wedge CK_{a_1,a_2} S_{a_1} o_3$

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. $S_{a_1} o_3 \wedge \neg S_{a_2} S_{a_1} o_3$: **move**$(-2,1)$, **move**$(-2,2)$, **move**$(-2,2)$, **move**$(-2,2)$, **move**$(-2,2)$, **move**$(-2,2)$
5. $\neg K_{a_1} S_{a_2} S_{a_1} o_3 \wedge S_{a_1} o_3$: **move**$(-2,1)$, **move**$(-2,2)$, **move**$(-2,2)$, **move**$(-2,2)$, **move**$(-2,2)$, **move**$(-2,2)$, **turn**$(-45)$

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 are not straightforward. To avoid a_2 seeing whether a_1 can see o_1, the cheapest plan returned by planner was for a_1 to move out of a_2's eye sight. The last one is the most difficult to solve. Not only should a_1 see o_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".

5.2.4 Results

Table 3 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. A plan length of "∞" means that the problem is unsolvable – no plan exists. While the perspective function in BBL depends on a *geometric model* based on agent's position, direction and facing angle, the results show that with proper usage of our F-STRIPS planner, we can represent continuous domains. Our epistemic solver is able to reason about other 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. This demonstrates that our PWP model can handle important problems in vision-based domains. 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 a great potential for application in hybrid-planning (Ramirez et al., 2018).

	Parameters				Performance					
	$\|Agt\|$	d	$\|\mathcal{G}\|$	$\|P\|$	$\|Gen\|$	$\|Exp\|$	$\|Calls\|$	TIME(s) *calls*	TIME(s) Total	Goal
BBL01	2	1	1	0	1	0	2	0.000	0.001	$K_{a_1}o_2$
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: Experimental Results for PWP on the BBL domain

5.3 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 friend's page and view their friend list. We extend two-way one-time communication channels from a classical gossip problem (Cooper et al., 2016) into two-way, all-time communication channels, and add the concept of secret messages. By decomposing secrets into messages and posting through an agent's friendship network, we model how secrets can be shared between a group of individuals not directly connected without anyone else on the network knowing the secret, and some secrets can be shared within a group excepting for some individuals. The former could be spies sharing information with each other through the resistance's personal page, and the latter could be a group arranging a surprise party for a mutual friend.

5.3.1 Example and Encoding in F-STRIPS

Let a, b, c, d, e be five agents in the SN, with friendship links shown in Figure 5. Their friend relations are represented by full lines between each agent. The dotted lines are referenced later for illustration purposes.

Let g be a friend of all agents that wants to share a secret. We assume the social network is in g's perspective directly, and the network is fixed. Let the epistemic queries be the set

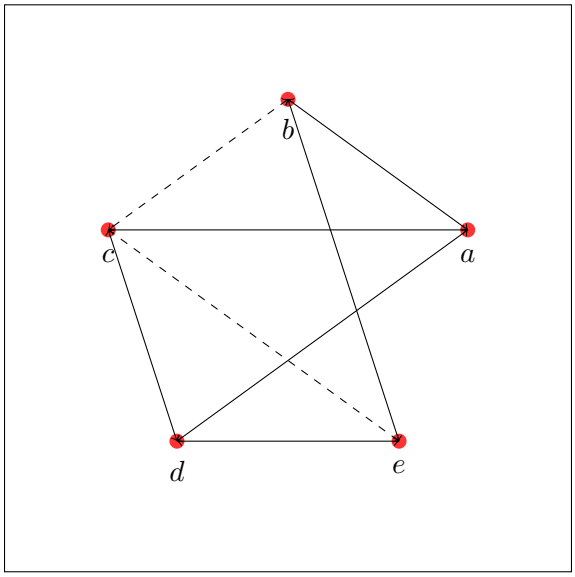

Figure 5: Example for Social-media Network

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 $i, j \in A,\ p \in P,\ q \in Q$
- $\mathcal{O}$: $\mathbf{post}(i, p)$, 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
- $\mathbb{F}$: $(\mathbf{@check}\ q) \mapsto \{\mathit{true}, \mathit{false}\}$

The variable $(\mathit{friended}\ i\ j)$ represents whether i and j are friends with each other. Action $(\mathbf{post}\ i\ p)$ specifies that the message p is posted on agent i's page. The initial state $\mathcal{I}$ represents the friendship relations in Figure 5, with no message posted yet. Similarly, the action **post** is the only source for epistemic relation changes.

5.3.2 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 "$K_a K_c p$".

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 5. 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++.

5.3.3 Goal Conditions

Goals that we tested are shown in Table 4. For some epistemic formulae between a and b, since they are friends, simply posting the message in 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, 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 organising 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 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 5, 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 4), the planner returns with the plan: **post**(e,p_1), **post**(e,p_2), **post**(e,p_3).

5.3.4 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	$K_i Ps \wedge \neg K_j Ps$
SN10	5	1	3	3	565	175	37644	2.484	2.530	$K_i Ps \wedge \neg K_{j,k} Ps$
SN11	5	1	5	3	816	251	90100	5.720	5.810	$K_i Ps \wedge \neg K_{other} Ps$
SN12	5	1	5	∞	2160	*all*	777600	53.191	54.004	$K_i Ps \wedge \neg K_{other} Ps$
SN13	5	1	2	∞	432	*all*	62208	9.809	9.895	$\neg K_i Ps \wedge K_{other} Ps$
SN14	5	1	2	3	216	92	13144	2.436	2.454	$\neg K_i Ps \wedge K_{other} Ps$

Table 4: Experiment Results for PWP on the Social-media Network domain.

Table 4 shows the result for our problems in the social-media network domain, where Ps represents $(p_1 \wedge p_2 \wedge p_3)$ 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 correlates with each other as well as total time.

5.4 Gossip

The Gossip problem is a canonical epistemic planning problem proposed by Baker and Shostak (1972). 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 knows that everyone knows all the secrets.

Different from the previous epistemic planning problems we experiment with in this paper, the knowledge generated in the gossip problem depends on the current knowledge of each agents, 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 Example 3.2, and a novel encoding based on actions.

5.4.1 State-based Approach Encoding

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

- $V = \{I_s\}$
- $D : dom(I_s) = \{0, \ldots, 2^{|Agt|^{|d|+1}} - 1\}$
- $\mathcal{O}$: **call**(x, y), where $x, y \in Agt$
- $\mathcal{I}$ = discussed below
- $\mathcal{G}$ = discussed below
- $\mathbb{F}$ = (**@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 Example 3.2. I_s represents 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. (2015a). Therefore, the external

function would update the state based on the current state and the current action. Consider the example of update in Table 5, where the number of agents is 3 and the depth is 2, and I_s represents 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	10000000000001000000000001	67117057
1	**call**(a,b)	11011000011011000000000001	113467393
2	**call**(a,c)	11111011111011000011110111	132080119
3	**call**(b,c)	11111011111111111111111111	132120575
4	**call**(a,b)	11111111111111111111111111	134217727

Table 5: 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. (2015b). 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.

5.4.2 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 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

- $V = \{A_s\}$
- $D : dom(A_s) = \{0, \ldots, (|Agt| \cdot (|Agt| - 1)/2)^{|p|} - 1\}$
- $\mathcal{O}$: **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

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, Herzig, Maffre, Maris, and Régnier (2019)'s proof.

node. Since Cooper et al. (2019) prove 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 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 with 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 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. (2019)'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.

5.4.3 Results

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.

From the Table 6, 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

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

	Parameters			State			Action (full)			Action (relative)			Gossip Generator		
					TIME(s)			TIME(s)			TIME(s)			TIME(s)	
	d	$\|g\|$	$\|p\|$	$\|calls\|$	*calls*	Total	$\|calls\|$	*calls*	Total	$\|calls\|$	*calls*	Total	$\|p\|$	*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 6: Experiment 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.

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. (2019)'s approach, our approach has similar performance on the problems with full goals, and perform slightly better on the problem with single goals, as it will not generate irrelevant epistemic relations.

5.5 Discussion

Overall, the experiment results show that our solution outperforms Muise et al. (2015a)'s encoding solution, which is the state-of-the-art for epistemic planning problems. As it can be seen from the results in both *Corridor* and *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 *BBL* and *SN* domains. In addition, even for the problem with epistemic relations embedded in the state, such as *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. Moreover, the amount of nodes involved in the search is affected by standard factors in search, such as plan length, the algorithm used by the planner, and the difficulty of the problem itself.

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

6. Conclusions

In this work, we introduce a new epistemic planning model called Planning with Preferences (PWP), driven by the intuition: 'seeing is believing'. The PWP model can evaluate epistemic formulae, even nested, distributed or common epistemic formulae, based on the simple concept of defining an agent's perspective. By separating the planning task from epistemic reasoning with F-STRIPS, we propose an expressive and flexible solution for most of the epistemic planning problems without an expensive pre-compilation step. We implement our model on well-known epistemic planning benchmarks and two new scenarios. The results not only show that our model can solve the epistemic benchmarks efficiently, but also demonstrate the variety of epistemic relations that can be handled. Our work is the first to delegate epistemic reasoning to an external solver.

For future work, there are three ways to extend our model. The first is to extend the model to belief rather than knowledge, which would increase the expressiveness of PWP. 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. However, to extend PWP to handle belief, including false belief, instead of only tracking the visible variables and their values (which are always the true values), we need to reason about the possible values for any variable (both visible and invisible) for any agent, including those that could be incorrect. Therefore, property (1) of perspective functions would not hold, which means we no longer get the nice properties of nested modal operators. Our intuition, which is similar to Scherl and Levesque (2003)'s work, is to maintain the agents' current knowledge as belief until it becomes updated. Second, we can improve our model by allowing simplified disjunctive knowledge in the state. This would allow PWP to support partial information on variables, such as $v = e_1 \vee v = e_2$ in the state of the world. Finally, investigating the relationship between event models in dynamic epistemic logic and our external functions is another direction for future work. Instead of seeing variables and evaluating epistemic relations

based on those variables, one could allow agents to 'seeing' actions and gain knowledge from observing actions, which would lead us to a different perspective on decomposing and solving epistemic planning problems.

Acknowledgements

The authors thank Andreas Herzig for his insightful discussions on the link between knowledge and seeing, and for inspiring the idea of the social network domain.

References

Bajada, J., Fox, M., & Long, D. (2015). Temporal planning with semantic attachment of non-linear monotonic continuous behaviours. In Yang, Q., & Wooldridge, M. J. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 1523–1529. AAAI Press.

Baker, B. S., & Shostak, R. E. (1972). Gossips and telephones. *Discrete Mathematics*, *2*(3), 191–193.

Baral, C., Gelfond, G., Pontelli, E., & Son, T. C. (2015). An action language for multi-agent domains: Foundations. *CoRR*, *abs/1511.01960*.

Bellemare, M., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents.. *JAIR*, *47*.

Bolander, T. (2014). Seeing is believing: Formalising false-belief tasks in dynamic epistemic logic. In *Proceedings of the European Conference on Social Intelligence (ECSI-2014), Barcelona, Spain, November 3-5, 2014.*, pp. 87–107.

Bolander, T. (2017). A gentle introduction to epistemic planning: The DEL approach. In *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017.*, pp. 1–22.

Bolander, T., & Andersen, M. B. (2011). Epistemic planning for single and multi-agent systems. *Journal of Applied Non-Classical Logics*, *21*(1), 9–34.

Bolander, T., Jensen, M. H., & Schwarzentruber, F. (2015). Complexity results in epistemic planning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 2791–2797.

Cooper, M. C., Herzig, A., Maffre, F., Maris, F., Perrotin, E., & Régnier, P. (2021). A lightweight epistemic logic and its application to planning. *Artif. Intell.*, *298*, 103437.

Cooper, M. C., Herzig, A., Maffre, F., Maris, F., & Régnier, P. (2016). A simple account of multi-agent epistemic planning. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, pp. 193–201.

Cooper, M. C., Herzig, A., Maffre, F., Maris, F., & Régnier, P. (2019). The epistemic gossip problem. *Discrete Mathematics*, *342*(3), 654–663.

Cooper, M. C., Herzig, A., Maris, F., Perrotin, E., & Vianey, J. (2020). Lightweight parallel multi-agent epistemic planning. In Calvanese, D., Erdem, E., & Thielscher, M. (Eds.), *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pp. 274–283.

Dornhege, C., Eyerich, P., Keller, T., Trüg, S., Brenner, M., & Nebel, B. (2009a). Semantic attachments for domain-independent planning systems. In *Proc. ICAPS*.

Dornhege, C., Eyerich, P., Keller, T., Trüg, S., Brenner, M., & Nebel, B. (2009b). Semantic attachments for domain-independent planning systems. In Gerevini, A., Howe, A. E., Cesta, A., & Refanidis, I. (Eds.), *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI.

Dornhege, C., Gissler, M., Teschner, M., & Nebel, B. (2009c). Integrating symbolic and geometric planning for mobile manipulation. In *2009 IEEE International Workshop on Safety, Security Rescue Robotics (SSRR 2009)*, pp. 1–6.

Engesser, T., Bolander, T., Mattmüller, R., & Nebel, B. (2017). Cooperative epistemic multi-agent planning for implicit coordination. In *Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017.*, pp. 75–90.

Fabiano, F., Burigana, A., Dovier, A., & Pontelli, E. (2020). EFP 2.0: A multi-agent epistemic solver with multiple e-state representations. In Beck, J. C., Buffet, O., Hoffmann, J., Karpas, E., & Sohrabi, S. (Eds.), *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, pp. 101–109. AAAI Press.

Fagin, R., Halpern, J. Y., Moses, Y., & Vardi, M. Y. (2003). *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA.

Fan, J., Wang, Y., & van Ditmarsch, H. (2015). Contingency and knowing whether. *Rew. Symb. Logic*, *8*(1), 75–107.

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

Frances, G., & Geffner, H. (2015). Modeling and computation in planning: Better heuristics for more expressive languages. In *Proc. ICAPS*.

Francès, G., Ramírez, M., Lipovetzky, N., & Geffner, H. (2017). Purely declarative action descriptions are overrated: Classical planning with simulators. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 4294–4301.

Gaschler, A., Petrick, R. P. A., Khatib, O., & Knoll, A. C. (2018). Kaboum: Knowledge-level action and bounding geometry motion planner. *J. Artif. Intell. Res.*, *61*, 323–362.

Gasquet, O., Goranko, V., & Schwarzentruber, F. (2014). Big brother logic: logical modeling and reasoning about agents equipped with surveillance cameras in the plane. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pp. 325–332.

Geffner, H. (2000). *Functional Strips: A More Flexible Language for Planning and Problem Solving*, pp. 187–209. Springer US, Boston, MA.

Geffner, H., & Bonet, B. (2013). *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Gregory, P., Long, D., Fox, M., & Beck, J. C. (2012). Planning modulo theories: Extending the planning paradigm. In McCluskey, L., Williams, B. C., Silva, J. R., & Bonet, B. (Eds.), *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI.

Hales, J., French, T., & Davies, R. (2012). Refinement quantified logics of knowledge and belief for multiple agents. *Advances in Modal Logic*, *9*, 317–338.

Haslum, P., Lipovetzky, N., Magazzeni, D., & Muise, C. (2019). *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*.

Herzig, A., & Maffre, F. (2015). How to share knowledge by gossiping. In *Multi-Agent Systems and Agreement Technologies - 13th European Conference, EUMAS 2015, and Third International Conference, AT 2015, Athens, Greece, December 17-18, 2015, Revised Selected Papers*, pp. 249–263.

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

Huang, X., Fang, B., Wan, H., & Liu, Y. (2017). A general multi-agent epistemic planner based on higher-order belief change. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 1093–1101.

Kaelbling, L. P., & Lozano-Pérez, T. (2012). 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*, pp. 2952–2959. IEEE.

Kominis, F., & Geffner, H. (2015). Beliefs in multiagent planning: From one agent to many. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, pp. 147–155.

Kominis, F., & Geffner, H. (2017). Multiagent online planning with nested beliefs and dialogue. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017.*, pp. 186–194.

Lakemeyer, G., & Lespérance, Y. (2012). 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*, pp. 498–503.

Le, T., Fabiano, F., Son, T. C., & Pontelli, E. (2018). EFP and PG-EFP: epistemic forward search planners in multi-agent domains. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018.*, pp. 161–170.

Levesque, H. J. (1996). What is planning in the presence of sensing?. In Clancey, W. J., & Weld, D. S. (Eds.), *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*, pp. 1139–1146. AAAI Press / The MIT Press.

Levesque, H. J. (1998). A completeness result for reasoning with incomplete first-order knowledge bases. In *KR*, pp. 14–23.

Lipovetzky, N., & Geffner, H. (2012). Width and serialization of classical planning problems. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pp. 540–545.

Lipovetzky, N., & Geffner, H. (2014). Width-based algorithms for classical planning: New results. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, pp. 1059–1060.

Lipovetzky, N., & Geffner, H. (2017). A polynomial planning algorithm that beats LAMA and FF. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017.*, pp. 195–199.

Miller, T., Felli, P., Muise, C. J., Pearce, A. R., & Sonenberg, L. (2016). 'Knowing whether' in proper epistemic knowledge bases. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pp. 1044–1050.

Muise, C. J., Belle, V., Felli, P., McIlraith, S. A., Miller, T., Pearce, A. R., & Sonenberg, L. (2015a). Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pp. 3327–3334.

Muise, C. J., Miller, T., Felli, P., Pearce, A. R., & Sonenberg, L. (2015b). Efficient reasoning with consistent proper epistemic knowledge bases. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pp. 1461–1469.

Nieuwenhuis, R., Oliveras, A., & Tinelli, C. (2006). Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll (t). *Journal of the ACM (JACM)*, *53*(6), 937–977.

Petrick, R. P. A., & Bacchus, F. (2002). A knowledge-based approach to planning with incomplete information and sensing. In Ghallab, M., Hertzberg, J., & Traverso, P. (Eds.), *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, pp. 212–222. AAAI.

Ramirez, M., Papasimeon, M., Lipovetzky, N., Benke, L., Miller, T., Pearce, A. R., Scala, E., & Zamani, M. (2018). Integrated hybrid planning and programmed control for real time uav maneuvering. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1318–1326. IFAAMAS.

Richter, S., & Westphal, M. (2010). The LAMA planner: Guiding cost-based anytime planning with landmarks. *J. Artif. Intell. Res.*, *39*, 127–177.

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

Scherl, R. B., & Levesque, H. J. (2003). Knowledge, action, and the frame problem. *Artif. Intell.*, *144*(1-2), 1–39.

Simon, H., & Newell, A. (1963). GPS: a program that simulates human thought. *Computers and Thought*, *1*(1), 279–293.

Van Ditmarsch, H., van Der Hoek, W., & Kooi, B. (2007). *Dynamic epistemic logic*, Vol. 337. Springer Science & Business Media.

Wan, H., Fang, B., & Liu, Y. (2021). A general multi-agent epistemic planner based on higher-order belief change. *Artif. Intell.*, *301*, 103562.

Wan, H., Yang, R., Fang, L., Liu, Y., & Xu, H. (2015). A complete epistemic planner without the epistemic closed world assumption. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 3257–3263.

7. Appendix

7.1 Code Examples

7.1.1 Corridor

Here is an code example for seeing function in Corridor:

```
bool sees(int agent_loc,int target_loc)
{
    return abs(agent_loc-target_loc)<=1;
};
```

7.1.2 Grapevine

Here is an code example for seeing function in Grapevine:

```
bool sees(int agent_loc,int target_loc)
{
    return agent_loc==target_loc;
};
```

7.1.3 BBL

Here is an code example for seeing function in BBL:

```
bool sees(int agent_x,int agent_y, int agent_dir,
          int agent_range,int target_x,int target_y)
{

    bool result = false; //default cannot sees
    int delta_x = target_x - agent_x;
    int delta_y = target_y - agent_y;
    int dir=0;

    if (delta_x == 0)
    {
        if (delta_y > 0)  dir = 90;
        else if (delta_y == 0) result = true;
        else dir =-90;
    }
    else
    {
        dir = (int) (atan2(delta_y,delta_x)*180/PI);
    }
    int delta_dir = abs(dir - agent_dir);
    if (delta_dir > 180) delta_dir = 360 - delta_dir;
    if (delta_dir <= agent_range/2) result = true;

    return result;
}
```

7.1.4 Social-media Network

Here is an code example for seeing function in SN:

```
bool sees(int agent,int target)
{
    return agent.friend_ids.count(target.parent_id)
};
```

7.2 Examples

7.2.1 Common Knowledge

The form of the common knowledge between two agents a and b does not only rely on the intersection on both agent's perspectives, but also rely on the intersection on both agent's nested perspective over that intersections, etc, until we reached a fixed point, which one intersection l of agent's nest perspectives is the same as the intersection on agent's perspectives over l. An common example is provided as follows:

The classic example is the Byzantine Generals: 2 generals who cannot directly communicate 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 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:
$f_a(s) = \{K_aK_ap_a, K_aK_ap_b, K_aK_bp_b, K_bK_bp_b, K_bK_ap_a, K_bK_bp_a, K_aK_bp_a, \}$
$f_b(s) = \{K_bK_bp_b, K_bK_bp_a, K_bK_ap_a, K_aK_ap_a\}$.

If we query common knowledge, we must evaluate their intersection s', which is:
$\{K_bK_bp_b, K_bK_bp_a, K_bK_ap_a, K_aK_ap_a\}$.

But for common knowledge, we need to apply perspective functions until we each termination. Applying another layer of perspective function on s':
$f_a(s') = \{K_ap_a\}$ and $f_b(s') = \{K_bp_b, K_bp_a, K_ap_a\}$

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 empty set, which is their common knowledge. As the intersection is empty, 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 force.