Artificial Intelligence 301 (2021) 103562

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

A general multi-agent epistemic planner based on higher-order belief change [☆]

Hai Wan, Biqing Fang, Yongmei Liu [*]

Dept. of Computer Science, Sun Yat-sen University, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 27 November 2020
Received in revised form 18 May 2021
Accepted 20 July 2021
Available online 29 July 2021

Keywords:
Epistemic planning
Multi-agent epistemic logic
Belief change

ABSTRACT

In recent years, multi-agent epistemic planning has received attention from both dynamic logic and planning communities. Existing implementations of multi-agent epistemic planning are based on compilation into classical planning and suffer from various limitations, such as generating only linear plans, restriction to public actions, and incapability to handle disjunctive beliefs. In this paper, we consider centralized multi-agent epistemic planning from the viewpoint of a third person who coordinates all the agents to achieve the goal. We treat contingent planning, resulting in nonlinear plans. We model private actions and hence handle beliefs, formalized with the multi-agent KD45 logic. We handle static propositional common knowledge, which we call constraints. For such planning settings, we propose a general representation framework where the initial knowledge base (KB) and the goal, the preconditions and effects of actions can be arbitrary KD45$_n$ formulas, and the solution is an action tree branching on sensing results. In this framework, the progression of KBs w.r.t. actions is achieved through the operation of belief revision or update on KD45$_n$ formulas, that is, higher-order belief revision or update. To support efficient reasoning and progression, we make use of a normal form for KD45$_n$ called alternating cover disjunctive formulas (ACDFs). We propose reasoning, revision and update algorithms for ACDFs. Based on these algorithms, adapting the PrAO algorithm for contingent planning from the literature, we implemented a multi-agent epistemic planner called MEPK. Our experimental results show the viability of our approach.

1. Introduction

In recent years, multi-agent epistemic planning (MEP) has received attention from both dynamic logic and planning communities. Essentially, multi-agent epistemic planning is planning that involves multiple agents, actions with epistemic preconditions and effects, and epistemic goals. For illustration, consider the following example from [31]. There is a corridor of several rooms, and a number of boxes that are in some of the rooms. Two agents can move along this corridor, and sense if a box is in a room. The task is to make the two agents collaborate via communication in finding out the positions of the boxes. Here, the goal is epistemic, the sensing action has epistemic effects, and the communication action has epistemic precondition and effect. In some cases, it is even necessary to reason about higher-order knowledge and beliefs, *i.e.*, knowledge and beliefs about other agents' knowledge and beliefs. For example, agent a may wish to know a secret s with agent

☆ This article belongs to Special Issue: Epistemic Planning.
* Corresponding author.
E-mail addresses: wanhai@mail.sysu.edu.cn (H. Wan), fangbq3@mail3.sysu.edu.cn (B. Fang), ymliu@mail.sysu.edu.cn (Y. Liu).

https://doi.org/10.1016/j.artint.2021.103562
0004-3702/© 2021 Elsevier B.V. All rights reserved.

b knowing she knows s, or without b knowing she knows s, or even with b believing she does not know s. Now consider a more practical application scenario: after an earthquake, two robots collaborate via communication in locating trapped people. In such a scenario, like in the above example, multi-agent epistemic planning is necessary, while propositional planning is not sufficient.

There are earlier works on single-agent epistemic planning. Herzig et al. [26] proposed a framework for epistemic planning where knowledge bases (KBs) are expressed as positive epistemic formulas, and showed how progression, regression and plan generation can be achieved in their framework. Bienvenu et al. [9] identified two normal forms of epistemic formulas called $\mathbf{S}_5$-CNF$_{\text{CNF,DNF}}$ and $\mathbf{S}_5$-DNF$_{\text{DNF,CNF}}$, and showed that progression and entailment are tractable when the KB is in $\mathbf{S}_5$-DNF$_{\text{DNF,CNF}}$ and the query is in $\mathbf{S}_5$-CNF$_{\text{CNF,DNF}}$. Petrick and Bacchus [39,40] presented a first-order epistemic planning system PKS based on the idea of progression. The kinds of knowledge they consider include literal knowledge, knowing-whether knowledge, and exclusive-or knowledge. However, their progression and reasoning algorithms are both incomplete, and hence their planner is incomplete. Wan et al. [47] presented a complete epistemic planner without the epistemic closed world assumption. They proposed normal forms for epistemic formulas which support tractable reasoning and progression, and adapted the Pruning AND/OR forward search (PrAO) algorithm for contingent planning [45].

Many efforts have gone into the theoretic studies of multi-agent epistemic planning (MEP). Bolander and Andersen [10] and Löwe et al. [34] were the first to propose to formalize MEP based on dynamic epistemic logic (DEL) [19]. In the DEL-based formalization, states are represented as Kripke models, actions are represented as action models, which are Kripke models of actions, describing agents' abilities to distinguish among actions, and by the product update operation, action models are used to update Kripke models. Bolander and Andersen showed that the solvability of such epistemic planning problems is undecidable in general. Further, Aucher and Bolander [5] showed that MEP is undecidable in the presence of only purely epistemic actions, and Charrier et al. [12] proved that the above holds even if the actions have preconditions of modal depth bounded by two. Very recently, Cong et al. [13] showed that MEP is undecidable even for two-agent S5 models, a fixed action, and a fixed goal. Nonetheless, Yu et al. [49] identified two important decidable fragments of multi-agent epistemic planning. The first one is MEP with propositional actions and goal not involving common knowledge, and the second one is MEP with a wide variety of special types of propositional purely epistemic actions. In contrast to centralized planning considered in the above works, Engesser et al. [20] investigated decentralized planning with implicit coordination. To investigate decidable fragments of MEP, Cooper et al. [14] formalized MEP based on a simple logic of knowledge that is grounded on the visibility of propositional variables and showed that it is decidable (PSPACE-complete). Meanwhile, Cooper et al. [15] studied a typical example of MEP, the so-called gossip problem [25], and showed that it is polynomial time while it becomes NP-complete in the presence of negative goals. When it comes to the specification of MEP domains, Baral et al. [7] proposed an action language $m\mathcal{A}$, whose semantics is based on DEL, and Son et al. [42] investigated finitary S5-theories, which can be characterized by finitely many finite Kripke models.

There are mainly two approaches for implementing multi-agent epistemic planning. Kominis and Geffner [31], and Muise et al. [38] showed how to exploit classical planning to solve restricted versions of MEP problems. By resorting to classical planning, both methods can only generate linear plans, doing conformant planning. Moreover, Kominis and Geffner assumed all actions are public, and hence dealt with knowledge. In contrast, Muise et al. focused on beliefs; however, they can only handle bounded-depth belief literals, disallowing disjunctive beliefs. Based on earlier works on the action language $m\mathcal{A}$ and finitary S5-theories, Le et al. [32] presented two multi-agent epistemic forward search planners: one uses simple breadth-first search, and the other employs heuristic search via an epistemic planning graph. The initial state is specified by a finitary S5-theory, which is an S5-theory containing only formulas of the following forms: ϕ, $CK_i\phi$, $C(K_i\phi \vee K_i\neg\phi)$, and $C(\neg K_i\phi \wedge \neg K_i\neg\phi)$, where C is the common knowledge operator, K_i is the knowledge operator, and ϕ is a propositional formula.

Belief change studies how an agent modifies her beliefs on receiving new information. However, so far research on belief change focuses on beliefs represented as formulas in propositional logic. Two main types of belief change are revision and update: revision concerns belief change about static environments due to partial and possibly incorrect information, whereas update concerns belief change about dynamic environments due to the performance of actions. Various guidelines for belief change have been proposed, and the most popular ones are the AGM postulates for belief revision [1], the KM postulates for belief update [30], and the DP postulates for iterated belief revision [17]. Katsuno and Mendelzon [30] briefly discussed how belief revision and update can be used for reasoning about actions: if a condition ϕ is found true, the KB is revised with ϕ; if an action with postcondition ψ is performed, the KB is updated with ψ. There have been preliminary works extending belief change from propositional logic to epistemic logic. Aucher [3] gave a semantic study of multi-agent belief revision incurred by private announcements. Aucher [4] explored the progression of KBs with respect to actions, where both KBs and actions are represented as canonical formulas in modal logics, which capture Kripke models up to certain depths [37]. Recently, Caridroit et al. [11] investigated several measures of distances between KD45$_n$ Kripke models, and use them to define the revision of beliefs represented by such models. Miller and Muise [36] studied belief update for KBs consisting of belief literals. Finally, Van Benthem [8] integrated belief revision into DELs, and Baltag and Smets [6] further presented a general framework for this integration: in line with the AGM approach of giving priority to new information, they proposed the action priority update operation: when updating a plausibility model by an action plausibility model, give priority to the action plausibility order.

In this paper, we consider centralized multi-agent epistemic planning from the viewpoint of a third person who coordinates all the agents to achieve the goal. We treat contingent rather than conformant planning, resulting in nonlinear plans.

We model private actions and hence handle beliefs, formalized with the multi-agent modal logic KD45$_n$ [21]. We do not support arbitrary common knowledge, but we handle static propositional common knowledge, which we call constraints. For such planning settings, we propose a general representation framework where the initial KB and the goal, the preconditions and effects of actions can be arbitrary KD45$_n$ formulas, and the solution is an action tree branching on sensing results. In this framework, the progression of KBs w.r.t. actions is achieved through the operation of belief revision or update on KD45$_n$ formulas, *i.e.*, higher-order belief revision or update. To support efficient reasoning and progression, we make use of a normal form for KD45$_n$ called alternating cover disjunctive formulas (ACDFs) [24]. We propose reasoning, revision and update algorithms for ACDF formulas. Based on these algorithms, adapting the PrAO algorithm, we implemented a multi-agent epistemic planner called MEPK. Our experimental results show the viability of our approach.

A preliminary version of this paper was published in IJCAI-2017 [28]. The main differences between this version and the conference paper are as follows:

- We revise the definition of progression w.r.t. deterministic actions (Definition 12) to resolve issues with conditional effects: a KB might be split into multiple copies, for which we apply all applicable effects.
- We revise the definition of progression w.r.t. sensing actions (Definition 14) to let the progression be false in the presence of impossible sensing results. Accordingly, we revise the definition of multi-agent epistemic solutions (Definition 17) to allow branches of impossible sensing results.
- We give semantic characterizations for our higher-order belief change operators for proper ACDFs, a fragment of ACDFs disallowing negative or disjunctive beliefs (Section 4.3.3). We define the concept of tree models and show that a proper ACDF has a model iff it has a tree model. By restricting attention to tree models, we show that for proper ACDFs, the semantic characterizations for propositional belief change nicely carry over to higher-order belief change.

Besides, in this version, we have extended and improved the presentation, added detailed complexity analysis, full proofs of all propositions, and illustrating examples, and updated the experimental results. We remark that Liu and Liu [33] have extended the work of this conference paper with the support of common knowledge.

The paper is organized as follows. In the next section, we introduce preliminaries, including ACDFs. Section 3 covers our modeling framework for multi-agent epistemic planning. In Section 4, we describe our reasoning and belief change algorithms. Section 5 is devoted to our implementation and experimentation results. Finally, we discuss related work and conclude the paper.

2. Preliminaries

In this section, we introduce the background work of our paper, *i.e.*, the multi-agent modal logic KD45$_n$, alternating cover disjunctive formulas, and belief revision and update.

2.1. Multi-agent modal logic KD45$_n$

Consider a finite set of agents $\mathcal{A}$ and a finite set of atoms $\mathcal{P}$. We use ϕ and ψ for formulas, Φ and Ψ for sets of formulas.

Definition 1. The language $\mathcal{L}_{KC}$ of multi-agent modal logic with common knowledge is generated by the BNF:

$$\varphi ::= p \mid \neg\phi \mid (\phi \wedge \psi) \mid K_a\phi \mid C\phi, \text{ where}$$

$p \in \mathcal{P}$, $a \in \mathcal{A}$, $\phi, \psi \in \mathcal{L}_{KC}$. We use $\mathcal{L}_K$ for the language without the C operator, and $\mathcal{L}_0$ for the propositional language.

As usual, "$\vee$" and "$\rightarrow$" are treated as abbreviations. To reduce the use of parentheses in formulas, we specify the following order of precedence for connectives: C, K_a, $\neg$, $\wedge$, $\vee$, and $\rightarrow$.

Intuitively, $K_a\phi$ means that agent a knows ϕ holds, and $C\phi$ means ϕ is *common knowledge* among all agents, *i.e.*, everybody knows ϕ, everybody knows everybody knows ϕ, everybody knows everybody knows everybody knows ϕ, and so on. In this paper, we restrict our attention to the case of propositional common knowledge, *i.e.*, $C\phi$ where $\phi \in \mathcal{L}_0$, and we call ϕ a *constraint*. We use $C^*\phi$ to denote $\phi \wedge C\phi$.

We let $L_a\phi$ stand for $\neg K_a\neg\phi$. We let $\top$ and $\bot$ represent *true* and *false* respectively. We let $\bigvee \Phi$ (resp. $\bigwedge \Phi$) denote the disjunction (resp. conjunction) of members of Φ; and we use $L_a\Phi$ to represent the conjunction of $L_a\phi$ where $\phi \in \Phi$. The length of a formula ϕ, denoted by $|\phi|$, is the number of atoms, logical connectives, and modal operators in ϕ. The *modal depth* of a formula ϕ in $\mathcal{L}_K$, denoted by $md(\phi)$, is the depth of nesting of modal operators in ϕ.

Definition 2. A *frame* is a pair (W, R), where W is a non-empty set of possible worlds; for each agent $a \in \mathcal{A}$, R_a is a binary relation on W, called the *accessibility relation* for a.

We say R_a is *serial* if for any $w \in W$, there is $w' \in W$ s.t. wR_aw'; we say R_a is *Euclidean* if whenever wR_aw_1 and wR_aw_2, we get $w_1R_aw_2$. A KD45$_n$ frame is a frame whose accessibility relations are serial, transitive and Euclidean.

Definition 3. A Kripke model is a triple $M = (W, R, V)$, where (W, R) is a frame, and V is a valuation map, which maps each $w \in W$ to a subset of $\mathcal{P}$. A pointed Kripke model is a pair $s = (M, w)$, where M is a Kripke model and w is a world of M, called the *actual world*.

Definition 4. Let $s = (M, w)$ be a Kripke model where $M = (W, R, V)$. We interpret formulas in $\mathcal{L}_{KC}$ by induction:

- $M, w \models p$ iff $p \in V(w)$;
- $M, w \models \neg\phi$ iff $M, w \not\models \phi$;
- $M, w \models \phi \wedge \psi$ iff $M, w \models \phi$ and $M, w \models \psi$;
- $M, w \models K_a\phi$ iff for all v s.t. $wR_a v$, $M, v \models \phi$;
- $M, w \models C\phi$ iff for all v s.t. $wR_{\mathcal{A}} v$, $M, v \models \phi$, where $R_{\mathcal{A}}$ is the transitive closure of the union of R_a for $a \in \mathcal{A}$.

A model of ϕ is a KD45$_n$ Kripke model (M, w) s.t. $M, w \models \phi$. We say ϕ is *satisfiable* if ϕ has a model. We say ϕ *entails* ψ, written $\phi \models \psi$, if any model of ϕ is also a model of ψ. We say ϕ and ψ are equivalent, written $\phi \Leftrightarrow \psi$, if $\phi \models \psi$ and $\psi \models \phi$. Note that we have $K_a\phi \models L_a\phi$, $K_a\bot \Leftrightarrow \bot$, and $L_a\top \Leftrightarrow \top$.

We say that ϕ is satisfiable w.r.t. constraint $\gamma \in \mathcal{L}_0$ if $\phi \wedge C^*\gamma$ is satisfiable; we say that ϕ entails ψ w.r.t. constraint γ, written $\phi \models_\gamma \psi$, if $\phi \wedge C^*\gamma \models \psi \wedge C^*\gamma$.

2.2. *Alternating cover disjunctive formulas*

In this section, we introduce alternating cover disjunctive formulas (ACDFs), which we use to support efficient reasoning and progression. We show that every multi-agent epistemic formula can be transformed to an equivalent ACDF whose length is singly exponential in the length of the original formula. Also, we show that it is tractable to check the satisfiability of ACDFs.

2.2.1. *Cover disjunctive formulas*

Janin and Walukiewicz [29] introduced the notion of disjunctive formulas for modal μ-calculus and showed that every formula is equivalent to a disjunctive formula. D'Agostino and Lenzi [16] gave the definition of disjunctive formulas for modal logics, using a *cover* modality. Ten Cate et al. [2006] showed that every formula in the multi-agent modal logic K$_n$ is equivalent to a disjunctive formula whose length is at most singly exponential in the length of the original formula. We slightly vary the definition, and use the name *cover disjunctive formulas*.

We first introduce the cover modality. Intuitively, $\nabla_a\Phi$ means that each world considered possible by agent a satisfies an element of Φ, and each element of Φ is satisfied by some world considered possible by agent a.

Definition 5. Let $a \in \mathcal{A}$, and Φ a finite set of formulas. The cover modality is defined as follows:

$$\nabla_a\Phi \doteq K_a(\bigvee \Phi) \wedge L_a\Phi.$$

The following are useful properties about the cover modality, which can help us to convert an arbitrary multi-agent epistemic formula to an ACDF. We use $\Phi \wedge \psi$ to denote the set $\{\phi \wedge \psi \mid \phi \in \Phi\}$.

Proposition 1.

1. $\nabla_a\{\top\} \Leftrightarrow \top$;
2. $K_a\psi \wedge L_a\Phi \Leftrightarrow \nabla_a(\{\psi\} \cup (\Phi \wedge \psi))$;
3. $\nabla_a\Phi \wedge \nabla_a\Phi' \Leftrightarrow \nabla_a[\Phi \wedge (\bigvee \Phi') \cup \Phi' \wedge (\bigvee \Phi)]$.

Proof. 1. $\nabla_a\{\top\} \Leftrightarrow K_a\top \wedge L_a\top \Leftrightarrow \top$.

2. $\nabla_a(\{\psi\} \cup (\Phi \wedge \psi)) \Leftrightarrow K_a\psi \wedge L_a\psi \wedge L_a(\Phi \wedge \psi) \Leftrightarrow K_a\psi \wedge L_a\Phi$, since $K_a\psi \models L_a\psi$, and $K_a\psi \wedge L_a\phi \models L_a(\phi \wedge \psi)$ for $\phi \in \Phi$.
3. $\nabla_a[\Phi \wedge (\bigvee \Phi') \cup \Phi' \wedge (\bigvee \Phi)]$
$\Leftrightarrow K_a\{\bigvee[\Phi \wedge (\bigvee \Phi')] \vee \bigvee[\Phi' \wedge (\bigvee \Phi)]\} \wedge L_a[\Phi \wedge (\bigvee \Phi')] \wedge L_a[\Phi' \wedge (\bigvee \Phi)]$
$\Leftrightarrow K_a[(\bigvee \Phi) \wedge (\bigvee \Phi')] \wedge L_a\Phi \wedge L_a\Phi' \Leftrightarrow K_a(\bigvee \Phi) \wedge L_a\Phi \wedge K_a(\bigvee \Phi') \wedge L_a\Phi'$
$\Leftrightarrow \nabla_a\Phi \wedge \nabla_a\Phi'$. □

Definition 6. The set of *cover disjunctive formulas* (CDFs) is inductively defined as follows:

1. A propositional term, *i.e.*, a conjunction of propositional literals, is a CDF.
2. If ϕ_0 is a propositional CDF, and for each $a \in \mathcal{B} \subseteq \mathcal{A}$, Φ_a is a finite set of CDFs, then $\phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\Phi_a$ is a CDF, called a *CDF term*.
3. If Φ is a non-empty finite set of CDF terms, then $\bigvee \Phi$ is a CDF, called a *disjunctive CDF*.

In this paper, we use DNF to mean propositional disjunctive normal form, *i.e.*, a DNF formula is a disjunction of propositional terms. So a propositional CDF is in DNF.

Moss [37] introduced the concept of canonical formulas, very similar to formulas introduced by Fine [22], and showed that every formula in K_n is equivalent to a disjunction of a finite set of canonical formulas. However, in the definition of canonical formulas, no disjunction is allowed. As a result, the conversion may cause a non-elementary blowup in size [37], as compared to a single exponential blowup for CDFs.

The following result shows that every formula in K_n is equivalent to a CDF whose length is singly exponential in the length of the original formula. The transformation part was first shown by Janin and Walukiewicz [29] in the context of modal μ-calculus, and the complexity part was first shown by ten Cate et al. [43] in the context of the ALC description logic. Since this result is the foundation for our compilation result concerning ACDFs, and the paper by ten Cate et al. doesn't include a detailed complexity analysis, we include a proof in the appendix.

Proposition 2 *(Janin and Walukiewicz [29], ten Cate et al. [43]). In K_n, every formula ϕ in $\mathcal{L}_K$ can be transformed to an equivalent CDF whose length is $O(2^{|\phi|^2})$.*

The idea of the proof is as follows: If ϕ is a propositional term, no transformation is needed. Otherwise, let $K_{a_1}\phi_1, \dots, K_{a_n}\phi_n$ be the modal atoms that appear in ϕ but not within the scope of any modal operator. Firstly, we treat $K_{a_1}\phi_1, \dots, K_{a_n}\phi_n$ as atoms and put ϕ into DNF. Each disjunct of the resulting DNF is of the form $\eta = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}}(K_a \bigwedge \Phi_a \wedge L_a\Psi_a)$, where $\mathcal{B} \subseteq \mathcal{A}$, and ϕ_0 is a propositional term. By Proposition 1 (2), $\eta \Leftrightarrow \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a(\{\bigwedge \Phi_a\} \cup \{\bigwedge \Phi_a \wedge \psi \mid \psi \in \Psi_a\})$. We repeat the process on each formula of $\bigwedge \Phi_a$ and $\bigwedge \Phi_a \wedge \psi$.

We illustrate the transformation with the following example. To improve readability, we use brackets "[]" or "{}" in place of "()".

Example 1. $K_1[q \wedge K_2(p \vee q) \wedge L_2 r \vee p] \wedge L_1 K_2 \neg q$
$\Leftrightarrow$ (by Proposition 1 (2))$\nabla_1\{q \wedge K_2(p \vee q) \wedge L_2 r \vee p, (q \wedge K_2(p \vee q) \wedge L_2 r \vee p) \wedge K_2\neg q\}$
$\Leftrightarrow$ (by converting to DNF) $\nabla_1\{q \wedge K_2(p \vee q) \wedge L_2 r \vee p, q \wedge K_2(p \vee q) \wedge L_2 r \wedge K_2\neg q \vee p \wedge K_2\neg q\}$
$\Leftrightarrow$ (by combining two K_2 atoms) $\nabla_1\{q \wedge K_2(p \vee q) \wedge L_2 r \vee p, q \wedge K_2(p \wedge \neg q) \wedge L_2 r \vee p \wedge K_2\neg q\}$
$\Leftrightarrow$ (by Proposition 1 (2)) $\nabla_1\{q \wedge \nabla_2\{p \vee q, (p \vee q) \wedge r\} \vee p, q \wedge \nabla_2\{p \wedge \neg q, p \wedge \neg q \wedge r\} \vee p \wedge \nabla_2\{\neg q\}\}$.

2.2.2. Alternating cover disjunctive formulas

Hales et al. [24] introduced the notion of alternating cover disjunctive formulas (ACDFs), and showed that in $KD45_n$, every formula in $\mathcal{L}_K$ is equivalent to such a formula. In this section, we introduce the definition of ACDFs, and present the compilation result together with a complexity analysis of the resulting formula, via introducing the notion of non-alternating factor.

Definition 7. The non-alternating factor of a formula ϕ, denoted by $na(\phi)$, is the number of modal operators of an agent which directly occur inside those of the same agent. We say that a formula is *alternating* if its non-alternating factor is 0.

Definition 8. We call an alternating CDF an ACDF (alternating cover disjunctive formula).

For example, $\nabla_a\{\top, q\} \wedge \nabla_b\{\top, \nabla_a\{\top, \neg q\}\}$ is an ACDF; but the CDF $\nabla_a\{\neg p \wedge q, \nabla_b\{p, q\}, \nabla_a\{\neg p \wedge q\}\}$ is not, and its non-alternating factor is 1.

The compilation result by Hales et al. makes use of the following proposition. To make the paper self-contained, we include a proof of the proposition in the appendix.

Proposition 3 *(van der Hoek and Meyer [27]). The following hold in $KD45_n$:*

1. $K_a(\pi \vee \alpha \wedge K_a\beta) \Leftrightarrow K_a(\pi \vee \alpha) \wedge K_a\beta \vee K_a\pi \wedge \neg K_a\beta$;
2. $K_a(\pi \vee \alpha \wedge L_a\beta) \Leftrightarrow K_a(\pi \vee \alpha) \wedge L_a\beta \vee K_a\pi \wedge \neg L_a\beta$.

The following result was proved by Hales et al. Here we enrich the result with a complexity analysis, and include a proof in the appendix.

Proposition 4 *(Hales et al. [24]). In $KD45_n$, every formula ϕ in $\mathcal{L}_K$ can be transformed to an equivalent alternating formula whose length is bounded by $2^{na(\phi)}|\phi|$.*

The idea of proof is as follows: from the outside in, iteratively apply the two equivalences from Proposition 3.

We illustrate the transformation with the following example.

Example 2. $K_1[q \wedge K_2(p \vee q) \wedge L_2\neg p \ \vee \ p \wedge K_1(p \vee L_1\neg p) \wedge L_1(p \wedge \neg q)] \Leftrightarrow$ (by Proposition 3(2))
$K_1[q \wedge K_2(p \vee q) \wedge L_2\neg p \ \vee \ p \wedge K_1(p \vee L_1\neg p)] \wedge L_1(p \wedge \neg q) \vee K_1[q \wedge K_2(p \vee q) \wedge L_2\neg p] \wedge \neg L_1(p \wedge \neg q)$, where
$K_1[q \wedge K_2(p \vee q) \wedge L_2\neg p \ \vee \ p \wedge K_1(p \vee L_1\neg p)] \Leftrightarrow$ (by Proposition 3(1))
$K_1[q \wedge K_2(p \vee q) \wedge L_2\neg p \vee p] \wedge K_1(p \vee L_1\neg p) \vee K_1[q \wedge K_2(p \vee q) \wedge L_2\neg p] \wedge \neg K_1(p \vee L_1\neg p)$, where
$K_1(p \vee L_1\neg p) \Leftrightarrow$ (by Proposition 3(2)) $L_1\neg p \vee K_1 p \wedge \neg L_1\neg p$.

Finally, we show the compilation result by Hales et al., enriched with a complexity analysis.

Theorem 1 *(Hales et al. [24]). In KD45$_n$, every formula ϕ in $\mathcal{L}_K$ can be transformed to an equivalent ACDF whose length is $O(2^{4^n l^2})$ where $n = na(\phi)$ and $l = |\phi|$.*

Proof. We first apply Proposition 4 to convert ϕ into an equivalent alternating formula ϕ' whose length is bounded by $2^n l$. Then we apply Proposition 2 to convert ϕ' into an equivalent CDF, which remains to be alternating and whose length is $O(2^{4^n l^2})$. □

So the length of the resulting formula is singly exponential in the size of ϕ, but doubly exponential in the non-alternating factor of ϕ.

2.2.3. Satisfiability checking of alternating cover disjunctive formulas

A *modal term* is a conjunction of propositional formulas and modal atoms of the form $K_a\phi$ or $L_a\phi$, where $\phi \in \mathcal{L}_K$. We call a modal term with the alternating agent modality property an *alternating modal term*. In the following, we present a result concerning how to check the satisfiability of alternating modal terms.

Proposition 5. *An alternating modal term $\delta = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}}(K_a\phi_a \wedge L_a\Psi_a)$ is satisfiable w.r.t. constraint γ iff the following hold:*

1. *$\phi_0 \wedge \gamma$ is propositionally satisfiable;*
2. *for each $a \in \mathcal{B}$, ϕ_a is satisfiable w.r.t. γ;*
3. *for each $a \in \mathcal{B}$, for each $\psi \in \Psi_a$, $\phi_a \wedge \psi$ is satisfiable w.r.t. γ.*

Proof. The only-if direction is easy. Since δ is satisfiable w.r.t. γ, *i.e.*, $\delta \wedge C^*\gamma$ is satisfiable, let (M, w) be a model of it. Then w satisfies $\delta \wedge \gamma$. Now let $a \in \mathcal{B}$. Since R_a is serial, there exists w_a s.t. wR_aw_a. Since $M, w \models K_a\phi_a \wedge C^*\gamma$, $M, w_a \models \phi_a \wedge C^*\gamma$. Thus ϕ_a is satisfiable w.r.t. γ. Now let $\psi \in \Psi_a$. Since $M, w \models K_a\phi_a \wedge L_a\psi \wedge C^*\gamma$, there exists w_ψ s.t. wR_aw_ψ and $M, w_\psi \models \phi_a \wedge \psi \wedge C^*\gamma$. Thus $\phi_a \wedge \psi$ is satisfiable w.r.t. γ.

We now prove the if direction. Since $L_a\top \Leftrightarrow \top$, without loss of generality, we assume that for each $a \in \mathcal{B}$, Ψ_a is not empty. Construct a model (M, w) as follows. By condition 1, create a new world w satisfying $\phi_0 \wedge \gamma$. By condition 3, for each $a \in \mathcal{B}$, for each $\psi \in \Psi_a$, there is a KD45$_n$ model (M_ψ, w_ψ) satisfying $\phi_a \wedge \psi \wedge C^*\gamma$, add a new copy of (M_ψ, w_ψ) into M, and let wR_aw_ψ; then add a-edges between all the a-children of w. Thus (M, w) is a KD45$_n$ model. Since δ is an alternating modal term, ϕ_a and $\psi \in \Psi_a$ do not use K_a or L_a as outmost modalities. So we get $M, w_\psi \models \phi_a \wedge \psi$. Also, all worlds of M satisfy γ. Thus $M, w \models \delta \wedge C^*\gamma$. Hence δ is satisfiable w.r.t. γ. □

The following are two easy corollaries of Proposition 5, which will be used later in our paper. The first characterizes whether an alternating modal term entails another, and the second characterizes the satisfiability of an ACDF.

Proposition 6. *Let $\delta = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}}(K_a\phi_a \wedge L_a\Psi_a)$ and $\delta' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}}(K_a\phi'_a \wedge L_a\Psi'_a)$ be two alternating modal terms satisfiable w.r.t. constraint γ. Then $\delta \models_\gamma \delta'$ iff the following hold:*

1. *$\phi_0 \wedge \gamma \models \phi'_0$ propositionally;*
2. *for each $a \in \mathcal{A}$, $\phi_a \models_\gamma \phi'_a$;*
3. *for each $a \in \mathcal{A}$, for every $\psi' \in \Psi'_a$ there is a $\psi \in \Psi_a$ s.t. $\phi_a \wedge \psi \models_\gamma \psi'$.*

Proof. We have $\delta \models_\gamma \delta'$ iff $\delta \wedge \neg\delta'$, which is $\delta \wedge [\neg\phi'_0 \vee \bigvee_{a\in\mathcal{A}}(L_a\neg\phi'_a \vee \bigvee_{\psi'\in\Psi'_a} K_a\neg\psi')]$, is not satisfiable w.r.t. γ iff the following hold:

1. $\delta \wedge \neg\phi'_0$ is not satisfiable w.r.t. γ;
2. for each $a \in \mathcal{A}$, $\delta \wedge L_a\neg\phi'_a$ is not satisfiable w.r.t. γ;
3. for each $a \in \mathcal{A}$, for every $\psi' \in \Psi'_a$, $\delta \wedge K_a\neg\psi'$ is not satisfiable w.r.t. γ.

Since we know δ and δ' are satisfiable w.r.t. γ, by Proposition 5, we have:

1. $\delta \wedge \neg\phi'_0$ is not satisfiable w.r.t. γ iff $\phi_0 \wedge \neg\phi'_0 \wedge \gamma$ is not propositional satisfiable iff $\phi_0 \wedge \gamma \models \phi'_0$ propositionally;

2. $\delta \wedge L_a \neg \phi'_a$ is not satisfiable w.r.t. γ iff $\phi_a \wedge \phi'_a$ is not satisfiable w.r.t. γ iff $\phi_a \models_\gamma \phi'_a$;
3. $\delta \wedge K_a \neg \psi'$ is not satisfiable w.r.t. γ iff there is a $\psi \in \Psi_a$ s.t. $\phi_a \wedge \neg \psi' \wedge \psi$ is not satisfiable w.r.t. γ, *i.e.*, $\phi_a \wedge \psi \models_\gamma \psi'$.

Thus the proposition is proved. □

Proposition 7. *An ACDF term $\delta = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \Phi_a$ is satisfiable w.r.t. a constraint γ iff the following hold:*

1. *$\phi_0 \wedge \gamma$ is propositionally satisfiable;*
2. *for each $a \in \mathcal{B}$, Φ_a is not empty;*
3. *for each $a \in \mathcal{B}$, for each $\phi \in \Phi_a$, ϕ is satisfiable w.r.t. γ.*

Proof. By Proposition 5, $\phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \Phi_a$, which is $\phi_0 \wedge \bigwedge_{a \in \mathcal{B}} K_a(\bigvee \Phi_a) \wedge L_a \Phi_a$, is satisfiable w.r.t. γ iff the following hold:

1. $\phi_0 \wedge \gamma$ is propositionally satisfiable;
2. for each $a \in \mathcal{B}$, $\bigvee \Phi_a$ is satisfiable w.r.t. γ;
3. for each $a \in \mathcal{B}$, for each $\phi \in \Phi_a$, $(\bigvee \Phi_a) \wedge \phi$ is satisfiable w.r.t. γ.

The above 3 conditions are equivalent to the conditions given in the proposition. □

The above proposition gives us a polynomial-time recursive algorithm to check the satisfiability of an ACDF, since the modal depth of ϕ is less than that of δ.

Proposition 8. *Whether an ACDF ϕ is satisfiable w.r.t. a DNF constraint γ can be checked in time $O(|\phi| \cdot |\gamma|)$.*

Proof. We prove by induction on ϕ.

1. ϕ is a propositional term. Let $\gamma = t_1 \vee \ldots t_n$. Then ϕ is satisfiable w.r.t. γ iff there is a term t_i of γ s.t. $\phi \wedge t_i$ is satisfiable iff there is a term t_i of γ s.t. $\phi \wedge t_i$ does not contain complementary literals. This can be checked in time $O(\Sigma_{i=1}^n(|\phi| + |t_i|))$, and hence in time $O(|\phi| \cdot |\gamma|)$.
2. $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \Phi_a$. By induction, whether $\phi_0 \wedge \gamma$ is propositionally satisfiable can be checked in time $O(|\phi_0| \cdot |\gamma|)$; for each $a \in \mathcal{B}$, for each $\phi_a \in \Phi_a$, whether ϕ_a is satisfiable w.r.t. γ can be checked in time $O(|\phi_a| \cdot |\gamma|)$. Thus by Proposition 7, whether ϕ is satisfiable w.r.t. γ can be checked in time $O(|\phi_0| \cdot |\gamma| + \Sigma_{a \in \mathcal{B}} \Sigma_{\phi_a \in \Phi_a} |\phi_a| \cdot |\gamma|)$, and hence in time $O(|\phi| \cdot |\gamma|)$.
3. $\phi = \bigvee \Phi$. Let $\Phi = \{\phi_1, \ldots, \phi_n\}$. Then ϕ is satisfiable w.r.t. γ iff there is ϕ_i which is satisfiable w.r.t. γ. By induction, this can be checked in time $O(\Sigma_{i=1}^n |\phi_i| \cdot |\gamma|)$, and hence in time $O(|\phi| \cdot |\gamma|)$. □

2.3. *Belief revision and update*

In our framework, we describe the world with KBs, and the progression of KBs w.r.t. actions is achieved through the operation of belief revision and update. In this section, we review propositional belief revision and update, which are the bases of higher-order belief revision and update proposed in Section 4.3.

We use $\circ$ to denote a revision operator, and $\diamond$ an update operator. Let ψ be the original formula, and μ the revision or update formula. Both revision and update are guided by the principle of minimal change. To formalize the distinction between revision and update, Katsuno and Mendelzon [30] presented model-theoretic definitions of them: intuitively, $\psi \circ \mu$ selects from the models of μ those that are closest to models of ψ, while $\psi \diamond \mu$ selects, for each model M of ψ, the set of models of μ that are closest to M. As easy properties of the definitions: when $\psi \wedge \mu$ is satisfiable, $\psi \circ \mu$ is equivalent to $\psi \wedge \mu$; update is distributive over the initial formula, *i.e.*, $(\psi_1 \vee \psi_2) \diamond \mu$ is equivalent to $(\psi_1 \diamond \mu \vee \psi_2 \diamond \mu)$.

Let's illustrate the difference between revision and update with an example. Take the notion of closeness based on set inclusion, *i.e.*, a model I is closer to a model M than a model J if $\mathrm{Diff}(I, M) \subseteq \mathrm{Diff}(J, M)$, where $\mathrm{Diff}(I, M)$ is the set of atoms where I and M assign different truth values. Then the above model-theoretic definitions give us Satoh's revision operator $\circ_s$ [41] and Winslett's update operator $\diamond_w$ [48]. For example, let $\psi = (a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)$, and $\mu = a \wedge c$. The models of ψ are $M_1 = \{a, b, c\}$ and $M_2 = \{a, \neg b, \neg c\}$; the models of μ are M_1 and $M_3 = \{a, \neg b, c\}$. Then $\psi \circ_s \mu = a \wedge b \wedge c$, equivalent to $\psi \wedge \mu$, since M_1 is closet to models of ψ. On the other hand, $\psi \diamond_w \mu = a \wedge c$, since M_1 is closest to M_1, but M_3 is closest to M_2.

Del Val [46] provided syntactic characterizations of belief change operators and algorithms based on them. Our higher-level belief change algorithms are recursive ones which as a basis resort to propositional belief change algorithms. To lay the foundation for complexity analysis of our high-level belief change algorithms, below we present his syntactic characterizations of Satoh's revision and Winslett's update operators (Propositions 9 and 11), and analyze the complexity of the algorithms based on these characterizations. The complexity analysis results (Propositions 10 and 12) are simple results not included in the original paper. We include the results here since they will be used later in Section 4.3, in the proofs of Propositions 5 and 7.

We begin with some notation. A DNF formula is treated as the set of its disjuncts, and a term is treated as the set of its literals. We assume that ψ and μ are both in DNF; we use ψ with subscripts to denote the disjuncts of ψ, and similarly for μ. For two terms t_1 and t_2, we use $Dif(t_1, t_2)$ to denote the set of literals in t_1 whose complement occurs in t_2. For a formula ϕ, we use $Prop(\phi)$ to denote the set of atoms occurring in ϕ. For a set S and a partial order $\leq$ on S, we use $Min(S, \leq)$ for the set of elements of S minimal under $\leq$.

First, define

$$\begin{aligned} revise(\psi_i, \mu_j) &= \bigwedge((\psi_i - Dif(\psi_i, \mu_j)) \cup \mu_j); \\ SynMinDif(\psi, \mu) &= Min(\{Prop(Dif(\psi_i, \mu_k)) \mid \psi_i \in \psi, \mu_k \in \mu\}, \subseteq); \\ MinPairs(\psi, \mu) &= \{\langle \psi_i, \mu_j \rangle \mid \psi_i \in \psi, \mu_j \in \mu, Prop(Dif(\psi_i, \mu_j)) \in SynMinDif(\psi, \mu)\}. \end{aligned}$$

The following two propositions show the propositional revision operation and its computational complexity.

Proposition 9 *(Theorem 7 in del Val [46]).*

$$\psi \circ_s \mu \Leftrightarrow \bigvee_{\langle \psi_i, \mu_j \rangle \in MinPairs(\psi, \mu)} revise(\psi_i, \mu_j).$$

Proposition 10. *Let ψ and μ be two DNF formulas. Then $\psi \circ_s \mu$ can be computed in time $O(|\psi|^2 \cdot |\mu|^2)$, and the resulting formula is of size $O(|\psi| \cdot |\mu|)$.*

The following two propositions show the propositional update operation and its computational complexity.

$$patch_{\psi_i}(\mu_j) = \bigwedge_{\mu_k \in \mu, Dif(\mu_j, \psi_i) \nsubseteq \mu_k} \neg \bigwedge(\mu_k - (\psi_i \cup \mu_j)).$$

Then we have:

Proposition 11 *(Theorem 1 in del Val [46]).*

$$\psi \diamond_w \mu \Leftrightarrow \bigvee_{\mu_j \in \mu, \psi_i \in \psi} revise(\psi_i, \mu_j) \wedge patch_{\psi_i}(\mu_j)$$

Proposition 12. *Let ψ and μ be two DNF formulas. The DNF formula of $\psi \diamond_w \mu$ can be computed in time $O(|\psi| \cdot 2^{|\mu|})$, and the resulting formula is of size $O(|\psi| \cdot 2^{|\mu|})$.*

3. Our modeling framework

In this section, we present our modeling framework for multi-agent epistemic planning (MEP), which is adapted from that for single-agent epistemic planning by Wan et al. [47].

We illustrate our framework with the collaboration via communication example from the introduction.

Example 3. As shown in Fig. 1, there is a corridor of three rooms p_1, p_2 and p_3. Two boxes b_1 and b_2 are located in some of the rooms. Two agents 1 and 2 can move back and forth along this corridor. When an agent gets into a room, she can see if a box is in the room. An agent can communicate information to another agent. Initially, the two agents are in p_2 and the two boxes are not there. The goal is for agent 1 to know the position of b_1, and for agent 2 to know the position of b_2.

We begin with the definition of MEP problems, then give the definition of different kinds of actions and the associated progression operations, and end with the definition of MEP solutions. The actions we consider include *ontic*, *communication* and *sensing* actions. The first two kinds share the same representation, and we call them *deterministic* actions.

3.1. Multi-agent epistemic planning problems

Definition 9. A multi-agent epistemic planning problem $\mathcal{Q}$ is a tuple $\langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$, where $\mathcal{A}$ is a set of agents; $\mathcal{P}$ is a set of atoms; $\mathcal{D}$ is a set of deterministic actions; $\mathcal{S}$ is a set of sensing actions; $\mathcal{I} \in \mathcal{L}_{\mathcal{K}}$ is the initial KB; $\mathcal{G} \in \mathcal{L}_{\mathcal{K}}$ is the goal; and $\gamma \in \mathcal{L}_0$ is the constraint.

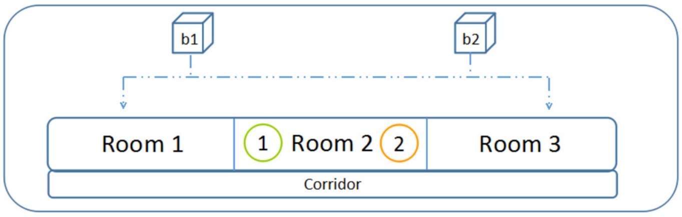

Fig. 1. Illustration for Example 3.

Note that we have $\mathcal{I} \in \mathcal{L}_{\mathcal{K}}$ and $\mathcal{G} \in \mathcal{L}_{\mathcal{K}}$. $\mathcal{I}$ and $\mathcal{G}$ actually describe the beliefs of a third person. Thus the propositional parts of $\mathcal{I}$ and $\mathcal{G}$ describe the third person's beliefs about the world. Since we model beliefs rather than knowledge, the subjective parts of $\mathcal{I}$ and $\mathcal{G}$ might not agree with the propositional parts.

The main reason we consider constraints in this paper is for natural modeling of planning domains: we use constraints to express static propositional common knowledge, as will be illustrated in our formalization of Example 3.

We now formalize Example 3.

- The atoms are: $at(i, p)$, meaning agent i is in room p; and $in(b, p)$, meaning box b is in room p.
- The ontic actions are: $left(i)$, agent i moves left; and $right(i)$, i moves right.
- The communication actions are: $tell(i, j, b, p)$, agent i tells j whether b is in p.
- The sensing actions are: $find(i, b, p)$: i sees if b is in p.
- The initial KB is $at(1, p_2) \wedge at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2) \wedge K_1(at(1, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2)) \wedge K_2(at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2))$;
- The goal is $\bigwedge_{i=1}^{2}(K_i in(b_i, p_1) \vee K_i in(b_i, p_2) \vee K_i in(b_i, p_3))$, meaning agent 1 knows the position of box b_1, and agent 2 knows the position of box b_2.
- The constraint is $\gamma_1 \wedge \gamma_2$, where $\gamma_1 = \bigwedge_{i=1}^{2}(at(i, p_1) \wedge \neg at(i, p_2) \wedge \neg at(i, p_3) \vee \neg at(i, p_1) \wedge at(i, p_2) \wedge \neg at(i, p_3) \vee \neg at(i, p_1) \wedge \neg at(i, p_2) \wedge at(i, p_3))$, meaning each agent is at exactly one room, and γ_2 is similar, representing each box is in exactly one room.

The reason that we have in the initial KB $at(1, p_2) \wedge K_1 at(1, p_2)$ instead of simply $K_1 at(1, p_2)$ is that we model beliefs rather than knowledge.

3.2. Actions and progression

Definition 10. A deterministic action is a pair $\langle pre, eff\rangle$, where $pre \in \mathcal{L}_K$ is the precondition; eff is a set of conditional effects, each of which is a pair $\langle con, cef\rangle$, where $con, cef \in \mathcal{L}_K$ indicate the condition and the effect, respectively. Let $eff = \{\langle c_1, e_1\rangle, \ldots, \langle c_n, e_n\rangle\}$. We require that eff should be consistent w.r.t. constraint γ, *i.e.*, for any non-empty $I \subseteq \{1, \ldots, n\}$, if $\bigwedge_{i\in I} e_i$ is unsatisfiable w.r.t. γ, so is $\bigwedge_{i\in I} c_i$.

Similarly to Definition 9, $pre, con, cef \in \mathcal{L}_K$, and they thus may contain propositional parts, which actually model the beliefs of a third person about the world.

For example, $left(i) = \langle pre, \{eff_1, eff_2\}\rangle$, where $pre = \neg at(i, p_1) \wedge K_i \neg at(i, p_1)$, $eff_1 = \langle at(i, p_2), at(i, p_1) \wedge K_i at(i, p_1)\rangle$, and $eff_2 = \langle at(i, p_3), at(i, p_2) \wedge K_i at(i, p_2)\rangle$. Here $\{eff_1, eff_2\}$ is consistent w.r.t. $\gamma_1 \wedge \gamma_2$, since $at(i, p_2) \wedge at(i, p_3)$ is unsatisfiable w.r.t. $\gamma_1 \wedge \gamma_2$.

For another example, $tell(i, j, b, p) = \langle pre, \{eff_1, eff_2\}\rangle$, where $pre = K_i in(b, p) \vee K_i \neg in(b, p)$, $eff_1 = \langle K_i in(b, p), K_j in(b, p)\rangle$, and $eff_2 = \langle K_i \neg in(b, p), K_j \neg in(b, p)\rangle$. Here $\{eff_1, eff_2\}$ is consistent, since $K_i in(b, p) \wedge K_i \neg in(b, p)$ is unsatisfiable.

Actually, when $K_i in(b, p)$, $tell(i, j, b, p)$ should result in common knowledge of $in(b, p)$ between the two agents. Since we do not support arbitrary common knowledge, we have to approximate common knowledge with higher-order knowledge. For example, we can express the conditional effect with $K_j in(b, p) \wedge K_i K_j in(b, p) \wedge K_j K_i in(b, p) \wedge K_i K_j K_i in(b, p) \wedge K_j K_i K_j in(b, p)$.

Definition 11. A sensing action is a triple $\langle pre, pos, neg\rangle$ of $\mathcal{L}_K$ formulas, where pre, pos, and neg indicate the precondition, the positive result, and the negative result, respectively. We require that $pos \wedge neg$ should be unsatisfiable w.r.t. γ.

Similarly to Definition 9, $pos, neg \in \mathcal{L}_K$. The fact that pos and neg may have propositional parts does not mean that sensing actions may change the world, but means that they may change the beliefs of a third person about the world.

For the example of $find(i, b, p)$, $pre = at(i, p) \wedge K_i at(i, p)$, $pos = in(b, p) \wedge K_i in(b, p)$, and $neg = \neg in(b, p) \wedge K_i \neg in(b, p)$. This is an example of an accurate sensing action, since beliefs agree with ground truth. If $pos = K_i in(b, p)$, and $neg = K_i \neg in(b, p)$, we have a possibly noisy sensing action.

An action a is *executable* w.r.t. a KB $\phi \in \mathcal{L}_K$ if $\phi \models_\gamma pre(a)$. This means that a is executable in each model of ϕ. For example, $left(1)$ is executable w.r.t. $\mathcal{I}$, since $\mathcal{I} \models_\gamma pre(left(1))$, which is $\neg at(1,p_1) \wedge K_1 \neg at(1,p_1)$. Suppose a is executable w.r.t. ϕ. The *progression* of ϕ w.r.t. a is defined by resorting to a revision operator $\circ_\gamma$ and an update operator $\diamond_\gamma$ (where γ is a constraint) for $\mathcal{L}_K$. Here, both $\circ_\gamma$ and $\diamond_\gamma$ are generic higher-order belief change operators. We will formally define our specific higher-order belief change operators in Section 4.3.

We use update for ontic actions, and revision for communication and sensing actions. We justify this as follows. It is well-accepted that propositional revision concerns belief change about static environments due to partial and possibly incorrect information, whereas propositional update concerns belief change about dynamic environments due to the performance of actions. Propositional belief revision and update are actually revision and update of first-order beliefs, *i.e.*, beliefs about the objective world. Thus, it is natural to expect $K_i\phi \bullet K_i\mu \Leftrightarrow K_i(\phi \bullet \mu)$, where $\bullet$ is $\circ$ or $\diamond$, ϕ and μ are propositional formulas. Note that when we write $K_i\phi \bullet K_i\mu$, we make the assumption that ϕ is all agent i knows. In general, it is natural to reduce higher-order belief revision and update to lower-order ones. Thus eventually, higher-order belief revision and update reduce to first-order ones. So higher-order belief revision and update concern belief change about static and dynamic environments (here by environments, we mean worlds), respectively. Thus we use update for ontic actions, and revision for communication and sensing actions.

Nonetheless, it might be controversial whether to use revision or update for communication or sensing actions. One argument is as follows: In our framework, a formula $\phi \in \mathcal{L}_K$ represents the belief of a third person. When sensing or communication actions happen, the belief states of the involved agents change, and hence the environment of the third person changes. Thus we should use update for sensing and communication actions. There is also an argument that both revision and update should be involved in communication: Suppose agent i is told ϕ. Then i should revise her beliefs, while other agents, including the third person, have to update their beliefs about what i believes. So it is a subtle and tricky issue of whether to use revision or update for communication or sensing actions. We will leave a more thorough exploration of this issue as future work.

We begin with progression of deterministic actions. Let $\phi \in \mathcal{L}_K$, and a be a deterministic action where $eff(a) = \{\langle c_1, e_1\rangle, \ldots, \langle c_n, e_n\rangle\}$. We follow To et al.'s way to process conditional effects [44]. There are two ideas behind our definition of progression of ϕ w.r.t. a. Firstly, we conjoin all applicable effects, and revise or update with the result. Secondly, to decide if effect e_i is applicable, we consider three cases:

1. if $\phi \models_\gamma c_i$, then e_i is applicable, since e_i is applicable in each model of ϕ;
2. if $\phi \models_\gamma \neg c_i$, then e_i is not applicable, since e_i is applicable in none model of ϕ;
3. otherwise, we split ϕ into $\phi \wedge c_i$, where e_i is applicable, and $\phi \wedge \neg c_i$, where e_i is not applicable.

Let $I = \{1, \ldots, n\}$. We use I^+ to denote the set of those i s.t. $\phi \models_\gamma c_i$, and I^- the set of i s.t. $\phi \models_\gamma \neg c_i$. For each $I' \subseteq I^* = I - I^+ - I^-$, we get a splitting of ϕ by conjoining c_i for $i \in I'$ and $\neg c_i$ for $i \in I^* - I'$. Thus we have the following definition.

Definition 12. Let $\phi \in \mathcal{L}_K$, and a be a deterministic action where $eff(a) = \{\langle c_1, e_1\rangle, \ldots, \langle c_n, e_n\rangle\}$. Let $I = \{1, \ldots, n\}$, $I^+ = \{i \in I \mid \phi \models_\gamma c_i\}$, $I^- = \{i \in I \mid \phi \models_\gamma \neg c_i\}$, and $I^* = I - I^+ - I^-$.

1. A splitting of ϕ w.r.t. $I' \subseteq I^*$ is $\phi_s = \phi \wedge \bigwedge\{c_i \mid i \in I'\} \wedge \bigwedge\{\neg c_i \mid i \in I^* - I'\}$ s.t. ϕ_s is satisfiable w.r.t. γ.
2. The progression of ϕ_s is $\phi_s \bullet_\gamma \bigwedge\{e_i \mid i \in I^+ \cup I'\}$, where $\bullet$ is $\diamond$ if a is ontic, and $\bullet$ is $\circ$ if a is a communication action.
3. The progression of ϕ w.r.t. a is the disjunction of progressions of all splittings of ϕ.

For example, let $\phi = r$, and $eff(a) = \{\langle p, q\rangle, \langle \neg p, q\rangle\}$. Then the progression of ϕ w.r.t. a is $((r \wedge p) \diamond_\gamma q) \vee ((r \wedge \neg p) \diamond_\gamma q)$.

In the above definition, in the worst case, $I^* = I$, so there are 2^n splitting of ϕ, where n is the number of conditional effects. However, in our experimental domains, usually, for each $i \in I$, either $\phi \models_\gamma c_i$ or $\phi \models_\gamma \neg c_i$, hence $I^* = \emptyset$, and there is no splitting of ϕ. Also, in our experimental domains, normally, I^+ is small, so a small number of conditional effects are applicable.

We now turn to sensing actions. Given a KB $\phi \in \mathcal{L}_K$, and a sensing action a, a certain sensing result of a might be impossible. For example, let $\phi = \neg in(b, p)$, and $a = find(i, b, p)$. Then $pos(a)$ is impossible, since it is contradictory to knowledge about the objective world. However, if $\phi = K_i \neg in(b, p)$, then $pos(a)$ is possible, since the current belief of agent i might be false. Whether $pos(a)$ is possible can be detected via checking if $\phi \wedge pos(a)$ is propositionally satisfiable, *i.e.*, the propositional part of $\phi \wedge pos(a)$ is satisfiable. When $pos(a)$ is impossible, we let the progression of ϕ w.r.t. $pos(a)$ be $\bot$, otherwise, we let it be $\phi \circ_\gamma pos(a)$. Thus we have

Definition 13. Let $\phi \in \mathcal{L}_K$. Then ϕ can be equivalently transformed to an ACDF formula ϕ'. In ϕ', we replace any occurrence of the $\nabla_a\Phi$ formula by $\top$. We say ϕ is propositionally satisfiable if the resulting propositional formula is satisfiable.

Definition 14. Let $\phi \in \mathcal{L}_K$, and a a sensing action. Then the progression of ϕ w.r.t. a with positive result is

$$\phi^+ = \begin{cases} \bot & \text{if } \phi \wedge pos(a) \text{ is propositionally unsatisfiable w.r.t. } \gamma \\ \phi \circ_\gamma pos(a) & \text{otherwise.} \end{cases}$$

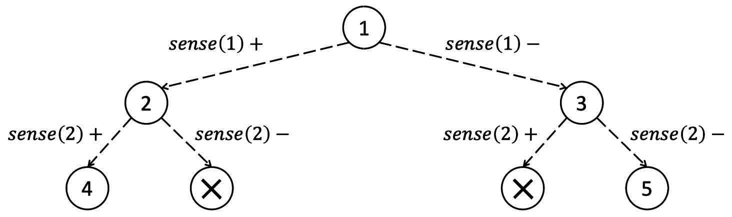

Fig. 2. A solution to an example with impossible sensing results.

The progression of ϕ w.r.t. a with negative result is similarly defined.

Note that by the above definition, the cases of impossible objective sensing results are handled properly. However, we are not able to handle impossible subjective sensing results. For example, in our framework, it is possible for agent i to first sense that j believes p, and then (immediately afterwards) that j doesn't believe p.

Definition 15. The progression of ϕ w.r.t. a sequence of actions (with sensing results for sensing actions) is inductively defined as follows: $prog(\phi, \epsilon) = \phi$, where ϵ represents the empty sequence; $prog(\phi, (a; \sigma)) = prog(prog(\phi, a), \sigma)$ if $\phi \models pre(a)$, and undefined otherwise.

3.3. Multi-agent epistemic planning solutions

A solution of an MEP problem is an action tree branching on sensing results such that the progression of the initial KB w.r.t. each branch in the tree entails the goal.

Definition 16. Let $\mathcal{Q}$ be an MEP problem $\langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$. The set $\mathcal{T}$ of action trees is inductively defined:

1. ϵ is in $\mathcal{T}$, here ϵ represents the empty tree;
2. if $a_d \in \mathcal{D}$ and $T \in \mathcal{T}$, then $a_d; T$ is in $\mathcal{T}$;
3. if $a_s \in \mathcal{S}$, $T^+, T^- \in \mathcal{T}$, then a_s; $(T^+ \mid T^-)$ is in $\mathcal{T}$.

Definition 17. Let $\mathcal{Q}$ be an MEP problem $\langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$. Let T be an action tree. We say a branch σ of T achieves the goal if $prog(\mathcal{I}, \sigma)$ is defined, and $prog(\mathcal{I}, \sigma) \models_\gamma \mathcal{G}$; and if $prog(\mathcal{I}, \sigma)$ is not $\bot$, we say σ properly achieves the goal. We say T is a solution of $\mathcal{Q}$ if each branch of T achieves the goal, and at least one branch properly achieves the goal.

Intuitively, we use a KB to model an epistemic state. When an action is performed, we revise or update the current KB with the action's effects. The problem is solved if for each possible sequence of actions (with sensing results for sensing actions), the final KB entails the goal.

Our definition is inspired by the PKS approach [39] to first-order epistemic planning. In the paper, the authors state that "The intuition behind our approach is that a planning agent operating under conditions of incomplete knowledge (and without a model of uncertainty) can only build plans based on what it knows and on how its knowledge will change as it executes actions – it has access to no other information at plan time." PKS uses a database collection **DB** to represent the agent's incomplete knowledge. When an action is chosen, its effects are applied to update **DB**. When branching on $K\alpha \vee K\neg\alpha$, the formula is removed from **DB**, and either α or $\neg\alpha$ is added to **DB**. A plan is found, if along each branch, the resulting **DB** satisfies the goal.

Note that since $\bot \models \phi$ for any ϕ, if a branch ends with an impossible sensing result, it achieves the goal. For example, there are two agents, and action $sense(i)$ senses the truth value of atom p. Suppose the initial KB is $\top$, and the goal is $K_1 p \wedge K_2 p \vee K_1 \neg p \wedge K_2 \neg p$. Then Fig. 2 shows a solution, where impossible sensing results are marked with $\times$.

Note that our definition of an MEP solution is a solution of centralized planning from the viewpoint of a third person who coordinates all the agents to achieve the goal. That is, with our MEP framework, a solution is computed offline, and then a third person monitors the execution of the plan, and instructs the agents to perform actions according to the execution states: if the currently executed action is a deterministic one, when it is done, the third person instructs the next action to be performed; if the currently executed action is a sensing action, according to the sensing result, the third-person instructs one of the successor actions to be performed. We illustrate this with an example below.

Fig. 3 shows a solution for Example 3. First, agent 1 is instructed to move left, and sense if b_1 is in p_1. If b_1 is in p_1, agent 2 is instructed to move right, and sense if b_2 is in p_3. If b_1 is not in p_1, agent 1 knows b_1 is in p_3, and she is instructed to sense if b_2 is in p_1, and tell the result to agent 2.

Finally, for illustration purpose, we present another example, which formalizes an instance of the classic Gossip problem [2].

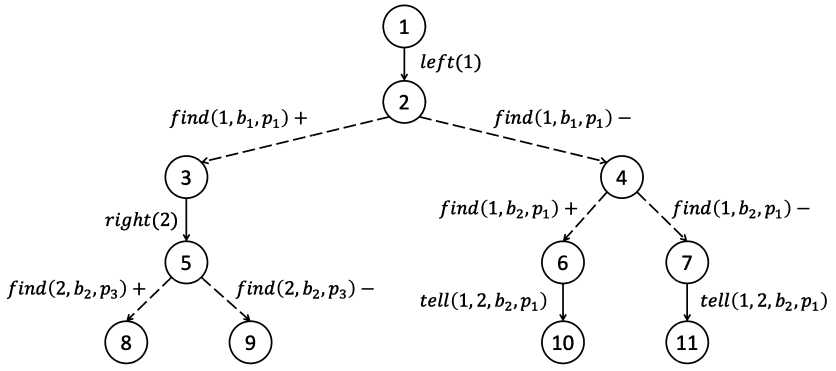

Fig. 3. A solution to Example 3.

Example 4. There are three agents 1, 2, and 3. Each of them has her own secret s_1, s_2, and s_3. Instead of sharing in public, they are only allowed to make a call to each other. In each call, they exchange all the secrets they know. The goal is that everyone knows all the secrets.

- The atoms are: $s(i)$, meaning the secret of agent i.
- The communication actions are: $share(i, j, k) = \langle pre, \{eff_1, eff_2, eff_3\}\rangle$, agent i calls agent j to exchange all the secrets they know, where
  - $pre = K_i(s_i \land \neg K_j s_i) \lor K_i(s_k \land \neg K_j s_k)$, meaning that i knows some secret that j does not know;
  - $eff_1 = \langle \top, K_i s_i \land K_j s_j\rangle$, meaning that each of i and j gets to know the secret of the other agent;
  - $eff_2 = \langle K_i s_k, K_j s_k\rangle$, meaning that j gets to know k's secret if i knows k's secret;
  - $eff_3 = \langle K_j s_k, K_i s_k\rangle$, meaning that i gets to know k's secret if j knows k's secret.
- The initial KB is

$$s_1 \land s_2 \land s_3 \land \neg K_1 s_2 \land \neg K_1 s_3 \land \neg K_2 s_1 \land \neg K_2 s_3 \land \neg K_3 s_1 \land \neg K_3 s_2 \land$$
$$K_1(s_1 \land \neg K_2 s_1 \land \neg K_3 s_1) \land K_2(s_2 \land \neg K_1 s_2 \land \neg K_3 s_2) \land K_3(s_3 \land \neg K_1 s_3 \land \neg K_2 s_3),$$

  meaning that initially each agent knows her own secret and knows that the other agents do not know her secret.
- The goal is $\bigwedge_{i=1}^{3} K_i(s_1 \land s_2 \land s_3)$, meaning that each agent knows all secrets.
- The constraint is $\top$.

3.4. Conformant vs contingent planning

Our framework can model both conformant and contingent planning problems. In contingent planning, we need sensing actions to generate branching plans. In conformant planning, the action that agent i senses the truth value of p is modeled as a deterministic action $\alpha = \langle pre, \{\langle p, K_i p\rangle, \langle \neg p, K_i \neg p\rangle\}\rangle$. The progression of a KB w.r.t. α is a single KB. The progression of a KB w.r.t. a sensing action results in different KBs for different sensing results.

3.5. Coincidence with propositional conformant planning

In this section, we show that for the fragment of propositional conformant planning, our MEP solution concept coincides with the standard one.

We first review propositional conformant planning, following the formal representations presented in [44]. A planning problem is a tuple $P = \langle F, A, I, G\rangle$ where F is a set of atoms, A is a set of actions, I is an initial condition, and G is a goal condition. Each action $a \in A$ is a pair $\langle pre(a), eff(a)\rangle$ where $pre(a)$ is the precondition, and $eff(a)$ is a set of conditional effects. Each conditional effect is a pair (c, e) where c is the condition and e is the effect. Each of the preconditions, conditions and effects is a set of literals. A literal is an atom p or its negation $\neg p$, which are complements to each other.

A state s can be treated as the set of literals holding in s. Action a is applicable in state s if $s \models pre(a)$, and the resulting set of triggered effects, written $eff(s, a)$, is the union of e such that $(c, e) \in eff(a)$ and $s \models c$. The result of applying a in s is a new state $\theta(s, a) = s/\neg eff(s, a) \cup eff(s, a)$, *i.e.*, the state obtained from s as follows: for each literal $l \in eff(s, a)$, first delete from s the complement of l and then add l.

A belief state S is a set of states. A formula ϕ represents S if S coincides with the set of all states satisfying ϕ. Action a is applicable in belief state S if it is applicable in every state in S. The result of applying a in S is a new belief state $\Theta(S, a) = \{\theta(s, a) \mid s \in S\}$.

A solution for a planning problem P is an action sequence $\pi = \langle a_1, ..., a_n\rangle$ that induces a belief state sequence $\langle S_0, S_1, ..., S_n\rangle$ such that S_0 is the set of all states satisfying I, every state in S_n satisfies G, and for each i such that $1 \le i \le n$, a_i is applicable in S_{i-1} and $S_i = \Theta(S_{i-1}, a_i)$.

Proposition 13. *Let s be a state, and t a subset of a state. Then $s \diamond_w t = s/\neg t \cup t$; and if $s \models \phi$, then $s \diamond_w t \models \phi \diamond_w t$.*

Proof. These follow from the model-theoretic definition of $\diamond_w$. $\square$

Proposition 14. *Suppose that the higher-order belief update operator coincides with Winslett's update operator on propositional formulas. Let a be an action. If a DNF formula ϕ represents a belief state S, then $prog(\phi, a)$ represents $\Theta(S, a)$.*

Proof. As in Definition 12, let $eff(a) = \{\langle c_1, e_1\rangle, \ldots, \langle c_n, e_n\rangle\}$, $I = \{1, \ldots, n\}$, $I^+ = \{i \in I \mid \phi \models_\gamma c_i\}$, $I^- = \{i \in I \mid \phi \models_\gamma \neg c_i\}$, and $I^* = I - I^+ - I^-$.

We first prove that for any $s \in S$, $\theta(s, a) \models prog(\phi, a)$. Let $I' = \{i \in I - I^+ \mid s \models c_i\}$. Since $s \models \phi$, for $i \in I^+$, we have $s \models c_i$. Then $eff(s, a) = \bigwedge\{e_i \mid i \in I^+ \cup I'\}$. Let ϕ_s be the splitting of ϕ w.r.t. I', *i.e.*, $\phi_s = \phi \wedge \bigwedge\{c_i \mid i \in I'\} \wedge \bigwedge\{\neg c_i \mid i \in I^* - I'\}$. Then $s \models \phi_s$. Then $\theta(s, a) = s \diamond_w eff(s, a) \models \phi_s \diamond_w \bigwedge\{e_i \mid i \in I^+ \cup I'\}$. Thus $\theta(s, a) \models prog(\phi, a)$.

We now prove that for any $s' \models prog(\phi, a)$, there exists $s \in S$ s.t. $s' = \theta(s, a)$. Since $s' \models prog(\phi, a)$, there exists a splitting $\phi_s = \phi \wedge \bigwedge\{c_i \mid i \in I'\} \wedge \bigwedge\{\neg c_i \mid i \in I^* - I'\}$ s.t. $s' \models \phi_s \diamond_w \bigwedge\{e_i \mid i \in I^+ \cup I'\}$. Thus there exists $s \models \phi_s$ s.t. $s' = s \diamond_w \bigwedge\{e_i \mid i \in I^+ \cup I'\}$. Since $s \models \phi_s$, $eff(s, a) = \bigwedge\{e_i \mid i \in I^+ \cup I'\}$. Thus $s' = s \diamond_w eff(s, a) = \theta(s, a)$. $\square$

For the fragment of propositional conformant planning, an MEP solution is an action sequence $\pi = \langle a_1, \ldots, a_n\rangle$ that induces a formula sequence $\langle \phi_0, \phi_1, \ldots, \phi_n\rangle$ such that ϕ_0 is the initial KB, $\phi_n \models \mathcal{G}$, and for each i such that $1 \le i \le n$, $\phi_{i-1} \models pre(a_i)$, and $\phi_i = prog(\phi_{i-1}, a_i)$. Thus, by the above proposition, we have

Theorem 2. *Suppose that the higher-order belief update operator coincides with Winslett's update operator on propositional formulas. Then for the fragment of propositional conformant planning, our MEP solution concept coincides with the standard one.*

4. Our algorithms

In this section, we present our reasoning and belief change algorithms.

To support efficient reasoning and belief change, we represent KBs as ACDFs, queries as the negation of ACDFs, revision or update formulas as ACDFs, and constraints as DNF formulas. Thus, for a planning problem, our planner first compiles the initial KB, the effects of conditional effects of deterministic actions, and the positive and negative results of sensing actions into ACDFs, the preconditions of actions, the conditions of conditional effects, and the goal into the negation of ACDFs, and finally the constraint into DNF formula. When progressing w.r.t. deterministic actions (see Definition 12), we compile each splitting of KB and each conjunction of applicable effects into ACDFs.

For Example 3, after compilation, we get

- $\mathcal{I} = at(1, p_2) \wedge at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2) \wedge$
  $\nabla_1\{at(1, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2)\} \wedge \nabla_2\{at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2)\}$;
- $pre(left(i)) = \neg(at(i, p_1) \vee \nabla_i\{\top, at(i, p_1)\})$; note this is the negation of an ACDF;
- $cef(eff_1(left(i))) = at(i, p_1) \wedge \nabla_i\{at(i, p_1)\}$;
- $pos(find(i, b, p)) = in(b, p) \wedge \nabla_i\{in(b, p)\}$;
- $\mathcal{G} = \neg \bigvee_{i=1}^{2} \nabla_i\{\top, \neg in(b_i, p_1), \neg in(b_i, p_2), \neg in(b_i, p_3)\}$;
- γ is a DNF formula of 81 terms.

4.1. Strong entailment and equivalence

Our planner searches through the space of KBs, represented as ACDFs, and performs loop detection during search to avoid generating duplicate KBs. Unfortunately, it is not tractable to check the equivalence of two ACDFs. Thus we introduce a stronger notion of equivalence which is computationally less demanding. When doing loop detection, we check if two ACDFs are strongly equivalent. In this way, we reduce the search space to a limited degree.

Our notion of strong equivalence is defined via defining the notion of strong entailment.

Definition 18. Let ϕ and ϕ' be two ACDFs, γ a DNF formula. The strong entailment relation $\phi \mapsto_\gamma \phi'$ is recursively defined:

1. For propositional terms ϕ and ϕ', $\phi \mapsto_\gamma \phi'$ if $\phi \wedge \gamma \models \phi'$.
2. When $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{A}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{A}} \nabla_a \Phi'_a$, $\phi \mapsto_\gamma \phi'$ if the following hold:
   (a) $\phi_0 \mapsto_\gamma \phi'_0$;
   (b) for each $a \in \mathcal{A}$, for each $\phi_a \in \Phi_a$ there exists $\phi'_a \in \Phi'_a$ s.t. $\phi_a \mapsto_\gamma \phi'_a$;
   (c) for each $a \in \mathcal{A}$, for each $\phi'_a \in \Phi'_a$ there exists $\phi_a \in \Phi_a$ s.t. $\phi_a \mapsto_\gamma \phi'_a$.
3. When $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$, $\phi \mapsto_\gamma \phi'$ if for all $\phi_i \in \Phi$ there exists $\phi'_j \in \Phi'$ s.t. $\phi_i \mapsto_\gamma \phi'_j$.

The following proposition shows the difference between strong entailment and entailment, which lies with Items 2(b) and 3.

Proposition 15. *Let ϕ and ϕ' be two ACDFs, γ a DNF formula.*

1. *When ϕ and ϕ' are propositional terms, $\phi \models_\gamma \phi'$ iff $\phi \wedge \gamma \models \phi'$.*
2. *When $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi'_a$, $\phi \models_\gamma \phi'$ iff the following hold:*
   (a) *$\phi_0 \models_\gamma \phi'_0$;*
   (b) *for each $a \in \mathcal{A}$, for every $\phi_a \in \Phi_a$, $\phi_a \models_\gamma \bigvee \Phi'_a$;*
   (c) *for each $a \in \mathcal{A}$, for every $\phi'_a \in \Phi'_a$ there is $\phi_a \in \Phi_a$ s.t. $\phi_a \models_\gamma \phi'_a$.*
3. *When $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$, $\phi \models_\gamma \phi'$ iff for all $\phi_i \in \Phi$, $\phi_i \models_\gamma \bigvee \Phi'$.*

Proof. (1) and (3) are easy. We prove (2). Recall $\nabla_a \Phi = K_a(\bigvee \Phi) \wedge L_a \Phi$. By Proposition 6, $\phi \models_\gamma \phi'$ iff the following hold:

1. $\phi_0 \models_\gamma \phi'_0$;
2. for each $a \in \mathcal{A}$, $\bigvee \Phi_a \models_\gamma \bigvee \Phi'_a$, *i.e.*, for every $\phi_a \in \Phi_a$, $\phi_a \models_\gamma \bigvee \Phi'_a$;
3. for each $a \in \mathcal{A}$, for every $\phi'_a \in \Phi'_a$ there is $\phi_a \in \Phi_a$ s.t. $\phi_a \models_\gamma \phi'_a$. □

By induction, it is easy to prove:

Proposition 16. *The strong entailment relation is reflexive and transitive.*

Proposition 17. *Let ϕ and ϕ' be two ACDFs, and γ a DNF formula. If $\phi \mapsto_\gamma \phi'$, then $\phi \models_\gamma \phi'$.*

Proof. We prove by induction.

1. ϕ and ϕ' are propositional terms. By Definition 18, $\phi \wedge \gamma \models \phi'$. Hence $\phi \wedge C^*\gamma \models \phi'$, *i.e.*, $\phi \models_\gamma \phi'$.
2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi'_a$. By Definition 18,
   (a) $\phi_0 \mapsto_\gamma \phi'_0$, *i.e.*, $\phi_0 \wedge \gamma \models \phi'_0$.
   (b) for each $a \in \mathcal{A}$, for each $\phi_a \in \Phi_a$ there exists $\phi'_a \in \Phi'_a$ s.t. $\phi_a \mapsto_\gamma \phi'_a$. By induction, $\phi_a \models_\gamma \phi'_a$. Hence $\phi_a \models_\gamma \bigvee \Phi'_a$.
   (c) for each $a \in \mathcal{A}$, for each $\phi'_a \in \Phi'_a$ there exists $\phi_a \in \Phi_a$ s.t. $\phi_a \mapsto_\gamma \phi'_a$. By induction, $\phi_a \models_\gamma \phi'_a$.
   Thus the 3 conditions of Proposition 15(2) are satisfied. Hence $\phi \models_\gamma \phi'$.
3. $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$. By Definition 18, for all $\phi \in \Phi$ there exists $\phi' \in \Phi'$ s.t. $\phi \mapsto_\gamma \phi'$. By induction, $\phi \models_\gamma \phi'$. Hence $\phi \models_\gamma \bigvee \Phi'$. So $\bigvee \Phi \models_\gamma \bigvee \Phi'$. □

The definition of strong entailment gives us a recursive algorithm to check strong entailment. As shown below, the complexity of the algorithm is exponential in $md(\phi \vee \phi')$, *i.e.*, the maximal modal depth of ϕ and ϕ', which is usually small.

Theorem 3. *Let ϕ and ϕ' be two ACDFs, γ a DNF formula. The strong entailment relation $\phi \mapsto_\gamma \phi'$ can be checked in time $O(2^{md(\phi\vee\phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$.*

Proof. We prove by induction.

1. ϕ and ϕ' are propositional terms. Then $\phi \mapsto_\gamma \phi'$ iff $\phi \wedge \gamma \models \phi'$ iff for each term t of γ, $\phi \wedge t \models \phi'$ iff for each term t of γ and for each literal l of ϕ', l is contained in $\phi \wedge t$. This can be checked in time $O(q)$, where $q = \Sigma_t \Sigma_l (|\phi| \cdot |t|) \leq |\phi| \cdot |\phi'| \cdot |\gamma|$.
2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi'_a$. By induction, whether $\phi \mapsto_\gamma \phi'$ can be checked in time $O(q)$, where

$$\begin{aligned} q &= |\phi| \cdot |\phi'| \cdot |\gamma| + 2\Sigma_a \Sigma_{\phi_a \in \Phi_a} \Sigma_{\phi'_a \in \Phi'_a} 2^{md(\phi_a \vee \phi'_a)} \cdot |\phi_a| \cdot |\phi'_a| \cdot |\gamma| \\ &\leq |\phi| \cdot |\phi'| \cdot |\gamma| + 2^{md(\phi\vee\phi')} \Sigma_a \Sigma_{\phi_a \in \Phi_a} \Sigma_{\phi'_a \in \Phi'_a} |\phi_a| \cdot |\phi'_a| \cdot |\gamma| \\ &\leq |\phi| \cdot |\phi'| \cdot |\gamma| + 2^{md(\phi\vee\phi')} \Sigma_a |\nabla_a \Phi_a| \cdot |\nabla_a \Phi'_a| \cdot |\gamma| \\ &\leq 2^{md(\phi\vee\phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|. \end{aligned}$$

3. $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$. By induction, whether $\phi \mapsto_\gamma \phi'$ can be checked in time $O(q)$, where $q = \Sigma_{\phi_1 \in \Phi} \Sigma_{\phi'_1 \in \Phi'} 2^{md(\phi_1 \vee \phi'_1)} \cdot |\phi_1| \cdot |\phi'_1| \cdot |\gamma| \leq 2^{md(\phi\vee\phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|$. □

Definition 19. We say that two ACDFs ϕ and ϕ' are strongly equivalent w.r.t. constraint γ, written $\phi \leftrightharpoons_\gamma \phi'$, if both $\phi \mapsto_\gamma \phi'$ and $\phi' \mapsto_\gamma \phi$.

The following proposition gives a characterization of the concept of strong equivalence. Essentially, two ACDFs are strongly equivalent if they can be made identical by removing redundant formulas from disjunctions and cover operations: for a disjunction $\bigvee \Phi$, an element is redundant if it strongly entails another one; for a cover formula $\nabla_a \Phi$, an element is redundant if it lies between two other elements in the order of strong entailment.

Proposition 18. *Let ϕ and ϕ' be two ACDFs, γ a DNF formula.*

1. *For propositional terms ϕ and ϕ', $\phi \leftrightharpoons_\gamma \phi'$ iff $\phi \wedge \gamma \Leftrightarrow \phi' \wedge \gamma$.*
2. *When $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi'_a$, $\phi \leftrightharpoons_\gamma \phi'$ iff the following hold:*
   (a) *$\phi_0 \leftrightharpoons_\gamma \phi'_0$;*
   (b) *for each $a \in \mathcal{A}$, for each $\phi_a \in \Phi_a$ there exist $\phi'_{a1}, \phi'_{a2} \in \Phi'_a$ and $\phi_{a1}, \phi_{a2} \in \Phi_a$ s.t. $\phi_{a1} \mapsto_\gamma \phi'_{a1} \mapsto_\gamma \phi_a \mapsto_\gamma \phi'_{a2} \mapsto_\gamma \phi_{a2}$.*
   (c) *for each $a \in \mathcal{A}$, for each $\phi'_a \in \Phi'_a$ there exists $\phi_a \in \Phi_a$ s.t. $\phi_a \leftrightharpoons_\gamma \phi'_a$ or there exist $\phi'_{a1}, \phi'_{a2} \in \Phi'_a - \{\phi'_a\}$ s.t. $\phi'_{a1} \mapsto_\gamma \phi'_a \mapsto_\gamma \phi'_{a2}$.*
   (d) *for each $a \in \mathcal{A}$, for each $\phi'_a \in \Phi'_a$ there exist $\phi_{a1}, \phi_{a2} \in \Phi_a$ and $\phi'_{a1}, \phi'_{a2} \in \Phi_a$ s.t. $\phi'_{a1} \mapsto_\gamma \phi_{a1} \mapsto_\gamma \phi'_a \mapsto_\gamma \phi_{a2} \mapsto_\gamma \phi'_{a2}$.*
3. *When $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$, $\phi \leftrightharpoons_\gamma \phi'$ iff for all $\phi_i \in \Phi$ there exist $\phi'_j \in \Phi'$ and $\phi_k \in \Phi$ s.t. $\phi_i \mapsto_\gamma \phi'_j \mapsto_\gamma \phi_k$, and for all $\phi'_j \in \Phi'$ there exist $\phi_i \in \Phi$ and $\phi'_k \in \Phi'$ s.t. $\phi'_j \mapsto_\gamma \phi_i \mapsto_\gamma \phi'_k$.*

Proof. This follows straightforwardly from Definitions 18 and 19. □

For example, $p \wedge q \vee p \leftrightharpoons p$, and $\nabla_a\{p, p \wedge q, p \vee q\} \leftrightharpoons \nabla_a\{p \wedge q, p \vee q\}$.

Proposition 19. *Suppose that $\phi \in \Phi$ and $\phi_1, \phi_2 \in \Phi - \{\phi\}$ s.t. $\phi_1 \mapsto_\gamma \phi \mapsto_\gamma \phi_2$. Then $\nabla_a \Phi \Leftrightarrow_\gamma \nabla_a(\Phi - \{\phi\})$.*

Proof. By Proposition 17, $\phi_1 \models_\gamma \phi \models_\gamma \phi_2$. We have $\nabla_a \Phi = K_a(\bigvee \Phi) \wedge L_a \Phi$. Since $\phi \models_\gamma \phi_2$, ϕ can be removed from Φ in $K_a(\bigvee \Phi)$. Since $\phi_1 \models_\gamma \phi$, ϕ can be removed from Φ in $L_a \Phi$. Hence $\nabla_a \Phi \Leftrightarrow_\gamma \nabla_a(\Phi - \{\phi\})$. □

By Proposition 16,

Proposition 20. *Strong equivalence is an equivalence relation.*

By Proposition 17,

Proposition 21. *Two strongly equivalent ACDFs are equivalent.*

However, two equivalent ACDFs might not be strongly equivalent, as shown in the following example.

Example 5.

1. Let $\phi = p \vee \neg p$, and $\phi' = q \vee \neg q$. Then $\phi \Leftrightarrow \phi'$ but $\phi \not\mapsto \phi'$ and $\phi' \not\mapsto \phi'$.
2. Let $\phi = \nabla_a\{p, q, p \vee q\}$, and $\phi' = \nabla_a\{p, q\}$. Then $\phi \Leftrightarrow \phi'$. However, $\phi \not\mapsto \phi'$, since $p \vee q \not\mapsto p$ and $p \vee q \not\mapsto q$.

4.2. Reasoning

As mentioned earlier, to support efficient reasoning, we represent KBs as ACDFs, and queries as the negation of ACDFs. The following result gives us a recursive algorithm to check if an ACDF entails the negation of another, or to check if the conjunction of two ACDFs is unsatisfiable. The proof is by applying Propositions 1(3) and 7.

Proposition 22. *Let ϕ and ϕ' be two ACDFs, γ a DNF formula.*

1. *If ϕ and ϕ' are propositional terms, then $\phi \models_\gamma \neg\phi'$ iff for each term t in γ, $\phi \wedge t \wedge \phi'$ has complementary literals.*
2. *If $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi'_a$, then $\phi \models_\gamma \neg\phi'$ iff one of the following holds:*
   (a) *$\phi_0 \models_\gamma \neg\phi'_0$;*
   (b) *there exists $a \in \mathcal{A}$ s.t. Φ_a or Φ'_a is empty;*
   (c) *there exist $a \in \mathcal{A}$ and $\phi_i \in \Phi_a$ s.t. for all $\phi'_j \in \Phi'_a$, $\phi_i \models_\gamma \neg\phi'_j$;*
   (d) *there exist $a \in \mathcal{A}$ and $\phi'_j \in \Phi'_a$ s.t. for all $\phi_i \in \Phi_a$, $\phi_i \models_\gamma \neg\phi'_j$.*
3. *If $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$, then $\phi \models_\gamma \neg\phi'$ iff for all $\phi_i \in \Phi$ and $\phi'_j \in \Phi'$, $\phi_i \models_\gamma \neg\phi'_j$.*

Proof. We prove by induction.

1. ϕ and ϕ' are propositional terms. Then $\phi \models_\gamma \neg\phi'$ iff $\phi \wedge \phi' \wedge \gamma$ is unsatisfiable iff for each term t in γ, $\phi \wedge t \wedge \phi'$ is unsatisfiable iff for each term t in γ, $\phi \wedge t \wedge \phi'$ has complementary literals.
2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi'_a$. We have $\phi \wedge \phi' \Leftrightarrow \phi_0 \wedge \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a \wedge \nabla_a \Phi'_a \Leftrightarrow \phi_0 \wedge \phi'_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a[\Phi_a \wedge (\bigvee \Phi'_a) \cup \Phi'_a \wedge (\bigvee \Phi_a)]$, by Proposition 1(3). Then $\phi \models_\gamma \neg\phi'$ iff $\phi \wedge \phi'$ is unsatisfiable w.r.t. γ iff by Proposition 7, one of the following holds:

(a) $\phi_0 \wedge \phi_0' \wedge \gamma$ is propositionally unsatisfiable, *i.e.*, $\phi_0 \models_\gamma \neg\phi_0'$;
(b) there exists $a \in \mathcal{A}$ s.t. Φ_a or Φ_a' is empty;
(c) there exist $a \in \mathcal{A}$ and $\phi \in \Phi_a \wedge (\bigvee \Phi_a') \cup \Phi_a' \wedge (\bigvee \Phi_a)$ s.t. ϕ is unsatisfiable w.r.t. γ, *i.e.*, there exist $a \in \mathcal{A}$ and $\phi \in \Phi_a$ s.t. $\phi \wedge (\bigvee \Phi_a')$ is unsatisfiable w.r.t. γ, or there exist $a \in \mathcal{A}$ and $\phi' \in \Phi_a'$ s.t. $\phi' \wedge (\bigvee \Phi_a)$ is unsatisfiable w.r.t. γ, *i.e.*, there exist $a \in \mathcal{A}$ and $\phi \in \Phi_a$ s.t. for all $\phi' \in \Phi_a'$, $\phi \wedge \phi'$ is unsatisfiable w.r.t. γ, or there exist $a \in \mathcal{A}$ and $\phi' \in \Phi_a'$ s.t. for all $\phi \in \Phi_a$, $\phi \wedge \phi'$ is unsatisfiable w.r.t. γ, *i.e.*, there exist $a \in \mathcal{A}$ and $\phi \in \Phi_a$ s.t. for all $\phi' \in \Phi_a'$, $\phi \models_\gamma \neg\phi'$, or there exist $a \in \mathcal{A}$ and $\phi' \in \Phi_a'$ s.t. for all $\phi \in \Phi_a$, $\phi \models_\gamma \neg\phi'$.

3. $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$. Then $\phi \models_\gamma \neg\phi'$ iff $(\bigvee \Phi) \wedge (\bigvee \Phi')$ is unsatisfiable w.r.t. γ iff for all $\phi \in \Phi$ and $\phi' \in \Phi'$, $\phi \wedge \phi'$ is unsatisfiable w.r.t. γ iff for all $\phi \in \Phi$ and $\phi' \in \Phi'$, $\phi \models_\gamma \neg\phi'$. □

Theorem 4. *Let ϕ and ϕ' be two ACDFs, γ a DNF formula. Whether $\phi \models_\gamma \neg\phi'$ can be checked in time $O(2^{md(\phi\vee\phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$.*

Proof. The proof is the same as that of Proposition 3. □

For Example 3, to check if $\mathcal{I} \models_\gamma pre(left(1))$, by Rule 3, we check if both $\mathcal{I} \models_\gamma \neg at(1, p_1)$ and $\mathcal{I} \models_\gamma \neg\nabla_1\{\top, at(1, p_1)\}$ hold. By Rule 1, the former holds. By Rule 2.a, the latter holds, since $at(1, p_2) \models_\gamma \neg at(1, p_1)$.

4.3. Higher-order belief change

In this section, we present our syntactic higher-order belief change operators, and we give semantic characterizations for our operators for a fragment of ACDFs called proper ACDFs. Intuitively, proper ACDFs only allow negation and disjunction for propositional formulas, *i.e.*, they disallow negative or disjunctive beliefs. A proper ACDF is equivalent to a formula of the form $\bigwedge_{p\in P} K_p\phi_p$, where P is a set of paths of agents, $K_{a_1a_2\ldots a_n}$ abbreviates for $K_{a_1}K_{a_2}\ldots K_{a_n}$, and each ϕ_p is a propositional formula. We show that for proper ACDFs, higher-order belief change nicely reduces to propositional belief change along each path. As a special case, we have $K_i\phi \bullet K_i\mu \Leftrightarrow K_i(\phi \bullet \mu)$, where $\bullet$ is $\circ$ or $\diamond$, ϕ and μ are propositional formulas. Based on the reduction result, we show that semantic characterizations for propositional belief change nicely carry over to higher-order belief change.

We do not yet have general semantic definitions of higher-order belief change operators. On the one hand, these are hard to come up with. As introduced in Section 2.3, Katsuno and Mendelzon's semantic definitions of propositional belief operators rely on the notion of closeness between two models. However, it is more complicated to define the distance between two Kripke models than that of two propositional models. Please refer to Caridroit et al. [11] for a comprehensive study of distances between Kripke models. On the other hand, our work on multi-agent epistemic planning tries to simulate the way humans do epistemic planning, and we think for humans to do higher-order belief change, which is more complicated than propositional belief change, the principle of efficient computability is more important than the principle of minimal change. So the principle of efficient computability guides our definition of higher-order belief change operators. We will leave for future work the hard issue of a general model-theoretic study of higher-order belief change.

The basic idea behind our belief change operators is to reduce the change of epistemic formulas to that of lower-order epistemic formulas, and as a basis we resort to change of propositional formulas. The essential difference between revision and update is: revision satisfies the conjunction property that when $\phi \wedge \phi'$ is satisfiable, $\phi \circ \phi' \Leftrightarrow \phi \wedge \phi'$, while update satisfies the distribution property that when both ϕ_1 and ϕ_2 are satisfiable, $(\phi_1 \vee \phi_2) \diamond \phi' \Leftrightarrow (\phi_1 \diamond \phi' \vee \phi_2 \diamond \phi')$.

To approximate the principle of minimal change, we mimic the definition of *MinPairs* from Section 2.3 as follows.

Definition 20. Let Φ and Φ' be sets of formulas, γ a DNF formula.

1. For $\phi \in \Phi$ and $\phi' \in \Phi'$,
$$dist(\phi, \phi') = \begin{cases} 0 & \text{if } \phi \wedge \phi' \text{ is satisfiable w.r.t. } \gamma \\ 1 & \text{otherwise.} \end{cases}$$
2. $dist(\Phi, \Phi') = \min\{dist(\phi, \phi') \mid \phi \in \Phi, \phi' \in \Phi'\}$.
3. The set of closest pairs of formulas from Φ and Φ' w.r.t. γ, denoted $\Phi *_\gamma \Phi'$, is the set $\{(\phi, \phi') \mid \phi \in \Phi, \phi' \in \Phi', dist(\phi, \phi') = dist(\Phi, \Phi')\}$.

Intuitively, whenever possible, $\Phi *_\gamma \Phi'$ restricts our attention to those consistent pairs of formulas. We let $\Phi \circ \phi'$ denote the set $\{\phi \circ \phi' \mid \phi \in \Phi\}$, and similarly for $\diamond$.

4.3.1. Revision

We first present the formal definition of our revision operator, and then give the intuitive explanation.

Definition 21. Let ϕ and ϕ' be ACDFs, γ a DNF formula. The revision of ϕ with ϕ' under γ, denoted $\phi \circ_\gamma \phi'$, is recursively defined as follows:

1. When ϕ and ϕ' are propositional, the result is $\phi \circ_s (\phi' \wedge \gamma)$, where $\circ_s$ is Satoh's revision operator.
2. When $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \Phi_a$, $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \Phi'_a$, and $\phi \wedge \phi'$ is satisfiable w.r.t. γ, $\phi \circ_\gamma \phi'$ is defined as:
$$(\phi_0 \circ_\gamma \phi'_0) \wedge \bigwedge_{a \in \mathcal{B}-\mathcal{B}'} \nabla_a \Phi_a \wedge \bigwedge_{a \in \mathcal{B}'-\mathcal{B}} \nabla_a \Phi'_a \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a [(\Phi_a \circ_\gamma \bigvee \Phi'_a) \cup (\Phi'_a \circ_\gamma \bigvee \Phi_a)].$$
3. When $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \Phi_a$, $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \Phi'_a$, and $\phi \wedge \phi'$ is unsatisfiable w.r.t. γ, $\phi \circ_\gamma \phi'$ is defined as:
$$(\phi_0 \circ_\gamma \phi'_0) \wedge \bigwedge_{a \in \mathcal{B}-\mathcal{B}'} \nabla_a \Phi_a \wedge \bigwedge_{a \in \mathcal{B}'-\mathcal{B}} \nabla_a \Phi'_a \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a [\Phi^*_a \cup (\Phi'_a - \Phi''_a)],$$
where $\Phi^*_a = \{\phi \circ_\gamma \phi' \mid (\phi, \phi') \in \Phi_a *_\gamma \{\bigvee \Phi'_a\}\}$, $\Phi''_a = \{\phi' \in \Phi'_a \mid$ there exists a $\phi \in \Phi^*_a$ s.t. $\phi \mapsto_\gamma \phi'\}$.
4. $(\bigvee \Phi) \circ_\gamma (\bigvee \Phi') = \bigvee \{\phi \circ_\gamma \phi' \mid (\phi, \phi') \in \Phi *_\gamma \Phi'\}$.

Rule 2 is for the purpose of the conjunction property: recall that by Proposition 1(3), $\nabla_a \Phi \wedge \nabla_a \Phi' \Leftrightarrow \nabla_a[\Phi \wedge (\bigvee \Phi') \cup \Phi' \wedge (\bigvee \Phi)]$. Rule 4 is to approximate the principle of minimal change: when there are consistent pairs of formulas ϕ and ϕ', we ignore those that are not. The intuition behind Rule 3 is as follows. Recall that $\nabla_a \Phi_a \Leftrightarrow K_a(\bigvee \Phi) \wedge L_a \Phi$, hence $\bigvee \Phi_a$ is the belief of agent a, and each $\phi \in \Phi_a$ is a possibility for agent a. To explain the definition of Φ^*_a, when there exist old possibilities that are consistent with the new belief $\bigvee \Phi'_a$, we just keep these possibilities and revise them with the new belief, otherwise we revise each old possibility with the new belief. Also, among all the new possibilities, we remove those which are strongly entailed by an element of Φ^*_a, getting $\Phi'_a - \Phi''_a$. This is because we would like to get a revision result as strong as possible. For example, suppose $\Phi^*_a = \{p \wedge q\}$, and $\Phi'_a = \{p\}$; then we obtain $\nabla_a\{p \wedge q\}$, which is strictly stronger than $\nabla_a\{p \wedge q, p\}$.

Our definition of revision relies on satisfiability checking. To complete the operation efficiently, we present the following recursive algorithm to check for satisfiability and return the revision result at the same time.

Definition 22. Let ϕ and ϕ' be ACDFs, γ a DNF formula. Procedure Sat-rev$_\gamma(\phi, \phi')$ returns a pair (s, ψ), where $s \in \{\top, \bot\}$, which is recursively defined as follows:

1. ϕ and ϕ' are propositional terms. Let s denote whether $\phi \wedge \phi' \wedge \gamma$ is satisfiable, and $\psi = \phi \circ_s (\phi' \wedge \gamma)$.
2. $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \Phi_a$, and $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \Phi'_a$. Let $(s_0, \psi_0) = \text{Sat-rev}_\gamma(\phi_0, \phi'_0)$.
For each $a \in \mathcal{B} \cap \mathcal{B}'$, for each $\phi_{ai} \in \Phi_a$, let $(s_{ai}, \psi_{ai}) = \text{Sat-rev}_\gamma(\phi_{ai}, \bigvee \Phi'_a)$; for each $\phi'_{aj} \in \Phi'_a$, let $(s'_{aj}, \psi'_{aj}) = \text{Sat-rev}_\gamma(\phi'_{aj}, \bigvee \Phi_a)$. If $s_0 = \top$, and for each $a \in \mathcal{B} \cap \mathcal{B}'$, for each $\phi_{ai} \in \Phi_a$ and each $\phi'_{aj} \in \Phi'_a$, $s_{ai} = \top$ and $s'_{aj} = \top$, then let $s = \top$ and
$$\psi = \psi_0 \wedge \bigwedge_{a \in \mathcal{B}-\mathcal{B}'} \nabla_a \Phi_a \wedge \bigwedge_{a \in \mathcal{B}'-\mathcal{B}} \nabla_a \Phi'_a \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a [\{\psi_{ai} \mid \phi_{ai} \in \Phi_a\} \cup \{\psi'_{aj} \mid \phi'_{aj} \in \Phi'_a\}].$$
Otherwise, let $s = \bot$. If there exists $\phi_{ai} \in \Phi_a$ s.t. $s_{ai} = \top$, let $\Phi^*_a = \{\psi_{ai} \mid \phi_{ai} \in \Phi_a, s_{ai} = \top\}$, otherwise let $\Phi^*_a = \{\psi_{ai} \mid \phi_{ai} \in \Phi_a\}$. Let $\Phi''_a = \{\phi' \in \Phi'_a \mid$ there exists a $\phi \in \Phi^*_a$ s.t. $\phi \mapsto_\gamma \phi'\}$. Now let
$$\psi = \psi_0 \wedge \bigwedge_{a \in \mathcal{B}-\mathcal{B}'} \nabla_a \Phi_a \wedge \bigwedge_{a \in \mathcal{B}'-\mathcal{B}} \nabla_a \Phi'_a \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a [\Phi^*_a \cup (\Phi'_a - \Phi''_a)].$$
3. $\phi = \bigvee_{i=1}^m \phi_i$ and $\phi' = \bigvee_{j=1}^n \phi'_j$. For each i and j, let $(s_{ij}, \psi_{ij}) = \text{Sat-rev}_\gamma(\phi_i, \phi'_j)$. If there exist i and j s.t. $s_{ij} = \top$, let $s = \top$ and $\psi = \bigvee\{\psi_{ij} \mid s_{ij} = \top\}$; otherwise, let $s = \bot$ and $\psi = \bigvee \psi_{ij}$.

Theorem 5. *Let ϕ and ϕ' be two ACDFs, γ a DNF formula. Sat-rev$_\gamma(\phi, \phi')$ can be computed in time $O(4^{md(\phi \vee \phi')} \cdot |\phi|^2 \cdot |\phi'|^2 \cdot |\gamma|^2)$ and the resulting formula is of size $O(2^{md(\phi \vee \phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$.*

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. Since ϕ, ϕ' and γ are all in DNF, whether $\phi \wedge \phi' \wedge \gamma$ is satisfiable can be checked in time $O(|\phi| \cdot |\phi'| \cdot |\gamma|)$. By Proposition 10, $\phi \circ_s (\phi' \wedge \gamma)$ can be computed in time $O(|\phi|^2 \cdot |\phi'|^2 \cdot |\gamma|^2)$, and the resulting formula is of size $O(|\phi| \cdot |\phi'| \cdot |\gamma|)$. Hence Sat-rev$_\gamma(\phi, \phi')$ can be computed in time $O(|\phi|^2 \cdot |\phi'|^2 \cdot |\gamma|^2)$.
2. $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \Phi_a$, and $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \Phi'_a$. Suppose that $s = \top$. By induction, it is easy to prove that ψ can be computed in time $O(2 \cdot 4^{md(\phi \vee \phi')-1} \cdot |\phi|^2 \cdot |\phi'|^2 \cdot |\gamma|^2)$, and ψ is of size $O(2^{md(\phi \vee \phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$. Now suppose $s = \bot$. We have each ψ_{ai} is of size $O(2^{md(\phi \vee \phi')-1} \cdot |\phi_{ai}| \cdot |\bigvee \Phi'_a| \cdot |\gamma|)$. By Proposition 3, for $\phi'_{aj} \in \Phi'_a$, whether $\psi_{ai} \mapsto_\gamma \phi'_{aj}$ can be checked in time $O(4^{md(\phi \vee \phi')-1} \cdot |\phi_{ai}| \cdot |\bigvee \Phi'_a| \cdot |\phi'_{aj}| \cdot |\gamma|^2)$. Thus Φ''_a can be computed in time $O(4^{md(\phi \vee \phi')-1} \cdot |\bigvee \Phi_a| \cdot |\bigvee \Phi'_a|^2 \cdot |\gamma|^2)$. So the total time is $O(4^{md(\phi \vee \phi')} \cdot |\phi|^2 \cdot |\phi'|^2 \cdot |\gamma|^2)$. Also, ψ is of size $O(2^{md(\phi \vee \phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$.
3. $\phi = \bigvee_{i=1}^m \phi_i$ and $\phi' = \bigvee_{j=1}^n \phi'_j$. By induction, Sat-rev$_\gamma(\phi_i, \phi'_j)$ can be computed in time $O(4^{md(\phi_i \vee \phi'_j)} \cdot |\phi_i|^2 \cdot |\phi'_j|^2 \cdot |\gamma|^2)$ and the resulting formula is of size $O(2^{md(\phi_i \vee \phi'_j)} \cdot |\phi_i| \cdot |\phi'_j| \cdot |\gamma|)$. Hence Sat-rev$_\gamma(\phi, \phi')$ can be computed in time $O(4^{md(\phi \vee \phi')} \cdot |\phi|^2 \cdot |\phi'|^2 \cdot |\gamma|^2)$ and the resulting formula is of size $O(2^{md(\phi \vee \phi')} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$. □

In the following, we analyze the properties of our revision operator. For an ACDF, we need a notion stronger than that of satisfiability. We say that an ACDF ϕ is disjunct-wise satisfiable if not only ϕ is satisfiable, but also for any disjunction in ϕ, each disjunct is satisfiable. We formally define the notion as follows:

Definition 23. We say that an ACDF ϕ is disjunct-wise satisfiable (d-satisfiable for short) w.r.t. γ if one of the following holds:

1. ϕ is a propositional term which is satisfiable w.r.t. γ.
2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{A}} \nabla_a \Phi_a$, for each $a \in \mathcal{A}$, each $\phi' \in \Phi_a$ is d-satisfiable w.r.t. γ, and ϕ is satisfiable w.r.t. γ.
3. $\phi = \bigvee \Phi$, and each $\phi' \in \Phi$ is d-satisfiable w.r.t. γ.

It is easy to prove by induction that if an ACDF ϕ is d-satisfiable w.r.t. γ, then it is satisfiable w.r.t. γ. The following result states the properties of our revision operator.

Theorem 6. *Let ϕ and ϕ' be ACDFs d-satisfiable w.r.t. γ. Let $\phi^* = \phi \circ_\gamma \phi'$. Then ϕ^* is d-satisfiable w.r.t. γ, and $\phi^* \models_\gamma \phi'$. Moreover, when $\phi \wedge \phi'$ is satisfiable w.r.t. γ, $\phi^* \Leftrightarrow_\gamma \phi \wedge \phi'$.*

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. The properties follow from the definition of Satoh's revision.

2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a \Phi_a$, $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a \Phi'_a$, $\phi \wedge \phi'$ is satisfiable w.r.t. γ. Let $a \in \mathcal{B} \cap \mathcal{B}'$. Since $\phi \wedge \phi'$ is satisfiable w.r.t. γ, by Proposition 5, for each $\phi_a \in \Phi_a$, $\phi_a \wedge \bigvee \Phi'_a$ is satisfiable w.r.t. γ. By induction, $\phi_a \circ_\gamma \bigvee \Phi'_a$ is d-satisfiable w.r.t. γ, and $\phi_a \circ_\gamma \bigvee \Phi'_a \Leftrightarrow_\gamma \phi_a \wedge \bigvee \Phi'_a$. Similarly, for each $\phi'_a \in \Phi'_a$, $\phi'_a \circ_\gamma \bigvee \Phi_a$ is d-satisfiable w.r.t. γ, and $\phi'_a \circ_\gamma \bigvee \Phi_a \Leftrightarrow_\gamma \phi'_a \wedge \bigvee \Phi_a$. By Proposition 1 (3), $\phi \circ_\gamma \phi' \Leftrightarrow_\gamma \phi \wedge \phi'$. It follows that $\phi \circ_\gamma \phi'$ is satisfiable w.r.t. γ and $\phi \circ_\gamma \phi' \models_\gamma \phi'$. Hence $\phi \circ_\gamma \phi'$ is d-satisfiable w.r.t. γ.

3. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a \Phi_a$, $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a \Phi'_a$, $\phi \wedge \phi'$ is unsatisfiable w.r.t. γ. By induction, $\phi_0 \wedge \phi'_0$ is satisfiable w.r.t. γ. Let $a \in \mathcal{B} \cap \mathcal{B}'$. By induction, for each $\phi_a \in \Phi_a$, $\phi_a \circ_\gamma \bigvee \Phi'_a$ is d-satisfiable w.r.t. γ, hence each element of Φ^*_a is d-satisfiable w.r.t. γ. Thus $\phi \circ_\gamma \phi'$ is satisfiable w.r.t. γ. Hence $\phi \circ_\gamma \phi'$ is d-satisfiable w.r.t. γ. By induction, $\phi_0 \circ_\gamma \phi'_0 \models_\gamma \phi'_0$, and for each $\phi^*_a \in \Phi^*_a$, $\phi^*_a \models_\gamma \bigvee \Phi'_a$. Also, by the definition of Φ''_a, for each $\phi'_a \in \Phi'_a$, there exists $\phi^*_a \in \Phi^*_a \cup (\Phi'_a - \Phi''_a)$ s.t. $\phi^*_a \models_\gamma \phi'_a$. By Proposition 6, $\phi \circ_\gamma \phi' \models_\gamma \phi'$.

4. $\phi = \bigvee \Phi$, $\phi' = \bigvee \Phi'$, $\phi \wedge \phi'$ is satisfiable w.r.t. γ. Then $\Phi *_\gamma \Phi' = \{(\phi_1, \phi'_1) \mid \phi_1 \in \Phi_1, \phi'_1 \in \Phi'_1, \phi_1 \wedge \phi'_1$ is satisfiable w.r.t. $\gamma\}$. By induction, for each $(\phi_1, \phi'_1) \in \Phi *_\gamma \Phi'$, $\phi_1 \circ_\gamma \phi'_1$ is d-satisfiable w.r.t. γ, and $\phi_1 \circ_\gamma \phi'_1 \Leftrightarrow_\gamma \phi_1 \wedge \phi'_1$. Thus $\phi \circ_\gamma \phi'$ is d-satisfiable w.r.t. γ, and $\phi \circ_\gamma \phi' = \bigvee\{\phi_1 \circ_\gamma \phi'_1 \mid (\phi_1, \phi'_1) \in \Phi *_\gamma \Phi'\} \Leftrightarrow_\gamma \bigvee\{\phi_1 \wedge \phi'_1 \mid (\phi_1, \phi'_1) \in \Phi *_\gamma \Phi'\} \Leftrightarrow_\gamma \phi \wedge \phi'$. It follows that $\phi \circ_\gamma \phi' \models_\gamma \phi'$.

5. $\phi = \bigvee \Phi$, $\phi' = \bigvee \Phi'$, $\phi \wedge \phi'$ is unsatisfiable w.r.t. γ. Then $\Phi *_\gamma \Phi' = \{(\phi_1, \phi'_1) \mid \phi_1 \in \Phi_1, \phi'_1 \in \Phi'_1\}$. Since both ϕ and ϕ' are d-satisfiable w.r.t. γ, for each $(\phi_1, \phi'_1) \in \Phi *_\gamma \Phi'$, both ϕ_1 and ϕ'_1 are d-satisfiable w.r.t. γ. By induction, $\phi_1 \circ_\gamma \phi'_1$ is d-satisfiable w.r.t. γ, and $\phi_1 \circ_\gamma \phi'_1 \models_\gamma \phi'_1$. Hence $\phi \circ_\gamma \phi'$ is d-satisfiable w.r.t. γ, and $\phi \circ_\gamma \phi' \models \phi'$. □

4.3.2. Update

We now present the formal definition of our update operator, followed by the intuitive explanation.

Definition 24. Let ϕ and ϕ' be ACDFs, γ a DNF formula. The update of ϕ with ϕ' under γ, denoted $\phi \diamond_\gamma \phi'$, is recursively defined as follows:

1. When ϕ and ϕ' are propositional, the result is $\phi \diamond_w (\phi' \wedge \gamma)$, where $\diamond_w$ is Winslett's update operator.
2. When $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a \Phi'_a$, $\phi \diamond_\gamma \phi'$ is defined as follows:
$$(\phi_0 \diamond_\gamma \phi'_0) \wedge \bigwedge_{a\in\mathcal{B}-\mathcal{B}'} \nabla_a \Phi_a \wedge \bigwedge_{a\in\mathcal{B}'-\mathcal{B}} \nabla_a \Phi'_a \wedge \bigwedge_{a\in\mathcal{B}\cap\mathcal{B}'} \nabla_a [\Phi^*_a \cup (\Phi'_a - \Phi''_a)],$$
where $\Phi^*_a = \Phi_a \diamond_\gamma \bigvee \Phi'_a$, $\Phi''_a = \{\phi' \in \Phi'_a \mid$ there exists a $\phi \in \Phi^*_a$ s.t. $\phi \mapsto_\gamma \phi'\}$.
3. $(\bigvee \Phi) \diamond_\gamma \phi' = \bigvee_{\phi\in\Phi} \phi \diamond_\gamma \phi'$.
4. When ϕ is a CDF term, $\phi \diamond_\gamma (\bigvee \Phi') = \bigvee\{\phi \diamond_\gamma \phi' \mid (\phi, \phi') \in \{\phi\} *_\gamma \Phi'\}$.

Rule 3 and the definition of Φ^*_a in Rule 2 are for the purpose of the distribution property. Rule 4 is to approximate the principle of minimal change: when there is $\phi' \in \Phi'$ s.t. ϕ' is consistent with ϕ, we simply ignore those which are not.

Theorem 7. *Let ϕ and ϕ' be two ACDFs, γ a DNF formula. The time to compute $\phi \diamond_\gamma \phi'$ is $O(2^{md(\phi\vee\phi')} \cdot 2^{|\phi'|\cdot|\gamma|} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$, and the resulting formula is of size $O(2^{|\phi'|\cdot|\gamma|} \cdot |\phi|)$.*

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. By Proposition 12, $\phi \diamond_w (\phi' \wedge \gamma)$ can be computed in time $O(2^{|\phi'|\cdot|\gamma|} \cdot |\phi|)$, and the resulting formula is of size $O(2^{|\phi'|\cdot|\gamma|} \cdot |\phi|)$.
2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a \Phi_a$, and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a \Phi'_a$. By induction, for each $\phi_{ai} \in \Phi_a$, $\psi_{ai} = \phi_{ai} \diamond_\gamma \bigvee \Phi'_a$ can be computed in time $O(2^{md(\phi\vee\phi')-1} \cdot 2^{|\bigvee \Phi'_a|\cdot|\gamma|} \cdot |\phi_{ai}| \cdot |\bigvee \Phi'_a| \cdot |\gamma|$, and the resulting formula is of size $O(2^{|\bigvee \Phi'_a|\cdot|\gamma|} \cdot |\phi_{ai}|)$. By Proposition 3, for $\phi'_{aj} \in \Phi'_a$, whether $\psi_{ai} \mapsto_\gamma \phi'_{aj}$ can be checked in time $O(2^{md(\phi\vee\phi')-1} \cdot 2^{|\bigvee \Phi'_a|\cdot|\gamma|} \cdot |\phi_{ai}| \cdot |\phi'_{aj}| \cdot |\gamma|)$. Thus both Φ^*_a and Φ''_a can be computed in time $O(2^{md(\phi\vee\phi')-1} \cdot 2^{|\bigvee \Phi'_a|\cdot|\gamma|} \cdot |\bigvee \Phi_a| \cdot |\bigvee \Phi'_a| \cdot |\gamma|)$. So the total time is $O(2^{md(\phi\vee\phi')} \cdot 2^{|\phi'|\cdot|\gamma|} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$. The resulting formula is of size $O(2^{|\phi'|\cdot|\gamma|} \cdot |\phi|)$.
3. $\phi = \bigvee_i \phi_i$. By induction, $\phi_i \diamond_\gamma \phi'$ can be computed in time $O(2^{md(\phi_i\vee\phi')} \cdot 2^{|\phi'|\cdot|\gamma|} \cdot |\phi_i| \cdot |\phi'| \cdot |\gamma|)$, and the resulting formula is of size $O(2^{|\phi'|\cdot|\gamma|} \cdot |\phi_i|)$. Thus $\phi \diamond_\gamma \phi'$ can be computed in time $O(2^{md(\phi\vee\phi')} \cdot 2^{|\phi'|\cdot|\gamma|} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$, and the resulting formula is of size $O(2^{|\phi'|\cdot|\gamma|} \cdot |\phi|)$.
4. ϕ is a CDF term, and $\phi' = \bigvee \phi'_i$. By Proposition 4, whether ϕ and ϕ'_i are consistent w.r.t. γ can be checked in time $O(2^{md(\phi\vee\phi'_i)} \cdot |\phi| \cdot |\phi'_i| \cdot |\gamma|)$. By induction, $\phi \diamond_\gamma \phi'_i$ can be computed in time $O(2^{md(\phi\vee\phi'_i)} \cdot 2^{|\phi'_i|\cdot|\gamma|} \cdot |\phi| \cdot |\phi'_i| \cdot |\gamma|)$, and the resulting formula is of size $O(2^{|\phi'_i|\cdot|\gamma|} \cdot |\phi|)$. Thus $\phi \diamond_\gamma \phi'$ can be computed in time $O(2^{md(\phi\vee\phi')} \cdot 2^{|\phi'|\cdot|\gamma|} \cdot |\phi| \cdot |\phi'| \cdot |\gamma|)$, and the resulting formula is of size $O(2^{|\phi'|\cdot|\gamma|} \cdot |\phi|)$. $\square$

The complexity of the algorithm is exponential in the size of ϕ', which is an action effect and hence usually small.

The following result states the properties of our update operator.

Theorem 8. *Let ϕ and ϕ' be ACDFs d-satisfiable w.r.t. γ. Let $\phi^* = \phi \diamond_\gamma \phi'$. Then ϕ^* is d-satisfiable w.r.t. γ, and $\phi^* \models_\gamma \phi'$. Moreover, $(\phi_1 \vee \phi_2) \diamond_\gamma \phi' \Leftrightarrow_\gamma (\phi_1 \diamond_\gamma \phi' \vee \phi_2 \diamond_\gamma \phi')$.*

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. The properties follow from the definition of Winslett's update.

2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a \Phi_a$, $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a \Phi'_a$. By induction, $\phi_0 \wedge \phi'_0$ is satisfiable w.r.t. γ. Let $a \in \mathcal{B} \cap \mathcal{B}'$. By induction, for each $\phi_a \in \Phi_a$, $\phi_a \diamond_\gamma \bigvee \Phi'_a$ is d-satisfiable w.r.t. γ, hence each element of Φ^*_a is d-satisfiable w.r.t. γ. Thus $\phi \diamond_\gamma \phi'$ is satisfiable w.r.t. γ. So $\phi \diamond_\gamma \phi'$ is d-satisfiable w.r.t. γ. By induction, $\phi_0 \diamond_\gamma \phi'_0 \models_\gamma \phi'_0$, and for each $\phi^*_a \in \Phi^*_a$, $\phi^*_a \models_\gamma \bigvee \Phi'_a$. Also, by the definition of Φ''_a, for each $\phi'_a \in \Phi'_a$, there exists $\phi^*_a \in \Phi^*_a \cup (\Phi'_a - \Phi''_a)$ s.t. $\phi^*_a \models_\gamma \phi'_a$. By Proposition 6, $\phi \diamond_\gamma \phi' \models_\gamma \phi'$.

3. $\phi = \bigvee \Phi$. Clearly, the disjunction property holds. Since ϕ is d-satisfiable (w.r.t. γ), each $\phi_1 \in \Phi$ is d-satisfiable. By induction, for each $\phi_1 \in \Phi$, $\phi_1 \diamond_\gamma \phi'$ is d-satisfiable and $\phi_1 \diamond_\gamma \phi' \models_\gamma \phi'$. Thus $\phi \diamond_\gamma \phi'$ is d-satisfiable w.r.t. γ and $\phi \diamond_\gamma \phi' \models_\gamma \phi'$.

4. ϕ is a CDF term, $\phi' = \bigvee \Phi'$. Since ϕ' is d-satisfiable (w.r.t. γ), each $\phi'_1 \in \Phi'$ is d-satisfiable. By induction, for each $\phi'_1 \in \Phi'$, $\phi \diamond_\gamma \phi'_1$ is d-satisfiable and $\phi \diamond_\gamma \phi'_1 \models_\gamma \phi'_1$. Thus $\phi \diamond_\gamma \phi'$ is d-satisfiable w.r.t. γ and $\phi \diamond_\gamma \phi' \models_\gamma \phi'$. $\square$

For Example 3, after doing action *left*(1), we get $\mathcal{I} \diamond_\gamma$ *cef*(*eff*$_1$(*left*(1)), equivalent to ϕ_1 under γ:

$\phi_1 = at(1, p_1) \wedge at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2) \wedge$
$\quad \nabla_1\{at(1, p_1) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2)\} \wedge \nabla_2\{at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2)\}$.

Now after doing *find*$(1, b_1, p_1)$ with a positive result, we get $\phi_1 \circ_\gamma$ *pos*(*find*$(1, b_1, p_1)$), equivalent to ϕ_2 under γ:

$\phi_2 = at(1, p_1) \wedge at(2, p_2) \wedge in(b_1, p_1) \wedge \neg in(b_2, p_2) \wedge$
$\quad \nabla_1\{at(1, p_1) \wedge in(b_1, p_1) \wedge \neg in(b_2, p_2)\} \wedge \nabla_2\{at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2)\}$.

4.3.3. Semantic characterizations for proper ACDFs

We now give semantic characterizations for our higher-order belief change operators for a fragment of ACDFs which we call proper ACDFs. Intuitively, proper ACDFs disallow negative or disjunctive beliefs. We define the concept of tree models: a tree model is a special KD45$_n$ model with an underlying tree structure such that each world has a unique a-successor for each agent a. Thus a tree model cannot represent negative or disjunctive beliefs. It is easy to show that a proper ACDF has a model iff it has a tree model. Hence for semantic characterizations for proper ACDFs, we can restrict our attention to tree models. Because tree models have a simple form, we are able to show that for proper ACDFs, the semantic characterizations for propositional belief change nicely carry over to higher-order belief change. For simplicity of the presentation, we ignore the constraint γ.

Definition 25. We say that an ACDF ϕ is proper if in any $\nabla_a \Phi_a$ subformula, Φ_a must be a singleton, and in ϕ disjunction can only be used for propositional formulas.

Proposition 23. $\nabla_a\{\phi\} \Leftrightarrow K_a\phi$.

Proof. $\nabla_a\{\phi\} \Leftrightarrow K_a\phi \wedge L_a\phi \Leftrightarrow K_a\phi \wedge L_a\top \Leftrightarrow K_a\phi$. $\square$

Let p be a path of agents $a_1, a_2, \ldots, a_n$. We use $K_p\phi$ to abbreviate for $K_{a_1} K_{a_2} \ldots K_{a_n}\phi$. We call K_p a path knowledge operator. In case p is the empty path ϵ, $K_p\phi$ simply represents ϕ. We let $L_p\phi$ stand for $\neg K_p \neg\phi$. We call p an alternating path if any adjacent agents on the path are different.

Proposition 24. *Any proper ACDF can be equivalently transformed to formula of the form $\bigwedge_{p \in P} K_p \phi_p$, where P is a set of alternating paths, and each ϕ_p is in DNF. We call such a formula an alternating path knowledge term.*

Proof. Let ϕ be a proper ACDF. We prove by induction on ϕ.

1. ϕ is a propositional term. The claim obviously holds.
2. $\phi = \bigvee \Phi$. By the definition of proper ACDFs, ϕ is a DNF.
3. $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \{\phi_a\}$. By the definition of proper ACDFs, each ϕ_a is a proper ACDF. By induction, each ϕ_a can be equivalently transformed to an alternating path knowledge term $\bigwedge_{p \in P_a} K_p \phi_{ap}$. Thus $\phi \Leftrightarrow \phi_0 \wedge \bigwedge_{a \in \mathcal{B}, p \in P_a} K_{ap} \phi_{ap}$, which is an alternating path knowledge term. □

Without loss of generality, we can assume an alternating path knowledge term $\bigwedge_{p \in P} K_p \phi_p$ of modal depth k takes the form of $\bigwedge_p K_p \phi_p$, where p ranges over all alternating paths of length $\leq k$, since for $p \notin P$, we can let ϕ_p be $\top$.

In the following, we show that for proper ACDFs, higher-order belief change nicely reduces to propositional belief change along each path. For each operator, we first prove a proposition which is used to prove the reduction result.

Proposition 25. *Let ϕ and ϕ' be proper ACDFs. We have $\phi \circ \phi' \mapsto \phi'$, and*

1. *When ϕ and ϕ' are propositional, we have $\phi \circ \phi' = \phi \circ_s \phi'$, and if $\phi \wedge \phi'$ is satisfiable, then $\phi \circ \phi' \Leftrightarrow \phi \wedge \phi'$, and $\phi \circ \phi' \mapsto \phi$.*
2. *When $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \{\phi_a\}$, $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \{\phi'_a\}$, and $\phi \wedge \phi'$ is satisfiable, we have $\phi \circ \phi' \Leftrightarrow \phi \wedge \phi'$, and $\phi \circ \phi' \mapsto \phi$.*
3. *When $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \{\phi_a\}$, $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \{\phi'_a\}$, and $\phi \wedge \phi'$ is unsatisfiable, we have*

$$\phi \circ \phi' = (\phi_0 \circ \phi'_0) \wedge \bigwedge_{a \in \mathcal{B} - \mathcal{B}'} \nabla_a \{\phi_a\} \wedge \bigwedge_{a \in \mathcal{B}' - \mathcal{B}} \nabla_a \{\phi'_a\} \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a \{\phi_a \circ \phi'_a\}.$$

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. By definition, $\phi \circ \phi'$ is $\phi \circ_s \phi'$. By Proposition 9,

$$\phi \circ_s \phi' = \bigvee_{\langle \phi_i, \phi'_j \rangle \in MinPairs(\phi, \phi')} revise(\phi_i, \phi'_j).$$

Recall that for a DNF ϕ, we use ϕ with subscripts to denote disjuncts of ϕ. Also recall that $revise(\phi_i, \phi'_j)$ is the result of revising term ϕ_i by term ϕ'_j. Thus for each term t of $\phi \circ_s \phi'$, $t \models \phi'_j$ for some j. By definition of strong entailment, $\phi \circ_s \phi' \mapsto \phi'$.

When $\phi \wedge \phi'$ is satisfiable, we have

$$\phi \circ_s \phi' = \bigvee_{\phi_i \in \phi, \phi'_j \in \phi', \phi_i \wedge \phi'_j \text{ is satisfiable}} \phi_i \wedge \phi'_j.$$

Thus $\phi \circ \phi' \Leftrightarrow \phi \wedge \phi'$, and we also have $\phi \circ \phi' \mapsto \phi$.

2. $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \{\phi_a\}$, $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \{\phi'_a\}$, and $\phi \wedge \phi'$ is satisfiable. By definition,

$$\phi \circ \phi' = (\phi_0 \circ \phi'_0) \wedge \bigwedge_{a \in \mathcal{B} - \mathcal{B}'} \nabla_a \{\phi_a\} \wedge \bigwedge_{a \in \mathcal{B}' - \mathcal{B}} \nabla_a \{\phi'_a\} \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a \{\phi_a \circ \phi'_a, \phi'_a \circ \phi_a\}.$$

Since $\phi \wedge \phi'$ is satisfiable, by Proposition 5, $\phi_0 \wedge \phi'_0$ is satisfiable, and for each $a \in \mathcal{B} \cap \mathcal{B}'$, $\phi_a \wedge \phi'_a$ is satisfiable. By induction, $\phi_0 \circ \phi'_0$ strongly entails both ϕ_0 and ϕ'_0, both $\phi_a \circ \phi'_a$ and $\phi'_a \circ \phi_a$ strongly entail both ϕ_a and ϕ'_a. Thus by definition of strong entailment, $\phi \circ \phi' \mapsto \phi'$ and $\phi \circ \phi' \mapsto \phi$. Also, by induction, $\phi_0 \circ \phi'_0 \Leftrightarrow \phi_0 \wedge \phi'_0$, $\phi_a \circ \phi'_a \Leftrightarrow \phi_a \wedge \phi'_a$, and $\phi'_a \circ \phi_a \Leftrightarrow \phi_a \wedge \phi'_a$. Thus

$$\phi \circ \phi' \Leftrightarrow (\phi_0 \wedge \phi'_0) \wedge \bigwedge_{a \in \mathcal{B} - \mathcal{B}'} \nabla_a \{\phi_a\} \wedge \bigwedge_{a \in \mathcal{B}' - \mathcal{B}} \nabla_a \{\phi'_a\} \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a \{\phi_a \wedge \phi'_a\}.$$

So $\phi \circ \phi' \Leftrightarrow \phi \wedge \phi'$.

3. $\phi = \phi_0 \wedge \bigwedge_{a \in \mathcal{B}} \nabla_a \{\phi_a\}$, $\phi' = \phi'_0 \wedge \bigwedge_{a \in \mathcal{B}'} \nabla_a \{\phi'_a\}$, and $\phi \wedge \phi'$ is unsatisfiable. By induction, $\phi_0 \circ \phi'_0 \mapsto \phi'_0$, and for each $a \in \mathcal{B} \cap \mathcal{B}'$, $\phi_a \circ \phi'_a \mapsto \phi'_a$. Thus by definition,

$$\phi \circ \phi' = (\phi_0 \circ \phi'_0) \wedge \bigwedge_{a \in \mathcal{B} - \mathcal{B}'} \nabla_a \{\phi_a\} \wedge \bigwedge_{a \in \mathcal{B}' - \mathcal{B}} \nabla_a \{\phi'_a\} \wedge \bigwedge_{a \in \mathcal{B} \cap \mathcal{B}'} \nabla_a \{\phi_a \circ \phi'_a\}.$$

So by definition of strong entailment, $\phi \circ \phi' \mapsto \phi'$. □

Note that in Section 3.2, we stated that we would like to have $K_i \phi \circ K_i \mu \Leftrightarrow K_i(\phi \circ \mu)$, where ϕ and μ are propositional formulas. Below, we prove a general version of this claim.

Theorem 9. *Let $\phi \Leftrightarrow \bigwedge_{p\in P} K_p\phi_p$ and $\phi' \Leftrightarrow \bigwedge_{p\in P'} K_p\phi'_p$ be proper ACDFs. Then*

$$\phi \circ \phi' \Leftrightarrow \bigwedge_{p\in P-P'} K_p\phi_p \wedge \bigwedge_{p\in P'-P} K_p\phi'_p \wedge \bigwedge_{p\in P\cap P'} K_p[\phi_p \circ_s \phi'_p].$$

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. The claim obviously holds.
2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\{\phi_a\}$, $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a\{\phi'_a\}$, and $\phi \wedge \phi'$ is satisfiable. By Proposition 5, for each $p \in P \cap P'$, $\phi_p \wedge \phi'_p$ is satisfiable. Thus by Proposition 25,

$$\begin{aligned}\phi \circ \phi' \Leftrightarrow \phi \wedge \phi' &\Leftrightarrow \bigwedge_{p\in P-P'} K_p\phi_p \wedge \bigwedge_{p\in P'-P} K_p\phi'_p \wedge \bigwedge_{p\in P\cap P'} K_p[\phi_p \wedge \phi'_p] \\ &\Leftrightarrow \bigwedge_{p\in P-P'} K_p\phi_p \wedge \bigwedge_{p\in P'-P} K_p\phi'_p \wedge \bigwedge_{p\in P\cap P'} K_p[\phi_p \circ_s \phi'_p]\end{aligned}$$

3. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\{\phi_a\}$, $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a\{\phi'_a\}$, and $\phi \wedge \phi'$ is unsatisfiable. By Proposition 25,

$$\phi \circ \phi' = (\phi_0 \circ \phi'_0) \wedge \bigwedge_{a\in\mathcal{B}-\mathcal{B}'} K_a\phi_a \wedge \bigwedge_{a\in\mathcal{B}'-\mathcal{B}} K_a\phi'_a \wedge \bigwedge_{a\in\mathcal{B}\cap\mathcal{B}'} K_a[\phi_a \circ \phi'_a].$$

By induction, for each $a \in \mathcal{B} \cap \mathcal{B}'$,

$$\phi_a \circ \phi'_a \Leftrightarrow \bigwedge_{p\in P_a-P'_a} K_p\phi_{ap} \wedge \bigwedge_{p\in P'_a-P_a} K_p\phi'_{ap} \wedge \bigwedge_{p\in P_a\cap P'_a} K_p[\phi_{ap} \circ_s \phi'_{ap}].$$

Thus

$$\begin{aligned}\phi \circ \phi' \Leftrightarrow\ &(\phi_0 \circ \phi'_0) \wedge \bigwedge_{a\in\mathcal{B}-\mathcal{B}'} K_a\phi_a \wedge \bigwedge_{a\in\mathcal{B}'-\mathcal{B}} K_a\phi'_a \wedge \\ &\bigwedge_{a\in\mathcal{B}\cap\mathcal{B}', p\in P_a-P'_a} K_{ap}\phi_{ap} \wedge \bigwedge_{a\in\mathcal{B}\cap\mathcal{B}', p\in P'_a-P_a} K_{ap}\phi'_{ap} \wedge \bigwedge_{a\in\mathcal{B}\cap\mathcal{B}', p\in P_a\cap P'_a} K_{ap}[\phi_{ap} \circ_s \phi'_{ap}] \\ \Leftrightarrow\ &\bigwedge_{p\in P-P'} K_p\phi_p \wedge \bigwedge_{p\in P'-P} K_p\phi'_p \wedge \bigwedge_{p\in P\cap P'} K_p[\phi_p \circ_s \phi'_p] \quad \square\end{aligned}$$

Proposition 26. *Let ϕ and ϕ' be proper ACDFs. We have $\phi \diamond \phi' \mapsto \phi'$, and*

1. *When ϕ and ϕ' are propositional, $\phi \diamond \phi'$ is $\phi \diamond_w \phi'$.*
2. *When $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\{\phi_a\}$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a\{\phi'_a\}$, $\phi \diamond \phi'$ is $(\phi_0 \diamond \phi'_0) \wedge \bigwedge_{a\in\mathcal{B}-\mathcal{B}'} \nabla_a\{\phi_a\} \wedge \bigwedge_{a\in\mathcal{B}'-\mathcal{B}} \nabla_a\{\phi'_a\} \wedge \bigwedge_{a\in\mathcal{B}\cap\mathcal{B}'} \nabla_a\{\phi_a \diamond \phi'_a\}$.*

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. By definition, $\phi \diamond \phi'$ is $\phi \diamond_w \phi'$. By Proposition 11,

$$\phi \diamond_w \phi' = \bigvee_{\phi'_j\in\phi', \phi_i\in\phi} revise(\phi_i, \phi'_j) \wedge patch_{\phi_i}(\phi'_j).$$

Thus for each term t of $\phi \diamond_w \phi'$, $t \models \phi'_j$ for some j. By definition, $\phi \diamond_w \phi' \mapsto \phi'$.

2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\{\phi_a\}$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a\{\phi'_a\}$. By induction, $\phi_0 \diamond \phi'_0 \mapsto \phi'_0$, and for each $a \in \mathcal{B} \cap \mathcal{B}'$, $\phi_a \diamond \phi'_a \mapsto \phi'_a$. Thus by definition,

$$\phi \diamond \phi' = (\phi_0 \diamond \phi'_0) \wedge \bigwedge_{a\in\mathcal{B}-\mathcal{B}'} \nabla_a\{\phi_a\} \wedge \bigwedge_{a\in\mathcal{B}'-\mathcal{B}} \nabla_a\{\phi'_a\} \wedge \bigwedge_{a\in\mathcal{B}\cap\mathcal{B}'} \nabla_a\{\phi_a \diamond \phi'_a\}.$$

So by definition of strong entailment, $\phi \diamond \phi' \mapsto \phi'$. $\square$

In the following, we prove a general version of the claim $K_i\phi \diamond K_i\mu \Leftrightarrow K_i(\phi \diamond \mu)$, where ϕ and μ are propositional formulas.

Theorem 10. *Let $\phi \Leftrightarrow \bigwedge_{p\in P} K_p\phi_p$ and $\phi' \Leftrightarrow \bigwedge_{p\in P'} K_p\phi'_p$ be proper ACDFs. Then*

$$\phi \diamond \phi' \Leftrightarrow \bigwedge_{p\in P-P'} K_p\phi_p \wedge \bigwedge_{p\in P'-P} K_p\phi'_p \wedge \bigwedge_{p\in P\cap P'} K_p[\phi_p \diamond_w \phi'_p].$$

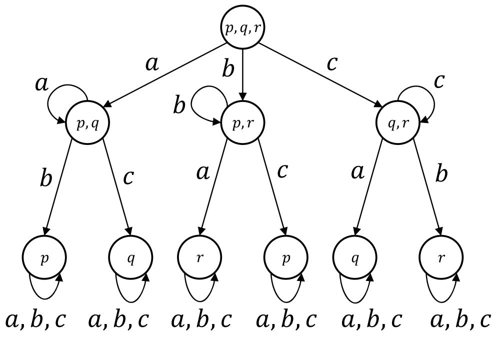

Fig. 4. An example tree model.

Proof. We prove by induction on $|\phi| + |\phi'|$.

1. ϕ and ϕ' are propositional. The claim obviously holds.
2. $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a\{\phi_a\}$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a\{\phi'_a\}$. By Proposition 26,

$$\phi \diamond \phi' = (\phi_0 \diamond \phi'_0) \wedge \bigwedge{}_{a\in\mathcal{B}-\mathcal{B}'} K_a\phi_a \wedge \bigwedge{}_{a\in\mathcal{B}'-\mathcal{B}} K_a\phi'_a \wedge \bigwedge{}_{a\in\mathcal{B}\cap\mathcal{B}'} K_a[\phi_a \diamond \phi'_a].$$

By induction, for each $a \in \mathcal{B} \cap \mathcal{B}'$,

$$\phi_a \diamond \phi'_a \Leftrightarrow \bigwedge_{p\in P_a-P'_a} K_p\phi_{ap} \wedge \bigwedge_{p\in P'_a-P_a} K_p\phi'_{ap} \wedge \bigwedge_{p\in P_a\cap P'_a} K_p[\phi_{ap} \diamond_w \phi'_{ap}].$$

Thus

$$\begin{aligned}\phi \diamond \phi' \Leftrightarrow\ & (\phi_0 \diamond \phi'_0) \wedge \bigwedge{}_{a\in\mathcal{B}-\mathcal{B}'} K_a\phi_a \wedge \bigwedge{}_{a\in\mathcal{B}'-\mathcal{B}} K_a\phi'_a \wedge \\ & \bigwedge{}_{a\in\mathcal{B}\cap\mathcal{B}', p\in P_a-P'_a} K_{ap}\phi_{ap} \wedge \bigwedge{}_{a\in\mathcal{B}\cap\mathcal{B}', p\in P'_a-P_a} K_{ap}\phi'_{ap} \wedge \bigwedge{}_{a\in\mathcal{B}\cap\mathcal{B}', p\in P_a\cap P'_a} K_{ap}[\phi_{ap} \diamond_w \phi'_{ap}] \\ \Leftrightarrow\ & \bigwedge_{p\in P-P'} K_p\phi_p \wedge \bigwedge_{p\in P'-P} K_p\phi'_p \wedge \bigwedge_{p\in P\cap P'} K_p[\phi_p \diamond_w \phi'_p] \quad \square\end{aligned}$$

We now show that for proper ACDFs, the semantic characterization for propositional belief change operators can nicely carry over to higher-order belief change operators. For this purpose, we consider specially designed KD45$_n$ models with an underlying tree structure such that each world has a unique a-successor for each agent a. We call such models tree models.

Definition 26. A tree model of depth $k \geq 0$ is a pointed Kripke model $t = (M, w)$ such that

1. After removal of loops at worlds, the underlying graph of t is a tree of depth k rooted at w.
2. w has a unique a-child for each agent $a \in \mathcal{A}$.
3. For each world v of t, if it is at level $j < k$ and it is the a-child of its parent, then there is an a-loop at v, and v has a unique b-child for each agent $b \neq a$.
4. For each leaf v of t, there is an a-loop at v for each agent $a \in \mathcal{A}$.

For example, Fig. 4 shows a tree model of depth 2 where there are three agents a, b, c and three atoms p, q, r.

A tree model has the following properties:

Proposition 27. *A tree model is a KD45$_n$ model such that each world has a unique a-successor for each agent a, and each world at level $j \geq 0$ is reachable from the root via a unique alternating path of length j.*

Proof. Let $a \in \mathcal{A}$. By Definition 26, obviously, each world has a unique a-successor. Now let wR_au and wR_av. Then we must have $u = v$. If $w = u$, uR_av. If $w \neq u$, by Condition 3 of Definition 26, uR_au, i.e., uR_av. Hence R_a is Euclidean. Now suppose wR_au and uR_av. By Definition 26, $w = u$ or $u = v$, hence wR_av. So R_a is transitive.

Finally, we prove by induction on j that each world at level $j \geq 0$ is reachable from the root via a unique alternating path of length j. Basis: $j = 0$. The path is the empty path. Induction: $j > 0$. Suppose that w is the a-child of some world v at level $j - 1$ for some agent a. By induction, v is reachable from the root by a unique alternating path p of length $j - 1$. By Condition 3, a must be different from the last agent of p. Thus w is reachable from the root via the unique alternating path pa of length j. $\square$

Thus we have the following properties of tree models:

Proposition 28. *Let $t = (M, w)$ be a tree model. Let p be a path of agents, $\phi, \psi \in \mathcal{L}_K$. Then we have $t \models K_p\phi \Leftrightarrow L_p\phi$; $t \models K_p(\phi \vee \psi) \Leftrightarrow K_p\phi \vee K_p\psi$.*

Proof. By Proposition 27, let v be the unique world of t reachable from w via p. Then we have $t \models K_p\phi$ iff $M, v \models \phi$ iff $t \models L_p\phi$; $t \models K_p(\phi \vee \psi)$ iff $M, v \models \phi \vee \psi$ iff $M, v \models \phi$ or $M, v \models \psi$ iff $t \models K_p\phi$ or $t \models K_p\psi$ iff $t \models K_p\phi \vee K_p\psi$. □

The above properties say that tree models cannot represent negative beliefs in the form of $L_p\phi \wedge \neg K_p\phi$ or disjunctive beliefs in the form of $K_p(\phi \vee \psi) \wedge \neg(K_p\phi \vee K_p\psi)$. Since proper ACDFs disallow negative or disjunctive beliefs, we will be able to easily show that if a proper ACDF has a model, then it has a tree model.

By Proposition 27, for a tree model, each world at level $j \geq 0$ is reachable from the root via a unique alternating path of length j. Thus a tree model of depth k can be represented as a tuple of valuations $\vec{V}$ where for each alternating path p of length $\leq k$, there is a valuation V_p.

Proposition 29. *Let $\phi = \bigwedge_{p \in P} K_p\phi_p$ be an alternating path knowledge term of modal depth k. Let $\vec{V}$ be a tree model of depth $\geq k$. Then $\vec{V} \models \phi$ iff for each $p \in P$, $V_p \models \phi_p$.*

Proof. In the tree model $\vec{V}$, for each path p, the root has a unique p-descendant, and its associated valuation is V_p. □

For example, the tree model in Fig. 4 is a model of

$$p \wedge q \wedge r \wedge K_c(p \wedge q \vee p \wedge r \vee q \wedge r) \wedge K_{ab}(p \vee q \vee r) \wedge K_{bc}(p \vee q \vee r).$$

Theorem 11. *Let $\phi = \bigwedge_{p \in P} K_p\phi_p$ be an alternating path knowledge term. Then ϕ is satisfiable iff it has a tree model.*

Proof. We only need to prove the only-if direction. By Proposition 5, an alternating modal term $\phi_0 \wedge \bigwedge_{a \in \mathcal{B}} K_a\phi_a$ is satisfiable iff ϕ_0 is propositionally satisfiable and for each $a \in \mathcal{B}$, ϕ_a is satisfiable. Now suppose that ϕ is satisfiable. By iteratively applying the above result, ϕ_p is satisfiable for each $p \in P$. Let $V_p \models \phi_p$ for each $p \in P$. We construct a tree model $\vec{V}$ from these V_p's. By Proposition 29, $\vec{V} \models \phi$. □

Because of the above property, for a proper ACDF, we can treat its tree models as its canonical models. Hence for semantic characterizations for proper ACDFs, we can restrict our attention to tree models.

Let ϕ be a DNF. We use $\text{Mod}(\phi)$ to denote the set of valuations that satisfy ϕ. Let ϕ be a proper ACDF of depth $\leq k$. We use $\text{TMod}_k(\phi)$ to denote the set of tree models of depth k which are models of ϕ.

In the following, we define order relations between tree models. Since a tree model can be represented as a tuple of valuations, we can easily define order relations between tree models as bitwise order relations between valuations.

Let $\leq$ be a binary relation on a set U. We say that $\leq$ is a partial preorder if it is reflexive and transitive. Let $\leq$ be a partial preorder on U. We write $x < y$ if $x \leq y$ but not $y \leq x$. Let $S \subseteq U$. We use $\text{Min}(S, \leq)$ for the set of elements of S minimal under $\leq$.

Definition 27.

1. Let V and V' be two valuations. The difference of V and V', written $\text{Diff}(V, V')$, is defined as $(V - V') \cup (V' - V)$.
2. A difference tree of depth k is a tuple $\vec{D}$ where for each alternating path p of length $\leq k$, D_p is a subset of $\mathcal{P}$.
3. Let $\vec{V}$ and $\vec{V}'$ be two tree models of depth k. We define $\text{Diff}(\vec{V}, \vec{V}')$ as $\vec{D}$ where for each alternating path p of length $\leq k$, $D_p = \text{Diff}(V_p, V'_p)$.

Definition 28.

1. Let V, V_1, V_2 be valuations. We define $V_1 \leq_V V_2$ if $\text{Diff}(V_1, V) \subseteq \text{Diff}(V_2, V)$.
2. Let $\vec{D}$ and $\vec{D}'$ be two different trees of depth k. We write $\vec{D} \leq_b \vec{D}'$ if for each path p, $D_p \subseteq D'_p$.
3. Let $\vec{V}, \vec{V}_1, \vec{V}_2$ be tree models of depth k. We define $\vec{V}_1 \leq_{\vec{V}} \vec{V}_2$ if $\text{Diff}(\vec{V}_1, \vec{V}) \leq_b \text{Diff}(\vec{V}_2, \vec{V})$.

The following Proposition is easy to prove.

Proposition 30. *$\leq_V$, $\leq_b$, and $\leq_{\vec{V}}$ are all partial preorders.*

Definition 29.

1. Let ϕ_0 and ϕ'_0 be two DNFs. We define $\mathrm{Diff}(\phi_0, \phi'_0) = \{\mathrm{Diff}(V, V') \mid V \in \mathrm{Mod}(\phi_0), V' \in \mathrm{Mod}(\phi'_0)\}$, and let $\mathrm{MinDiff}(\phi_0, \phi'_0) = \mathrm{Min}(\mathrm{Diff}(\phi_0, \phi'_0), \subseteq)$.
2. Let ϕ and ϕ' be two proper ACDFs of modal depth $\leq k$. We define $\mathrm{Diff}_k(\phi, \phi') = \{\mathrm{Diff}(\vec{V}, \vec{V}') \mid \vec{V} \in \mathrm{TMod}_k(\phi), \vec{V}' \in \mathrm{TMod}_k(\phi')\}$, and let $\mathrm{MinDiff}_k(\phi, \phi') = \mathrm{Min}(\mathrm{Diff}_k(\phi, \phi'), \leq_b)$.

The following proposition shows that the set of minimal differences of two proper ACDFs is the Cartesian product of the minimal differences of path formulas.

Let $\vec{V}$ be a tree model of depth k. Let p be an alternating path of length $\leq k$, and let V' be a valuation. We use $\vec{V}[V'/V_p]$ to denote the tree model which is the same as $\vec{V}$ except that the valuation associated with p is V'.

Proposition 31. *Let $\phi \Leftrightarrow \bigwedge_p K_p\phi_p$, and $\phi' \Leftrightarrow \bigwedge_p K_p\phi'_p$, where p ranges over all alternating paths of length $\leq k$. Let $\vec{V} \in \mathrm{TMod}_k(\phi)$ and $\vec{V}' \in \mathrm{TMod}_k(\phi')$. Then $\mathrm{Diff}(\vec{V}, \vec{V}') \in \mathrm{MinDiff}_k(\phi, \phi')$ iff for each p, $\mathrm{Diff}(V_p, V'_p) \in \mathrm{MinDiff}(\phi_p, \phi'_p)$.*

Proof. We prove $\mathrm{Diff}(\vec{V}, \vec{V}') \notin \mathrm{MinDiff}(\phi, \phi')$ iff there exists p s.t. $\mathrm{Diff}(V_p, V'_p) \notin \mathrm{MinDiff}(\phi_p, \phi'_p)$. $\Rightarrow$: Suppose that there exist $\vec{V}_1 \in \mathrm{TMod}_k(\phi)$ and $\vec{V}_2 \in \mathrm{TMod}_k(\phi')$ s.t. $\mathrm{Diff}(\vec{V}_1, \vec{V}_2) <_b \mathrm{Diff}(\vec{V}, \vec{V}')$. Then there exists p s.t. $\mathrm{Diff}(V_{1p}, V_{2p}) \subset \mathrm{Diff}(V_p, V'_p)$. Thus $\mathrm{Diff}(V_p, V'_p) \notin \mathrm{MinDiff}(\phi_p, \phi'_p)$. $\Leftarrow$: Suppose that there is p and there exist $V_{1p} \models \phi_p$ and $V_{2p} \models \phi'_p$ s.t. $\mathrm{Diff}(V_{1p}, V_{2p}) \subset \mathrm{Diff}(V_p, V'_p)$. Now let $\vec{V}_1 = \vec{V}[V_{1p}/V_p]$ and $\vec{V}_2 = \vec{V}[V_{2p}/V_p]$. Then $\mathrm{Diff}(\vec{V}_1, \vec{V}_2) <_b \mathrm{Diff}(\vec{V}, \vec{V}')$. By Proposition 29, $\vec{V}_1 \models \phi$ and $\vec{V}_2 \models \phi'$. Thus $\mathrm{Diff}(\vec{V}, \vec{V}') \notin \mathrm{MinDiff}(\phi, \phi')$. $\square$

The following is a semantic characterization for our higher-order revision operator. Like propositional revision, $\phi \circ \phi'$ selects from the tree models of ϕ' those that are closest to tree models of ψ.

Theorem 12. *Let ϕ and ϕ' be proper ACDFs of modal depth $\leq k$. Then $\mathrm{TMod}_k(\phi \circ \phi') =$*

$$\{\vec{V}' \in \mathrm{TMod}_k(\phi') \mid \exists \vec{V} \in \mathrm{TMod}_k(\phi) \text{ s.t. } \mathrm{Diff}(\vec{V}, \vec{V}') \in \mathrm{MinDiff}_k(\phi, \phi')\}.$$

Proof. Let $\phi \Leftrightarrow \bigwedge_p K_p\phi_p$, and $\phi' \Leftrightarrow \bigwedge_p K_p\phi'_p$, where p ranges over all alternating paths of length $\leq k$. By Theorem 9, $\phi \circ \phi' \Leftrightarrow \bigwedge_p K_p[\phi_p \circ_s \phi'_p]$. Thus $\vec{V}' \in \mathrm{TMod}_k(\phi \circ \phi')$ iff for each path p, $V'_p \models \phi_p \circ_s \phi'_p$ iff (by the definition of $\circ_s$) for each p, $V'_p \models \phi'_p$ and there exists $V_p \models \phi_p$ s.t. $\mathrm{Diff}(V_p, V'_p) \in \mathrm{MinDiff}(\phi_p, \phi'_p)$ iff (by Proposition 31) $\vec{V}' \models \phi'$ and there exists $\vec{V} \models \phi$ s.t. $\mathrm{Diff}(\vec{V}, \vec{V}') \in \mathrm{MinDiff}_k(\phi, \phi')$. $\square$

The following proposition shows that the set of minimal elements under $\leq_{\vec{V}}$ is the Cartesian product of the set of minimal elements under $\leq_{V_p}$ for each path p.

Proposition 32. *Let $\phi \Leftrightarrow \bigwedge_p K_p\phi_p$, where p ranges over all alternating paths of length $\leq k$. Let $\vec{V}$ and $\vec{V}'$ be tree models of depth k. Then $\vec{V}' \in \mathrm{Min}(\mathrm{TMod}_k(\phi), \leq_{\vec{V}})$ iff for each path p, $V'_p \in \mathrm{Min}(\mathrm{Mod}(\phi_p), \leq_{V_p})$.*

Proof. We prove that $\vec{V}' \notin \mathrm{Min}(\mathrm{TMod}_k(\phi), \leq_{\vec{V}})$ iff there is p s.t. $V'_p \notin \mathrm{Min}(\mathrm{Mod}(\phi_p), \leq_{V_p})$. $\Rightarrow$: Suppose there is $\vec{V}'' \in \mathrm{TMod}_k(\phi)$ s.t. $\vec{V}'' <_{\vec{V}} \vec{V}'$. Then there exists path p s.t. $V''_p <_{V_p} V'_p$. Thus $V'_p \notin \mathrm{Min}(\mathrm{Mod}(\phi_p), \leq_{V_p})$. $\Leftarrow$: Suppose that there exist p and $V''_p \models \phi_p$ s.t. $V''_p <_{V_p} V'_p$. Let $\vec{V}'' = \vec{V}'[V''_p/V'_p]$. Then $\vec{V}'' \models \phi$ (by Proposition 29) and $\vec{V}'' <_{\vec{V}} \vec{V}'$. Thus $\vec{V}' \notin \mathrm{Min}(\mathrm{TMod}_k(\phi), \leq_{\vec{V}})$. $\square$

The following is a semantic characterization for our higher-order update operator. Like propositional update, $\phi \diamond \phi'$ selects, for each tree model $\vec{V}$ of ϕ, the set of tree models of ϕ' that are closest to $\vec{V}$.

Theorem 13. *Let ϕ and ϕ' be proper ACDFs of modal depth $\leq k$. Then we have*

$$\mathrm{TMod}_k(\phi \diamond \phi') = \bigcup_{\vec{V} \in \mathrm{TMod}_k(\phi)} \mathrm{Min}(\mathrm{TMod}_k(\phi'), \leq_{\vec{V}}).$$

Proof. Let $\phi \Leftrightarrow \bigwedge_p K_p\phi_p$, and $\phi' \Leftrightarrow \bigwedge_p K_p\phi'_p$, where p ranges over all alternating paths of length $\leq k$. By Theorem 10, $\phi \diamond \phi' \Leftrightarrow \bigwedge_p K_p[\phi_p \diamond_w \phi'_p]$. Thus $\vec{V}' \in \mathrm{TMod}_k(\phi \diamond \phi')$ iff for each path p, $V'_p \models \phi_p \diamond_w \phi'_p$ iff (by the definition of $\diamond_w$) for each p, there is $V_p \in \mathrm{Mod}(\phi_p)$ s.t. $V'_p \in \mathrm{Min}(\mathrm{Mod}(\phi'_p), \leq_{V_p})$ iff (by Proposition 32) there is $\vec{V} \in \mathrm{TMod}_k(\phi)$ s.t. $\vec{V}' \in \mathrm{Min}(\mathrm{TMod}_k(\phi'), \leq_{\vec{V}})$. $\square$

5. Implementation and experimentation

Based on the theoretic work, we have developed EPDDL – an epistemic extension of PDDL [35], to describe multi-agent epistemic planning problems, and implemented a multi-agent epistemic planner MEPK.[1]

In this section, we introduce EPDDL, describe the overall architecture of MEPK, and give our experimental evaluation of MEPK.

5.1. EPDDL

An MEP problem specified in EPDDL consists of two parts: a domain file for specifying types, predicates and actions, and a problem file for specifying objects, agents, the initial KB and goal.

Below is the domain file for Example 3, where the detail for action *right* is omitted.

```
(define (domain ctc2)
  (:types agent room box)
  (:predicates (at ?ag - agent ?p - room) (in ?b - block ?p - room))

  (:action left
   :category (ontic)
   :parameters (?i - agent)
   :precondition (and (not(at ?i p1)) (K_?i (not(at ?i p1))))
   :effect (<{at ?i p2} {and (at ?i p1) (K_?i (at ?i p1))}>
            <{at ?i p3} {and (at ?i p2) (K_?i (at ?i p2))}>))

  (:action right ...)

  (:action find
   :category (sensing)
   :parameters (?i - agent ?b - box ?p - room)
   :precondition (and (at ?i ?p) (K_?i (at ?i ?p)))
   :observe_pos (and (in ?b ?p) (K_?i (in ?b ?p)))
   :observe_neg (and (not (in ?b ?p)) (K_?i (not (in ?b ?p)))))

  (:action tell
   :category (communication)
   :parameters (?i - agent ?j - agent ?b - box ?p - room)
   :precondition (or (K_?i (in ?b ?p)) (K_?i (not (in ?b ?p))))
   :effect (<{K_?i (in ?b ?p)} {K_?j (in ?b ?p)}>
            <{K_?i (not (in ?b ?p))} {K_?j (not (in ?b ?p))}>))
```

The domain name is `ctc2`. There are three types: `agent`, `room`, and `box`. There are two predicates: binary predicate `at` whose first parameter `?ag` is of type `agent` and second parameter `?p` is of type `room`, and binary predicate `in`. There are three categories of actions: ontic, communication, and sensing. The preconditions of actions, the conditions and effects of conditional effects, and the positive and negative results of sensing actions are represented with arbitrary multi-agent epistemic formulas. For example, `(and (not (at ?i p1)) (K_?i (not (at ?i p1))))` represents the formula $\neg at(i, p_1) \wedge K_i \neg at(i, p_1)$.

Then the following is the problem file for Example 3:

```
(define (problem ctc2)
  (:domain ctc2)
  (:objects b1 b2 - box p1 p2 p3 - room)
  (:agents a b)

  (:init (and (at a p2) (at b p2) (not (in b1 p2)) (not (in b2 p2))
              (K_a (and (at a p2) (not (in b1 p2)) (not (in b2 p2))))
              (K_b (and (at b p2) (not (in b1 p2)) (not (in b2 p2))))))

  (:constraint (and
                 (or (and (in b1 p1) (not (in b1 p2)) (not (in b1 p3)))
                     (and (not (in b1 p1)) (in b1 p2) (not (in b1 p3)))
                     (and (not (in b1 p1)) (not (in b1 p2)) (in b1 p3))
                 (or (and (in b2 p1) (not (in b2 p2)) (not (in b2 p3)))
                     (and (not (in b2 p1)) (in b2 p2) (not (in b2 p3)))
                     (and (not (in b2 p1)) (not (in b2 p2)) (in b2 p3)))
                 ...)
```

[1] The link to our planner and domain sources is: https://github.com/sysulic/MEPK.

```
  (:goal (and (or (K_a (in b1 p1)) (K_a (in b1 p2)) (K_a (in b1 p3)))
              (or (K_b (in b2 p1)) (K_b (in b2 p2)) (K_b (in b2 p3))))))
```

5.2. Overall architecture

The MEPK system consists of two modules: Compiler and Planner. The Compiler module parses the input, *i.e.*, an MEP problem described with EPDDL. Then it does the following conversion: the initial KB, the effects of conditional effects of deterministic actions, and the positive and negative results of sensing actions into ACDFs, the preconditions of actions, the conditions of conditional effects, and the goal into the negation of ACDFs, and the constraint into a DNF formula. Finally, the compiler grounds the operators, namely parameterized actions, into actions, and grounds predicates into atoms. For the `ctc2` input, the operator $left(i)$ will be grounded into $left(a)$ and $left(b)$.

Like [47], we adapt the PrAO algorithm for contingent planning [45] as our planning algorithm. PrAO extends AND/OR forward search with pruning techniques. It uses the so-called minimal DNF formulas to represent states and employs the following basic operations: reasoning, equivalence checking, update w.r.t. ontic actions, and update w.r.t. sensing actions. We use ACDFs to represent states, and for the above operations, we plug in our operations of reasoning, strong equivalence checking, the progression w.r.t. deterministic actions, and progression w.r.t. sensing actions.

Algorithm 1 presents our main procedure. Each node n is a knowledge base in ACDF. We use $state(n)$ to indicate the state of n, which is one of the following values: *goal* meaning that the node is goal achievable, *dead* meaning that the node is not goal achievable, *unexplored*, and *explored*. We use $connected(n)$ to denote whether the node n can be reached from the initial node. Lines 1-5 return an empty tree if $\mathcal{I} \models \mathcal{G}$; otherwise, initialize the search graph, *i.e.*, set the initial node n_0 to the initial KB, and let $connected(n_0) = true$. Lines 6-14 build up the search graph until a solution is found or it is known that no solution exists. If there is no unexplored and connected node, the search graph cannot be further expanded and hence *null* is returned; otherwise, choose an unexplored and connected node n and explore it by calling $Explore(n)$. With loop detection, $Explore(n)$ generates the children of node n by applying any executable deterministic or sensing action. If $state(n_0)$ becomes *goal*, return a solution built from $\mathcal{T}$ by calling $BuildPlan(\mathcal{T})$; if $state(n_0)$ becomes *dead*, return *null*. See [45] for details of $Explore(n)$ and $BuildPlan(\mathcal{T})$.

In line 9, we provide with two strategies to choose nodes: breadth-first search (BFS) and heuristic search. For heuristic search, we use greedy BFS, *i.e.*, we choose an unexplored and connected node with the largest heuristic function value.

To define the heuristic function, we first define the degree of inconsistency of two ACDFs. The definition is inspired by the recursive algorithm to check if an ACDF entails the negation of another (see Proposition 22). Recall that if ϕ and ϕ' are propositional terms, then $\phi \models_\gamma \neg\phi'$ iff for each term t in γ, $\phi \wedge t \wedge \phi'$ has complementary literals.

Definition 30. Let ϕ and ϕ' be two ACDFs, γ a DNF formula. The *degree of inconsistency* of ϕ and ϕ' w.r.t. γ, denoted by $d_\gamma(\phi, \phi')$, is recursively defined as follows:

1. For propositional terms ϕ and ϕ', $d_\gamma(\phi, \phi')$ is the number of terms t in γ s.t. $\phi \wedge t \wedge \phi'$ has complementary literals.
2. When $\phi = \phi_0 \wedge \bigwedge_{a\in\mathcal{B}} \nabla_a \Phi_a$ and $\phi' = \phi'_0 \wedge \bigwedge_{a\in\mathcal{B}'} \nabla_a \Phi'_a$, $d_\gamma(\phi, \phi') \doteq d_\gamma(\phi_0, \phi'_0) + \sum_{a\in\mathcal{B}\cap\mathcal{B}'} \sum_{\phi_a\in\Phi_a, \phi'_a\in\Phi'_a} d_\gamma(\phi_a, \phi'_a)$.
3. When $\phi = \bigvee \Phi$ and $\phi' = \bigvee \Phi'$, $d_\gamma(\phi, \phi') \doteq \sum_{\phi_i\in\Phi, \phi_j\in\Phi'} d_\gamma(\phi_i, \phi_j)$.

Let ϕ be a node generated from a node ϕ'. The heuristic value of ϕ is measured by the gain of the degree of inconsistency with $\neg\mathcal{G}$ when we move from ϕ' to ϕ.

Definition 31. Let ϕ be a node generated from a node ϕ', γ a constraint, and $\mathcal{G}$ the goal. The heuristic value of ϕ (w.r.t. γ and $\mathcal{G}$) is defined as: $h(\phi) = d_\gamma(\phi, \neg\mathcal{G}) - d_\gamma(\phi', \neg\mathcal{G})$.

5.3. Experimental evaluation

We evaluate MEPK with Selective-communication (SC) and Collaboration-and-communication (CC) domains adapted from [31], Coin-in-the-box adapted from [32], and Grapevine from [38], where SC is called "Corridor". We also made up three domains: Assembly-line (AL), and domains adapted from the classic Gossip problem [2] and the knowledge game Hexa [18]. We did not consider three domains from [31]: MuddyChildren, Sum, and WordRooms. The first two involve public announcements, which result in common knowledge, and so we are not able to model them. The third is a variant of CC. Below we give the description of each domain we use.

Selective-communication: SC(n). There are n rooms in a corridor. The agents can move from a room to a neighboring room. When agent i tells some information, all the other agents in the same room or in a neighboring room can hear what was told. Initially, each agent is in one of the rooms. The goal is that some agents get to know some information while some other agents do not. Versions of SC(n) with higher modal depth result from an approximation of common knowledge with higher-order knowledge.

Collaboration-and-communication: CC(k,n), *CC(k,n), †CC(k,n). These are variants of Example 3 where there are k boxes and n rooms. The *CC(k, n) variant uses sensing actions as in Example 3. The CC(k, n) variant imitates the version from [31]

Algorithm 1: $Plan(\mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma)$.

```
Input: An MEP problem $\mathcal{Q} = \langle \mathcal{A}, \mathcal{P}, \mathcal{D}, \mathcal{S}, \mathcal{I}, \mathcal{G}, \gamma \rangle$.
Output: A solution or null.
1  if $\mathcal{I} \models \mathcal{G}$ then
2      return an empty tree
3  else
4      Let $n_0 = \mathcal{I}$, $state(n_0) = unexplored$,
5      $connected(n_0) =$ true, and $\mathcal{T} = \{n_0\}$
6  while true do
7      if there is no node $n$ s.t. $state(n) = unexplored$ and $connected(n) = true$ then
8          return null
9      Choose an unexplored and connected node $n$
10     $Explore(n)$
11     if $state(n_0) = goal$ then
12         return $BuildPlan(\mathcal{T})$
13     if $state(n_0) = dead$ then
14         return null
```

in that with one action, an agent can see which boxes are in a room. So in this version, we replace sensing actions with ontic actions whose effect is knowing whether each box is in a room. We comment that this replacing is possible since this example has linear plans. Furthermore, we add a *cheat* action to *CC(k, n), leading to the variant †CC(k,n). The $cheat(i, j, b, r)$ action means agent i misleads agent j about whether box b is in room r.

Finding-the-truth: FT(k,n). There are k boxes and n rooms. Each box is placed in a room. The agents start with wrong beliefs of the positions of the boxes. The agents can move between the rooms and check if a box is in a room. The goal is for the agents to find out the true locations of the boxes.

Coin-in-the-box: Coin(k). There are three agents and a box containing a coin. Only one agent has the key to the box and one can peek into the box to check the face of the coin if the box is open. Agents can distract (resp. signal) others so that they won't (resp. will) look at the box. One can announce that the coin is showing head or tail. Initially, some of the agents look at the box. The goal is to make certain agents know the face of the coin while others don't. Different values of the parameter k correspond to different settings of the initial KB and goal.

Grapevine(n). A few guests attend a meeting in a villa with n rooms. Each guest has her own secret to share with others. Each guest can move between the rooms, and broadcast her secret to the guests in the same room. The goal is that only some of the guests obtain the designated secrets.

Hexa Game. There are k agents and k cards, each with a unique color. Initially, everyone is holding a card, and can only see the color of her own card. A player can ask a question to another player whether her card is of a certain color. The question should always be honestly answered. The goal is for some agents to know the cards of all players.

Assembly-line (AL). There are two agents, each responsible for processing a part of a product. It is possible that an agent fails in processing her part. An agent can inform the other agent of the status of her task. Two agents decide to assemble the product or restart depending on their knowledge of the status of the agents' tasks.

Gossip. Each of several friends has her own secret to share. Instead of sharing in public, they are only allowed to make a call to each other. In each call, they exchange all the secrets they know. The goal is that everyone knows all the secrets of other friends.

Our experiments were run on a Linux machine with 3.60 GHz Intel Core i7 and 8 GB RAM. Our experimental results are shown in Table 1. The first 3 columns indicate the domain, the number of agents, and the maximal modal depth of the KBs. Their values uniquely determine a problem. The next two columns represent the number of sensing and deterministic actions, and the number of atoms. If the number of sensing actions is 0, the planning problem is conformant, else it is contingent. In the MEPK columns (one for BFS and one for heuristic search), A-$B(X/Y/Z)$ stands for A seconds of total time, B seconds spent on search, depth X and Y nodes of solution tree, and Z nodes searched. The results show the viability of our approach. Nonetheless, our planner does not perform well on the CC domain due to complicated constraints and action effects. Also, our planner does not scale well on Finding-the-truth and Hexa Game. This is because the search performance is greatly influenced by the number of actions, the depth of the shallowest solutions, and the complexity of the goals. For example, in Hexa Game, there are 48 (resp. 100) sensing actions when there are 4 (resp. 5) agents.

We compare the performance of MEPK with those of the planner by Kominis & Geffner[2] [31] and the RP-MEP planner by Muise et al.[3] [38]. We reran the corresponding experiments of the two planners using the FF-X planner with their domain sources, and the comparison results are shown in Table 2. In the last two columns, (X) stands for plan length X, and "na"

[2] Available at https://bitbucket.org/filcom/fromonetomany.
[3] Available at https://bitbucket.org/haz/pdkb-planning/src/default.

Table 1
Experimental results of MEPK with BFS search and MEPK with heuristic search.

Domain	$\|\mathcal{A}\|$	d	$\|\mathcal{S}\|+\|\mathcal{D}\|$	$\|\mathcal{P}\|$	MEPK (BFS)	MEPK (Heu)
CC(2,4)	2	1	0+28	16	22.56-22.56 (6/7/263)	**1.73-1.72** (8/9/45)
CC(3,4)	2	1	0+36	20	257.14-257.13 (6/7/263)	**23.67-23.66** (8/9/45)
CC(4,4)	2	1	0+44	24	–	**514.31-514.29** (8/9/45)
*CC(2,3)	2	1	8+16	12	0.48-0.47 (4/9/119)	**0.15-0.15** (4/10/51)
*CC(2,3)	3	1	12+42	15	2.53-2.52 (5/11/1223)	**0.07-0.07** (5/13/156)
*CC(2,4)	2	1	12+20	16	6.42-6.42 (7/20/1160)	**2.23-2.22** (21/40/751)
*CC(3,3)	3	1	18+60	18	10.72-10.71 (5/13/1429)	**0.19-0.19** (7/15/185)
†CC(2,3)	2	1	8+28	12	19.17-19.16 (4/10/642)	**7.14-7.13** (6/11/471)
†CC(2,3)	3	1	12+78	15	0.22-0.21 (4/7/343)	**0.06-0.05** (6/11/130)
†CC(2,4)	2	1	12+36	16	**0.06-0.05** (3/6/49)	2.31-2.30 (8/14/850)
†CC(3,3)	3	1	18+114	18	0.65-0.63 (4/7/312)	**0.37-0.36** (6/12/238)
FT(1,2)	1	1	2+2	4	**0.01-0.01** (1/3/4)	**0.01-0.01** (1/3/4)
FT(2,3)	1	1	6+2	9	0.08-0.08 (4/8/106)	**0.03-0.03** (4/8/39)
FT(2,3)	2	1	12+4	12	7.42-7.42 (6/11/1423)	**0.32-0.32** (10/16/251)
SC(4)	3	1	1+3	13	0.02-0.01 (5/10/26)	**0.02-0.02** (5/10/22)
SC(4)	7	1	1+3	29	0.04-0.04 (5/10/26)	**0.03-0.03** (5/10/22)
SC(4)	8	1	1+3	33	**0.04-0.03** (5/10/30)	0.05-0.05 (5/10/32)
SC(4)	3	3	1+3	13	**0.02-0.01** (5/10/26)	0.04-0.03 (5/10/32)
SC(4)	3	4	1+3	13	**0.03-0.02** (5/10/26)	0.04-0.03 (5/10/23)
SC(8)	3	1	1+3	25	**0.05-0.04** (10/19/40)	**0.05-0.04** (10/19/43)
Coin(1)	3	1	0+25	8	**0.01-0.01** (2/3/9)	**0.01-0.01** (2/3/9)
Coin(2)	3	1	0+25	8	0.02-0.01 (3/4/31)	**0.01-0.01** (3/4/18)
Coin(3)	3	1	0+25	8	**0.05-0.04** (5/6/79)	**0.05-0.04** (16/17/94)
Grapevine(2)	3	2	0+18	9	**0.01-0.01** (2/3/15)	**0.01-0.01** (2/3/15)
Grapevine(2)	4	1	0+40	12	1.80-1.80 (5/6/2607)	**0.01-0.01** (6/7/61)
Grapevine(2)	4	2	0+56	12	**0.03-0.01** (2/3/22)	0.05-0.02 (2/3/20)
Grapevine(2)	4	3	0+56	12	**0.05-0.02** (2/3/22)	0.09-0.05 (2/3/20)
Grapevine(2)	4	4	0+56	12	**0.06-0.03** (2/3/22)	0.09-0.05 (2/3/20)
Grapevine(3)	4	1	0+152	16	**0.07-0.01** (2/3/27)	0.66-0.60 (16/17/472)
	3	1	18+0	9	**0.01-0.01** (1/3/3)	**0.01-0.01** (1/3/3)
Hexa Game	4	1	48+0	16	**0.02-0.02** (3/11/42)	**0.02-0.02** (3/11/50)
	5	1	100+0	25	**9.69-9.68** (6/47/1670)	–
	2	2	2+4	4	**0.01-0.01** (5/12/32)	0.02-0.02 (5/18/32)
	2	3	2+4	4	**0.02-0.02** (5/12/32)	0.03-0.03 (5/18/32)
Assemble Line	2	4	2+4	4	**0.04-0.04** (5/12/32)	**0.04-0.04** (5/18/32)
	2	5	2+4	4	**0.07-0.07** (5/12/32)	0.08-0.08 (5/18/32)
	2	7	2+4	4	**0.36-0.36** (5/12/32)	0.45-0.44 (5/18/32)
	2	10	2+4	4	**5.53-5.53** (5/12/32)	6.62-6.62 (5/18/32)
	3	2	0+6	3	0.03-0.03 (3/4/11)	**0.02-0.01** (3/4/7)
Gossip	4	2	0+24	4	1.84-1.83 (4/5/133)	**0.20-0.20** (5/6/30)
	5	2	0+120	5	–	**1.10-1.09** (7/8/99)

means that the planning problem is not considered by the planner. For example, CC is not encoded by RP-MEP; as to SC, K&G only encodes SC(4) with 3 agents and $d = 1$. The results show that the searching performance of MEPK is reasonable, but still worse than the two other planners. This is because MEPK uses more general and complex KBs, and hence spends more time on reasoning and progression. However, when it comes to total time, MEPK performs better than the two other planners except for the CC domain. This is because MEPK saves from the expensive compilation into classical planning.

We also compare the performance of MEPK with those of the EFP and PG-EFP planners[4] by Le et al. [32]. Since (PG-)EFP is only executable on macOS, the comparison experiments were run on macOS with 2.30 GHz Intel Core i5 and 8 GB RAM. We use the problem settings of the two planners,[5] and the comparison results are illustrated in Table 3. We set the timeout bound to 30 minutes, and "–" means timeout. The results show that MEPK performs better in most instances.

Finally, we should note that the four planners use different encodings of MEP problems. Thus even for the same domain, the exact MEP problem considered by a planner might be different from that considered by another planner. For example, for the CC domain, the K&G and MEPK encodings do not specify the exact positions of boxes in the initial state, while the RP-MEP encoding has to since it cannot represent disjunctive beliefs. More details of the different encodings will be given in the next section.

[4] Available at https://github.com/tiep/EpistemicPlanning.
[5] Note the differences in the MEPK data for the CC(2,4) and SC(4) domains in Tables 2 and 3.

Table 2
Comparison results with planners by Kominis & Geffner and Muise et al..

Domain	$\lvert\mathcal{A}\rvert$	d	MEPK (BFS)	MEPK (Heu)	K&G	RP-MEP
CC(2,4)	2	1	22.56-22.56 (6/7/263)	1.73-1.72 (8/9/45)	**0.02-0.01** (8)	na
CC(3,4)	2	1	257.14-257.13 (6/7/263)	23.67-23.66 (8/9/45)	**4.17-0.03** (8)	na
CC(4,4)	2	1	–	**514.31-514.29** (8/9/45)	726.14-0.71 (8)	na
	3	1	0.02-0.02 (5/10/27)	**0.01-0.01** (5/10/27)	0.01-0.01 (9)	0.04-0.01 (5)
SC(4)	7	1	0.04-0.04 (5/10/26)	**0.03-0.03** (5/10/22)	na	0.05-0.01 (5)
	3	3	**0.02-0.01** (5/10/26)	0.04-0.03 (5/10/32)	na	0.12-0.01 (5)
	3	2	**0.01-0.01** (2/3/15)	**0.01-0.01** (2/3/15)	na	0.18-0.01 (2)
Grapevine(2)	4	1	1.80-1.80 (5/6/2607)	**0.01-0.01** (6/7/61)	na	0.11-0.01 (5)
	4	2	**0.03-0.01** (2/3/22)	0.05-0.02 (2/3/20)	na	0.53-0.02 (2)

Table 3
Comparison results with EFP and PG-EFP by Le et al..

Domain	$\lvert\mathcal{A}\rvert$	d	MEPK (BFS)	MEPK (Heu)	EFP	PG-EFP
CC(2,4)	2	1	0.16-0.14 (3/4/15)	**0.09-0.07** (3/4/9)	–	42.85-40.86 (7)
CC(2,3)	3	1	0.68-0.59 (4/5/35)	**0.37-0.28** (4/5/23)	12.92-12.34 (4)	2.42-1.84 (4)
CC(3,3)	3	1	47.01-46.70 (4/5/105)	6.43-6.12 (4/5/25)	770.57-770.43 (6)	**2.35-2.19** (6)
SC(4)	3	1	**0.03-0.03** (5/10/26)	0.04-0.03 (5/10/28)	0.06-0.06 (5)	–
SC(6)	5	1	0.06-0.05 (6/12/55)	**0.04-0.03** (7/13/34)	0.36-0.35 (6)	0.24-0.24 (6)
SC(8)	7	1	**0.45-0.44** (9/18/159)	1.35-1.34 (11/20/253)	15.04-15.02 (9)	–
	3	2	**0.02-0.01** (2/3/15)	0.10-0.10 (4/5/64)	0.20-0.17 (2)	–
Grapevine(2)	4	1	8.08-8.08 (5/6/2607)	**0.16-0.15** (10/11/209)	–	–
	4	2	**0.07-0.04** (2/3/22)	1.17-1.14 (4/5/179)	1.25-1.08 (2)	–
Coin(1)	3	2	**0.01-0.01** (2/3/9)	0.02-0.01 (2/3/9)	0.04-0.02 (2)	0.21-0.20 (2)
Coin(2)	4	2	0.03-0.02 (3/4/31)	**0.02-0.01** (3/4/18)	0.17-0.16 (3)	0.88-0.87 (3)
Coin(3)	5	2	0.09-0.09 (5/6/79)	**0.06-0.05** (6/7/50)	2.71-2.70 (5)	1.17-1.16 (5)
Assemble Line	2	2	0.03-0.03 (5/12/32)	**0.03-0.03** (5/12/30)	1.48 - 1.47 (5)	0.70 - 0.69 (5)

Table 4
Comparison of four MEP systems.

System	Approach	Solution	Action	Knowledge/Belief	Language
K&G	compilation	linear	public	knowledge	$\mathcal{L}_K$
RP-MEP	compilation	linear	private	belief	RML^d
MEPK	native (KB)	branching	private	belief	$\mathcal{L}_K$
(PG-)EFP	native (model)	linear	private	belief	$\mathcal{L}_K$

6. Comparison of four MEP systems

In this section, we compare the four MEP systems: K&G [31], RP-MEP [38], MEPK, and (PG-)EFP [32].

Table 4 illustrates a comparison of basic aspects of the four systems. Both K&G and RP-MEP rely on compilation into classical planning, while MEPK and (PG-)EFP are native planners that search in the space of knowledge bases and Kripke models, respectively. MEPK generates branching solutions as action trees, while the other systems all generate linear plans. K&G assumes all actions are public and hence handles knowledge, while the other systems consider private actions and hence beliefs. Finally, K&G, MEPK and (PG-)EFP all use the multi-agent epistemic language, while RP-MEP uses a restricted language RML^d, denoting restricted modal literals with bounded modal depth.

In the following, for each of K&G, RP-MEP and (PG-)EFP, we illustrate how the approach encodes a variant of Example 3, due to the difference in expressibility of different approaches.

We begin with K&G summarized that they deal with knowledge rather than belief. All the agents start with a common initial belief. In this approach, a sensing action $sense(A, \Phi)$ represents that agents in A sense in parallel the truth value of each formula of Φ, which is a set of multi-agent epistemic formulas. We formalize a variant of Example 3 with 4 rooms:

- The ontic actions are: $left(i)$ and $right(i)$, where
  $left(i) = \langle pre, \{eff_1, eff_2, eff_3\}\rangle$, with $pre = \neg at(i, p_1)$,
  $eff_1 = \langle at(i, p_2), \{at(i, p_1), \neg at(i, p_2), \neg at(i, p_3), \neg at(i, p_4)\}\rangle$,
  $eff_2 = \langle at(i, p_3), \{\neg at(i, p_1), at(i, p_2), \neg at(i, p_3), \neg at(i, p_4)\}\rangle$, and
  $eff_3 = \langle at(i, p_4), \{\neg at(i, p_1), \neg at(i, p_2), at(i, p_3), \neg at(i, p_4)\}\rangle$.

- The communication actions are: $sense(i, \{K_j in(b, p)\})$, meaning agent j communicates to agent i whether j knows $in(b, p)$.
- The sensing actions are: $sense(i, \{in(b_1, p), in(b_2, p)\})$ with precondition $at(i, p)$, meaning agent i finds out in parallel whether each block is in p.
- The common initial belief is $at(1, p_2) \wedge at(2, p_2) \wedge \neg in(b_1, p_2) \wedge \neg in(b_2, p_2) \wedge \gamma_2$, where γ_2 expresses that each box has a unique location, as in Example 3.
- The goal is $\bigwedge_{i=1}^{2} \bigvee_{k=1}^{4} K_i in(b_i, p_k)$.
- A solution is: $left(1), right(2), sense(1, \{in(b_1, p_1), in(b_2, p_1)\}), sense(2, \{in(b_1, p_3), in(b_2, p_3)\}), sense(1, \{K_2 in(b_1, p_3)\}), sense(2, \{K_1 in(b_2, p_1)\})$. This is because: If agent 1 senses b_1 is in p_1, then she knows the position of p_1. Otherwise, she knows that b_1 is not in p_1. If she senses $K_2 in(b_1, p_3)$, then she also knows that b_1 is in p_3. Otherwise, she knows that b_1 is not in p_3. Since the common initial belief is that b_1 is not in p_2, she gets to know that b_1 is in p_4. The case for agent 2 is similar.

The RP-MEP approach makes use of restricted modal literals (RMLs) from the perspective of a single root agent $\star$. Thus RP-MEP cannot represent the constraint $in(b_1, p_1) \vee in(b_1, p_2) \vee in(b_1, p_3)$. Therefore, we formulate a variant of Example 3 where b_1 is in p_1 and b_2 is in p_3, which are represented by RMLs $B_\star in(b_1, p_1)$ and $B_\star in(b_2, p_3)$.

- The actions are (with action $right(i)$ omitted):
  $left(i) = \langle \neg at(i, p_1), \{eff_1, eff_2\}\rangle$, where $eff_1 = \langle at(i, p_2), \{B_\star at(i, p_1), B_\star \neg at(i, p_2)\}\rangle$, and
  $eff_2 = \langle at(i, p_3), \{B_\star at(i, p_2), \neg B_\star at(i, p_3)\}\rangle$;
  $tell(i, j, b, p) = \langle \top, \{eff_1, eff_2\}\rangle$, where $eff_1 = \langle B_i in(b, p), \{B_\star B_j in(b, p)\}\rangle$, and $eff_2 = \langle B_i \neg in(b, p), \{B_\star B_j \neg in(b, p)\}\rangle$;
  $find(i, b, p) = \langle pre, \{eff_1, eff_2\}\rangle$, where $pre = at(i, p)$, $eff_1 = \langle in(b, p), \{B_\star B_i in(b, p)\}\rangle$, and $eff_2 = \langle \neg in(b, p), \{B_\star B_i \neg in(b, p)\}\rangle$.
- The initial KB is $at(1, p_2) \wedge at(2, p_2) \wedge in(b_1, p_1) \wedge in(b_2, p_3) \wedge B_1 at(1, p_2) \wedge B_2 at(2, p_2)$.
- The goal is $B_1 in(b_1, p_1) \wedge B_2 in(b_2, p_3)$.

(PG-)EFP encodes an MEP problem using $m\mathcal{A}$ [7], a multi-agent epistemic extension of the action language $\mathcal{A}$ [23], and finitary S5-theories. The initial state is specified by a finitary S5-theory, which is an S5-theory contains only formulas of the following forms: ϕ, $CK_i\phi$, $C(K_i\phi \vee K_i\neg\phi)$, where C is the common knowledge operator, and ϕ is a propositional formula. We formulate a variant of Example 3 as follows:

- Action preconditions (with action $right(i)$ omitted):
  executable $left(i)$ **if** $\neg at(i, p_1)$,
  executable $find(i, b, p)$ **if** $K_i at(i, p)$,
  executable $tell(i, j, b, p)$ **if** $K_i in(b, p)$.
- Action effects (with action $right(i)$ omitted):
  $left(i)$ **causes** $at(i, p_1)$ **if** $at(i, p_2)$, $left(i)$ **causes** $\neg at(i, p_2)$ **if** $at(i, p_2)$,
  $left(i)$ **causes** $at(i, p_2)$ **if** $at(i, p_3)$, $left(i)$ **causes** $\neg at(i, p_3)$ **if** $at(i, p_3)$,
  $tell(i, j, b, p)$ **announces** $in(b, p)$, meaning $in(b, p)$ becomes the common knowledge of the observers,
  $find(i, b, p)$ **determines** $in(b, p)$, meaning the observers get to know the truth value of $in(b, p)$.
- Action observability statements:
  $\{i, j\}$ **observes** $tell(i, j, b, p)$,
  i **observes** $find(i, b, p)$.
- The initial epistemic state is specified by a finitary S5-theory:
  initially $at(1, p_2) \wedge at(2, p_2) \wedge in(b_1, p_1) \wedge in(b_2, p_3) \wedge Cat(1, p_2) \wedge Cat(2, p_2)$.
- The goal is expressed as: **goal** $\bigwedge_{i=1}^{2} \bigvee_{k=1}^{3} K_i in(b_i, p_k)$.

7. Conclusions

In this paper, we have proposed a novel multi-agent epistemic planning framework based on higher-order belief change. In this framework, the initial KB and the goal, the preconditions and effects of actions can be arbitrary KD45$_n$ formulas, the progression of KBs w.r.t. actions is achieved through the operation of belief revision or update, and the solution is an action tree branching on sensing results.

Now we would like to compare the modeling approach of our paper with the DEL-based approach. In the DEL-based approach, an epistemic state is modeled by a Kripke model, an action is modeled by an action model, and the progression of an epistemic state w.r.t. an action is modeled by the product update of the Kripke model by the action model. Whereas in our approach, an epistemic state is modeled by a KB in multi-agent epistemic logic, an action is modeled by its preconditions and effects like in classical planning, and progression is modeled by high-level belief revision or update. In comparison with the DEL-based approach, we think ours is closer to the representation framework from the planning community, and more suitable for representing multi-agent epistemic planning problems.

To support efficient reasoning and progression, we resort to alternating cover disjunctive formulas (ACDFs). Any KD45$_n$ formula ϕ can be transformed to an equivalent ACDF whose length is singly exponential in the length of ϕ, but doubly exponential in the non-alternating factor of ϕ, *i.e.*, the number of modal operators of an agent which directly occur inside those of the same agent. We propose reasoning, strong equivalence checking, and basic revision and update algorithms for ACDFs. The complexities of our algorithms are polynomial in the size of the formulas but exponential in the modal depth of the formulas, which we expect to be small, except that the complexity of the update algorithm is also exponential in the size of the update formulas, which are effects of actions and hence usually small formulas. Based on the theoretic work, we have implemented a native multi-agent epistemic planner MEPK, which does not rely on compilation into classical planning. Experimental results have demonstrated the viability of our approach.

In this paper, we give syntactic approaches for higher-order belief revision and update. The operators have basic properties that both return satisfiable formulas entailing the revision or update formula, revision has the conjunction property, and update has the distribution property. For proper ACDFs, we show that semantic characterizations for propositional belief change nicely carry over to higher-order belief change. This is achieved via introducing the concept of tree models, showing that a proper ACDF has a model iff it has a tree model, and then restricting our attention to tree models. We do not yet have general semantic definitions of higher-order belief change. Despite this, we think that we have made an important first step towards the study of higher-level belief change. In the future, we are interested in doing a general model-theoretic study of higher-order belief change and improving our revision and update algorithms.

Finally, by basing multi-agent epistemic planning on higher-order belief change, this paper exposes an interesting challenge – how to use revision and/or update to model sensing and communication actions. In this paper, we use revision for sensing and communication actions. However, there are arguments that update should be used for these actions, and arguments that both revision and update should be used for communication. In the future, we are interested in a more thorough exploration of this challenging issue.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We are deeply grateful to the anonymous reviewers whose comments and suggestions have helped us significantly improve the paper. We thank Liangda Fang for helpful discussions on the paper, and Hans van Ditmarsch for introducing us to the normal form of alternating cover disjunctive formulas. Yongmei Liu acknowledges support from National Natural Science Foundation of China under Grant No. 61572535. Hai Wan and Biqing Fang acknowledge support from National Natural Science Foundation of China under Grant No. 61976232, and Guangzhou Science and Technology Project under Grant No. 201804010435.

Appendix A. Proofs

Proof of Proposition 2. We prove by induction on $|\phi|$. Basis: ϕ is p. No transformation is needed. Induction step: Let $p_1,\ldots,p_m$ be the atoms that appear in ϕ but not within the scope of any modal operator. Let $K_{a_1}\phi_1,\ldots,K_{a_n}\phi_n$ be the modal atoms that appear in ϕ but not within the scope of any modal operator. Let $\Phi=\{\phi_1,\ldots,\phi_n\}$. For $i=1,\ldots,n$, let $l_i=|\phi_i|$. Let $l=\Sigma_{i=1}^{n} l_i$. It is easy to prove by induction that $|\phi|\ge 2(m+n)+l-1$. Firstly, we treat $K_{a_1}\phi_1,\ldots,K_{a_n}\phi_n$ as atoms and put ϕ into DNF. Let ϕ' be the resulting formula. Then there are at most 2^{m+n} disjuncts in ϕ', and each disjunct is of the form $\eta=\phi_0\wedge\bigwedge_{a\in\mathcal{B}}(K_a\bigwedge\Phi_a\wedge L_a\Psi_a)$, where $\mathcal{B}\subseteq\mathcal{A}$, ϕ_0 is a term of $p_1,\ldots,p_m$, Φ_a and Ψ_a are disjoint subsets of Φ. By Proposition 1 (2), $\eta\ \Leftrightarrow\phi_0\wedge\bigwedge_{a\in\mathcal{B}}\nabla_a(\{\bigwedge\Phi_a\}\cup\{\bigwedge\Phi_a\wedge\psi\mid\psi\in\Psi_a\})$. By induction, each formula β of $\bigwedge\Phi_a$ and $\bigwedge\Phi_a\wedge\psi$ can be transformed to an equivalent CDF whose length is $O(2^{|\beta|^2})$. Clearly, $|\beta|\le l+n-1$. In total, there are at most n such formulas. Thus we obtain a CDF ϕ'', equivalent to ϕ', and $|\phi''|\in O(2^{m+n}n2^{(l+n-1)^2})$, hence in $O(2^{(l+2(m+n)-1)^2})$, which is $O(2^{|\phi|^2})$. □

Proof of Proposition 3. 1. $\Leftarrow$: If $M,w\models K_a\pi$, obviously, $M,w\models K_a(\pi\vee\alpha\wedge K_a\beta)$. So let $M,w\models K_a(\pi\vee\alpha)\wedge K_a\beta$. Now let v satisfy wR_av. Then $M,v\models\pi\vee\alpha$. Now let u satisfy vR_au. Since R_a is transitive, we have wR_au. From $M,w\models K_a\beta$, we get $M,u\models\beta$. Thus $M,v\models K_a\beta$. So $M,w\models K_a(\pi\vee\alpha\wedge K_a\beta)$.
$\Rightarrow$: Let $M,w\models K_a(\pi\vee\alpha\wedge K_a\beta)$. There are two cases: (1) $M,w\models K_a\beta$. It is easy to show $M,w\models K_a(\pi\vee\alpha)$. (2) $M,w\models\neg K_a\beta$. Then there exists u s.t. wR_au and $M,u\models\neg\beta$. Now let v satisfy wR_av. Since R_a is Euclidean, vR_au. Hence $M,v\models\neg K_a\beta$. Since $M,v\models\pi\vee\alpha\wedge K_a\beta$, we get $M,v\models\pi$. Hence $M,w\models K_a\pi$.

2. $\Leftarrow$: If $M,w\models K_a\pi$, obviously, $M,w\models K_a(\pi\vee\alpha\wedge L_a\beta)$. So let $M,w\models K_a(\pi\vee\alpha)\wedge L_a\beta$. Then there exist v s.t. wR_av and $M,v\models\beta$. Now let u satisfy wR_au. Since R_a is Euclidean, uR_av. So $M,u\models L_a\beta$. Since $M,u\models\pi\vee\alpha$, we get $M,w\models K_a(\pi\vee\alpha\wedge L_a\beta)$.
$\Rightarrow$: Let $M,w\models K_a(\pi\vee\alpha\wedge L_a\beta)$. There are two cases: (1) $M,w\models L_a\beta$. It is easy to show $M,w\models K_a(\pi\vee\alpha)$. (2) $M,w\models\neg L_a\beta$. Now let u satisfy wR_au. Then $M,u\models\pi\vee\alpha\wedge L_a\beta$. Then for all v s.t. uR_av, since R_a is transitive, wR_av, and hence $M,v\models\neg\beta$ because $M,w\models\neg L_a\beta$. So $M,u\models\neg L_a\beta$. Thus $M,u\models\pi$. Therefore, $M,w\models K_a\pi$. □

Proof of Proposition 4. We prove by induction on $|\phi|$. When ϕ is p, $\neg\psi$, or $(\psi \wedge \psi')$, the proof is easy. Now let ϕ be $K_a\psi$. If $na(\phi) = 0$, no transformation is needed. Otherwise, $K_a\psi$ must be of the form $K_a(\pi \vee \alpha \wedge K_a\beta)$ or $K_a(\pi \vee \alpha \wedge L_a\beta)$, where π might be $\bot$ and α might be $\top$. We only prove the first case, the second case is similar. By Proposition 3, $K_a\psi \Leftrightarrow K_a(\pi \vee \alpha) \wedge K_a\beta \vee K_a\pi \wedge \neg K_a\beta$. Let $n_1 = na(K_a\pi)$, $n_2 = na(K_a\alpha)$, and $n_3 = na(K_a\beta)$; let $l_1 = |\pi|, l_2 = |\alpha|$, and $l_3 = |\beta|$. Then $na(K_a\psi) = n_1 + n_2 + n_3 + 1$, and $|K_a\psi| = l_1 + l_2 + l_3 + 4$. By induction, each of $K_a(\pi \vee \alpha)$, $K_a\beta$, $K_a\pi$ is equivalent to an alternating formula. Hence $K_a\psi$ is equivalent to an alternating formula η, and

$$\begin{aligned}
|\eta| \leq & 2^{n_1+n_2}(l_1 + l_2 + 2) + 2 \cdot 2^{n_3}(l_3 + 1) + 2^{n_1}(l_1 + 1) + 4 \\
= & (2^{n_1+n_2} + 2^{n_1})l_1 + 2^{n_1+n_2}l_2 + 2^{n_3+1}l_3 + 2^{n_1+n_2+1} + 2^{n_3+1} + 2^{n_1} + 4 \\
\leq & 2^{n_1+n_2+n_3+1}(l_1 + l_2 + l_3 + 4) = 2^{na(\phi)}|\phi|. \quad \square
\end{aligned}$$

Proof of Proposition 10. $\{Diff(\psi_i, \mu_j) \mid \psi_i \in \psi, \mu_j \in \mu\}$ can be computed in time $O(|\psi| \cdot |\mu|)$, hence $MinPairs(\psi, \mu)$ and so $\psi \circ_s \mu$ can be computed in time $O(|\psi|^2 \cdot |\mu|^2)$. In the worst case, each of $Diff(\psi_i, \mu_j)$ is empty, and so $\psi \circ_s \mu$ is of size $O(|\psi| \cdot |\mu|)$. $\square$

Proof of Proposition 12. $revise(\psi_i, \mu_j)$ and $Diff(\mu_j, \psi_i)$ can be computed in time $O(|\psi_i| \cdot |\mu_j|)$. $patch_{\psi_i}(\mu_j)$ can be computed in time $O(|\mu| \cdot (|\psi_i| + |\mu_j|))$. Note that $patch_{\psi_i}(\mu_j)$ is a sub-expression of $\neg \bigwedge_{k \neq j} \mu_k$. Hence $revise(\psi_i, \mu_j) \wedge patch_{\psi_i}(\mu_j)$ can be put into DNF in $O(|\psi_i| \cdot 2^{|\mu|})$. Therefore, the DNF formula of $\psi \diamond_w \mu$ can be computed in time $O(|\psi| \cdot 2^{|\mu|})$, and the resulting formula is of size $O(|\psi| \cdot 2^{|\mu|})$. $\square$

References

[1] C.E. Alchourrón, P. Gärdenfors, D. Makinson, On the logic of theory change: partial meet contraction and revision functions, J. Symb. Log. 50 (1985) 510–530.
[2] M. Attamah, H. van Ditmarsch, D. Grossi, W. van der Hoek, Knowledge and gossip, in: Proceedings of the 21st European Conference on Artificial Intelligence (ECAI-2014), 2014, pp. 21–26.
[3] G. Aucher, Generalizing AGM to a multi-agent setting, Log. J. IGPL 18 (2010) 530–558.
[4] G. Aucher, DEL-sequents for progression, J. Appl. Non-Class. Log. 21 (2011) 289–321.
[5] G. Aucher, T. Bolander, Undecidability in epistemic planning, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-2013), 2013, pp. 27–33.
[6] A. Baltag, S. Smets, A qualitative theory of dynamic interactive belief revision, in: Logic and the foundations of game and decision theory (LOFT 7), vol. 3, Amsterdam University Press, 2008, pp. 9–58.
[7] C. Baral, G. Gelfond, E. Pontelli, T.C. Son, An action language for reasoning about beliefs in multi-agent domains, in: Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR-2012), 2012.
[8] J. van Benthem, Dynamic logic for belief revision, J. Appl. Non-Class. Log. 17 (2007) 129–155.
[9] M. Bienvenu, H. Fargier, P. Marquis, Knowledge compilation in the modal logic S5, in: Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI-2010), 2010, pp. 261–265.
[10] T. Bolander, M.B. Andersen, Epistemic planning for single and multi-agent systems, J. Appl. Non-Class. Log. 21 (2011) 9–34.
[11] T. Caridroit, S. Konieczny, T. de Lima, P. Marquis, On distances between KD45n Kripke models and their use for belief revision, in: Proceedings of the Twenty-second European Conference on Artificial Intelligence (ECAI-2016), 2016, pp. 1053–1061.
[12] T. Charrier, B. Maubert, F. Schwarzentruber, On the impact of modal depth in epistemic planning, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-2016), 2016, pp. 1030–1036.
[13] S.L. Cong, S. Pinchinat, F. Schwarzentruber, Small undecidable problems in epistemic planning, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-2018), 2018, pp. 4780–4786.
[14] M.C. Cooper, A. Herzig, F. Maffre, F. Maris, P. Régnier, A simple account of multi-agent epistemic planning, in: Proceedings of the Twenty-second European Conference on Artificial Intelligence (ECAI-2016), 2016, pp. 193–201.
[15] M.C. Cooper, A. Herzig, F. Maffre, F. Maris, P. Régnier, Simple epistemic planning: generalised gossiping, in: Proceedings of the Twenty-second European Conference on Artificial Intelligence (ECAI-2016), 2016, pp. 1563–1564.
[16] G. D'Agostino, G. Lenzi, On modal μ-calculus with explicit interpolants, J. Appl. Log. 4 (2006) 256–278.
[17] A. Darwiche, J. Pearl, On the logic of iterated belief revision, Artif. Intell. 89 (1997) 1–29.
[18] H. van Ditmarsch, Knowledge games, Bull. Econ. Res. (2001) 249–273.
[19] H. van Ditmarsch, W. van der Hoek, B.P. Kooi, Dynamic Epistemic Logic, Springer, 2007.
[20] T. Engesser, T. Bolander, R. Mattmüller, B. Nebel, Cooperative epistemic multi-agent planning for implicit coordination, in: Proceedings of the Ninth Workshop on Methods for Modalities (M4M-2017), 2017, pp. 75–90.
[21] R. Fagin, J. Halpern, Y. Moses, M. Vardi, Reasoning about Knowledge, MIT Press, 1995.
[22] K. Fine, Normal forms in modal logic, Notre Dame J. Form. Log. 16 (1975) 229–237.
[23] M. Gelfond, V. Lifschitz, Representing action and change by logic programs, J. Log. Program. 17 (1993) 301–321.
[24] J. Hales, T. French, R. Davies, Refinement quantified logics of knowledge and belief for multiple agents, in: Proceedings of the Ninth Conference on Advances in Modal Logic (AiML-2012), 2012, pp. 317–338.
[25] S.M. Hedetniemi, S.T. Hedetniemi, A.L. Liestman, A survey of gossiping and broadcasting in communication networks, Networks 18 (1988) 319–349.
[26] A. Herzig, J. Lang, P. Marquis, Action representation and partially observable planning using epistemic logic, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003), 2003, pp. 1067–1072.
[27] J.-J.Ch. Meyer, W. van der Hoek, Epistemic Logic for AI and Computer Science, Cambridge University Press, 2004.
[28] X. Huang, B. Fang, H. Wan, Y. Liu, A general multi-agent epistemic planner based on higher-order belief change, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-2017), 2017, pp. 1093–1101.
[29] D. Janin, I. Walukiewicz, Automata for the modal mu-calculus and related results, in: Proceedings of the Twentieth International Symposium on Mathematical Foundations of Computer Science (MFCS-1995), 1995, pp. 552–562.

[30] H. Katsuno, A.O. Mendelzon, On the difference between updating a knowledge base and revising it, in: Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR-1991), 1991, pp. 387–394.
[31] F. Kominis, H. Geffner, Beliefs in multiagent planning: from one agent to many, in: Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS-2015), 2015, pp. 147–155.
[32] T. Le, F. Fabiano, T.C. Son, E. Pontelli, EFP and PG-EFP: epistemic forward search planners in multi-agent domains, in: Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS-2018), 2018, pp. 161–170.
[33] Q. Liu, Y. Liu, Multi-agent epistemic planning with common knowledge, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-2018), 2018, pp. 1912–1920.
[34] B. Löwe, E. Pacuit, A. Witzel, DEL planning and some tractable cases, in: Proceedings of the Third International Workshop on Logic, Rationality, and Interaction (LORI-2011), 2011, pp. 179–192.
[35] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL—The Planning Domain Definition Language, Technical Report CVC TR98003/DCS TR1165, Yale Center for Computational Vision and Control, New Haven, CT, 1998.
[36] T. Miller, C.J. Muise, Belief update for proper epistemic knowledge bases, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-2016), 2016, pp. 1209–1215.
[37] L.S. Moss, Finite models constructed from canonical formulas, J. Philos. Log. 36 (2007) 605–640.
[38] C.J. Muise, V. Belle, P. Felli, S.A. McIlraith, T. Miller, A.R. Pearce, L. Sonenberg, Planning over multi-agent epistemic states: a classical planning approach, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-2015), 2015, pp. 3327–3334.
[39] R.P.A. Petrick, F. Bacchus, A knowledge-based approach to planning with incomplete information and sensing, in: Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS-2002), 2002, pp. 212–222.
[40] R.P.A. Petrick, F. Bacchus, Extending the knowledge-based approach to planning with incomplete information and sensing, in: Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS-2004), 2004, pp. 613–622.
[41] K. Satoh, Nonmonotonic reasoning by minimal belief revision, in: Proceedings of the First International Conference on Fifth Generation Computer Systems (FGCS-1988), 1988, pp. 455–462.
[42] T.C. Son, E. Pontelli, C. Baral, G. Gelfond, Finitary s5-theories, in: Logics in Artificial Intelligence - Proceedings of the 14th European Conference (JELIA-2014), 2014, pp. 239–252.
[43] B. ten Cate, W. Conradie, M. Marx, Y. Venema, Definitorially complete description logics, in: Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2006), 2006, pp. 79–89.
[44] S.T. To, E. Pontelli, T.C. Son, A conformant planner with explicit disjunctive representation of belief states, in: Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS-2009), 2009.
[45] S.T. To, T.C. Son, E. Pontelli, Contingent planning as and/or forward search with disjunctive representation, in: Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS-2011), 2011, pp. 258–265.
[46] A. del Val, Syntactic characterizations of belief change operators, in: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-1993), 1993, pp. 540–545.
[47] H. Wan, R. Yang, L. Fang, Y. Liu, H. Xu, A complete epistemic planner without the epistemic closed world assumption, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-2015), 2015, pp. 3257–3263.
[48] M. Winslett, Reasoning about action using a possible models approach, in: Proceedings of the Seventh Conference on Artificial Intelligence (AAAI-1988), 1988, pp. 89–93.
[49] Q. Yu, X. Wen, Y. Liu, Multi-agent epistemic explanatory diagnosis via reasoning about actions, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI-2013), 2013, pp. 1183–1190.