University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Muise, C;Belle, V;Felli, P;McIlraith, S;Miller, T;Pearce, AR;Sonenberg, L

Title:
Efficient multi-agent epistemic planning: Teaching planners about nested belief

Date:
2022-01-01

Citation:
Muise, C., Belle, V., Felli, P., McIlraith, S., Miller, T., Pearce, A. R. & Sonenberg, L. (2022). Efficient multi-agent epistemic planning: Teaching planners about nested belief. Artificial Intelligence, 302, https://doi.org/10.1016/j.artint.2021.103605.

Persistent Link:
https://hdl.handle.net/11343/297370

Efficient Multi-agent Epistemic Planning: Teaching Planners About Nested Belief

Christian Muise[a,*], Vaishak Belle[b], Paolo Felli[c], Sheila McIlraith[d], Tim Miller[e], Adrian R. Pearce[e], Liz Sonenberg[e]

[a]*School of Computing*
Queen's University, Kingston, Canada
[b]*University of Edinburgh, UK &*
Alan Turing Institute, UK
[c]*Faculty of Computer Science*
University of Bozen-Bolzano, Bolzano, Italy
[d]*Department of Computer Science*
University of Toronto, Toronto, Canada
[e]*School of Computing and Information Systems*
University of Melbourne, Melbourne, Australia

Abstract

Many AI applications involve the interaction of multiple autonomous agents, requiring those agents to reason about their own beliefs, as well as those of other agents. However, planning involving nested beliefs is known to be computationally challenging. In this work, we address the task of synthesizing plans that necessitate reasoning about the beliefs of other agents. We plan from the perspective of a single agent with the potential for goals and actions that involve nested beliefs, non-homogeneous agents, co-present observations, and the ability for one agent to reason *as if* it were another. We formally characterize our notion of planning with nested belief, and subsequently demonstrate how to automatically convert such problems into problems that appeal to classical planning technology for solving efficiently. Our approach represents an important step towards applying the well-established field of automated planning to the challenging task of planning involving nested beliefs of multiple agents.

*Corresponding author
Email addresses: christian.muise@queensu.ca (Christian Muise), vaishak@ed.ac.uk (Vaishak Belle), pfelli@unibz.it (Paolo Felli), sheila@cs.toronto.edu (Sheila McIlraith), tmiller@unimelb.edu.au (Tim Miller), adrianrp@unimelb.edu.au (Adrian R. Pearce), l.sonenberg@unimelb.edu.au (Liz Sonenberg)

Keywords: automated planning, epistemic planning, knowledge and belief

1. Introduction

AI applications increasingly involve the interaction of multiple agents – be they intelligent user interfaces that interact with human users, gaming systems, or multiple autonomous robots interacting together in a factory setting. In the absence of prescribed coordination, it is often necessary for individual agents to synthesize their own plans, taking into account not only their own capabilities and beliefs about the world but also their beliefs about other agents, including what each of the agents will come to believe as the consequence of the actions of others. To illustrate, consider the scenario where Larry and Moe meet on a regular basis at the local diner to swap the latest gossip. Larry has come to know that Nancy (Larry's daughter) has just received a major promotion in her job, but unbeknownst to him, Moe has already learned this bit of information through the grapevine. Before they speak, both believe Nancy is getting a promotion, Larry believes Moe is unaware of this (and consequently wishes to share the news), and Moe assumes Larry must already be aware of the promotion but is unaware of Moe's own knowledge of the situation. Very quickly we can see how the nesting of (potentially incorrect) belief can be a complicated and interesting setting to model.

In this paper, we examine the problem of synthesizing plans in such settings. In particular, given a finite set of agents, each with: (1) (possibly incomplete and incorrect) beliefs about the world and about the beliefs of other agents; and (2) differing capabilities including the ability to perform actions whose outcomes are unknown to other agents; we are interested in synthesizing a plan to achieve a goal condition. Planning is at the belief level and as such, while we consider the execution of actions that can change the state of the world (ontic actions) as well as an agent's state of knowledge or belief (epistemic or more accurately doxastic actions, including communication actions), all outcomes are with respect to belief. Further, those beliefs respect the KD_n and $\mathrm{KD45}_n$ axioms

of epistemic logic [25]. Finally, we take a *perspectival view*, planning from the viewpoint of a single agent. While the planning agent presumes control of all the other agents actions (i.e., we do not model the possibility of other agents acting in an uncertain and unobserved manner), there is *not* presumption of complete knowledge over the nested belief of other agents. We contrast this with traditional multi-agent planning which generates a coordinated plan to be executed by multiple agents under an assumption of complete knowledge of the reasoner (e.g., [13]).

Solutions to the epistemic planning problem can be roughly divided into two classes: (1) the approach taken in *Dynamic Epistemic Logic* (DEL) of using Kripke structures to maintain knowledge and then using models, such as event models, to update Kripke structures as events occur, for example, Bolander and Andersen [10]; or (2) to maintain a set of formulae in a database and using belief update and revision to progress the database, for example, Huang et al. [33]. In this paper, we adopt the latter approach.

We define epistemic planning for a particular class of problems in which the knowledge base is a *proper epistemic knowledge base* (PEKB) [40]. A PEKB is defined as finite set of epistemic literals, meaning that: (1) they do not allow for disjunctive belief; and (2) the depth of nested belief is bounded. We show that this provides certain theoretical guarantees on the computational complexity of progression of actions that have PEKBs as preconditions and effects, and directly offers an efficient means of planning using known methodologies.

We propose a means of encoding our definition of epistemic planning as a classical planning problem, enabling us to exploit state-of-the-art classical planning techniques to synthesize plans for these challenging planning problems. A key aspect of our encoding is the use of *ancillary conditional effects* – additional conditional effects of actions which enforce desirable properties such as epistemic modal logic axioms (cf. Section 4), and allow domain modellers to encode conditions under which agents are mutually aware of actions (cf. Section 5.1). By encoding modal logic axioms as effects of actions, we are using the planner to perform epistemic reasoning in addition to the standard reasoning

about action and change. We implement this encoding as a compilation from epistemic planning problems to classical planning problems, and show that it handles a rich variety of problems.

Computational machinery for epistemic reasoning has historically appealed to theorem proving or model checking (e.g., [21]), while epistemic planning, originally popularized within the DEL community, has shifted from a sole focus on theoretical concerns (e.g., [45]) to a growing array of practical approaches and implementations. Previously, the problem of planning with epistemic goals has received limited attention in the automated planning community (e.g., [58, 5] and most recently with multi-agent beliefs [37, 33]). The work presented here is an important step towards leveraging state-of-the-art planning technology as a black-box sub-procedure to address rich epistemic planning problems of the sort examined by the DEL community. Indeed, we can readily solve existing examples in the DEL literature (cf. Section 7). We further discuss the relationship of our work to other modern epistemic planning work in Section 8.

Parts of this paper have been published in earlier work, notably the theory of proper epistemic knowledge bases [47, 48, 52] and parts of the encoding to classical planning [53]. However, the work in this paper extends that body of work in three key areas: (1) we provide a formal definition of epistemic planning over PEKBs, with earlier work [53] informally defining the problem; (2) we extend the encoding to deal with a restricted class of common knowledge, called *always known*, which are propositions for which every agent will always know the value, such as static knowledge; and (3) we significantly expand the evaluation, defining new benchmark problems and presenting applications on which we have used our planner.

After presenting the background notation required in Section 2, we detail the syntax and semantics of the Restricted Perspectival Multi-agent Epistemic Planning model in Section 3. We follow with the detailed encoding to classical planning in Section 4. Extensions to greatly reduce the burden of modelling and improve planner efficiency are presented in Section 5 with grounded examples in Section 6. We investigate the empirical nature of our approach in Section 7 and

provide a discussion of related work and concluding remarks in Sections 8-9. For reader convenience, Appendix A provides a list of the common acronyms used throughout the paper. To illustrate the PDKB Description Language, a variant of PDDL [29], Appendix B details the *Grapevine* and *Envelope* domains. In Appendix C we demonstrate the encoding of ancillary conditional effects. Finally, for completeness, Appendix D outlines the theory of PEKBs in the context of related work, emphasising the nice logical and computational properties that make them a suitable representation for extending classical planning over belief bases, in a $\mathrm{KD}_n/\mathrm{KD45}_n$ context.

Example 1 (Grapevine)**.** We will use a common example to explain the concepts introduced throughout the paper. Consider a scenario where a group of agents each have their own secret to (possibly) share with one another. Each agent can move freely between a pair of rooms, and broadcast any secret they currently believe to everyone in the room. Initially they only believe their own unique secret. Goals we might pose include the universal spread of information (everyone believes every secret), misconception (an agent holds a false belief about someone else's belief), etc. We will use $1, 2, \cdots$ to represent the agents, and $s_1, s_2, \cdots$ to represent their secrets, respectively. Given our perspectival view of the setting, these may be cast either in terms of one of the acting agents in the environment, or as a third-party observer that has partial information about the world.

2. Preliminaries

2.1. Epistemic Logic

In this section, we introduce the concepts of epistemic logic, and in particular, the model of epistemic logic dealing with belief. We remain true to the terminology used in epistemic logic, so it is worth noting for those unfamiliar with the subject area that terms similar to those found in the planning literature are used. In particular, *state* represents a configuration of what is

true/false in the world, but *reachability* is not defined in terms of actions applicability. Rather, it is a way to capture the space of possible states an agent believes to be possible.

Let $\mathcal{P}$ and Ag respectively be finite sets of propositions and agents. The set of well-formed formulae, Ł, is obtained from the following grammar:

$$\phi ::= p \mid \phi \wedge \phi' \mid \Box_i \phi \mid \neg\phi \mid \top \mid \bot$$

in which $p \in \mathcal{P}$ and $i \in Ag$. $\Box_i \phi$ should be interpreted as "agent i believes ϕ", and we will suppress the agent index when the formula holds for all agents. We will also use $\Diamond\phi$ as a syntactic shorthand for $\neg\Box\neg\phi$ and $\Diamond_i\phi$ should be interpreted as "agent i considers it possible that ϕ".

The semantics is given using *Kripke structures* [25]. Each Kripke structure is a tuple $M = (\mathcal{W}, \pi, R_1, \ldots R_n)$, in which $\mathcal{W}$ is the set of all worlds considered in a model, $\pi \in \mathcal{W} \to 2^{\mathcal{P}}$ is a function that maps each world to the set of propositions that hold in that world, and each $R_i \subseteq \mathcal{W} \times \mathcal{W}$ (for each $i \in Ag$) is a belief accessibility relation. Each relation R_i captures the uncertainty of agent i such that, given the actual world w, the set $R_i(w) = \{w' \mid R_i(w, w')\}$ is the set of worlds that agent i considers possible, i.e., indistinguishable from w.

Given these definitions, the satisfaction of a formula ϕ in a Kripke structure M and a world w is denoted as $M, w \vDash \phi$, and it is defined inductively over the structure of ϕ:

$$\begin{array}{lll}
M, w \vDash \top & & \\
M, w \nvDash \bot & & \\
M, w \vDash p & \text{iff} & p \in \pi(w) \\
M, w \vDash \varphi \wedge \psi & \text{iff} & M, w \vDash \varphi \text{ and } M, w \vDash \psi \\
M, w \vDash \neg\varphi & \text{iff} & M, w \nvDash \varphi \\
M, w \vDash \Box_i \varphi & \text{iff} & \text{for all } v \in R_i(w),\ M, v \vDash \varphi
\end{array}$$

We define entailment as: $\phi \vDash \psi$ if and only if for every model M and world w such that $M, w \vDash \phi$, we have $M, w \vDash \psi$. The pointed model (M, w) defines an actual world in the model.

As discussed by Fagin et al. [25], constraints on Kripke structures lead to particular properties of belief. If the Kripke structure is *serial*, *transitive*, and *Euclidean* we obtain (arguably) the most common properties of belief:

K	$\Box\phi \wedge \Box(\phi \supset \psi) \supset \Box\psi$	(Distribution)
D	$\Box\phi \supset \Diamond\phi$	(Consistency)
4	$\Box\phi \supset \Box\Box\phi$	(Positive introspection)
5	$\Diamond\phi \supset \Box\Diamond\phi$	(Negative introspection)

These axioms collectively form the system referred to as $KD45_n$, where n specifies that there are multiple agents in the environment. From the axioms, additional theorems can be derived. For example, in this work, we use the following theorems from Hughes and Cresswell [34] for reducing neighbouring belief modalities involving the same agent into a single belief modality:

$$\Box_i\Box_i\phi \equiv \Box_i\phi \qquad \Box_i\Diamond_i\phi \equiv \Diamond_i\phi$$
$$\Diamond_i\Box_i\phi \equiv \Box_i\phi \qquad \Diamond_i\Diamond_i\phi \equiv \Diamond_i\phi$$

2.2. Proper epistemic knowledge bases

Not surprisingly, reasoning (and planning) in these logical frameworks is computationally challenging [25, 4]. To mitigate this, Lakemeyer and Lespérance [40] define a *proper epistemic knowledge base* (PEKB) as a set of restricted formulae, called *restricted modal literals* (RMLs), of the form:

$$\phi ::= \top \mid \bot \mid p \mid \neg p \mid \Box_i\phi \mid \Diamond_i\phi$$

where $p \in \mathcal{P}$ and $i \in Ag$. Thus, a PEKB contains no disjunctive formulae.

The depth of an RML is defined as:

$$depth(\phi) = \begin{cases} 1 + depth(\psi) & \text{if } \phi = \Box_i(\psi) \\ 1 + depth(\psi) & \text{if } \phi = \Diamond_i(\psi) \\ 0 & \text{otherwise} \end{cases}$$

where $p \in \mathcal{P}$. We will view a conjunction of RMLs equivalently as a set, and denote the set of all RMLs with bounded depth d for a group of agents Ag as $\mathcal{L}^{Ag,d}_{RML}(\mathcal{P})$ (we drop the $\mathcal{P}$ qualifier when it is obvious from the context).

Because RMLs are represented as a sequence of $\Box$ and $\Diamond$ operators, ending in a propositional literal, they are in *negation normal form* (NNF); i.e., negation appears only in front of propositional variables. Any standard modal literal with negations interleaved can be re-written into NNF using the equivalences,

$$\neg\Box_i\varphi \equiv \Diamond_i\neg\varphi \qquad \neg\Diamond_i\varphi \equiv \Box_i\neg\varphi \qquad \neg\neg p \equiv p$$

To simplify exposition throughout the paper, we will take the negation of an RML to indicate its equivalent NNF (i.e., by using the repeated application of the above equivalences so that negations appear only at the literal level).

We use $Lit(\phi)$ to refer to the literal at the end of the RML ϕ:

$$Lit(\phi) = \begin{cases} Lit(\psi) & \text{if } \phi = \Box_i\psi \text{ or } \phi = \Diamond_i\psi \\ \phi & \text{otherwise} \end{cases}$$

Lakemeyer and Lespérance [40] show how to compile a PEKB into *prime implicate normal form* (PINF) in exponential time and space, and how to check entailment of this PINF formula in polynomial time. This compares to double-exponential time and space for non-restricted problems [9]. Thus, by sacrificing expressiveness, some computational cost can be reduced. Their entailment algorithm is sound for arbitrary formulae, and complete for PINF formulae. The consistency of a PEKB is defined using the semantics of epistemic logic, where the PEKB is seen as the conjunction of the elements in the PEKB [40].

Muise et al. [52] showed that for the logics KD_n, compilation to PINF is not required: the PEKB can be queried directly in polynomial time. Querying is sound for any arbitrary KD_n formula, and is complete for formulae in a particular normal form. Miller and Muise [48] then further define belief update for PEKBs that can be calculated in polynomial time.

Extending to the KD45_n case — that is, adding positive and negative introspection (axioms 4 and 5 respectively) — is straightforward using the equivalences in Section 2.1. Lakemeyer and Lespérance [40] define an *i-objective* formula as a formula that is about the world and agents other than i. For example, $\Box_j(p \wedge \Box_i\neg p)$ is i-objective, but $\Box_j p \wedge \Box_i\neg p$ is not. A formula is *i-reduced*

iff for all sub-formulae $\Box_i\varphi$ and $\Diamond_i\varphi$, φ is i-objective.

One can see that any RML $\Box_i\varphi$ or $\Diamond_i\varphi$ can be i-reduced by repeatedly applying the equivalences above to strip out consecutive occurrences of modal operators of the same agent. Therefore, one can reduce both a KD45_n PEKB and query into a KD_n PEKB and query respectively, allowing application of the approaches in this section to KD45_n. We restrict our discussion in the remainder of this paper to the more simplified case of KD_n, assuming that for KD45_n, consecutive modal operators of the same agent have been removed.

Expressivenes of PEKBs. Given that PEKBs restrict more general epistemic logic by removing disjunction, it raises the question whether they are too restrictive. The answer depends on which perspective is taken. On the one hand, omitting disjunction from the language means that some interesting properties around disjunctive preconditions and goals need to be encoded manually, and this manual encoding needs to carefully ensure all logical properties are maintained. On the other hand, contemporary classical and non-deterministic planning tools typically do not support disjunction on even propositional formulae, so expanding the planning language to a PEKBs does not limit the planning at all – it makes modelling some problems more straightforward. In Section 7.5, we highlight an example of where the lack of handling disjunctive knowledge limits what we can represent. Further, we dive deeper on the expressive differences between our approach and related works in Section 8.

PEKBs have been shown to be expressive enough for many applications, such as collaborative filtering [40] and team formation [50]. The key contribution of this paper is to show how epistemic planning can be done *efficiently*. What is clear is that to solve planning problems efficiently, trade-offs in expressiveness must be made. The inclusion of disjunction would immediately rule out the approach of compiling to classical or non-deterministic planning without explicitly reifying sub-formulae. Importantly, if classical planners did support disjunction, then including disjunctive epistemic knowledge bases would require a double-exponential compilation step just on the epistemic formulae themselves

[8], rather than polynomial.

As the results in this paper later show, we can indeed solve several standard benchmarks in epistemic planning more efficiently than other epistemic planners (both for expressiveness and computational reasons). In our view, to advance the field of epistemic planning to the point where planners are viable for solving large-scale planning tasks rather than epistemic puzzles, restricted representations like PEKBs will be essential.

2.3. Classical and FOND Planning

A classical planning problem consists of a tuple $\langle F, I, G, O \rangle$, where F is a set of fluent atoms, I is the initial state, G is the goal, and O is a set of operators. A *complete state* (or just *state*) s is a subset of the fluents F with the interpretation that fluents not in s are false (equivalently, this can be seen as a conjunction between the literals found in s and the negated literals not found in s). A *partial state* s is similarly a subset of the fluents F, but has the interpretation that literals not found in s can take on any value (thus equivalent to a conjunction of just the literals in s). I is a complete state while G is a partial state. Every operator $o \in O$ is a tuple $\langle Pre_o, \mathit{eff}_o^+, \mathit{eff}_o^- \rangle$, and we say that o is applicable in s iff $Pre_o \subseteq s$. The set eff_o^+ (resp. eff_o^-) contains conditional effects describing the fluent atoms that should be added (resp. removed) from the state when applying the operator. Finally, every conditional effect in eff_o^+ or eff_o^- is of the form $(\mathcal{C} \to l)$ where $\mathcal{C}$ is the condition for the effect and l is a fluent that is the result of the effect. The condition $\mathcal{C}$ consists of a tuple $\langle \mathcal{C}^+, \mathcal{C}^- \rangle$ where $\mathcal{C}^+$ is the set of fluents that must hold and $\mathcal{C}^-$ the set of fluents that must not hold. A conditional effect $(\langle \mathcal{C}^+, \mathcal{C}^- \rangle \to l)$ *fires* in state s iff $\mathcal{C}^+ \subseteq s$ and $\mathcal{C}^- \cap s = \emptyset$. Assuming o is applicable in s, and $\mathit{eff}_o^+(s)$ (resp. $\mathit{eff}_o^-(s)$) are the positive (resp. negative) conditional effects that fire in state s, the state of the world s' after applying o is defined as follows:

$$\begin{aligned} s' = s \setminus \{l \mid (\mathcal{C} \to l) \in \mathit{eff}_o^-(s)\} \\ \cup \{l \mid (\mathcal{C} \to l) \in \mathit{eff}_o^+(s)\} \end{aligned}$$

Our account of classical planning mirrors the standard representation (see, for example, [26]), with the exception that we make explicit the fluent atoms that are added, deleted, required to be in, or required to be absent from the state of the world. This simplifies the exposition when we encode nested beliefs as a classical planning problem.

Fully-observable non-deterministic (FOND) planning extends classical planning with non-deterministic operators. That is, operators can have one *or more* effects, and exactly one of these effects occurs when the action is executed. The planning agent does not know which effect will occur prior to execution, but can observe which effect occurred immediately after execution. To extend classical planning, instead of each operator containing a single pair of positive and negative effects, an operator has a *set* of non-deterministic effects, with each containing positive and negative effects: $\langle Pre_o, \mathit{Eff}_o \rangle$, in which every non-deterministic effect $E \in \mathit{Eff}_o$ consists of two sets of conditional effects, eff_E^+ and eff_E^-. A solution to a FOND planning problem is a *policy* π, which maps states to operators. A policy represents a set of possible sequences of actions. If every sequence in a policy reaches the goal, it is said to be *strong*; otherwise, if only some sequences reach the goal, it is *weak*.

3. Restricted Perspectival Multi-agent Epistemic Planning

In this section, we define the syntax and semantics of *restricted perspectival multi-agent epistemic planning* (RP-MEP). At a high-level, this is defined simply as planning over states of the world where states are PEKBs, instead of collections of propositional fluents. The choice of PEKBs is suitable for two reasons:

1. The syntactic restrictions imposed by PEKBs that prevent disjunction and infinite nesting are consistent with the syntactic restrictions employed by STRIPS-based planners, in which arbitrary disjunction is not permitted, and the set of literals (fluents) in a planning problem is finite. In this

sense, using PEKBs increases the expressiveness of the STRIPS language to include epistemic formulae.

2. PEKBs come with nice logical and computational properties, as we outline in this section, and elaborate in Appendix D. First, a *consistent* PEKB — that is, one with no contradictory statements — is *logically separable*, which means the literals in the knowledge base do not interact to produce new formulae. Further, a consistent PEKB can be queried in polynomial time without a pre-compilation step such as the one used by Lakemeyer and Lespérance [40], and can be queried in constant time with an exponential pre-compilation step. Finally, a consistent PEKB can be updated with new literals and remain consistent using a polynomial-time algorithm.

We assume that the state represents the mental model of a particular agent that perceives an environment that includes all other agents. All reasoning is from the perspective of this single agent. The fluents that are true in a state correspond to the RMLs that the agent believes, while the fluents that are false correspond to the RMLs that the agent does not believe. Action execution, then, is predicated on the agent *believing* that the preconditions are satisfied. Similarly, the mental model of the agent is updated according to the effects of an action. Note that we do not need to enforce a separation of ontic and epistemic effects – the same action can update belief about propositions as well as RMLs. This is due to the interpretation that the state of the world represents the mental model of a given agent: every effect is epistemic in this sense.

3.1. Syntax

In describing the actions for a RP-MEP problem, we consider both conditional and non-deterministic effects; combined they allow for a rich variety of epistemic problems to be described. The non-determinism, however, is of a restricted form in that it is fully observable, meaning that the executing agent does not know what the outcome of an action will be, however, it will observe the outcome immediately after executing the action. While we hope to remove

this restriction in the future, it allows us to account for various contingencies during the planning process, and model aspects such as questions asked to other agents in the environment.

The basic atomic action in RP-MEP is a PEKB planning action. Essentially, an action is just a classical planning action, but in which the preconditions and effects can be PEKBs instead of just propositions.

Definition 1. RP-MEP Planning Action

A RP-MEP Planning action a is a pair $\langle Pre_a, \mathit{Eff}_a \rangle$, in which Pre_a is a PEKB capturing the preconditions of a and Eff_a is the set of non-deterministic effects of a, defined as follows. Every non-deterministic effect $E \in \mathit{Eff}_a$ is constituted by a set of *conditional effects* $\{(\gamma_1, \varphi_1), \ldots, (\gamma_k, \varphi_k)\}$, in which each γ_i is a PEKB called the *condition* of the conditional effect, and each φ_i is a RML called the *effect* of the conditional effect. A conditional effect (γ_i, φ_i) is read informally to say that if the condition is true before the action executes, then the effect will be true after the action executes.

We can now define a planning problem as follows:

Definition 2. Multi-Agent Epistemic Planning Problem

A ***m**ulti-agent **e**pistemic **p**lanning* (MEP) problem $\mathcal{D}$ is a tuple of the form $\langle \mathcal{P}, \mathcal{A}, Ag, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{P}$ and Ag are as above, $\mathcal{A}$ is a set of RP-MEP planning actions, $\mathcal{I}$ is the PEKB representing the *initial theory* and $\mathcal{G}$ is a PEKB capturing the *goal condition.*

Following Reiter [63] and van Ditmarsch et al. [19], the above action formalization can be expressed as standard *precondition* and *successor state axioms*, which would then define the meaning of achieving the goal. Using PDL-like syntax, we interpret $[\alpha]\mathcal{G}$ as "$\mathcal{G}$ holds after executing action α". We formally define this below in Section 3.2.2.

We define a ***r**estricted **p**erspectival* multi-agent epistemic planning problem

(RP-MEP problem) for depth bound d and the root agent $\star \in Ag$ as a MEP problem with the additional restrictions that: (1) every RML is from the perspective of the root agent – i.e., it is from the following set:

$$\{\Box_\star \varphi \mid \varphi \in \mathcal{L}^{Ag,d}_{RML}\} \cup \{\Diamond_\star \varphi \mid \varphi \in \mathcal{L}^{Ag,d}_{RML}\},$$

and (2) there is no disjunctive belief: the initial theory, goal specification, and every precondition are all PEKBs, every effect is a single RML, and every effect condition is a PEKB.

We address the planning problem from the view of an acting agent, where the designated root agent $\star$ is the one for which we plan. Intuitively, this means that conditional effects are formulated in the context of the root agent; e.g., we would have a conditional effect of the form $(\{\Box_\star \gamma\}, \Box_\star \varphi)$ for action a in a RP-MEP problem to capture the fact that the root agent will believe φ if it believed γ before a occurred (as formalised in the next section).

This admits a rich class of planning problems; e.g., it is reasonable to assume that the root agent's view of the world differs from what a particular agent i believes, and so another conditional effect of a might be $(\{\Box_\star \gamma\}, \Box_\star \Box_i \neg \varphi)$ – even though the root agent believes doing a would make φ true if γ holds, the root agent believes that i will believe $\neg\phi$ if γ holds. In particular, this is easily shown to generalize a standard assumption in the non-epistemic multi-agent planning literature [44] that all agents hold the same view of what changes after actions occur.

In the next section, we show how a restricted perspectival multi-agent epistemic planning problem can be represented as a classical planning problem, where the key insight is to encode reasoning features (such as deduction in KD_n) as *ramifications* realized using ordinary planning operators.

3.2. Semantics

To formally define the semantics of epistemic planning over PEKBs, we use the notation defined by Miller and Muise [48] for belief update in PEKBs. The theoretical justification for this process is explored in Appendix D, but for

clarity we restrict the rhetoric in this section to only those concepts required to describe our encoding.

3.2.1. Action Progression with PEKBs

While syntactic treatments of belief and knowledge have been proposed in the past; e.g. [20, 38], PEKBs place restrictions on the syntax of formulae to provide desirable computational properties. The key result by Lakemeyer and Lespérance [40] is that by eliminating disjunction from formulae, PEKBs can be compiled in into a set of prime implicates, similar to the method employed by [8, 9] for the logic K_n. This allows entailment queries to be answered in polynomial time by structurally traversing the prime implicates instead of querying the original belief base. The cost is that the compilation into prime implicates is exponential, rather than double exponential in the case of epistemic logic with disjunction. One could reason efficiently (polynomial time) at the level of prime implicates, but we opt to further process the PEKBs so that they are deductively closed – this will greatly simplify the propositional encoding later in Section 4.

In this section, we define the semantics of epistemic planning over PEKBs. For this reason, in order to keep the terminology intuitive, we call these *PEKB states*. For simplicity, we define a PEKB state P as a *deductively-closed* subset of formulae. We use $\overline{P}$ to denote the PEKB that contains the negation of every RML in P; that is $\overline{P} = \{\neg\varphi \mid \varphi \in P\}$. Following Definitions 1 and 2, we can define the procedures to modify PEKBs as follows:

Definition 3. Belief update and erasure in PEKBs

Given deductively-closed PEKBs P and Q, we define $P \blacklozenge Q$ and $P \diamond Q$ as the *belief erasure* and *belief update* of P and Q respectively, as follows:

$$P \blacklozenge Q = P \setminus Q$$
$$P \diamond Q = (P \blacklozenge \overline{Q}) \cup Q.$$

Belief update of P with Q is defined as the standard 'forget (erase) then conjoin': anything in P disagreeing with Q is forgotten, then everything from Q is added.

Miller and Muise [48] present definitions of belief update and erasure for PEKBs and evaluate them against the Katsuno and Mendelzon [36] postulates for belief update [36]. They also show that belief update and erasure in PEKBs can be computed in polynomial time.

Definition 4. RP-MEP Progression

We define the progression of a single deterministic effect E applied to a deductively-closed PEKB state P, labelled $\mathit{Progress}\,(P, E)$, as:

$$\begin{aligned}
\mathit{Progress}\,(P, E) &= (P \blacklozenge (R \cup U)) \diamond Q \\
Q &= \bigcup_{(\gamma,\varphi)\in E} \{\psi \mid \gamma \subseteq P \wedge \varphi \vDash \psi\} \\
R &= \bigcup_{(\gamma,\neg\varphi)\in E} \{\psi \mid \gamma \subseteq P \wedge \varphi \vDash \psi\} \\
U &= \bigcup_{(\gamma,\varphi)\in E} \{\neg\psi \mid \overline{\gamma} \cap P = \emptyset \wedge \neg\varphi \vDash \neg\psi\}.
\end{aligned}$$

Q defines the set of literals to be added, R defines the set of literals to be deleted, while U defines the set of *uncertain firing* literals to be deleted. Uncertain firing captures instances in which an agent is unsure whether a conditional effect is true, denoted $\overline{\gamma} \cap P = \emptyset$. If an agent is unsure, then it should not believe the effect (unless it was already true), but must admit that it could be true. Therefore, it must not believe the opposite, and we should remove $\neg\varphi$ and anything that follows from it deductively.

From this, we can define the set of possible progressions of a non-deterministic RP-MEP action $a = \langle \mathit{Pre}_a, \mathit{Eff}_a \rangle$ in a deductively-closed PEKB state P:

$$\mathit{Progress}\,(P, a) \stackrel{\text{def}}{=} \{\mathit{Progress}\,(P, E) \mid E \in \mathit{Eff}_a \text{ and } \mathit{Pre}_a \subseteq P\}$$

Intuitively, $\mathit{Progress}\,(P, a)$ is defined as a set of PEKB states, each corresponding to the application of a non-deterministic effect of the RP-MEP action

a. In case the RP-MEP action a is not executable in P, i.e. $Pre_a \not\subseteq P$, then the set is empty.

A (RP-MEP) *policy* is a function $\alpha : \mathcal{P} \to \mathcal{A}$ mapping PEKB states to RP-MEP actions ($\mathcal{P}$ is used to denote the set of all possible PEKB states). Valid policies can be partial functions, because their terminating states can be undefined.

A *trajectory* is a sequence of PEKB states, and we say that a trajectory $P_0, \ldots, P_n$ is *induced* by a policy α from a PEKB state P iff $P = P_0$ and for $i \in [0, n-1]$ we have that $\alpha(P_i)$ is defined and that $P_{i+1} \in \mathit{Progress}\,(P_i, \alpha(P_i))$. Intuitively, as actions have non-deterministic effects, there are in general many trajectories which are induced by the same policy.

A PEKB state P is said to be an *end state* of a policy α from a PEKB state P_0, denoted $P \in \mathit{end}(P_0, \alpha)$, iff there exists a trajectory $P_0, \ldots, P$ induced by α from P_0 such that $\alpha(P)$ is either not defined or empty.

We further assume that a valid policy does not yield inescapable cycles. In other words, for every trajectory T induced by policy α from a PEKB P, there exists a trajectory T' so that $T \cdot T'$ is induced by α from P and the last state of $T \cdot T'$ is an end state of α. This yields policies that correspond to strong cyclic plans from FOND planning.

It is worth pointing out the implicit assumption of "fairness". Loosely speaking, fairness refers to a property of the non-determinism in an environment that captures the notion of all possible outcomes having some chance of success: if the agent finds themselves repeatedly in the same situation and executes the same action, every possible outcome should occur infinitely often in the limit. Due to our definition of a valid policy, we are assuming fairness. This may be an abstraction of the true nature of the environment (e.g., a non-deterministic action that "asks" another agent a question is inherently unfair), but it is still a useful abstraction for reasoning with both uncertainty in action outcomes and uncertainty in agent beliefs. There are encoding techniques to solve unfair non-deterministic planning problems with solvers assuming fairness (e.g., [14]), but those notions are out of scope for this work.

3.2.2. Plan Verification and Generation

In this section, we formally define the problems of plan validation and plan verification for RP-MEP.

Given a conjunction of RMLs ϕ, we use $[\alpha]\phi$ to represent that ϕ holds in all trajectories defined by policy α, given some start state. Formally, this is defined as:

$$P \vDash [\alpha]\phi \quad \text{iff} \quad Q \vDash \phi \text{ for all } Q \in end(P, \alpha),$$

in which P is a deductively-closed PEKB. If $[\alpha]\phi$ holds in P, then ϕ holds for all end states of the policy α from P and α is said to be a *strong* policy for ϕ from P. We define $\langle\alpha\rangle\phi$ as shorthand for $\neg[\alpha]\neg\phi$, which means that ϕ holds on at least one end trajectory of α. If $\langle\alpha\rangle\phi$, then α is said to be a *weak* policy for ϕ from P.

The notions of plan verification and generation rely on the idea of a *valid policy* for a goal: a policy that achieves the goal from the initial state. Formally, we have the following type of valid policies.

Definition 5. Plan Verification and Generation

Given an RP-MEP problem, $\langle \mathcal{P}, \mathcal{A}, Ag, \mathcal{I}, \mathcal{G} \rangle$, a policy α is a valid *weak* policy for $\mathcal{G}$ iff the following holds:

$$\mathcal{I} \models \langle\alpha\rangle\mathcal{G}$$

A policy α is a valid strong policy for $\mathcal{G}$ iff the following holds:

$$\mathcal{I} \models [\alpha]\mathcal{G}$$

The task of *plan verification* is to determine if a policy is a valid (weak or strong) policy for $\mathcal{G}$. The task of *plan generation* is to generate such a policy.

Our notion of plan generation and verification follows closely that of traditional fully-observable non-deterministic planning, only with our states of the world being represented by PEKBs in lieu of an assignment of true/false to the

fluents in the domain. It is unique in relation to other formalisms of epistemic planning, in that our restriction to PEKBs is not an assumption that the other related systems adopt. The more fundamental differences are discussed later in Sections 7 and 8.

4. Propositional Encoding of RP-MEP

In this section, we show how to encode an RP-MEP into a standard propositional planning problem, as either classical planning or FOND planning as required. The novelty in this work is that, given an RP-MEP planning problem, we convert this into a propositional planning problem that can be solved with any off-the-shelf classical/FOND planning tool that supports conditional effects. As FOND planning is a generalisation of classical planning, we define our encoding for FOND planning only.

The basic framework that we take is as follows. For every RML in the domain problem, create a standard propositional fluent to represent that RML. Then, for every action in the original problem, replace each RML with its propositional fluent, and add new conditional effects to the action to handle the semantics of RP-MEP. The result is a FOND planning problem in which a solution to the FOND problem is a solution for the RP-MEP problem.

The remainder of the section will proceed as follows:

1. We present the framework for our encoding of an RP-MEP problem into FOND planning.
2. We specify how the belief update operator defined in Section 3 is encoded.
3. We specify how to correctly update if an agent is uncertain whether a conditional effect fires.

4.1. Base Encoding

First, using the theorems for reducing neighbouring modalities outlined in Section 2.1 (e.g. $\Box_i\Box_i\phi \equiv \Box_i\phi$), remove any neighbouring modalities from each

RML that appears in any part of the planning problem, including preconditions, effects, fluents, etc. This preserves the semantics of the original problem, but simplifies the encoding.

Then, we transform the problem into an equivalent propositional planning problem. Intuitively, every RML in $\mathcal{L}^{Ag,d}_{RML}$ will correspond to a single fluent in F (e.g., both $\Box_1 p$ and $\Diamond_1 \neg p$ will become fluents), and the operators will describe how the mental model of our root agent should be updated. Formally, we define the encoding of a RP-MEP problem as follows:

Definition 6. Encoding of RP-MEP

Let $\mathcal{B}_i$ and $\mathcal{N}_i$ be functions that map i's positive (resp. negative) belief from a PEKB P to the respective fluents:

$$\mathcal{B}_i(P) = \{l_\phi \mid \Box_i \phi \in P\}$$
$$\mathcal{N}_i(P) = \{\neg l_\phi \mid \neg\Box_i \phi \in P\}$$

Given a RP-MEP problem, $\langle \mathcal{P}, \mathcal{A}, Ag, \mathcal{I}, \mathcal{G}\rangle$ and a bound d on the depth of nested belief we wish to consider, we define the propositional encoding as the tuple $\langle F, I, G, O\rangle$ such that:

$$F \stackrel{\text{def}}{=} \{l_\phi \mid \phi \in \mathcal{L}^{Ag,d}_{RML}\} \qquad I \stackrel{\text{def}}{=} \mathcal{B}_\star(Cl\mathcal{I}) \qquad G \stackrel{\text{def}}{=} \mathcal{B}_\star(\mathcal{G})$$

and for every action $\langle Pre_a, \mathit{Eff}_a\rangle$ in $\mathcal{A}$, we have a corresponding operator $\langle Pre_o, \mathit{Eff}_o\rangle$ in O such that:

$$Pre_o \stackrel{\text{def}}{=} \mathcal{B}_\star(Pre_a)$$
$$\mathit{Eff}_o \stackrel{\text{def}}{=} \{\langle \mathit{eff}^+_E, \mathit{eff}^-_E\rangle \mid E \in \mathit{Eff}_a\}$$
$$\mathit{eff}^+_E \stackrel{\text{def}}{=} \{(\langle \mathcal{B}_\star(\gamma), \overline{\mathcal{N}_\star(\gamma)}\rangle \rightarrow l_\phi) \mid (\gamma, \Box_\star \phi) \in E\}$$
$$\mathit{eff}^-_E \stackrel{\text{def}}{=} \{(\langle \mathcal{B}_\star(\gamma), \overline{\mathcal{N}_\star(\gamma)}\rangle \rightarrow l_\phi) \mid (\gamma, \neg\Box_\star \phi) \in E\}$$

Whenever clear from the context, we will use eff^+_o and eff^-_o to refer to the effects associated with an action with a single outcome.

4.2. Maintaining Logical Closure

Because of the direct correspondence, we will use the RML notation and terminology for the fluent atoms in F. The encoding, thus far, is a straightforward adaptation of the RP-MEP definition that hinges on two properties: (1) there is a finite bound on the depth of nested belief; and (2) we restrict ourselves to representing RMLs and not arbitrary formulae. Crucially, however, we wish to maintain the assumption that the agents are internally consistent with respect to KD_n, and would like to do so in a pragmatic/syntactic manner. To accomplish this, we define a systematic closure procedure, Cl, that deduces a new set of RMLs from an existing one under KD_n, allowing us to pre-compute all of the necessary inferences *in advance of planning*:

Definition 7. RML Closure

Given an RML l, we define $Cl(l)$ to be the set RMLs that are KD_n logical consequences of l. $Cl(l)$ is computed by repeatedly applying the D axiom ($\Box_i\psi \supset \Diamond_i\psi$) to the RML, resulting in the set of all RMLs that follow logically from ϕ using the D axiom. This can be done by simply replacing all combinations of occurrences of $\Box_i$ with $\Diamond_i$; e.g., for the RML $\Box_i\Diamond_j\Box_k p$, the resulting set would be $\{\Diamond_i\Diamond_j\Box_k p,\ \Box_i\Diamond_j\Diamond_k p,\ \Diamond_i\Diamond_j\Diamond_k p\}$.

Similarly, the closure of a PEKB is simply the union of the closure of its elements, i.e., $Cl(P) = \bigcup_{l\in P} Cl(l)$.

Given the finite nature of a single RML, the closure is also finite.

Theorem 1 (Soundness & completeness of Cl)**.** Given a PEKB P in KD_n, $Cl(P) = \{\varphi \mid P \vDash \varphi$, where φ is an RML$\}$.

Proof. As shown by Miller and Muise [48], a PEKB in KD_n is *logically separable*. Intuitively, this means that we cannot infer any new RML from a PEKB by combining two RMLs taken from the PEKB that we cannot already infer from just one RML. Therefore, to compute $Cl(P)$ we can just apply the axioms K and D to individual RMLs in the PEKB P. The repeated application of axioms

K and D to each of the RMLs in the PEKB, when computing Cl, is a sound inference. Note that only a finite number of applications are possible given the finite nesting of each RML and the finite size of the PEKB.

For completeness, note that because any PEKB P is logically separable, we can compute $\{\varphi \mid P \vDash \varphi\}$, where φ is an RML, by taking the union of $\{\varphi \mid l \vDash \varphi\}$ for each $l \in P$. We can then prove via induction that each φ will be in $Cl(l)$:

Case $l \equiv \Box_i\psi$: Axiom D results in $\Diamond_i\psi$, which is an RML because ψ is an RML. Further, $\Diamond_i\psi$ is in $Cl(l)$ from the definition of Cl (Definition 7). By induction, any further application of KD_n axioms to $\Diamond_i\psi'$ are in $Cl(l)$. Because ψ is itself an RML and cannot contain disjunction or implication, axiom K does not apply.

Case $l \equiv \Diamond_i\psi$: Axiom D does not apply directly, and by induction, further application of KD_n axioms to $\Diamond_i\psi$ will be in $Cl(l)$. As with the previous case, axiom K does not apply.

Therefore, Cl is both sound and complete. □

Along with the requirement that an agent should never believe an RML and its negation, we have two further constraints on the PEKBs that we are encoding:

$$\phi \in s \Rightarrow \neg\phi \notin s$$

$$\phi \in s \Rightarrow \forall\psi \in Cl(\phi), \psi \in s$$

The enforcement of such state constraints can either be achieved procedurally within the planner, or representationally. We choose the latter, appealing to a solution to the well-known ramification problem (e.g., [60, 43]), representing these state constraints as *ancillary conditional effects* of actions that enforce the state constraints. The correctness of the resulting encoding is predicated on the assumption that the domain modeller provided a consistent problem formulation (i.e., there are no inherent inconsistencies in the goal, initial state, or action effects). This mirrors the common assumption in modelling for planning

more generally. The ancillary conditional effects for operator o are as follows:

$$(\mathcal{C} \rightarrow l) \in \mathit{eff}_o^+ \Rightarrow (\mathcal{C} \rightarrow \neg l) \in \mathit{eff}_o^- \tag{1}$$

$$(\mathcal{C} \rightarrow l) \in \mathit{eff}_o^+ \Rightarrow \forall l' \in Cl(l),\ \ (\mathcal{C} \rightarrow l') \in \mathit{eff}_o^+ \tag{2}$$

Example 2. Returning to our example, consider the effect of agent 1 telling secret s_1 to agent 2. Assuming there is no positive or negative condition for this effect to fire, the effect would be $(\langle\emptyset,\emptyset\rangle \rightarrow \Box_2 s_1) \in \mathit{eff}^+$. Using (1) would create $(\langle\emptyset,\emptyset\rangle \rightarrow \neg\Box_2 s_1) \in \mathit{eff}^-$ and (2) would create $(\langle\emptyset,\emptyset\rangle \rightarrow \neg\Box_2\neg s_1) \in \mathit{eff}^+$. Subsequently, (1) would fire again creating $(\langle\emptyset,\emptyset\rangle \rightarrow \Box_2\neg s_1) \in \mathit{eff}^-$. We can see already, with this simple example, that effects may cascade to create new ones.

The second issue is to ensure the state remains consistent under KD_n when removing beliefs. If we remove an RML l, we should also remove any RML that could be used to deduce l. To compute the set of such RMLs, we use the contrapositive: $\neg l'$ will deduce l if and only if $\neg l$ deduces l' (i.e., $l' \in Cl(\neg l)$). We thus have the following additional conditional effects for operator o:

$$(\mathcal{C} \rightarrow l) \in \mathit{eff}_o^- \Rightarrow \forall l' \in Cl(\neg l),\ (\mathcal{C} \rightarrow \neg l') \in \mathit{eff}_o^- \tag{3}$$

Example 3. Consider the effect of an action informing us that agent 2 should no longer believe that agent 1 does not believe agent 2's secret: $(\langle\emptyset,\emptyset\rangle \rightarrow \neg\Box_1 s_2) \in \mathit{eff}^-$. Using (3), we would have the additional effect $(\langle\emptyset,\emptyset\rangle \rightarrow \Box_1\neg s_2) \in \mathit{eff}^-$. If $\Box_1\neg s_2$ remained in our knowledge base, then so should $\neg\Box_1 s_2$ assuming that our knowledge base is deductively closed.

The effect of these two rules together is to encode the problem such that entailment is effectively done by computing the deductive closure of the PEKB each time an action effect is evaluated, matching the definition of progression in Section 3. All possible RMLs are pre-compiled into a planning problem, and during planning, these are used multiple times, meaning that the pre-compilation is a valuable step. Given the indexing used in most modern planners, entailment for checking preconditions and effect conditions in planning actions becomes a constant-time operation.

4.3. Uncertain Firing

To complete the faithful transformation of a RP-MEP problem to a FOND problem, we must also consider the third case of RP-MEP progression defined in Definition 4: when an agent is uncertain whether a conditional effect should fire due to the uncertainty of beliefs.

To encode this, we appeal to a common technique in planning under uncertainty (e.g., [59, 55]): when the conditions of a positive conditional effect are not believed to be false, the negation of the effect's result can no longer be believed. Intuitively, if an agent is unsure whether a conditional effect fires then it must consider the condition's effect possible, and thus no longer believe the negation of the effect. We create the following additional conditional effects:

$$(\langle \mathcal{C}^+, \mathcal{C}^- \rangle \rightarrow l) \in \mathit{eff}_o^+ \Rightarrow$$
$$(\langle \emptyset, \{\neg\phi \mid \phi \in \mathcal{C}^+\} \cup \mathcal{C}^- \rangle \rightarrow \neg l) \in \mathit{eff}_o^- \tag{4}$$

Example 4. Consider a conditional effect for the action of agent 1 sharing their secret that stipulates that if agent 2 believes that agent 1 is trustworthy (denoted as $\Box_2 t_1$), then agent 2 would believe agent 1's secret: $(\langle \{\Box_2 t_1\}, \emptyset \rangle \rightarrow \Box_2 s_1) \in \mathit{eff}^+$. Using (4), we would derive the new negative effect $(\langle \emptyset, \{\neg\Box_2 t_1\} \rangle \rightarrow \neg\Box_2 s_1) \in \mathit{eff}^-$. Rule 3 would then fire, which would remove $\Box_2 \neg s_1$. Intuitively, although agent 2 is unsure about agent 1's trustworthiness, it should no longer believe that the secret is false.

In what follows, we denote by $s' = \mathit{result}(s, \langle \mathit{eff}_o^+, \mathit{eff}_o^- \rangle)$ the new state resulting from the application of a non-deterministic effect $\langle \mathit{eff}_o^+, \mathit{eff}_o^- \rangle$ of an operator o, as formally defined in Section 2.3. With this notation at hand, we give the following main result.

Theorem 2. Given an RP-MEP action $a = \langle \mathit{Pre}_a, \mathit{Eff}_a \rangle$ and $E \in \mathit{Eff}_a$, let $\langle \mathit{Pre}_o, \mathit{Eff}_o \rangle$ be the operator corresponding to a and $E_o = \langle \mathit{eff}_E^+, \mathit{eff}_E^- \rangle$ its conditional effects corresponding to E, which are obtained according to Definition 6 and rules (1)-(4). Then, for any consistent PEKB P we have that $Q = \mathit{Progress}\,(P, E)$ iff $\mathit{Cl}(\mathcal{B}_\star(Q) \cup \mathcal{N}_\star(Q)) = \mathit{result}(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P), E_o)$.

Proof. First, recall that $I \stackrel{\text{def}}{=} \mathcal{B}_\star(Cl(\mathcal{I}))$ by Def. 6, hence $Cl(\mathcal{B}_\star(\mathcal{I}) \cup \mathcal{N}_\star(\mathcal{I})) = I$ as $\mathcal{N}_\star(\mathcal{I})$ is empty for any RP-MEP problem (the initial theory is restricted to positive RMLs only – see Section 3.1). Also, note that the encoded FOND problem is such that states are in fact always closed, hence we can consider $Cl(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P))$ in the above theorem. Consider a l_ϕ for which the equality in the theorem does not hold. Since by Def. 6 $l_\phi \in \mathcal{B}_\star(P)$ iff $\Box_\star\phi \in P$ (and similarly for $\mathcal{N}_\star$), Theorem 1 implies there are four cases to consider, which we prove by contradiction.

Assume l_ϕ is both in $Cl(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P))$ and in $result(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P), E_o)$, but not in $Cl(\mathcal{B}_\star(Q) \cup \mathcal{N}_\star(Q))$. Then by the definition of *Progress* (P, E) it must be that l_ϕ is in $(R \cup U)$ as defined in Def. 4, i.e., it is a fluent to be deleted due to a conditional effect of E or it is an uncertain firing fluent. Then we have that $(\gamma, \neg\Box_\star\phi) \in E$ and $\Box_\star\phi \in P$, with $\gamma \in P$, or $(\gamma, \Box_\star\overline{\phi}) \in E$ and $\overline{\gamma} \notin P$. Then by the definition of eff^-_E in Def. 6, rules (3)-(4) and the definition of $result$, it must be the case that l_ϕ is not in $result(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P), E_o)$ either.

Assume now l_ϕ is in $Cl(\mathcal{B}_\star(Q) \cup \mathcal{N}_\star(Q))$ and $Cl(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P))$ but not in $result(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P), E_o)$. This means that $(\mathcal{C} \to l_\phi) \in \mathit{eff}^-_E(s)$ for some $\mathcal{C}$, with $s = Cl(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P))$. Then, by the definition of eff^-_E in Def. 6, it must be the case that $(\gamma, \neg\Box_\star\phi) \in E$. As a consequence, ϕ is in the set R as defined in Def. 4, which implies that it cannot be in $Cl(\mathcal{B}_\star(Q) \cup \mathcal{N}_\star(Q))$, by definition of $Q = \mathit{Progress}\,(P, E)$.

The cases in which l_ϕ is not in $Cl(\mathcal{B}_\star(Q) \cup \mathcal{N}_\star(Q))$ nor in $Cl(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P))$ but in $result(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P), E_o)$, or conversely not in $Cl(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P))$ nor in $result(\mathcal{B}_\star(P) \cup \mathcal{N}_\star(P), E_o)$ but in $Cl(\mathcal{B}_\star(Q) \cup \mathcal{N}_\star(Q))$ can be easily ruled out by the definition of eff^+_E in Def. 6 and the closure rule (2), as $Cl(P) = \{\varphi \mid P \vDash \varphi\}$. Indeed, all and only the positive conditional effect of a RP-MEP action are encoded as positive conditional effects in the FOND planning problem. □

With these extra conditional effects, we have a faithful encoding of the orig-

inal RP-MEP problem. In other words, a policy α will be found[1] for initial state $\mathcal{I}$ and goal $\mathcal{G}$: $\mathcal{I} \models [\alpha]\mathcal{G}$.

Theorem 3. Our encoding is sound and complete with respect to RP-MEP.

Proof. The proof is a straightforward extension of the proof for Theorem 2 and the fact that Definition 6 (together with the rules for ancillary conditional effects) faithfully encode *Progress* (P, E). Any policy that is derived using a sound and complete planner on the encoded problem will thus generate a sound and complete policy. □

5. Extensions

Given the core compilation, there are a variety of extensions we have explored to improve the task of domain modeling. Here, we present some of the key ones that have had the largest impact.

5.1. Conditioned Mutual Awareness

Our specification of a RP-MEP problem and the subsequent encoding into classical planning allow us to specify a rich set of actions. Unlike traditional approaches that compile purely ontic action theories into ones that deal with belief (e.g., the work on conformant planning by Palacios and Geffner [55]), we allow for arbitrary conditional effects that include nested belief both as conditions and as effects.

While expressive, manually encoding effects with nested belief can be involved due to the cascading of ancillary conditional effects. Here, we extend the scope of ancillary conditional effects to safely capture a common phenomenon in planning with nested belief: mutual awareness of the effects of actions.

Example 5. In our running example, if an agent enters a room, then we realize this as an effect: e.g., $(\langle\emptyset, \emptyset\rangle \rightarrow at_1_loc1) \in eff^+$. In many applications, other

[1] This assumes a sound and complete planning algorithm for the encoded problem.

agents may also be aware of this: e.g., $(\langle\emptyset,\emptyset\rangle \rightarrow \Box_2 at_1_loc1) \in \mathit{eff}^+$. Perhaps we wish to predicate this effect on the second agent believing that it is also in this room: e.g., $(\langle\{\Box_2 at_2_loc1\},\emptyset\rangle \rightarrow \Box_2 at_1_loc1) \in \mathit{eff}^+$. It is this kind of behaviour of conditioned mutual awareness that we would like to capture in a controlled and automated manner.

By appealing to ancillary conditional effects, we will create new effects from existing ones. We have already demonstrated the ancillary conditional effects required for a faithful encoding to adhere to the axioms and state constraints we expect from our agent. We extend this idea here to capture the appealing property of *conditioned mutual awareness*.

Definition 8. Conditioned Mutual Awareness

An RP-MEP planning action $a \in \mathcal{A}$ is a tuple $\langle Pre_a, \mu_i^a, \mathit{Eff}_a\rangle$, in which Pre_a and Eff_a are the same as in Definition 1, and $\mu_i^a \in F$ is the condition for agent i to be aware of the effects of action a. Note that μ_i^a can be a unique fluent that is either always believed or never believed by a given agent.

Intuitively, agent i is aware of every conditional effect of a only when agent i believes μ_i^a.

For a given set of fluents T, we define the shorthand $\Box_i T = \{\Box_i l \mid l \in T\}$ and $\neg\Box_i T = \{\neg\Box_i l \mid l \in T\}$ and model conditioned mutual awareness through the following two encoding rules for every agent $i \in Ag$ to derive new conditional effects:

$$(\langle\mathcal{C}^+,\mathcal{C}^-\rangle \rightarrow l) \in \mathit{eff}_a^+ \Rightarrow$$
$$(\langle\Box_i\mathcal{C}^+ \cup \neg\Box_i\mathcal{C}^- \cup \{\Box_i\ \mu_i^a\ \},\emptyset\rangle \rightarrow \Box_i l) \in \mathit{eff}_a^+ \tag{5}$$
$$(\langle\mathcal{C}^+,\mathcal{C}^-\rangle \rightarrow l) \in \mathit{eff}_a^- \Rightarrow$$
$$(\langle\Box_i\mathcal{C}^+ \cup \neg\Box_i\mathcal{C}^- \cup \{\Box_i\ \mu_i^a\ \},\emptyset\rangle \rightarrow \neg\Box_i l) \in \mathit{eff}_a^+ \tag{6}$$

Note that each form of ancillary conditional effect adds a new positive conditional effect. In the positive case, we believe that the agent i has a new belief

$\Box_i l$ if we believe that agent i had the prerequisite belief for the effect to fire. In the negative case, we would believe that the agent no longer holds the belief, but because we take a perspectival view, it is encoded as a positive conditional effect – i.e., we would believe $\neg\Box_i l$. For instance, the ancillary conditional effect from our working example says that we should no longer believe the negation of agent 1's secret if we do not believe agent 1 is untrustworthy (see Example 4), which would create the following ancillary conditional effect:

$$(\langle\emptyset, \{\neg t_1\}\rangle \rightarrow \neg s_1) \in \mathit{eff}^- \Rightarrow$$
$$(\langle\{\neg\Box_2\neg t_1\}, \emptyset\rangle \rightarrow \neg\Box_2\neg s_1) \in \mathit{eff}^+.$$

We restrict the application of the above rules by applying them only if the following two conditions are met: (1) every RML in the newly created effect has a nested depth smaller than or equal to our bound d; and (2) if we are applying the above rule for agent i to a conditional effect $(\mathcal{C} \rightarrow l) \in \mathit{eff}_o^-$, then $l \notin \{\Box_i l', \neg\Box_i l'\}$. The first restriction bounds the number of conditional effects while the second prevents unwanted outcomes from introspection. To see why this exception is required, consider the example of a pair of conditional effects for an action where we discover agent 1 may or may not believe s_2 (i.e., we should forget any belief about what agent 1 believes regarding s_2). Omitting μ_1^o for clarity, we have the following *negative* conditional effects:

$$(\langle\emptyset, \emptyset\rangle \rightarrow \neg\Box_1 s_2) \quad (\langle\emptyset, \emptyset\rangle \rightarrow \Box_1 s_2)$$

If we were to apply the above rules with agent 1, we would add two *positive* ancillary conditional effects:

$$(\langle\emptyset, \emptyset\rangle \rightarrow \neg\Box_1\neg\Box_1 s_2) \quad (\langle\emptyset, \emptyset\rangle \rightarrow \neg\Box_1\Box_1 s_2)$$

which subsequently would simplify to the following conditional effects (given that we combine successive modalities of the same agent index under KD_n):

$$(\langle\emptyset, \emptyset\rangle \rightarrow \Box_1 s_2) \quad (\langle\emptyset, \emptyset\rangle \rightarrow \neg\Box_1 s_2)$$

Thus, the resulting effects would indicate that the agent reaches an inconsistency with its own belief. To avoid this issue, we apply rule (6) only when the effect is not a belief (negative or positive) of the corresponding agent.

Because we can assume that conditioned mutual awareness is given and computed in the original RP-MEP specification, Theorem 3 continues to hold.

5.2. Always Known Fluents

Some of the time, some things are universal. This is the philosophy behind our powerful second extension: *always known fluents* (or AK fluents for short). These specially designated fluents are such that *every* agent *always* knows their value (true or false), and it is common knowledge among all agents that this is the case. Having this capability allows us to effectively isolate the epistemic fragment of a planning problem from the rest of the combinatorics involved – AK fluents behave just as regular fluents in planning and require no special treatment to capture nested belief surrounding them. As a common example, static fluents describing the layout of a map are always commonly known among the agents, and need not have nested belief associated with them.

Frequently, syntactic sugar for a planning language simplifies the formulation of a model but does not result in an improved efficiency for solving. This is not the case with AK fluents. Not only does it simplify the modeling – allowing the domain designer to focus on only those aspects which should be epistemic in isolation – it further improves the planner's capability of solving the problem because we do not require encoding new fluents for the nested belief of AK fluents, nor do we require additional effects to handle the nested beliefs. There is no impact on the space of valid plans, but without the AK treatment, the classical planner must maintain the nested belief of all agents for these fluents which is largely redundant. We show how AK fluents are incorporated to seamlessly work with the encoding presented in Section 4, and the following

updated version of Definition 6 highlights the key differences in **bold** font.

Definition 9. Encoding of RP-MEP with AK Fluents

Given a RP-MEP problem, $\langle \mathcal{P}, \mathcal{A}, Ag, \mathcal{I}, \mathcal{G} \rangle$ where $\mathcal{P} = \mathcal{P}_{reg} \cup \mathcal{P}_{AK}$ is composed of regular fluents $\mathcal{P}_{reg}$ and AK fluents $\mathcal{P}_{AK}$ (where $\mathcal{P}_{reg} \cap \mathcal{P}_{AK} = \emptyset$), and a bound d on the depth of nested belief we wish to consider, we define the classical encoding as the tuple $\langle F, I, G, O \rangle$ such that:

$$\boldsymbol{AK(KB)} \stackrel{\textbf{def}}{=} \boldsymbol{\{l_\phi \mid \phi \in (KB \cap \mathcal{P}_{AK})\}}$$

$$\boldsymbol{\overline{AK}(KB)} \stackrel{\textbf{def}}{=} \boldsymbol{\{l_\phi \mid \neg\phi \in KB} \text{ and } \boldsymbol{\phi \in \mathcal{P}_{AK})\}}$$

$$F \stackrel{\text{def}}{=} \{l_\phi \mid \phi \in \mathcal{L}^{Ag,d}_{RML}(\mathcal{P}_{reg})\} \cup \boldsymbol{AK(\mathcal{P})}$$

$$I \stackrel{\text{def}}{=} \mathcal{B}_\star(Cl(\mathcal{I})) \cup \boldsymbol{AK(Cl(\mathcal{I}))}$$

$$G \stackrel{\text{def}}{=} \mathcal{B}_\star(\mathcal{G}) \cup \boldsymbol{AK(\mathcal{G})}$$

and for every action $\langle Pre_a, \mathit{Eff}_a \rangle$ in $\mathcal{A}$, we have a corresponding operator $\langle Pre_o, \mathit{Eff}_o \rangle$ in O such that:

$$Pre_o \stackrel{\text{def}}{=} \mathcal{B}_\star(Pre_a) \cup \boldsymbol{AK(Pre_a)}$$

and for every set of conditional effects $E \in \mathit{Eff}_a$:

$$\mathit{eff}^+_E \stackrel{\text{def}}{=} \{(\langle \mathcal{B}_\star(\gamma) \cup \boldsymbol{AK(\gamma)}, \overline{\mathcal{N}_\star(\gamma)} \cup \boldsymbol{\overline{AK}(\gamma)} \rangle \rightarrow l_\phi) \mid (\gamma, \Box_\star\phi) \in E\} \cup$$
$$\{(\langle \boldsymbol{AK(\gamma)}, \boldsymbol{\overline{AK}(\gamma)} \rangle \rightarrow l_\phi) \mid \boldsymbol{(\gamma, \phi) \in E} \textbf{ and } \boldsymbol{\phi \in \mathcal{P}_{AK}}\}$$

$$\mathit{eff}^-_E \stackrel{\text{def}}{=} \{(\langle \mathcal{B}_\star(\gamma) \cup \boldsymbol{AK(\gamma)}, \overline{\mathcal{N}_\star(\gamma)} \cup \boldsymbol{\overline{AK}(\gamma)} \rangle \rightarrow l_\phi) \mid (\gamma, \neg\Box_\star\phi) \in E\} \cup$$
$$\{(\langle \boldsymbol{AK(\gamma)}, \boldsymbol{\overline{AK}(\gamma)} \rangle \rightarrow l_\phi) \mid \boldsymbol{(\gamma, \neg\phi) \in E} \textbf{ and } \boldsymbol{\phi \in \mathcal{P}_{AK}}\}$$

Note that the AK fluents can be used as conditions for regular effects, but if they are changed as part of any effect then only AK fluents may appear as conditions. We make this assumption as part of the AK extension, as otherwise uncertainty could propagate from non-AK fluents to AK fluents. With the restriction in place, every agent will know the true value of every fluent in $\mathcal{P}_{AK}$ (and all have common knowledge that this is the case).

Generally speaking, we maintain the property that AK fluents are treated as if they are common knowledge, and use the absence of any particular AK fluent $\phi \in \mathcal{P}_{AK}$ in the state to represent the fact that ϕ is commonly known among the agents to be false.

Aside from the core encoding, we also must consider how the fluents are treated in the definition of ancillary conditional effects. Depending on the type of effect, and where the AK fluents exist, we have the following:

- Ancillary effects (1)-(3) are applied as normal if the literal being changed is from $\mathcal{P}_{reg}$, and not applied at all if it is from $\mathcal{P}_{AK}$.
- For ancillary effect (4), if the original effect is adding a $\mathcal{P}_{AK}$ literal, then it does not need to fire (recall that the negation of a literal from $\mathcal{P}_{AK}$ will never appear in the state, as the absence of the positive literal means that it's commonly known to be false). If, on the other hand, the original effect is adding a $\mathcal{P}_{reg}$ literal, then the ancillary effect remains unchanged – negated $\mathcal{P}_{AK}$ literals may be placed in the $\mathcal{C}^-$ set as a result, but this is benign as they will never appear in the agent's knowledge base.
- Finally, ancillary effects (5) and (6) are modified to treat the $\mathcal{P}_{AK}$ literals uniquely, and can only be used for effects that add literals from $\mathcal{P}_{reg}$. The following is an updated version of (5), and (6) is analogous:

$$ak(\mathcal{C}) \stackrel{\text{def}}{=} \{l_\phi \mid l_\phi \in \mathcal{C} \text{ and } \phi \in \mathcal{P}_{AK}\}$$
$$\Box_i\mathcal{C} \stackrel{\text{def}}{=} \{\Box_i l_\phi \mid l_\phi \in \mathcal{C} \text{ and } \phi \in \mathcal{L}^{Ag,d}_{RML}(\mathcal{P}_{reg})\}$$

$(\langle \mathcal{C}^+, \mathcal{C}^- \rangle \to l) \in \mathit{eff}^+_o \Rightarrow$

$(\langle \Box_i\mathcal{C}^+ \cup \neg\Box_i\mathcal{C}^- \cup \Box_i\{\mu^o_i\} \cup \boldsymbol{ak(\{\mu^o_i\})} \cup \boldsymbol{ak(\mathcal{C}^+)}, \boldsymbol{ak(\mathcal{C}^-)} \rangle \to \Box_i l) \in \mathit{eff}^+_o$

Note that the condition for mutual awareness may be from $\mathcal{P}_{reg}$ or $\mathcal{P}_{AK}$.

It is worth reiterating the prevailing assumption regarding the literals from $\mathcal{P}_{AK}$: *if the positive literal is not in the agent's knowledge base, then it is commonly known to be false.* As a result, the above restrictions on the ancillary effects will never add the negation of a literal from $\mathcal{P}_{AK}$ as an effect.

We do not detail a formal proof analogous to that of Theorem 3, but generally the soundness and correctness can be seen as a direct result of treating the $\mathcal{P}_{AK}$ fluents as common knowledge among all of the agents, and the modifications listed above simply following that assumption through proper bookkeeping.

6. Action Examples

To give a better sense of what is possible with our epistemic planning framework, we detail a few commonly used action types in the context of nested belief. This list is by no means exhaustive, but simply serves to illustrate the modeling possibilities.

6.1. Partially and Fully Observable Ontic Actions

Consider a simple ontic action to turn the lights on, which can be captured as follows, which has the positive effect of enabling *lights_on*, and the negative effect of enabling *lights_off*:

$$\begin{aligned} Pre &= \{lights_off\} \\ \mu_i &= \top \\ Eff &= \{\langle eff^+, eff^- \rangle\} \\ (\langle \emptyset, \emptyset \rangle \rightarrow lights_on) &\in eff^+ \\ (\langle \emptyset, \emptyset \rangle \rightarrow lights_off) &\in eff^- \end{aligned}$$

Suppose instead we were interested in a general flip action. If we assume that *lights_on* is an AK fluent (otherwise, we would need negative preconditions), then we could use:

$$Pre = \emptyset$$
$$\mu_i = \top$$
$$Eff = \{\langle eff^+, eff^- \rangle\}$$
$$(\langle \{lights_on\}, \emptyset \rangle \rightarrow lights_on) \in eff^-$$
$$(\langle \emptyset, \{lights_on\} \rangle \rightarrow lights_on) \in eff^+$$

Note that both of these examples consider a fully observable environment, and so to capture partial observability, we could use:

$$Pre = \{lights_off\}$$
$$\mu_i = in_room_i$$
$$Eff = \{\langle eff^+, eff^- \rangle\}$$
$$(\langle \emptyset, \emptyset \rangle \rightarrow lights_on) \in eff^+$$
$$(\langle \emptyset, \emptyset \rangle \rightarrow lights_off) \in eff^-$$

in which the condition on mutual awareness is that the agent is in the room: $\mu_i = in_room_i$.

6.2. Public and Semi-private Announcements

Let us now consider the case of announcements. Interestingly, we capture both truthful and untruthful announcements in a simple fashion. For example, the following captures a truthful public announcement that the door is open:

$$Pre = \{door_open\}$$
$$\mu_i = \top$$
$$Eff = \{\langle eff^+, \emptyset \rangle\}$$
$$\forall i \in Ag \setminus \{\star\}, (\langle \emptyset, \emptyset \rangle \rightarrow \Box_i door_open) \in eff^+$$

If we now remove the precondition (that is, set it to the empty set), the announcement could be an untruthful one.

Suppose we are now interested in making a a semi-private truthful announcement, in the sense that the announcement is heard by all agents present in a certain room, we could use:

$$\begin{aligned} Pre &= \{door_open\} \\ \mu_i &= in_room_i \\ Eff &= \{\langle eff^+, \emptyset\rangle\} \\ \forall i \in Ag, (\langle\{in_room_i\}, \emptyset\rangle &\rightarrow \Box_i door_open) \in eff^+ \end{aligned}$$

6.3. Yes/No Questions

Suppose we are interested in inquiring whether i believes the lights are on. Such an action is captured as follows:

$$\begin{aligned} Pre &= \emptyset \\ \mu_i &= \top \\ Eff &= \{\langle eff_1^+, \emptyset\rangle, \langle eff_2^+, \emptyset\rangle\} \\ (\langle\emptyset, \emptyset\rangle &\rightarrow \Box_i lights_on) \in eff_1^+ \\ (\langle\emptyset, \emptyset\rangle &\rightarrow \Box_i \neg lights_on) \in eff_2^+ \end{aligned}$$

Such interactions have appeared in works such as [51] on teamwork formation, where negotiation between agents is conducted through a series of yes/no questions similar to the one presented here.

7. Evaluation

In this section, we present an evaluation of our ideas over several planning domains — some inspired from previous literature and others in which our planner has been used to generate solutions. We evaluate over a series of parameters, including number of agents and maximum depth of a restricted modal literal. The vast majority of domains explore settings where inconsistent or incorrect

belief play a role – a key strength of our planner compared to many epistemic reasoning systems available.

We implemented the scheme above to convert a RP-MEP planning problem into a classical planning problem, which can be subsequently solved by any planner capable of handling negative preconditions and conditional effects. The compiler consumes a custom format for the RP-MEP problems and can either simulate the execution of a given action sequence or invoke a classical planner built using a configuration of the LAPKT planning library [62]. The source code, benchmarks, and demo of the compilation process can be found online at:

`http://pdkb.haz.ca/`

Throughout this section, we will use the following notation:

- Ag: the set of the agents in the problem
- d: the maximum depth of nested reasoning
- F: the fluents in the encoding
- $\vec{o}$: the computed sequential plan

Further, to emphasize one key strength of our choice to build on top of classical planners, we delineate the difference between the planner originally used when we introduced the system [53] with the latest incarnation of the same planner. In some instances, we find a substantial improvement in computation efficiency, and this is directly attributed to using classical planning as a black-box technology: as the field progresses, so does the strength of our approach.

7.1. Thief

The *Thief* [45] problem is one of deception. A simplified version of a computer game, the thief agent must steal an item while avoiding being detected by guards. The thief agent can use actions to misdirect the guards' 'attention'. The actions in the domain model events in which the one party can make a noise and the other party notices; one can see the other from behind, learning of their

location; and both can see each other face-to-face, simultaneously learning of each others' locations. In the problem instances, the locations of each is initially unknown.

We have verified the model of the pre-existing *Thief* problem, and all of the existing queries considered in the previous literature posed to demonstrate the need for nested reasoning (e.g., those found in [45]) are trivially solved in a fraction of a second.

7.2. Corridor

The *Corridor* problem involves agents that can walk back and forth between rooms of a corridor, and state information that they believe to be true within earshot of the neighbouring rooms [37]. Specifically, an agent broadcasting proposition q in a room will then be believed by all agents in the same room as well as neighbouring rooms. As the setting is for a framework of common knowledge, the belief of proposition q is common among all agents that hear the statement, and the position of all agents is universally known as well.

This domain was modified to allow for the broadcasting agent to spread false belief (i.e., to lie about the secret). All those agents within earshot of the broadcast will adopt whatever is announced, and the goals contain a mix of agent belief over what is believed to be correct / incorrect.

We varied some of the discussed parameters and report on the results in Table 1 (the first Corridor problem corresponds to the one presented by Kominis and Geffner [37] with the exception of the ability to lie). As the results show, as the depth and number of agents increase, the compilation time increases exponentially, while the planning time is standard for a classical planning problem of this scale. There was little difference between the old and new planners, and only one column for plan size is included as the two planners coincided in this regard for every problem.

At the extreme end, we found the limit for both what the planning procedure and our preprocessing approach can accomplish. With 5 agents and depth 5, we have over 200k fluents created. The compilation to classical planning completes

$\|Ag\|$	d	$\|F\|$	$\|\vec{o}\|$	Time (s)			
				Solve$_{old}$	**Solve**	**Compile**	**Total**
3	1	54	8	0.01	0.03	0.10	0.13
5	1	62	8	0.01	0.03	0.14	0.17
7	1	70	8	0.01	0.03	0.18	0.20
3	3	558	8	0.02	0.04	0.78	0.82
5	3	2262	8	0.10	0.13	4.64	4.77
7	3	5950	8	0.60	0.68	15.65	16.33
3	5	18702	8	3.34	3.33	55.69	59.02
5	5	222262	MO	MO	MO	1776.10	MO
7	5	MO	MO	MO	MO	MO	MO

Table 1: Results for encoding and planning time for the Corridor problem.

(in just under half an hour), but the planner runs out of memory trying to solve the problem. Scaling further (with 7 agents), we are not even able to compile the problem. While it is useful to examine the limits of our approach, we should emphasize that the majority of interesting use cases we have found for planning with nested belief is restricted to depth 1-2.

7.3. Grapevine

As a more challenging test-bed, we modelled a setting that combines the *Corridor* problem [37] and the classic *Gossip* problem [23]. In the new problem, *Grapevine*, there are three rooms with all agents starting in the first. Every agent believes their own secret to begin with, and the agents can either move between rooms or broadcast a secret they believe (or its negation if they wish to lie). Movement is always observed by all, but through the use of conditioned mutual awareness the sharing of a secret is only observed by those in the same room. Unlike the corridor domain above, agents adopt a belief only if they do not already have an assumption about it. That is, we modelled the notion that an agent "cannot change their mind". This problem allows us to pose

a variety of interesting goals ranging from private communication (similar to the Corridor problem) to goals of misconception/deception in the agent's belief (e.g., $G = \{\Box_a s_b, \Box_b \neg \Box_a s_b\}$).

$\|Ag\|$	$\|g\|$	d	$\|F\|$	$\|\vec{o}\|_{old}$	$\|\vec{o}\|$	**Time (s)**			
						Solve$_{old}$	**Solve**	**Compile**	**Total**
4	2	1	116	4	4	0.07	0.08	0.81	0.89
4	4	1	116	6	6	0.10	0.07	0.83	0.91
4	8	1	116	16	12	0.06	0.08	0.84	0.93
4	2	2	628	6	5	2.31	1.39	13.37	14.75
4	4	2	628	7	7	12.13	1.37	13.24	14.61
4	8	2	628	22	27	216.15	2.35	13.02	15.37
8	2	1	340	4	4	1.02	0.65	5.49	6.14
8	4	1	340	9	11	2.45	1.95	5.47	7.42
8	8	1	340	16	16	2.36	0.83	5.52	6.35
8	2	2	4436	6	5	350.23	309.79	253.69	563.48
8	4	2	4436	12	9	650.78	303.81	260.27	564.07
8	8	2	4436	TO	17	TO	337.48	254.20	591.69

Table 2: Results for encoding and planning time for the Grapevine problem.

Table 2 shows the results of our system in this more involved setting. We additionally report the size of the goal posed to the planner as $|g|$. Note that the results on plan length between the old and new planner are roughly similar, but there are a number of problems were the new planner finds the solution in far less time (e.g., two orders of magnitude improvement on the 6th problem).

We discuss the most related epistemic planners in Section 8, and more importantly why they cannot be used as a comparison on these problems. However, unlike the Corridor domain above and the Grid domain below, the Grapevine domain can be modelled in the language used by the newly introduced epistemic planner EFP2.0 [24]. Table 3 shows a comparison for the first 6 problems (those

with 8 agents are not solvable due to memory violation). The resource limits for this evaluation were 1 hour and 32Gb of memory using a stronger machine than the other evaluations.[2] The plan lengths were equivalent, and the discrepancy on this measure for lines 4-5 stem from the additional "initialize" action that is used in the RP-MEP encoding.

$\|Ag\|$	$\|g\|$	d	$\|\vec{o}\|$		Time (s)		
			RPMEP	EFP2	RPMEP	EFP2	EFP2$_{simp}$
4	2	1	4	4	0.46	39.40	0.67
4	4	1	6	6	0.47	2698.30	100.22
4	8	1	12	TO	0.46	TO	-
4	2	2	5	4	9.28	48.43	0.91
4	4	2	7	6	9.26	3413.02	14.37
4	8	2	27	TO	9.76	TO	-

Table 3: Results for encoding and planning time for EFP2.0 on the Grapevine problem.

The column for "simplified" demonstrates the solve time if the problem is simplified so fluents irrelevant to the goal are manually pruned. This is a fairly straightforward process to approximate, and captures the performance if this simple preprocessing were done automatically.

We find that as the general plan length grows, as driven by the complexity of the goals and nesting used to achieving them, the scalability of EFP2.0 suffers. In some sense, this is to be expected given the richness of the formalism handled by EFP2.0: complex formulae (including those with disjunction) can be used which is not the case for RP-MEP. Further, the representation is built using accessibility relations which means that scaling of nested depth has little to no impact on the performance.

This domain serves as a prime example of the orthogonal nature of RP-MEP

[2]We thank the lead author of EFP2.0, Francesco Fabiano, for help running this evaluation.

and other epistemic planners capable of handling false belief: the scalability and efficiency seen in RP-MEP comes from its ability to reason over complex action theories in large state spaces, while its limitation stems from the depth of nesting and type of formulae that is permitted.

7.4. Selective Communication

This is a collection of domains and problems in the area of *selective communication* [66]. In a multi-agent system, selective communication is the task of deciding when and what to communicate between agents to improve the outcomes of a collective task. For example, in a disaster response scenario, the environment will be partially-observable and initially, much of it will be unknown. As individual agents survey the area, they will obtain knowledge about the domain and can communicate this information to update other agents. However, communicating all information can be non-optimal if communication comes at a cost; e.g. if some agents are human who can become easily overwhelmed, or if there is a cost of communicating, such as a risk of giving away one's location in an adversarial environment. Using epistemic planning allows us to model communication as a natural action: communicating knowledge/belief to another agent updates their beliefs. We further consider the possibility of false beliefs propagating (e.g., through the malicious spread of misinformation or faulty communication).

The selective communication domains originally come from Alshehri [3], and define five different domains in which survivors must be located in a set of rooms and brought back to a medical area. The five domains are: (1) a simple scenario in which there is an unknown number of survivors whose location is initially unknown, and each survivor's location must be believed by at least one agent (distributed belief); (2) same as scenario 1 except that all agents must believe that all other agents believe the locations of all survivors (higher-order epistemic goal); (3) same as 1 except agents can broadcast belief and a single agent (the commander) needs to believe the location of all survivors, while all other agents can be ignorant of their location; (4) same as 2 except communication is one-

to-one instead of using a broadcast; and (5) same as 1 except in a cluttered environment with unknown obstacles preventing certain plans, and so sharing belief about these plans can improve the agents' navigation.

The five different domains have several problems, by varying the number of agents and number of rooms to search, based on the well-known *Blocks World For Teams* (BW4T) scenario [35]. In our evaluation, we take only the selective communication model outlined by Alshehri [3], omitting the two baselines of no communication and communicating all new information. Actions include moving between rooms, sensing survivors (epistemic), communicating with other agents (epistemic action), and transporting survivors back to the medical area. All problems in this set make use of 'always knowing' fluents (cf. Section 5.2) to model static parts of the problem, such as room IDs / locations.

Table 4 shows the results for these. The scenario labels **!epgoal**, **epgoal**, **broad**, **!broad**, and **blocked** respectively correspond to scenarios (1)-(5) described above. As with the other domains, we see that the compilation takes a bulk of the time. However, what this shows different to the Corridor and Grapevine problems is that the size of the compilation does not affect the classical planner. That is, given a compilation, long, non-trivial plans can be generated, in most case in under one second, despite the larger state space. We also find a substantial improvement in planner performance compared to the old system used.

7.5. Hattari

Hattari is a board game where partial information and nested belief plays a crucial role in the strategy and game mechanics [54]. Players secretly look at a card with a unique number written on it, and then pass the card to the right for the next person to look at. After this initial phase, there is common knowledge shared among every successive pair of players, and the game unfolds through a mix of deduction and bluffing centered around the ability for individuals to reason about nested belief.

Scenario	$\|Ag\|$	$\|F\|$	$\|\vec{o}\|_{old}$	$\|\vec{o}\|$	Time (s)			
					Solve$_{old}$	**Solve**	**Comp.**	**Total**
!epgoal: 1	3	594	15	15	0.03	0.05	0.70	0.75
!epgoal: 2	3	864	21	21	0.05	0.07	1.11	1.18
!epgoal: 3	4	720	14	14	0.05	0.07	1.10	1.16
!epgoal: 4	4	1032	20	20	0.07	0.09	1.78	1.87
epgoal: 1	3	594	37	42	17.52	0.08	0.70	0.78
epgoal: 2	3	864	46	53	10.88	0.13	1.14	1.27
epgoal: 3	4	720	61	36	659.21	10.67	1.09	11.76
epgoal: 4	4	1032	65	42	1607.87	18.29	1.77	20.06
broad: 1	3	600	19	19	0.08	0.12	0.59	0.71
broad: 2	3	600	19	19	0.03	0.05	0.89	0.93
broad: 3	3	870	25	25	0.07	0.06	1.23	1.29
broad: 4	4	1040	25	25	0.06	0.09	1.84	1.93
!broad: 1	3	732	23	23	0.13	0.14	6.18	6.32
!broad: 2	3	732	23	23	0.13	0.15	6.39	6.54
!broad: 3	4	1264	30	30	0.61	0.60	25.54	26.14
!broad: 4	4	1264	30	30	0.61	0.60	25.31	25.91
blocked: 1	3	1008	50	51	1.39	0.13	2.81	2.93
blocked: 2	3	1416	58	55	0.23	0.23	5.12	5.35
blocked: 3	4	1242	50	53	2.68	0.20	4.68	4.88
blocked: 4	4	1728	TO	73	TO	0.59	8.57	9.16

Table 4: Results for encoding and planning time for the Selective Communication problem.

We modeled the mechanics of the game using the framework presented in this paper, and embedded the belief maintenance as part of a web application:[3]

[3]Source code for the site: `https://bitbucket.org/jekegren/hattari-project`

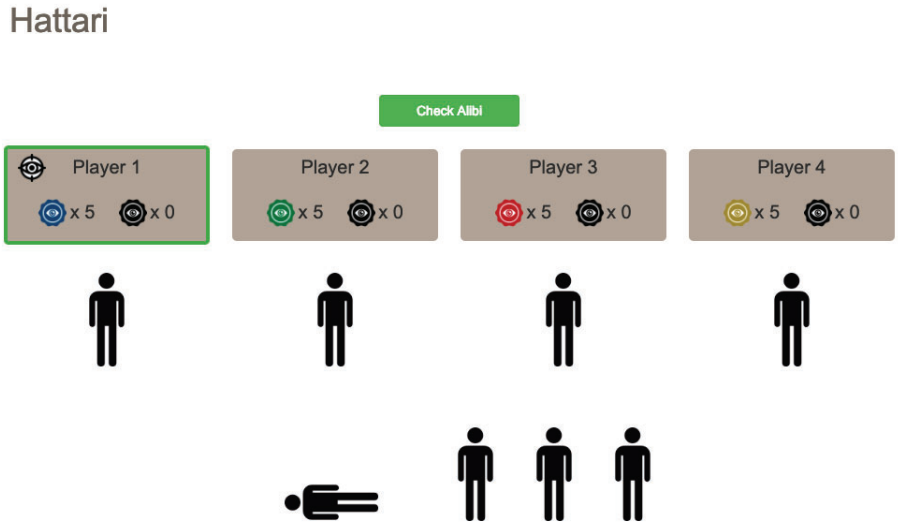


Figure 1: Screenshot of the online Hattari game

`http://hattari.haz.ca/`

A screenshot of the online game can be seen in Figure 1.

As planning is not a component to this scenario, we do not report on the evaluation time (which was negligible for the task of belief update). However, to get a sense of the encoding for the game, the following action demonstrates the perspectival model for agent ag looking at a card c:

$$\mu_i = \top$$
$$Eff = \{\langle eff^+, \emptyset \rangle\}$$
$$(\langle \{me_{ag}\}, \emptyset \rangle \rightarrow holding_ag_c) \in eff^+$$
$$(\langle \{holding_ag_c\}, \emptyset \rangle \rightarrow \Box_{ag} holding_ag_c) \in eff^+$$

We have the following properties of the action:

- The precondition is not listed, as we are only doing belief maintenance and not planning (the game logic implemented elsewhere dictates what actions are possible to execute).

- The condition for mutual awareness is true, meaning that nested ancillary effects of the form in equation (5) will apply.
- me_{ag} is a special fluent believed only by agent ag.
- The perspective of this action may not be from ag's point of view.

The last point is of particular interest: when the perspective is from another agent, the first effect will not fire (as me_{ag} is not believed), but the second one may. Other actions in the domain cause the cards to be rotated between players, and so a sequence such as $[look_ag_c, pass_cards, look_ag2_c]$ from the perspective of ag will cause the belief base to first contain $holding_ag_c$ from the first effect, and then $holding_ag2_c$ from the effects of the $pass_cards$ action, and then $\Box_{ag2} holding_ag2_c$ from the second effect of the third action.

This basic form of epistemic update demonstrates the scope of what our framework for epistemic planning can capture.

7.6. Results Discussion

The results in this section demonstrate that our approach can solve a wide variety of epistemic planning scenarios, from toy problems in the literature to larger-scale problems that contain longer plans. The trend shown in the results, though, is that the compilation time is the bottleneck in the approach; increasing exponentially in the number of agents and the depth of reasoning. The planning process is typically fast, and moving forward we hope to reduce the compilation time by only generating fluents and conditional effects that are relevant to achieving the goal.

8. Related Work

8.1. Comparison to Other Planners

We contrast our work with the most closely related epistemic planners. In particular, two planners – EFP and MEPK – have the theoretical capability to

handle the problems explored in Section 7. However, in practice, modelling and solving those settings are not feasible for a variety of reasons we detail here.[4]

8.1.1. EFP

A recent approach to epistemic planning is the EFP planner [41]. The authors present a pair of planners for generating multi-agent epistemic plans: EFP and PG-EFP. Both are forward-search planners, differing with respect to their search mechanisms. EFP uses breadth-first search, while PG-EFP uses heuristic search based on a novel epistemic plan graph. These planners were originally inspired by and built on the ideas presented in this paper.

Similar to RP-MEP, they syntactically restrict the epistemic formulae they can represent, however the encoding is different. The authors compare their planners to the Huang et al. [33] planner and a version of RP-MEP that lacked the support for AK fluents, all on domains that required only notions of knowledge and not belief. Performance varies on different benchmarks. In general, RP-MEP demonstrates superior performance at shallow depth, but as d increases, EFP planners demonstrate superior performance. This is because the size of the RP-MEP problem grows exponentially with increased depth, which is not the case with the EFP planners.

Most crucially, while the language EFP is built on the $m\mathcal{A}^*$ action language, which is capable of representing incorrect beliefs of agents, the implementation of EFP focuses exclusively on nested knowledge. Core aspects of the benchmarks explored in Section 7, such as agents that share false information, is thus something we cannot model in EFP1.0 for solver comparison. We should emphasize, however, that this is a restriction only with the implementation of EFP1.0 and not with the $m\mathcal{A}^*$ language itself. Further, while RP-MEP is restricted to formulae that are disjunction-free, the EFP system has no such restriction.

Very recently, the EFP1.0 planner was extended to handle a richer class of

[4]We would like to thank the authors of both systems for working with us to understand these core differences fully.

epistemic problems in the EFP2.0 system [24]. The new system builds on the previous by defining an improved transition function, and includes a host of planner improvements such as duplicate state detection and reduced state size representation. Unlike EFP1.0, the EFP2.0 implementation permits certain forms of inconsistent belief to be modeled. In particular, untruthful announcements are feasible, as long as it does not override prior belief of the agents that observe the announcement.

For the three main benchmarks explored in Section 7, only Grapevine follows this style of inconsistent belief.[5] The other two domains (Corridor and Grid) propagate incorrect belief in a manner that cannot yet be modelled by EFP2.0. On the other hand, the language handled by the EFP2.0 planner captures disjunction, and thus offers a far richer formalism in that regard. Further, as noted in the evaluation discussion, EFP2.0 is not impacted by the required depth of nested reasoning, whereas RP-MEP faces an exponential increase in the compiled representation size depending on the nested depth.

8.1.2. MEPK

The most closely related planner to our work is MEPK [33]. They take a different approach to epistemic planning by defining algorithms for belief revision and update over arbitrary $KD45_n$ formulae. Their approach exploits the use of *alternating cover disjunctive formulas* (ACDF), first defined by Hales et al. [27]. ACDF is used to represent the knowledge base and all other formulae in the planning problem, thus requiring a translation from arbitrary formula to ACDF. The length of an ACDF formula is single exponentially bound by the length of the original formula. Checking entailment between two ACDF formulae is untractable, so Huang et al. [33] introduce a stronger form of entailment, computable in polynomial time. This notion is then used to define polynomial-time belief revision and update operators. A standard search algorithm is then

[5] See the link below in the discussion of the MEPK planner for the EFP2.0 model for the first Grapevine problem as an illustrative example.

used to find plans, with the belief revision and update operators used to define progression. Similar to our notion of always known fluents, Huang et al. [33] support static common knowledge, which are propositional formulae that never change and are common knowledge to all agents. Our always known fluents are more flexible in one regard, as they can be arbitrarily modified during execution, however they cannot contain disjunction.

By supporting more expression formulae, Huang et al. [33] are effectively trading efficiency for expressiveness. Their planner can handle arbitrary KD_n formulae, while ours cannot, but their compilation in ACDF and search are more expensive than classical or FOND planning.

Unlike the majority of related work listed below, MEPK can handle true doxastic reasoning, and thus are able to handle notions such as agents that lie in communication. However, a full empirical comparison between the planners is not feasible due to the nature of our ancillary effects – in particular those for uncertain firing and conditioned mutual awareness.

Appendix C demonstrates the core ancillary effects, and the differences in representations used by RP-MEP and MEPK (PDKBDDL and EPDDL respectively). To further illustrate the incompatible nature, the following is the smallest problem in the grapevine domain for both languages (it also includes the format used by EFP2.0 as discussed above):

`http://editor.planning.domains/#read_session=UusNZTzIs3`

The PDKBDDL representation is under 100 lines in total, while the EPDDL is a semi-automated conversion from the compiled form of the PDKBDDL resulting in over 12,000 lines for the EPDDL encoding. The reason the EPDDL representation is so large is a consequence of having to compile all of the epistemic inferences into the model similar to RP-MEP. This is necessary since some key inference steps are not handled natively by the latest version of MEPK, and thus the full range of ramifications must be provided in the encoding.

RP-MEP takes under a second to solve the problem from start to finish including the preprocessing (see Section 7 for the details), while MEPK runs out

of memory in a matter of seconds (both with and without heuristics enabled). The expected plan length is 4, and the classical planner finds this in 0.06 seconds.

It should be emphasized that *this is by no means a fair comparison.* It is essentially asking MEPK to reason about a classical encoding with all of the advanced doxastic reasoning compiled into the domain; a comparison between the expressive MEPK planner and a state-of-the-art classical planner. Many of the ancillary effects we use are handled natively by MEPK, but must be expanded due to their interaction with the other ancillary effects as a chain of reasoning. We detail these further in Appendix C.

Conversely, there is no natural re-encoding that avoids this explosion in representation. The main reason is that some of the automatic inferences made by RP-MEP – namely the conditioned mutual awareness and uncertain firing effects – are interleaved with the remaining inferences that both planners handle. Thus, any effect added by RP-MEP's compilation that requires a chain of reasoning with at least one ancillary effect unique to RP-MEP will not be captured by the native MEPK reasoning. It must be encoded by hand.

Finally, we should emphasize that there is a large space of models that can be captured by MEPK that cannot be readily modelled and solved by RP-MEP. In particular, any domain that requires reasoning over disjunctions is something that distinguishes the two planners from one another. Additionally, the manner with which we model nested belief means that the scalability of RP-MEP is largely restricted to small depths, while MEPK does not face such issues.

Ultimately, the two planners offer complementary strengths, and the problems readily modelled in both PDKBDDL and EPDDL fail to exhibit the true strengths of either RP-MEP or MEPK. Combining the strengths of both approaches is an extremely lucrative area for future work.

8.2. Additional Related Work

There is a variety of research related to the ideas we have presented, which we briefly summarize below.

Reasoning about knowledge and belief has a long history in philosophy, but

has been gaining increasing prominence in computer science and AI [25] since the work of Kripke, Hintikka, Prior, among others [39, 31, 61]. In particular, at least since the eighties, the extension of epistemic logic to reason about actions has received considerable attention, in languages such as the situation calculus and propositional modal logic [49, 63, 19].

Our approach builds on the DEL family of languages. Research into DEL [19], and more recently DEL planning (e.g., [10]), deals with how to reason about knowledge or belief in a setting with multiple agents, in the interest of achieving individual or joint goals. Until recently, focus in this area has primarily been on the logical foundations (e.g., semantic considerations) for updating an epistemic state according to physical (ontic) and non-physical (epistemic) actions, as well as identifying the classes of restricted reasoning that are tractable from a computability standpoint [45]. A preliminary version of our article [53] was among the first approaches to consider the implementation perspective on epistemic planning. In particular, we leveraged intuitions from state-of-the-art approaches for automated planning with partial observability, discussed further below. This necessitated some restrictions on the specification language. While the full language of DEL is, therefore, clearly more expressive than our approach in terms of the logical reasoning that an agent can achieve in theory, this expressiveness increases the computational complexity of the reasoning. In particular, the practical synthesis of DEL plans remains a challenging problem. See, for example, [22] on notable progress on this front where the planner manipulates DEL models directly. We refer interested readers to international workshops such as [6] for recent advancements, as well as disparate communities that approach epistemic planning from the perspective of game-theoretic strategies, linguistics, and so on. (For example, outside of the reasoning about actions community, the treatment of knowledge and time has also received a lot of attention in the form of variants of Alternating Temporal Epistemic Logic, e.g., [56, 32].)

Also adopting the philosophy of trading off expressive power with computational efficiency, the EL-O framework takes a syntactic restriction on the general

problem of epistemic planning [17, 18, 16]. Similar to our approach, EL-O is restricted to conjunctions of epistemic literals and the representation is compiled to a planning formalism. The key distinctions between our work and that of Cooper *et al.* are: (1) they choose "knowing whether" as a primitive; (2) our style of compilation (i.e., using ancillary effects) is unique; and, most crucially, (3) we capture a form of doxastic reasoning whereas they capture a restricted form of S5 epistemic knowledge. This final point is a fundamental distinguishing factor between the approaches – but we note there are situations where either one would be the preferred setting.

As mentioned above, our technique was inspired by recent, state-of-the-art approaches to planning with partial observability [12, 11]. Of course, planning with partial observability is a flavor of epistemic planning – in that the uncertainty captures an implicit belief state – albeit a limited one in which beliefs are not nested. Approaches such as [11] consider the problem of how beliefs can be represented as classical states, by "compilation." Thus, from an epistemic planning standpoint, only individual knowledge of facts about the world (as opposed to belief) are handled: the agent can "know p holds" (i.e., Kp), "know p does not hold" (i.e., $K\neg p$), or "not know the value of p" (i.e., $\neg Kp \wedge \neg K\neg p$). (The multi-agent case is not considered.) The use of a knowledge modality was extended to be predicated on assumptions about the initial state, leading to effective techniques for conformant and contingent planning [55, 1].

Kominis and Geffner [37] also take their inspiration from this lineage, and is the work that is most related to ours. They too share the general motivation of bridging the rich fields of epistemic reasoning and automated planning by using classical planning over multi-agent epistemic states. However, the two approaches are fundamentally different and as a result each comes with its own strengths and shortcomings. The largest difference is our choice to focus on belief rather than knowledge – for us, modeling the possibility of incorrect belief is essential. In contrast, Kominis and Geffner [37] assume that all agents start with common initial knowledge, and further assume that all action effects are commonly known to all agents. (We can easily incorporate this setting as

a special case, but it is not necessary.) Conversely, they are able to handle arbitrary formulae, including disjunctive knowledge, while we are restricted to reasoning with RMLs. Moving forward, we hope to explore how we can combine ideas from both approaches.

The restriction on our specification language builds on a restricted fragment studied in [40]. They introduce so-called *Proper Epistemic Knowledge Bases* (PEKBs), where disjunctive knowledge is not permitted, enabling computationally tractable reasoning. We leverage that fragment in that the preconditions, goals, and states of our work can be viewed as PEKBs. In that spirit, approaches such as the 0-approximation semantics [7] are alternate candidates for achieving tractability in reasoning. See, for example, Son [64]. Our approach can be seen as having the "Epistemic Closed World Assumption" [65]. In essence, after compilation, the states contain explicit representation of everything that can be proved (restricted to our language).

9. Concluding Remarks

We have presented a model of planning with nested belief, and the key contribution of this paper is to show that epistemic planning within this model can be done efficiently.

We have demonstrated how a syntactically restricted class of problems involving planning with nested belief can be compiled into classical planning problems. Despite the restricted form, we are able to model complex phenomena such as public or private communication, commonly observed action effects, and non-homogeneous agents (each with their own view of how the world changes). Our focus on belief (as opposed to knowledge) provides a realistic framework for an agent to reason about a changing environment where knowledge cannot be presumed.

To solve this expressive class of problems, we appeal to existing techniques for dealing with ramifications, and compile the problem into a form that classical planning can handle. We show that our approach can solve a wide variety of

epistemic planning scenarios, from toy problems in the literature to larger-scale problems that contain longer plans – indeed, for problems that can be modeled in the language of other epistemic planners, we solve them more efficiently by orders of magnitude. Further, since we use classical planning as a black-box technology, as the field progresses, so does the strength of our approach. We have illustrated as such by comparing the performance of our approach using a modern classical planner versus one from only a few years ago.

In the future, we hope to expand the work in three key directions. First, we would like to explore other forms of ancillary conditional effects similar to the conditioned mutual awareness to give the designer greater flexibility during modelling (e.g., with concepts such as teamwork protocols or social realities). Second, we want to formalize the connection between general multi-agent epistemic planning and the syntactic restriction that we focus on encoding. We hope to provide an automated sound (but incomplete) approximation of an arbitrary MEP problem into a RP-MEP problem. Finally, we would like to increase the expressiveness of our problems, in particular, by introducing a restricted form of disjunctive reasoning allowing modellers to express the concept of an agent a *knowing whether* agent a knows p is true, without agent a knowing whether p is true or not; for example, agent a sees agent b look at the clock, so a knows that b knows whether it is passed midnight, but agent a does not know themselves. While more general disjunction is more expressive, it is more computationally challenging. 'Knowing whether' is by far the most common type of disjunctive knowledge we have encountered in our applications.

Acknowledgements

This research is partially funded by Australian Research Council Discovery Grant DP130102825, *Foundations of Human-Agent Collaboration: Situation-Relevant Information Sharing*, and by the Natural Sciences and Engineering Research Council of Canada (NSERC). We would like to also thank Biqing Fang, Yongmei Liu, Son Tran, and Francesco Fabiano for extensive help in understanding their systems and the differences between them and RP-MEP. We

also thank Maayan Shvo for his help in characterizing the space of existing epistemic planners.

References

[1] Albore, A., Palacios, H., Geffner, H., 2009. A translation-based approach to contingent planning, in: IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009, pp. 1623–1628.

[2] Alchourrón, C.E., Gärdenfors, P., Makinson, D., 1985. On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic 50, 510–530. URL: http://dx.doi.org/10.2307/2274239, doi:10.2307/2274239.

[3] Alshehri, A.D.G., 2017. Improving performance of multi-agent cooperation using epistemic planning. Master thesis. School of Computing and Information Systems, University of Melbourne.

[4] Aucher, G., Bolander, T., 2013. Undecidability in epistemic planning, in: IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, pp. 27–33.

[5] Baier, J.A., Mombourquette, B., McIlraith, S.A., 2014. Diagnostic problem solving via planning with ontic and epistemic goals, in: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR), pp. 388–397.

[6] Baral, C., Bolander, T., van Ditmarsch, H., McIlraith, S., 2017. Epistemic Planning (Dagstuhl Seminar 17231). Dagstuhl Reports 7, 1–47. doi:10.4230/DagRep.7.6.1.

[7] Baral, C., Son, T.C., 1997. Approximate reasoning about actions in presence of sensing and incomplete information, in: Logic Programming, Pro-

ceedngs of the 1997 International Symposium, Port Jefferson, Long Island, NY, USA, October 13-16, 1997, pp. 387–401.

[8] Bienvenu, M., 2008. Prime implicate normal form for ALC concepts, in: The 22nd AAAI Conference on Artificial Intelligence, pp. 412–417.

[9] Bienvenu, M., 2009. Prime implicates and prime implicants: From propositional to modal logic. Journal of Artificial Intelligence Research 36, 71–128.

[10] Bolander, T., Andersen, M.B., 2011. Epistemic planning for single and multi-agent systems. Journal of Applied Non-Classical Logics 21, 9–34. doi:10.3166/jancl.21.9-34.

[11] Bonet, B., Geffner, H., 2014. Belief tracking for planning with sensing: Width, complexity and approximations. Journal of Artificial Intelligence Research (JAIR) 50, 923–970. doi:10.1613/jair.4475.

[12] Brafman, R.I., Shani, G., 2012. Replanning in domains with partial information and sensing actions. Journal of Artificial Intelligence Research (JAIR) 45, 565–600. doi:10.1613/jair.3711.

[13] Brenner, M., Nebel, B., 2009. Continual planning and acting in dynamic multiagent environments. Autonomous Agents and Multi-Agent Systems 19, 297–331. doi:10.1007/s10458-009-9081-1.

[14] Camacho, A., McIlraith, S.A., 2016. Strong-cyclic planning when fairness is not a valid assumption, in: Proceedings of the Workshop on Knowledge-based Techniques for Problem Solving and Reasoning co-located with 25th International Joint Conference on Artificial Intelligence (IJCAI 2016), pp. 1–9.

[15] Chellas, B.F., 1980. Modal Logic - An Introduction. Cambridge University Press. URL: https://doi.org/10.1017/CBO9780511621192, doi:10.1017/CBO9780511621192.

[16] Cooper, M.C., Herzig, A., Maffre, F., Maris, F., Perrotin, E., Régnier, P., 2021. A lightweight epistemic logic and its application to planning. Artificial Intelligence 298, 103437.

[17] 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, IOS Press. pp. 193–201.

[18] Cooper, M.C., Herzig, A., Maris, F., Perrotin, E., Vianey, J., 2020. Lightweight parallel multi-agent epistemic planning, in: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, pp. 274–283.

[19] van Ditmarsch, H., van der Hoek, W., Kooi, B.P., 2007. Dynamic epistemic logic. volume 337. Springer.

[20] Eberle, R.A., 1974. A logic of believing, knowing, and inferring. Synthese 26, 356–382.

[21] van Eijck, J., 2004. Dynamic epistemic modelling. Manuscript, CWI, Amsterdam .

[22] Engesser, T., Bolander, T., Mattmüller, R., Nebel, B., 2017. Cooperative epistemic multi-agent planning for implicit coordination. arXiv preprint arXiv:1703.02196 .

[23] Entringer, R.C., Slater, P.J., 1979. Gossips and telegraphs. Journal of the Franklin Institute 307, 353–360. doi:http://dx.doi.org/10.1016/0016-0032(79)90004-8.

[24] Fabiano, F., Burigana, A., Dovier, A., Pontelli, E., 2020. EFP 2.0: A multi-agent epistemic solver with multiple e-state representations, in: Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020, AAAI Press. pp. 101–109.

[25] Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y., 1995. Reasoning about knowledge. volume 4. MIT press Cambridge.

[26] Ghallab, M., Nau, D., Traverso, P., 2004. Automated planning: theory & practice. Elsevier.

[27] Hales, J., French, T., Davies, R., 2012. Refinement quantified logics of knowledge and belief for multiple agents. Advances in Modal Logic 9, 317–338.

[28] Harper, W.L., 1976. Rational conceptual change, in: PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association, JSTOR. pp. 462–494.

[29] Haslum, P., Lipovetzky, N., Magazzeni, D., Muise, C., 2019. An Introduction to the Planning Domain Definition Language. Morgan & Claypool.

[30] Herzig, A., Rifi, O., 1999. Propositional belief base update and minimal change. Artificial Intelligence 115, 107–138.

[31] Hintikka, J., 1962. Knowledge and belief: an introduction to the logic of the two notions. Cornell University Press.

[32] van der Hoek, W., Wooldridge, M., 2002. Tractable multiagent planning for epistemic goals, in: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3, ACM, New York, NY, USA. pp. 1167–1174. doi:10.1145/545056.545095.

[33] 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 26th International Joint Conference on Artificial Intelligence, AAAI Press. pp. 1093–1101.

[34] Hughes, G.E., Cresswell, M.J., 1996. A new introduction to modal logic. Psychology Press.

[35] Johnson, M., Jonker, C., Van Riemsdijk, B., Feltovich, P.J., Bradshaw, J.M., 2009. Joint activity testbed: Blocks world for teams (bw4t), in: International Workshop on Engineering Societies in the Agents World, Springer. pp. 254–256.

[36] Katsuno, H., Mendelzon, A.O., 1991. On the difference between updating a knowledge base and revising it, in: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, pp. 387–394.

[37] 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.

[38] Konolige, K., 1983. A deductive model of belief, in: International Joint Conference on Artificial Intelligence, pp. 377–381.

[39] Kripke, S., 1959. A completeness theorem in modal logic. Journal of Symbolic Logic 24, 1–14.

[40] 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. doi:10.3233/978-1-61499-098-7-498.

[41] 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, pp. 161–170.

[42] Levi, I., 1978. Subjunctives, dispositions and chances, in: Dispositions. Springer, pp. 303–335.

[43] Lin, F., Reiter, R., 1994. State constraints revisited. J. Log. Comput. 4, 655–678. doi:10.1093/logcom/4.5.655.

[44] Liu, Y., Wen, X., 2011. On the progression of knowledge in the situation calculus, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, pp. 976–982.

[45] Löwe, B., Pacuit, E., Witzel, A., 2011. DEL planning and some tractable cases, in: Logic, Rationality, and Interaction - Third International Workshop, LORI 2011, Guangzhou, China, October 10-13, 2011. Proceedings, pp. 179–192. doi:10.1007/978-3-642-24130-7_13.

[46] Makinson, D., 1987. On the status of the postulate of recovery in the logic of theory change. Journal of Philosophical Logic 16, 383–394.

[47] 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.

[48] Miller, T., Muise, C.J., 2016. Belief update for proper epistemic knowledge bases, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pp. 1209–1215.

[49] Moore, R.C., 1985. A Formal Theory of Knowledge and Action, in: Formal Theories of the Commonsense World. Ablex, Norwood, NJ, pp. 319–358.

[50] Muise, C., Dignum, F., Felli, P., Miller, T., Pearce, A.R., Sonenberg, L., 2015a. Towards team formation via automated planning, in: International Workshop on Coordination, Organisation, Institutions and Norms in Multi-Agent Systems, Springer. pp. 282–299.

[51] Muise, C., Dignum, F., Felli, P., Miller, T., Pearce, A.R., Sonenberg, L., 2016. Towards team formation via automated planning. Lecture Notes in

Computer Science 9628, 282–299. Special Issue on Coordination, Organizations, Institutions, and Norms in Agent Systems XI.

[52] Muise, C., Miller, T., Felli, P., Pearce, A., Sonenberg, L., 2015b. Efficient reasoning with consistent proper epistemic knowledge bases, in: The International Conference on Autonomous Agents and Multiagent Systems, pp. 1461–1469.

[53] Muise, C.J., Belle, V., Felli, P., McIlraith, S.A., Miller, T., Pearce, A.R., Sonenberg, L., 2015c. Planning over multi-agent epistemic states: A classical planning approach., in: AAAI, pp. 3327–3334.

[54] Oink Games Inc., . In A Grove. `https://oinkgms.com/en/in-a-grove`. Accessed: 2019-02-22.

[55] Palacios, H., Geffner, H., 2009. Compiling uncertainty away in conformant planning problems with bounded width. Journal of Artificial Intelligence Research (JAIR) 35, 623–675. doi:`10.1613/jair.2708`.

[56] Penczek, W., Lomuscio, A., 2003. Verifying epistemic properties of multiagent systems via bounded model checking, in: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM. pp. 209–216.

[57] Perrussel, L., Marchi, J., Thévenin, J.M., Zhang, D., 2012. Relevant minimal change in belief update, in: European Workshop on Logics in Artificial Intelligence, Springer. pp. 333–345.

[58] Petrick, R.P.A., 2006. A Knowledge-level approach for effective acting, sensing, and planning. Ph.D. thesis. Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.

[59] Petrick, R.P.A., Levesque, H.J., 2002. Knowledge equivalence in combined action theories, in: Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002, pp. 303–314.

[60] Pinto, J., 1999. Compiling ramification constraints into effect axioms. Computational Intelligence 15, 280–307. doi:10.1111/0824-7935.00094.

[61] Prior, A., 1967. Past, present and future. Oxford University Press.

[62] Ramirez, M., Lipovetzky, N., Muise, C., 2015. Lightweight Automated Planning ToolKiT. http://lapkt.org/. Accessed: 2015-01-17.

[63] Reiter, R., 2001. Knowledge in action: logical foundations for specifying and implementing dynamical systems. volume 16. MIT press Cambridge.

[64] Son, T.C., 2017. Answer set programming and its applications in planning and multi-agent systems, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer. pp. 23–35.

[65] 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.

[66] Wu, F., Zilberstein, S., Chen, X., 2011. Online planning for multi-agent systems with bounded communication. Artificial Intelligence 175, 487–511.

Appendix A. Glossary of Key Terms

Acronym	Extended form	Sec.	Description
RML	restricted modal literal	2.2	literals of the form $\phi ::= p \mid \Box_i \phi \mid \neg\phi$
PEKB	proper epistemic knowledge base	2.2	a set of RMLs
PINF	prime implicate normal form	2.2	a PEKB representation with polynomial entailment
FOND	fully-observable non-deterministic planning	2.3	classical planning, with non-deterministic operators
MEP	multi-agent epistemic planning	3	planning problem with bounded-depth RMLs as fluents
RP-MEP	restricted-perspective multi-agent planning	3	a MEP with a root agent and no negated belief (RMLs $\neg\Box_i\phi$)
AK	always known fluents	5.2	there is common knowledge about the value of these fluents

Appendix B. Exemplary Domains

Here we detail two domains in the file format used by RP-MEP – the PDKB Domain Description Language (PDKBDDL). The language is a variant of the Planning Domain Definition Language (PDDL) [29] which allows for the expressiveness of nested agent belief, conditioned mutual awareness, always known fluents, etc. The first domain demonstrates some of the features of the language using the grapevine domain as an example. The second demonstrates the drawback of RP-MEP not being able to reason with disjunctive beliefs.

Appendix B.1. Grapevine

The following snippets of PDKBDDL demonstrate the features of the specification language beyond PDDL, and corresponds to the Grapevine problem

listed in Section 7 where there are 4 agents, 2 goals, and depth 2 reasoning. Note that while we list four separate examples, they correspond to the same problem and are connected through the `include` functionality.

prob-4ag-2g-2d.pdkbddl

```
{include:domain-4ag.pdkbddl}

(define (problem prob-4ag-2g-2d)

    ; PDKBDDL allows for including files in order to
    ;  compose common elements.
    {include:problem-setup-2d.pdkbddl}

    ; This indicates the restricted depth of nesting
    ;  that will be compiled
    (:depth 2)

    ; [ag]XYZ corresponds to agent ag believing XYZ

    ; <ag>XYZ corresponds to agent ag thinking XYZ is possible

    ; (!fluent) is used in lieu of (not (fluent)) to make
    ;  the task of parsing easier.

    (:goal
        [b][c](!secret a)
        [c](secret a)
    )
)
```

domain-4ag.pdkbddl

```
(define (domain grapevine)
```

```
    ; This specifies the finite number of agents
    (:agents a b c d)

    {include:domain.pdkbddl}

)
```

domain.pdkbddl

```
    (:types loc)
    (:constants )

    ; Predicates marked with {AK} are "Always Known".
    ;  The remaining predicates will be believed to
    ;  some nesting by the agents.
    (:predicates
            (secret ?agent)
        {AK}(at ?agent - agent ?l - loc)
        {AK}(connected ?l1 ?l2 - loc)
        {AK}(initialized)
    )

    (:action move

        ; The derive-condition specifies the condition for
        ;  mutual awareness. "always" translates to True,
        ;  while "never" translates to False.
        :derive-condition   always

        :parameters         (?a - agent ?l1 ?l2 - loc)
```

```
    :precondition       (and (at ?a ?l1)
                             (connected ?l1 ?l2)
                             (initialized))

    ; Note again that we use (!at ...) rather than
    ;  the common PDDL style of (not (at ...))
    :effect             (and (at ?a ?l2) (!at ?a ?l1))
)


(:action share

    ; This condition stipulates that agents are aware
    ;  of this action when they are "at" the location.
    ;  The parameter ?l is bound to the ?l listed in
    ;  the :parameters section, and $agent$ is a stand-in
    ;  for every agent in the domain.
    :derive-condition   (at $agent$ ?l)

    :parameters         (?a ?as - agent ?l - loc)

    ; Note that belief can be part of the precondition.
    :precondition       (and (at ?a ?l)
                             (initialized)
                              [?a](secret ?as))

    ; Quantification will include all agents, including
    ;  the acting one.
    :effect         (and
                        (forall ?a2 - agent
```

```
                            (when
                               (and      (at ?a2 ?l)
                                    <?a2>(secret ?as))
                                [?a2](secret ?as)))
                    )
)

(:action fib
    :derive-condition   (at $agent$ ?l)
    :parameters         (?a ?as - agent ?l - loc)
    :precondition       (and     (at ?a ?l)
                                 (initialized)
                            [?a](secret ?as))
    :effect         (and
                      (forall ?a2 - agent
                            (when
                               (and      (at ?a2 ?l)
                                    <?a2>(!secret ?as))
                               [?a2](!secret ?as)))
                    )
)

(:action initialize
    :derive-condition   never
    :precondition       (and)
    :effect             (and
                            (initialized)
                            (forall ?ag - agent
                                [?ag](secret ?ag))
                        )
```

```
)
```

problem-setup-2d.pdkbddl

```
(:domain grapevine)

(:objects l1 l2 l3 - loc)

; This allows us to project to individual agents, and is
;  not discussed in this paper.
(:projection )

; The task of valid_generation is to create a plan. To
;  confirm a plan instead, valid_assessment can be used,
;  along with a list of actions in a :plan field.
(:task valid_generation)

; The :init-type indicates the assumption of the root
;  agent. Here, it means that every RML not listed is
;  presumed to be possible.
(:init-type complete)
    (:init

    ; Map
    (connected l1 l2)
    (connected l2 l1)
    (connected l2 l3)
    (connected l3 l2)

    ; Agents all in l1
    (forall ?ag - agent (at ?ag l1))
```

```
        ; Agents all believe others think secrets are possible
        (forall ?ag1 - agent
          (forall ?ag2 - agent
            (forall ?s - agent
                (and
                  [?ag1]<?ag2>(secret ?s)
                  [?ag1]<?ag2>(!secret ?s)
                ))))
    )
```

Appendix B.2. Envelope

The structure of this domain is that two agents observe a secret in an envelope, and they update their nested belief about the truth/falsehood of that secret. We consider the viewpoint of both the first and second agent to open the envelope in this setting. In PDKBDDL:

PDKBDDL

```
(define (domain envelope)

    (:agents alice bob)
    (:types )
    (:predicates (secret) )

    (:action check
        :derive-condition always
        :parameters       (?ag - agent)
        :precondition     (and)
        :effect           (and (when  (secret) [?ag](secret))
                               (when (!secret) [?ag](!secret)))
    )
)
```

```
(define (problem future-reasoning)
    (:domain envelope)
    (:projection )
    (:depth 2)
    (:task valid_assessment)

    (:init-type complete)
    (:init
        (forall ?ag1 - agent (and
            <?ag1>(secret)
            <?ag1>(!secret)
            (forall ?ag2 - agent (and
                [?ag2]<?ag1>(secret)
                [?ag2]<?ag1>(!secret)))))
        (secret)
    )

    (:goal (and [bob][alice](secret)))

    (:plan
        (check bob)
        (check alice)
    )
)
```

Notice that the agents begin believing the secret (and its negation) is possible. Also, they believe that the other agents thinks it possible as well. The action to check is extremely compact and simple: for the agent that checks, they'll learn the true value of the secret.

With the mutual awareness aspect of RP-MEP, we have a further ramifi-

cation that if the agent believes the secret (resp. its negation) when the other agents checks, then they'll come to believe the other agent believes (resp. doesn't believe) it as well. The syntax used above is to verify that the plan [(check bob), (check alice)] achieves the goal of [bob][alice](secret).

This is readily handled by RP-MEP. After the first action, bob believes the secret, and through the ancillary effects bob comes to believe alice does as well after the second action. However, the reversed plan no longer works:

PDKBDDL

```
...
  (:plan
      (check alice)
      (check bob)
  )
...
```

Here, bob would need to retain the information that alice believes either the secret or its negation, and then reconcile that disjunctive fact with their own discovery of the true value. We also see this phenomenon in the Hattari domain.

Appendix C. Encoding Ancillary Effects

Here we demonstrate the encoding of the core concepts of RP-MEP – ancillary effects – in both our modelling language (PDKBDDL), as well as that of MEPK (EPDDL). Most of the ancillary effects that we handle as part of the preprocessing are natively handled by the MEPK planner, and these examples serve to illustrate the commonalities.

Appendix C.1. Negation Removal

The first example is a simple domain to demonstrate the impact of negation removal, corresponding to Eqn (1). We list the entire domain and problem for both languages, but subsequent examples will just show the relevant action descriptions for the particular phenomenon.

PDKBDDL

```
(define (domain negation-removal)
    (:agents a)
    (:types )
    (:constants )
    (:predicates (p) (q))

    (:action apply
        :derive-condition   always
        :precondition       (and )
        :effect             (and [a](p))
    )

    (:action check
        :derive-condition   always
        :precondition       (and (not <a>(!p)))
        :effect             (q)
    )
)

(define (problem prob)
    (:domain negation-removal)
    (:projection )
    (:depth 2)
    (:task valid_generation)
    (:init-type complete)
    (:init
        <a>(p)
        <a>(!p)
    )
    (:goal (q))
```

```
)
```

Note that the general structure of these examples are to make the goal fluent (q) true, which can only occur in the action sequence [apply,check]. It is the 'check' action that checks (through its precondition) that the effect of 'apply' is correctly handled. Here is the equivalent problem for EPDDL used by MEPK:

EPDDL

```
(define (domain negation-removal)
    (:objects)
    (:agents a )
    (:predicates  (p) (q) )

    (:action apply
        :category (ontic)
        :parameters ()
        :precondition (True)
        :effect (<{(True)} {(K_a (p))}>))
    (:action check
        :category (ontic)
        :parameters ()
        :precondition (not (DK_a (not (p))))
        :effect             (<{(True)} {(q)}>))

    (:init (and (DK_a (p)) (DK_a (not (p)))))
    (:constraint (True))
    (:goal (q))
)
```

The correspondence between the languages is fairly direct. For every '[agent]' in PDKBDDL, we have 'K_agent' in EPDDL, etc. Both planners successfully

handle this problem, as it only requires removing the negated information of an effect from the knowledge base.

Appendix C.2. Closure

Next, we consider the application of logical closure to effects (cf. Eqn (2)).

PDKBDDL

```
...
  (:action apply
      :derive-condition   always
      :precondition       (and )
      :effect             (and [a](p))
  )

  (:action check
      :derive-condition   always
      :precondition       (and <a>(p))
      :effect             (q)
  )
...
```

EPDDL

```
...
  (:action apply
      :category (ontic)
      :parameters ()
      :precondition (True)
      :effect (<{(True)} {(K_a (p))}>)
  )

  (:action check
      :category (ontic)
```

```
      :parameters ()
      :precondition (DK_a (p))
      :effect       (<{(True)} {(q)}>)
  )
...
```

Again, we find that both RP-MEP and MEPK readily handle this situation.

Appendix C.3. Inverted Closure

Corresponding to Eqn (3), the inverted closure ensures that the knowledge base doesn't contain information that would entail something we are removing. Once again, both RP-MEP and MEPK handle this situation equally well.

PDKBDDL

```
...
  (:action apply
      :derive-condition   always
      :precondition       (and )
      :effect             (and (not <a>(p)))
  )

  (:action check
      :derive-condition   always
      :precondition       (and (not [a](p)))
      :effect             (q)
  )
...
  (:init [a](p) )
...
```

EPDDL

```
...
  (:action apply
      :category (ontic)
      :parameters ()
      :precondition (True)
      :effect (<{(True)} {(not (DK_a (p)))}>)
  )

  (:action check
      :category (ontic)
      :parameters ()
      :precondition (not (K_a (p)))
      :effect             (<{(True)} {(q)}>)
  )
...
  (:init (K_a (p)))
...
```

Appendix C.4. Uncertain Firing

The final ancillary effect, corresponding to Eqn (4), highlights where the two approaches diverge. "Uncertain firing", as identified in the text, is a known phenomenon for belief update in action theories with partial observability. The distinguishing factor is that this inference *involves reasoning about how an action affects the world*. The previous three examples simply refer to ramifications of maintaining a consistent KD_n knowledge base.

From the PDKBDDL, notice that the agent begins thinking (q) must be false, but then comes to think it may be possible.

PDKBDDL

```
...
  (:action apply
      :derive-condition   always
      :precondition       (and )
      :effect             (and (when (p) (q)))
  )

  (:action check
      :derive-condition   always
      :precondition       (and <a>(q))
      :effect             (r)
  )
...
  (:init [a](!q) )
  (:goal (r))
...
```

The corresponding EPDDL, with an added effect to ensure that the agent knows about the conditional effect (i.e., "K_a (p) → K_a (q)") is as follows:

EPDDL

```
...
  (:action apply
      :category (ontic)
      :parameters ()
      :precondition (True)
      :effect (<{(p)} {(q)}>
               <{(K_a (p))} {(K_a (q))}>)
  )

  (:action check
```

```
      :category (ontic)
      :parameters ()
      :precondition (DK_a (q))
      :effect       (<{(True)} {(r)}>)
  )
...
  (:init (K_a (not (q))))
  (:goal (r))
...
```

MEPK does not handle this case. The reason being is that any ramification that involves *knowledge about the impact an action has on the world is not captured by the planner natively.* In order to cover the case of uncertain firing, the ramification must be written manually. This would correspond to the domain author specifying an effect of the form:

EPDDL

```
  <{DK_a (p)} {DK_a (q)}>
```

While this is a minor modification to make the domain work correctly with MEPK, note that it is a ramification of sorts that needs to be covered automatically. Otherwise, all ramifications become essentially those that must be written by hand. See Section 8.1.2 for further discussion.

While we do not detail an example of conditioned mutual awareness, the impact is the same. MEPK does not have a native treatment of such phenomenon (i.e., subsets of agents that are mutually aware of the impact an action has), and thus must be modelled by hand.

Appendix D. PEKBs for Efficient Planning

Here, we outline a theory of Proper Epistemic Knowledge Bases (PEKB) that are suitable for epistemic planning in a KD_n/$\mathrm{KD45}_n$ context.

Appendix D.1. PEKBs

In this work, we expand the theory of PEKBs for the logic of KD_n and $\mathrm{KD45}_n$, suitable for representing epistemic planning problems on top of STRIPS-based planners. The choice of PEKBs is suitable for two reasons:

1. The syntactic restrictions imposed by PEKBs that prevent disjunction and infinite nesting are consistent with the syntactic restrictions employed by STRIPS-based planners, in which arbitrary disjunction is not permitted, and the set of literals (fluents) in a planning problem are finite. In this sense, using PEKBs increases the expressiveness of the STRIPS language to include epistemic formulae.

2. PEKBs come with nice logical and computational properties, as we show here. First, a *consistent* PEKB — that is, one with no contradictory statements — is *logically separable*, which means the literals in the knowledge base do not interact to produce new formulae. Further, a consistent PEKB can be queried in polynomial time without a pre-compilation step such as the one used by Lakemeyer and Lespérance [40], and can be queried in constant time with an exponential pre-compilation step. Finally, a consistent PEKB can be updated with new literals and remain consistent using a polynomial-time algorithm.

Thus, given these two above properties, PEKBs make a suitable representation for extending classical planning over belief bases. In the remainder of this appendix, we prove these properties of PEKBs and analyse their complexity.

Appendix D.2. Logical Separability in PEKBs

The property of logical separability of formulae in PEKBs is a key property in the complexity analysis of PEKBs.

Definition 10. Logical Separability [40]

The set of RMLs P is *logically separable* if and only if for every consistent set

of RMLs P' the following holds:

$$\text{if} \quad P \cup P' \models \bot \quad \text{then} \quad \exists \varphi \in P, \text{ s.t. } P' \cup \{\varphi\} \models \bot$$

Intuitively, a set of formulae is logically separable if we cannot infer anything by combining two or more formulae from the set. E.g., $\{\Box_i p, \Box_i(p \supset q)\}$ is not logically separable, because we can infer $\Box_i q$ from the combination of the two formulae in the set. The set $\{\Box_i p, \Box_i(p \supset q)\} \cup \{\Diamond_i \neg q\}$ is inconsistent, but $\Diamond_i \neg q$ is consistent with both other formulae individually. Logical separability plays an important role later, as examples such as the one above are forbidden. The core issue is the use of disjunction (manifest in the example as an implication) which opens the door to case-based reasoning and far more complex issues when it comes computing all possible ramifications.

To simplify the notation, we will denote a PEKB as a conjunction

$$\begin{aligned} & \Gamma \\ \wedge \;\; & \Diamond_i \psi_1^i \wedge \ldots \wedge \Diamond_i \psi_m^i \wedge \Box_i \chi_1^i \wedge \ldots \wedge \Box_i \chi_n^i \\ \wedge \;\; & \Diamond_j \psi_1^j \wedge \ldots \wedge \Diamond_j \psi_m^j \wedge \Box_j \chi_1^j \wedge \ldots \wedge \Box_j \chi_n^j \\ \wedge \;\; & \cdots \end{aligned}$$

Here, Γ is a conjunctive propositional formula, and we have grouped the $\Box$ and $\Diamond$ operators for each particular agent so that each ψ and χ symbols can be used to unambiguously identify the outermost operator.

Theorem 4. Given a PEKB P in KD_n assumed to be in the form above, we have that $P \models \bot$ *iff* at least one of the following holds:

(a) $\Gamma \models \bot$

(b) $\psi_k^i \wedge \chi_1^i \wedge \ldots \wedge \chi_n^i \models \bot$ (for some agent i and index k)

(c) $\chi_1^i \wedge \ldots \wedge \chi_n^i \models \bot$ (for some agent i).

Proof. A proof for the logic K_n (for which only parts (a) and (b) above are required) is presented by Bienvenu [9] (see Theorem 1, part (3)) for the single agent case. It is straightforward to see that with the addition of the axiom D, that part (c) must be added as a consistent PEKB cannot contain the formula $\Box_i \bot$. Finally, we note that Bienvenu's [9] proof extends to the multi-agent case KD_n because two or more agents can believe contradictory propositions in KD_n— it is only an agent's internal beliefs that must be consistent. □

Following directly from KD_n, we can use the following lemma (cf. [34, 15]):

Lemma 1. *$\varphi \models \psi$ implies $\Diamond_i \varphi \models \Diamond_i \psi$ and $\Box_i \varphi \models \Box_i \psi$*

Theorem 5. Given a consistent PEKB P and RML ψ in KD_n, if $P \models \psi$ then $\exists \varphi \in P$, s.t. $\varphi \models \psi$

Proof. We prove this inductively on the structure of ψ. Assume $P = (\Gamma \wedge \Diamond_i \psi_1 \wedge \ldots \wedge \Diamond_i \psi_m \wedge \Box_i \chi_1 \wedge \ldots \wedge \Box_i \chi_n)$.

The case of $\psi \equiv \gamma$, with γ being some conjunct in Γ, is straightforward because PEKBs contain no disjunction. For the $\psi \equiv \Box_i \psi'$ case, $P \models \Box_i \psi'$ iff $P \wedge \Diamond_i \neg \psi' \models \bot$. From Theorem 4 and the assumption that the PEKB is consistent, it follows that $\chi_1 \wedge \ldots \wedge \chi_n \models \psi'$. By induction, there must be some $\chi_k \in \{\chi_1, \ldots, \chi_n\}$ such that $\chi_k \models \psi'$. From Lemma 1, we know that $\Box_i \chi_k \models \Box_i \psi'$ and that $\Box_i \chi_k \in P$, so this case holds.

The case of $\psi \equiv \Diamond_i \psi'$ is similar. If $P \models \Diamond_i \psi'$, then from Theorem 4, it follows that either for some $\psi_j \in \{\psi_1, \ldots, \psi_m\}$, we have that $\psi_j \wedge \chi_1 \wedge \ldots \wedge \chi_n \models \psi'$, or that $\chi_1 \wedge \ldots \wedge \chi_n \models \psi'$. By induction, it must be that $\psi_j \models \psi'$ or $\chi_k \models \psi'$ for some $\chi_k \in \{\chi_1, \ldots, \chi_n\}$. From Lemma 1 and axiom D, we know that either $\Diamond_i \psi_j \models \Diamond_i \psi'$ (where $\Diamond_i \psi_j \in P$), or $\Box_i \chi_k \models \Diamond_i \psi'$ (where $\Box_i \chi_k \in P$); so this case holds. From the three cases, the theorem holds. □

From the above theorem, it is clear to see that if a PEKB is inconsistent, this inconsistency can be detected by checking all pairwise RMLs. Theorem 5 provides us with a very direct and tractable procedure for inference with PEKBs. The following corollary follows directly from Definition 10 and Theorem 5.

Corollary 1. A consistent PEKB in KD_n is logically separable.

Corollary 1 is an important property in the context of the computational complexity of RP-MEP: given a PEKB P in KD_n with n elements, to update it with a new RML φ, we only need to check φ against each element in P to check if it is inconsistent with the belief base, which is in $O(n \cdot d)$ (with d being the depth of the RML). Thus, detecting inconsistency (and performing belief update), requires only a pairwise check of all elements in the PEKB, which has time complexity of just $O((n \cdot d)^2)$.

Appendix D.3. Entailment in Consistent PEKBs

For definition purposes, we consider the space of all PEKB 's as a partially-ordered set (poset) $(R, \models)$, in which R is the set of all RMLs. We use P and Q to refer to subsets of R (that is, P and Q are PEKBs), and we use $\overline{P}$ to denote the PEKB that contains the negation of every RML in P; that is $\overline{P} = \{\neg\varphi \mid \varphi \in P\}$. Figure D.2 shows a Hasse diagram of a poset. The poset is bounded, with the top element $\Diamond\Diamond\Diamond p$ and the bottom element $\Box\Box\Box p$. We only focus on posets across RMLs *with the exact same prefix of agent modalites*. That is, sequences of either $\Box$ or $\Diamond$ operators that use the some ordering of agents. The appropriateness of this simplification will become clear in Definition 12 and Theorem 6.

Definition 11. Upwards and downwards closure

Given an RML φ, we define the *upward closure* $\uparrow\varphi$ of φ as the upward closure with respect to its poset, defined as:

$$\uparrow\varphi = \{\psi \mid \varphi \models \psi\}$$

The *downward closure* of l, denoted $\downarrow\varphi$, is just $\overline{\uparrow\neg\varphi}$; that is $\downarrow\varphi = \{\psi \mid \psi \models \varphi\}$.

The upward closure of a PEKB P is $\uparrow P = \bigcup_{\varphi\in P} \uparrow\varphi$. The downward closure of a PEKB P is defined as $\downarrow P = \bigcup_{\varphi\in P} \downarrow\varphi$, or alternatively $\downarrow P = \overline{\uparrow\overline{P}}$.

Intuitively, the upward closure of a PEKB $\uparrow P$ is all RMLs that are entailed by P; and the downwards closure $\downarrow P$ is all RMLs that entail an RML in P. Thus, $\uparrow\Box_i p = \{\Box_i p, \Diamond_i p\}$; and $\downarrow\Diamond_i p = \{\Box_i p, \Diamond_i p\}$.

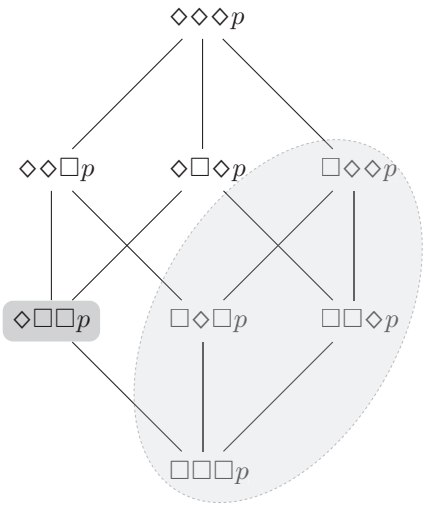


Figure D.2: A representative Hasse diagram with bottom element $\Box\Box\Box p$. The shaded set represents $\downarrow\Box\Diamond\Diamond p$, which would be removed if $\Box\Diamond\Diamond p$ was erased from a PEKB containing $\Box\Box\Box p$. The shaded formula $\Diamond\Box\Box p$ should be the RML that remains, because it is the maximal RML in $\uparrow\Box\Box\Box p$ that is not in with $\downarrow\Box\Diamond\Diamond p$.

We introduce the new notation for closure here to unify the notation of upward/downward closure that the main text did not require ($\uparrow$ is analogous to Cl, and $\downarrow$ is newly introduced). The upward or downward closure of any PEKB is finite. The logical separability of PEKBs means that combining two RMLs cannot produce any new RMLs that we cannot get from inferring from just one. For each RML, the number of RMLs that it entails is finite: we can only repeatedly apply the axiom D, thus traversing up the poset lattice, until we reach the top element.

Definition 12. Algorithm for PEKB Entailment in KD_n

Given a consistent PEKB P and a query φ that is a conjunction of RMLs, we define KD_n entailment $P \models_{PEKB} \varphi$ as:

$P \models_{PEKB} p$	iff	$p \in P$
$P \models_{PEKB} \varphi \wedge \psi$	iff	$P \models_{PEKB} \varphi$ and $P \models_{PEKB} \psi$
$P \models_{PEKB} \neg\varphi$	iff	$P \not\models_{PEKB} \varphi$
$P \models_{PEKB} \Box_i\varphi$	iff	for some $\Box_i\psi \in P$, $\{\psi\} \models_{PEKB} \varphi$
$P \models_{PEKB} \Diamond_i\varphi$	iff	for some $\Box_i\psi \in P$, $\{\psi\} \models_{PEKB} \varphi$ or
		for some $\Diamond_i\psi \in P$, $\{\psi\} \models_{PEKB} \varphi$

Theorem 6. The entailment algorithm defined by $\models_{PEKB}$ is sound and complete in KD_n: i.e., $P \models_{PEKB} \varphi$ iff $P \vDash \varphi$, where $\vDash$ is the standard entailment defined in Section 2.

Proof. The cases for the propositional primitive, conjunction and negation are straightforward from their definition. Given that we assume the query is a conjunction of RMLs, the conjunction in Definition 12 is the only inductive argument required – this follows the standard notion of entailment for a conjunctive query. Therefore, we must only consider $P \models_{PEKB} \varphi$ where φ is an RML.

In this case, we can appeal to Theorem 1 (capturing the ways one RML entails another) and Theorem 5 (capturing logical separability) to ensure that $P \models_{PEKB} \varphi$ if and only if $P \models \varphi$, where φ is an RML.

□

Theorem 7. Entailment for consistent PEKBs in KD_n has worst-case complexity of $O(|P| \cdot |\varphi| \cdot d)$, in which $|P|$ is the PEKB size, $|\varphi|$ is the number of conjuncts in the query, and d is the longest RML in the PEKB or query.

Proof. Given the logical separability, we simply need to compare each element in the PEKB with each element of the conjunction in the query, which is $|P| \cdot |\varphi|$. For each query, the worst case is to compare along the length of the RML using the last two rules until a propositional literal is encountered, thus the worst case complexity is $O(|P| \cdot |\varphi| \cdot d)$. □

Definition 13. Entailment as closure

Given a consistent PEKB P, and a query φ in KD_n that is a conjunction of RMLs, we define entailment as: $P \models_{PEKB} \varphi$ iff for all $\psi \in \varphi$, $\psi \in \uparrow P$. In other words, if we compute the closure of P, we need only check containment in the closure for every RML in the conjunction φ.

The soundness and completeness of this holds trivially from the definition of closure.

Theorem 8. Entailment as (upward) closure for consistent PEKBs has worst-case complexity of $O(|P| \cdot |\varphi| \cdot 2^d)$, in which $|P|$ is the size of the PEKB, $|\varphi|$ is the number of conjuncts in the query, and d is the longest RML in the PEKB or query. However, if each RML is indexed using, for example, hashing when it is computed as part of a the closure, complexity is $O(|\varphi|)$ for any query once the closure is computed.

Proof. The size of $\uparrow P$ is $|P| \cdot 2^d$, because for each RML of length d, there are 2^d RMLs that follow from it by applying axioms K and D. Thus, the closure is computed in $O(|P| \cdot 2^d)$ time, and then φ queries must be run against each RML in $\uparrow P$, which has a worst case of $O(|P| \cdot |\varphi| \cdot 2^d)$. If each RML in $\uparrow P$ is indexed using hashing and the depth is bounded, the lookup becomes constant, so the complexity is just looking up $|\varphi|$ number of queries. □

Using indexing still requires a compilation of $\uparrow P$, which has a worst-case complexity of $O(|P| \cdot 2^d)$. However, if there are multiple queries to be run against the knowledge base, using compilation and hashing as a way of ammortizing the cost is valuable. In Section 4, we showed how this compilation is encoded within a planning model, so the lookup provides significant value.

Appendix D.4. Belief Erasure and Update in PEKBs

Here, we outline a polynomial-time algorithm for belief update — a key property required for PEKBs to be useful in planning.

Our definition of belief update in PEKBs uses the 'forget-then-conjoin' approach of first removing any beliefs that conflict with the update, and then adding the update. This 'forgetting' process, which Katsuno and Mendelzon [36] term *belief erasure*, is not simply the act of subtracting the negation of the RMLs in the update, because we must also remove any RML that implies the negated update. For example, given the PEKB $\{\Diamond_i p, \Box_i p\}$, removing $\Diamond_i p$ should also remove $\Box_i p$; otherwise, the belief base would still entail $\Diamond_i p$. Further, a belief erasure operator should follow the principle of *minimal change*: when removing belief from an existing belief base, we should remove only what we must so that the belief base no longer entails the removed belief.

Definition 14. Prime PEKBs

A PEKB P is *prime* if and only if all elements in P are prime implicates of P (maximal elements); that is, for all $\varphi, \psi \in P$, if $\varphi \models \psi$ then $\psi \models \varphi$. The set of maximal elements of a PEKB is denoted $\max(P)$.

Note that for any PEKB P (prime or otherwise) and an RML φ, we have that $P \models \varphi$ iff $\varphi \in \uparrow P$.

Definition 15. Belief erasure in PEKBs

Given PEKBs P and Q, we define $P \blacklozenge Q$ as the *belief erasure* of Q from Q as follows:

$$P \blacklozenge Q = \max(\uparrow P \setminus \downarrow Q)$$

That is, take the upward closure of P and remove the downward closure of Q, removing any non-prime RMLs. This removes Q and anything that implies it, leaving those things that are in the upward closure of P that do not entail Q.

Theorem 9. The complexity of $P \blacklozenge Q$, where P and Q are both prime, is $O(|P| \cdot |Q| \cdot d)$, in which d is the depth of the longest RML in P or Q. If instead we calculate $P \blacklozenge Q$ by calculating $\uparrow P$ and $\downarrow Q$, and the subtracting their

difference, the complexity is $O(2^{|P|} \cdot 2^{|Q|} \cdot d)$.

Proof. Given the logical separability of PEKBs, we need to only calculate the erasure of each pair in $P \times Q$. This erasure can be done linearly in the depth of the RMLs. For each modal operator index that is $\Box_i$ in both φ and ψ, create a new RML that is equivalent to φ but with $\Diamond_i$ at that index, and then take the top elements of this set. This can be calculated by traversing up the poset until we reach the top elements (see Figure D.2 for an example), which has a maximum depth of d.

If we instead calculate the upward and downward closure of P and Q respectively, we need to just iterate through the $2^{|P|} \times 2^{|Q|}$ pairs and remove any from $\uparrow P$ that occurs in $\downarrow Q$. Checking each pair has linear complexity in d, although as with entailment, this operation can be done in constant time with indexing. □

Given a belief erasure operator, belief update for a PEKB is straightforward: update is forget (erase) then conjoin, eliminating any RML that is not prime.

Definition 16. Belief Update in PEKBs

Given PEKBs P and Q, we define belief update of P with Q, denoted as $P \diamond Q$, as follows:

$$P \diamond Q = \max((P \blacklozenge \overline{Q}) \cup Q)$$

That is, remove anything that conflicts with Q, then add the elements in Q, and take the maximal elements from the result.

The complexity of this is just the complexity of belief erasure, with the added overhead of adding the new elements in Q into the PEKBs.

This update operator observes the property of relevant minimal change [57]. Because PEKBs are logically separable, it is clear to see that any RMLs that are not related to the new RMLs remain in the knowledge base.

Thus, we have defined belief erasure and belief update for PEKBs.

Appendix D.5. Analysis of Belief Update and Erasure Operators

In this section, we extend the results by Miller and Muise [48] that analyse these operators with respect to the classic belief update postulates from Katsuno and Mendelzon [36]. Specifically, we include proofs that did not appear in the published version. These results show that the belief update and erasure operators are satisfiable and correct.

Appendix D.5.1. KM Postulates for Belief Update

Katsuno and Mendelzon [36] propose a set of postulates for belief update called the Katsuno-Mendelzon (KM) postulates. These postulates, which echo the AGM postulates for belief revision [2], specify eight properties that a belief update operator should have to be an appealing update mechanism (phrased using our notation):

U1 $P \diamond Q \models Q$

U2 If $P \models Q$ then $P \diamond Q \equiv P$

U3 If P and Q are satisfiable, then $P \diamond Q$ is satisfiable

U4 If $P \equiv P'$ and $Q \equiv Q'$ then $P \diamond Q \equiv P' \diamond Q'$

U5 $(P \diamond Q) \sqcup R \models P \diamond (Q \sqcup R)$

U6 If $P \diamond Q \models R$ and $P \diamond R \models Q$ then $P \diamond Q \equiv P \diamond R$

U7 If P is complete then $(P \diamond Q) \sqcup (P \diamond R) \models P \diamond (Q \vee R)$

U8 $(P \vee Q) \diamond R \equiv (P \diamond R) \vee (Q \diamond R)$

Because PEKBs do not permit disjunction, U7 and U8 are not relevant for our belief update operator.

Despite their widespread use, it is not commonly accepted that all postulates are desirable for all belief update operators. Herzig and Rifi [30] argue that only postulates U1, U3, U8, and (possibly) U4 should be satisfied by all update operators.

Theorem 10. KM postulates U1, and U3-U6 hold for PEKB belief update operator $\diamond$. U2 holds if P is satisfiable.

Proof. The proofs for U1 and U4 are straightforward.

U2: From the definitions of $\diamond$ and $\blacklozenge$, $P \diamond Q$ is equivalent to $\max(\uparrow P \setminus \downarrow \overline{Q}) \sqcup Q$. If $P \models Q$ and P is satisfiable, we know that $\uparrow P \cap \downarrow \overline{Q} = \emptyset$, and therefore $\uparrow P \setminus \downarrow \overline{Q} = \uparrow P$. Further, if $P \models Q$, we know that $\uparrow Q \subseteq \uparrow P$, and therefore, $\uparrow P \cup \uparrow Q = \uparrow P$, and therefore $\max(\uparrow P \cup \uparrow Q) = \max(\uparrow P)$, meaning that postulate U2 holds.

U3: Assume that the postulate does not hold, so $P \not\models \bot$ and $Q \not\models \bot$, but $P \diamond Q \models \bot$. If this is the case, then there is some ϕ such that $\uparrow P \setminus \downarrow \overline{Q} \models \phi$ and $\uparrow Q \models \neg\phi$. However, from the definition of downwards closure, we know that $\downarrow \overline{Q} = \overline{\uparrow Q}$, meaning that $\uparrow P \setminus \overline{\uparrow Q} \models \phi$. If $\uparrow Q \models \neg\phi$ then it must be that $\overline{\uparrow Q} \models \phi$, and as a result, $\uparrow P \setminus \overline{\uparrow Q} \not\models \phi$, violating our assumption. Therefore, postulate U3 holds.

U5: Assume that the postulate does not hold. This implies that there is some ϕ such that $\uparrow P \setminus \downarrow \overline{Q \sqcup R} \sqcup (Q \sqcup R) \models \phi$ and $\uparrow P \setminus \downarrow \overline{Q} \sqcup (Q \sqcup R) \not\models \phi$. Because $Q \sqcup R$ occurs in the latter, ϕ cannot be entailed by $Q \sqcup R$. This implies that: (a) $\uparrow P \setminus \downarrow \overline{Q \sqcup R} \models \phi$; but (b) $\uparrow P \setminus \downarrow \overline{Q} \not\models \phi$. From (a), we know that that $\uparrow P \models \phi$, and so for (b) to hold, it must be that $\downarrow \overline{Q} \models \phi$. However, $\downarrow \overline{Q \sqcup R} \models \downarrow \overline{Q}$, meaning that $\uparrow P \setminus \downarrow \overline{Q \sqcup R} \not\models \phi$, which contradicts (a). Therefore, postulate U5 holds.

U6: Assume that the postulate does not hold, then this means that if: (a) $\uparrow P \setminus \downarrow \overline{Q} \sqcup Q \models R$ and (b) $\uparrow P \setminus \downarrow \overline{R} \sqcup R \models Q$, then there exists some ϕ such that $\uparrow P \setminus \downarrow \overline{Q} \sqcup Q \models \phi$ but $\uparrow P \setminus \downarrow \overline{R} \sqcup R \not\models \phi$ (or vice-versa, but the cases are symmetric). It cannot be that case that $Q \models \phi$, otherwise from (b) it would follow that $\uparrow P \setminus \downarrow \overline{R} \sqcup R \models \phi$, violating our assumption. Therefore, it must be that $P \models \phi$ and $\overline{Q} \not\models \phi$, and $(\uparrow P \setminus \downarrow \overline{R}) \sqcup R \not\models \phi$. Because $P \models \phi$, it must be that $\downarrow \overline{R} \models \phi$, and therefore $R \models \neg\phi$. But from (a), this implies that $\uparrow P \setminus \downarrow \overline{Q} \sqcup Q \models \neg\phi$, contradicting our assumptions. Therefore postulate U6 holds. □

Katsuno and Mendelzon [36] present the so-called *representation theorem*, which shows the completeness of the belief update operator. It is clear that the pre-order on interpretations defined by Katsuno and Mendelzon can simply be defined over the poset corresponding to the elements in the PEKB. Due to the logical separability of PEKBs, Definition 3 amounts to an equivalent notion of Katsuno and Mendelzon's representation theorem.

Appendix D.5.2. KM Postulates for Belief Erasure

Katsuno and Mendelzon [36] also propose a set of postulates for belief erasure based on the principle of minimal change: when removing belief from an existing belief base, we should remove only what we must so that the belief base no longer entails the removed belief. These postulates phrased using our notation are:

E1 $P \models P \blacklozenge Q$

E2 If $P \models \overline{Q}$ then $P \blacklozenge Q \equiv P$

E3 If P is satisfiable then $P \blacklozenge Q \not\models Q$

E4 If $P \equiv P'$ and $Q \equiv Q'$ then $P \blacklozenge Q \equiv P' \blacklozenge Q'$

E5 $(P \blacklozenge Q) \sqcup Q \models P$

E8 $(P \vee Q) \blacklozenge R \equiv (P \blacklozenge R) \vee (Q \blacklozenge R)$

E8 does not make sense because PEKBs cannot contain disjunctive formulae.

Katsuno and Mendelzon [36] define an identity, a mirror of the identity Harper introduced that expresses belief contraction in terms of set operations and belief contraction [28]. Katsuno and Mendelzon's identity can be expressed as:

$$P \blacklozenge Q \equiv P \sqcap (P \diamond \overline{Q}) \tag{D.1}$$

in which $P \sqcap Q = \max(\uparrow P \cap \uparrow Q)$. Intuitively, this identity stipulates that erasing Q should be the same as restricting the belief base to what would hold if the negation of Q was added.

Katsuno and Mendelzon [36] show that if the identity in Equation D.1 holds and the $\diamond$ operator satisfies postulates U1-U4 and U8, then the $\blacklozenge$ operator satisfies postulates E1-E5 and E8. The following counterexample demonstrates that Equation D.1 does not hold on our operators: $P = \{\Box_i p\}$ and $Q = \{\Diamond_i p, \Diamond_i \neg p\}$. From this, $P \blacklozenge Q = \{\}$, while $P \diamond \overline{Q} = \{\Diamond_i p, \Diamond_i \neg p\}$, which when intersected with $\uparrow P$ leaves $\{\Diamond_i p\}$.

Katsuno and Mendelzon [36] define a second identity between update and erasure:

$$P \diamond Q \equiv (P \blacklozenge \overline{Q}) \sqcup Q \tag{D.2}$$

This mirrors the Levi identity for belief revision and contraction [42]. They show that if this identify holds and the $\blacklozenge$ operator satisfies E1-E4 and E8, then $\diamond$ satisfies U1-U4 and U8. Equation D.2 is just our definition of belief update, and Katsuno and Mendelzon's theorem about the relationship between the two sets of postulates holds:

Theorem 11. KM postulates E1, E3, and E4 hold for the PEKB belief erasure operator. E2 holds if P is satisfiable.

Proof. The proofs for E1, E3 and E4 are straightforward, so are omitted.

E2: Assume that the postulate does not hold. Then, there is some ϕ such that $P \blacklozenge Q \not\models \phi$ but $P \models \phi$ (the reverse cannot hold because $\blacklozenge$ is defined as set complement). From the definitions of $\blacklozenge$ and downwards closure, this implies that $\uparrow P \setminus \overline{\uparrow \overline{Q}} \not\models \phi$. For this to hold, it must be that $\overline{\uparrow \overline{Q}} \models \phi$, which implies that $\overline{Q} \models \neg\phi$. However, from the premise $P \models \overline{Q}$, this implies that $P \models \neg\phi$, which contradicts the assumption $P \models \phi$, so postulate E2 holds if P is satisfiable.

If P is unsatisfiable, postulate E2 not hold. A counterexample is $P = \{p, \neg p\}$ and $Q = \{p\}$. □

The E5 postulate does not hold. As a simple counterexample to this, consider the PEKBs $P = \{\Box_i p\}$ and $Q = \{\Diamond_i p\}$. Erasing Q from P will result in an empty set, and then adding Q will result in $\{\Diamond_i p\}$, which does not entail $\Box_i p$.

Finally, we note briefly on a controversial[6] postulate — the *recovery postulate*:

$$(P \blacklozenge Q) \diamond Q \models P$$

This extends postulate E5, using $\diamond$ instead of $\sqcup$. The counterexample for E5 serves to show this postulate does not hold. However, we can characterise precisely when postulate E5 and the recovery postulate are satisfied: when $Q = {\downarrow}Q$. This follows directly from the definition of $\blacklozenge$.

[6]See a discussion of the issues surrounding the postulate in Makinson [46].