An Action Language for Multi-Agent Domains

Chitta Baral[†], Gregory Gelfond[‡], Enrico Pontelli[◇], Tran Cao Son[◇⋆]

[†]*Department of Computer Science and Engineering, Arizona State University*
[‡]*Department of Computer Science, University of Nebraska Omaha*
[◇]*Department of Computer Science, New Mexico State University*
[⋆]*Corresponding author*

Abstract

In multi-agent domains, the actions performed by an agent may not only modify the world and the agent's knowledge and beliefs about the world, but may also change the knowledge and beliefs of other agents about the world and their own knowledge and beliefs about other agents' knowledge and beliefs about the world. Similarly, the goals of an agent in a multi-agent domain may involve manipulating the knowledge and beliefs of other agents' and, again, not just their knowledge[1] about the world, but also their knowledge about other agents' knowledge about the world.

The goal of this paper is to investigate an action language, called m$\mathcal{A}^*$, that provides the necessary features to address the above aspects in representing and reasoning about actions and change in multi-agent domains. The language, as designed, can also serve as a specification language for epistemic planning, thereby addressing an important issue in the development of multi-agent epistemic planning systems. The m$\mathcal{A}^*$ action language is a generalization of the single-agent action languages, extensively studied in the literature, to the case of *multi-agent domains.* The language allows the representation of different types of actions that an agent can perform in a domain where many other agents might be present—such as world-altering actions, sensing actions, and communication actions. The action language also allows the specification of agents' *dynamic awareness* of action occurrences—which has implications on what agents' know about the world and other agents' knowledge about the world. These features are embedded in a language that is simple, yet powerful enough to address a large variety of knowledge manipulation scenarios in multi-agent domains.

The semantics of m$\mathcal{A}^*$ relies on the notion of *state,* which is described by a *pointed Kripke model* and is used to encode the agents' knowledge and the real state of the world. The semantics is defined by a transition function that maps pairs of actions and states into sets of states. The paper presents a number of properties of the action theories, including properties that guarantee finiteness of the set of initial states and their practical realization in a system. Finally, the paper relates m$\mathcal{A}^*$ to other relevant formalisms in the area of reasoning about actions in multi-agent domains.

Keywords: Action Languages, Epistemic Planning, Reasoning about Knowledge

Email address: chitta@asu.edu, ggelfond@unomaha.edu, epontell@cs.nmsu.edu, tson@cs.nmsu.edu (Chitta Baral[†], Gregory Gelfond[‡], Enrico Pontelli[◇], Tran Cao Son[◇⋆])

[1]We will use the term "knowledge" to mean both "knowledge" and "beliefs" when clear from the context.

1. Introduction

1.1. Motivations

Reasoning about Actions and Change (RAC) has been a research focus since the early days of artificial intelligence (McCarthy, 1959). Languages for representing actions and their effects have been proposed soon after (Fikes and Nilson, 1971). Although the early papers on this topic by Fikes and Nilson (1971) did not include formal semantics, papers with formal semantics came some years after, with leading efforts by Lifschitz (1987). The approach adopted in this paper is predominantly influenced by the methodology for representing and reasoning about actions and change proposed by Gelfond and Lifschitz (1993). In this approach, actions of agents are described in a high-level language, with an English-like syntax and a transition function-based semantics. Action languages offer several benefits, including a succinct way for representing dynamic domains. The approach proposed in this paper is also related to action description languages developed for planning, such as (Pednault, 1989; Ghallab et al., 1998).

Over the years, several action languages (e.g., $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$) have been developed, as discussed in (Gelfond and Lifschitz, 1998). Each of these languages addresses some important problems in RAC, e.g., the ramification problem, concurrency, actions with duration, knowledge of agents. Action languages have also provided the foundations for several successful approaches to automated planning; for example, the language $\mathcal{C}$ is used in the planner C-PLAN (Castellini et al., 2001) and the language $\mathcal{B}$ is used in CPA (Son et al., 2005). The *Planning Domain Definition Language (PDDL)* (Ghallab et al., 1998), the de-facto standard language for many planning systems, could also be viewed as a type of action language (a generalization of STRIPS). However, the primary focus of all these research efforts has been about reasoning within *single-agent domains.*

In single-agent domains, reasoning about actions and change mainly involves reasoning about what is true in the world, what the agent knows about the world, how the agent can manipulate the world (using world-changing actions) to reach particular states, and how the agent (using sensing actions) can learn unknown aspects of the world. In *multi-agent domains* an agent's action may not just change the world and the agent's knowledge about the world, but also may change other agents' knowledge. Similarly, the goals of an agent in a multi-agent world may involve manipulating the knowledge of other agents—in particular, this may involve not just their knowledge about the world, but also their knowledge about other agents' knowledge about the world. Although there is a large body of research on multi-agent planning (see, e.g., (Durfee, 1999; de Weerdt et al., 2003; de Weerdt and Clement, 2009; Allen and Zilberstein, 2009; Bernstein et al., 2002; Goldman and Zilberstein, 2004; Guestrin et al., 2001; Nair et al., 2003; Peshkin and Savova, 2002)), relatively few efforts address the above aspects of multi-agent domains, which offer a number of new research challenges in representing and reasoning about actions and change. The following simple example illustrates some of these issues.

Example 1 (Three Agents and the Coin Box)**.** *Three agents, A, B, and C, are in a room. In the middle of the room there is a box containing a coin. It is common knowledge that:*

- *None of the agents knows whether the coin lies heads or tails up;*

- *The box is locked and one needs a key to open it; agent A has the key of the box and everyone knows this;*
- *In order to learn whether the coin lies heads or tails up, an agent can peek into the box—but this requires the box to be open;*
- *If one agent is looking at the box and a second agent peeks into the box, then the first agent will observe this fact and will be able to conclude that the second agent knows the status of the coin; on the other hand, the first agent's knowledge about which face of the coin is up does not change;*
- *Distracting an agent causes that agent to not look at the box;*
- *Signaling an agent to look at the box causes such agent to look at the box;*
- *Announcing that the coin lies heads or tails up will make this a common knowledge among the agents that are listening.*

Suppose that the agent A would like to know whether the coin lies heads or tails up. She would also like to let the agent B know that she knows this fact. However, she would like to keep this information secret from C. Please, observe that the last two sentences express goals that are about agents' knowledge about other agents' knowledge. Intuitively, she could achieve her goals by:

1. *Distracting C from looking at the box;*
2. *Signaling B to look at the box if B is not looking at the box;*
3. *Opening the box; and*
4. *Peeking into the box.* □

This simple story presents a number of challenges for research in representing and reasoning about actions and their effects in multi-agent domains. In particular:

- The domain contains several types of actions:
  - Actions that allow the agents to change the state of the world (e.g., opening the box);
  - Actions that change the knowledge of the agents (e.g, peeking into the box, announcing heads/tails);
  - Actions that manipulate the beliefs of other agents (e.g., peeking while other agents are looking); and
  - Actions that change the observability of agents with respect to awareness about future actions (e.g., distract and signal actions before peeking into the box).

  We observe that the third and fourth types of actions are not considered in single agent systems.
- The reasoning process that allows agent A to verify that steps (1)-(4) will indeed achieve her goal requires A's ability to reason about the effects of actions on several aspects:
  - *The state of the world*—e.g., opening the box causes the box to become open;

- *The agents' awareness of the environment and of other agents' actions*—e.g., distracting or signaling an agent causes this agent not to look or to look at the box, respectively; and
- *The knowledge of other agents about her own knowledge*—e.g., someone following her actions would know what she knows.

While the first requirement is the same as for an agent in single-agent domains, the last two are specific to multi-agent domains.

With respect to planning, the above specifics of multi-agent systems raise an interesting problem:

> *"How can one generate a plan for the agent A to achieve her goal, given the description in Example 1?"*

The above problem is an *Epistemic Planning problem in a Multi-agent domain (EPM)* (Bolander and Andersen, 2011) which refers to the generation of plans for multiple agents to achieve goals which can refer to the state of the world, the beliefs of agents, and/or the knowledge of agents. EPM has recently attracted the attention of researchers from various communities such as planning, dynamic epistemic logic, and knowledge representation. The Dagstuhl seminars on the subject (Agotnes et al., 2014; Baral et al., 2017) provided the impetus for the development of several epistemic planners (Kominis and Geffner, 2015; Huang et al., 2017; Muise et al., 2015; Wan et al., 2015; Liu and Liu, 2018; Le et al., 2018) and extensive studies of the theoretical foundations (e.g., decidability and computational complexity) of EPM (Aucher and Bolander, 2013; Bolander et al., 2015). In spite of all these efforts, to the best of our knowledge, only two systems have been proposed that address the complete range of issues mentioned in Example 1: the dynamic epistemic modeling system called DEMO van Eijck (2004) and the recently proposed system described in Le et al. (2018). This is in stark contrast to the landscape of automated planning for single-agent domains, where we can find several efficient automated planners capable of generating plans consisting of hundreds of actions within seconds—especially building on recent advances in search-based planning.

Among the main reasons for the lack of planning systems capable of dealing with the issues like those shown in Example 1 are: **(i)** the lack of action-based formalisms that can address the above mentioned issues and that can actually be orchestrated, and **(ii)** the fact that logical approaches to reasoning about knowledge of agents in multi-agent domains are mostly model-theoretical, and not amenable to an implementation in search-based planning systems. Indeed, both issues were raised in the recent Dagstuhl seminar (Baral et al., 2017). The issue **(i)** is considered as one of the main research topics in EPM, while **(ii)** is related to the practical and conceptual knowledge representation challenges—discussed by Herzig[2] at the second Dagstuhl seminar (Baral et al., 2017). We will discuss these issues in more detail in the next sub-section.

[2] `http://materials.dagstuhl.de/files/17/17231/17231.AndreasHerzig.Slides.pdf`

1.2. Related Work

In terms of related work, multi-agent actions have been explored in *Dynamic Epistemic Logics (DEL)* (e.g., Baltag and Moss (2004); Herzig et al. (2005); van Benthem (2007); van Benthem et al. (2006); van Ditmarsch et al. (2007)). However, as discussed later in the paper, DEL does not offer an intuitive view of how to orchestrate or execute a single multi-agent action. In addition, the complex representation of multi-agent actions—similar to a Kripke structure—drastically increases the number of possible multi-agent actions—thus, making it challenging to adopt a search-based approach in developing multi-agent action sequences to reach a given goal. It can be observed that several approaches to epistemic planning in multi-agent domains with focus on knowledge and beliefs of agents did employ an extension of PDDL rather than using DEL (Kominis and Geffner, 2015; Huang et al., 2017; Muise et al., 2015; Wan et al., 2015; Liu and Liu, 2018).

The research in DEL has also not addressed some critical aspects of multi-agent search-based planning, such as the determination of the initial state of a planning domain instance. Moreover, research in DEL did not explore the link between the state of the world and the observability encoded in multi-agent actions, and hence preventing the dynamic evolution of the observational capabilities and awareness of the agents with respect to future actions. In some ways, the DEL approach is similar to the formulation of belief updates (e.g., (**?**Katsuno and Mendelzon, 1992; del Val A. and Shoham, 1994)), and most of the differences and similarities between belief updates and reasoning about actions carry over to the differences and similarities between DEL and our formulation of RAC in multi-agent domains. We will elaborate on these differences in a later section of the paper.

1.3. Contributions and Assumptions

Our goal in this paper is to develop a framework that allows reasoning about actions and their effects in a multi-agent domain; the framework is expected to address the above-mentioned issues, e.g., actions' capability to modify agents' knowledge and beliefs about other agents' knowledge and beliefs. To this end, we propose a high-level action language for representing and reasoning about actions in multi-agent domains. The language provides the fundamental components of a planning domain description language for multi-agent systems. The main contributions of the paper are:

- The action language m$\mathcal{A}^*$, which allows the representation of different types of actions—such as world-altering actions, announcement actions, and sensing actions—for formalizing multi-agent domains; the language explicitly supports actions that allow the dynamic modification of the awareness and observation capabilities of the agents;
- A transition function-based semantics for m$\mathcal{A}^*$, that enables hypothetical reasoning and planning in multi-agent domains. This, together with the notion of a *finitary*-**S5** *theories* for representing the initial state, introduced in (Son et al., 2014), provides a foundation for the implementation of a heuristic search-based planner for domains described in m$\mathcal{A}^*$; and
- Several theoretical results relating the semantics of m$\mathcal{A}^*$ to multi-agent actions characterizations using the notion of update models from DEL Baltag and Moss (2004).

In developing m$\mathcal{A}^*$, we make several design decisions. The key decision is that actions in our formalism can be effectively executed and the outcome can be effectively determined. This is not the

case, for example, in DEL (van Ditmarsch et al., 2007), where actions are complex graph structures, similar to Kripke structures, possibly representing a multi-modal formula, and it is not clear if and how such actions can be executed. We also assume that actions are deterministic, i.e., the result of the execution of a world altering action is unique. This assumption can be lifted in a relatively simple manner—by generalizing the techniques for handling non-deterministic actions studies in the context of single-agent domains.

Although we have mentioned both knowledge and beliefs, in this paper we will follow van Ditmarsch et al. (2007); Baltag and Moss (2004) and focus only on formalizing the changes of *beliefs* of agents after the execution of actions. Following the considerations by van Benthem (2007), the epistemic operators used in this paper can be read as *"to the best of my information."* Note that, in a multi-agent system, there may be a need to distinguish between the *knowledge* and the *beliefs* of an agent about the world. Let us consider Example 1 and let us denote with p the proposition *"nobody knows whether the coin lies heads or tails up."* Initially, the three agents know that p is true. However, after agent A executes the sequence of actions (1)-(4), A will know that p is false. Furthermore, B also knows that p is false, thanks to her awareness of A's execution of the actions of opening the box and looking into it. However, C, being unaware of the execution of the actions performed by A, will still believe that p is true. If this were considered as a part of C's knowledge, then C would result in having false knowledge.

The investigation of the discrepancy between knowledge and beliefs has been an intense research topic in dynamic epistemic logic and in reasoning about knowledge, which has lead to the development of several modal logics (e.g., (Fagin et al., 1995; van Ditmarsch et al., 2007)). Since our main aim is the development of an action language for hypothetical reasoning and planning, we will be primarily concerned with the beliefs of agents. Some preliminary steps in this direction have been explored in the context of the DEL framework (Herzig et al., 2005; Son et al., 2015). We leave the development of an action-based formalism that takes into consideration the differences between beliefs and knowledge as future work.

1.4. Paper Organization

The rest of the paper is organized as follows. Section 2 reviews the basics definitions and notation of a modal logic with belief operators and the update model based approach to reasoning about actions in multi-agent domains. This section also reviews the definition of finitary **S5**-theories whose models are finite. It also includes a short discussion for the development of m$\mathcal{A}^*$. Section 3 presents the syntax of m$\mathcal{A}^*$. Section 4 explores the modeling of the semantics of m$\mathcal{A}^*$ using the update models approach; we define the transition function of m$\mathcal{A}^*$ which maps pairs of actions and states into states; the section also presents the entailment relation between m$\mathcal{A}^*$ action theories and queries along with relevant properties. Section 5 provides an analysis of m$\mathcal{A}^*$ with respect to the existing literature, including a comparison with DEL. Section 6 provide some concluding remarks and directions for future work. For simplicity of presentation, the proofs of the main theorems are placed in Appendix A. Appendix B provides a set of m$\mathcal{A}^*$ examples.

2. Preliminaries

We begin with a review of the basic notions from the literature on formalizing knowledge and reasoning about effects of actions in multi-agent systems. Subsection 2.1 presents the notion of Kripke structures. Subsection 2.2 reviews the notion of update models developed by the dynamic epistemic logic community for reasoning about effects of actions in multi-agent systems.

2.1. Belief Formulae and Kripke Structures

Let us consider an environment with a set $\mathcal{AG}$ of n agents. The *real state of the world* (or *real state*, for brevity) is described by a set $\mathcal{F}$ of propositional variables, called *fluents*. We are concerned with the beliefs of agents about the world and about the beliefs of other agents. For this purpose, we adapt the logic of knowledge and the notations used in Fagin et al. (1995); van Ditmarsch et al. (2007). We associate to each agent $i \in \mathcal{AG}$ a modal operator $\mathbf{B}_i$ (to indicate a belief of agent i) and represent the beliefs of an agent as belief formulae in a logic extended with these operators. Formally, we define belief formulae as follows.

Fluent formulae: a fluent formula is a propositional formula built using the propositional variables in $\mathcal{F}$ and the traditional propositional operators $\vee$, $\wedge$, $\rightarrow$, $\neg$, etc. In particular, a *fluent atom* is a formula containing just an element $f \in \mathcal{F}$, while a *fluent literal* is either a fluent atom $f \in \mathcal{F}$ or its negation $\neg f$. We will use $\top$ and $\bot$ to denote *true* and *false*, respectively.

Belief formulae: a belief formula is a formula in one of the following forms:

- a fluent formula;
- a formula of the form $\mathbf{B}_i\varphi$ where φ is a belief formula;
- a formula of the form $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \Rightarrow \varphi_2$, or $\neg\varphi_1$ where φ_1, φ_2 are belief formulae;
- a formula of the form $\mathbf{E}_\alpha\varphi$ or $\mathbf{C}_\alpha\varphi$, where φ is a formula and $\emptyset \neq \alpha \subseteq \mathcal{AG}$.

Formulae of the form $\mathbf{E}_\alpha\varphi$ and $\mathbf{C}_\alpha\varphi$ are referred to as *group formulae.* Whenever $\alpha = \mathcal{AG}$, we simply write $\mathbf{E}\varphi$ and $\mathbf{C}\varphi$ to denote $\mathbf{E}_\alpha\varphi$ and $\mathbf{C}_\alpha\varphi$, respectively. Let us denote with $\mathcal{L}_{\mathcal{AG}}$ the language of the belief formulae over $\mathcal{F}$ and $\mathcal{AG}$.

Intuitively, belief formulae are used to describe the beliefs of one agent concerning the state of the world as well as about the beliefs of other agents. For example, the formula $\mathbf{B}_1\mathbf{B}_2 p$ expresses the fact that "*Agent 1 believes that agent 2 believes that p is true*," while $\mathbf{B}_1 f$ states that "*Agent 1 believes that f is true*."

In what follows, we will simply talk about "formulae" instead of "belief formulae," whenever there is no risk of confusion. In order to define the semantics of such logic language, we need to introduce the notion of a Kripke structure.

Definition 1 (Kripke Structure)**.** *A Kripke structure is a tuple $\langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_n\rangle$, where*

- *S is a set of worlds,*
- *$\pi : S \mapsto 2^{\mathcal{F}}$ is a function that associates an interpretation of $\mathcal{F}$ to each element of S, and*

- *For* $1 \leq i \leq n$, $\mathcal{B}_i \subseteq S \times S$ *is a binary relation over* S.

A pointed Kripke structure *is a pair* (M, s) *where* $M = \langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$ *is a Kripke structure and* $s \in S$. *In a pointed Kripke structure* (M, s)*, we refer to* s *as the* real *(or* actual*) world.*

For the sake of readability, we use $M[S]$, $M[\pi]$, and $M[i]$ to denote the components S, π, and $\mathcal{B}_i$ of M, respectively. We write $M[\pi](u)$ to denote the interpretation associated to u via π and $M[\pi](u)(\varphi)$ to denote the truth value of a fluent formula φ with respect to the interpretation $M[\pi](u)$. In keeping with the tradition of action languages, we will often refer to $M[\pi](u)$ as the set of fluent literals defined by[3]

$$\{f \mid f \in \mathcal{F}, M[\pi](u)(f) = \top\} \cup \{\neg f \mid f \in \mathcal{F}, M[\pi](u)(f) = \bot\}.$$

Given a consistent and complete set of literals X, i.e., $|\{f, \neg f\} \cap X| = 1$ for every $f \in \mathcal{F}$, we write $M[\pi](u) = X$ to indicate that the interpretation $M[\pi](u)$ is defined in such a way that $M[\pi](u) = X$.

Intuitively, a Kripke structure describes the possible worlds envisioned by the agents—and the presence of multiple worlds identifies uncertainty and the existence of different beliefs. The relation $(s_1, s_2) \in \mathcal{B}_i$ denotes that the belief of agent i about the real state of the world is insufficient for her to distinguish between the world described by s_1 and the one described by s_2. The world s in the state (M, s) identifies the world in $M[S]$ that corresponds to the actual world.

We will often view a Kripke structure $M = \langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$ as a directed labeled graph, whose set of nodes is S and whose set of edges contains (s, i, t)[4] if and only if $(s, t) \in \mathcal{B}_i$. (s, i, t) is referred to as an edge coming out of (resp. into) the world s (resp. t).

Following van Ditmarsch et al. (2007), we will refer to a pointed Kripke structure (M, s) as a *state* and often use these two terms interchangeably.

The satisfaction relation between belief formulae and a state is defined as next.

Definition 2. *Given a formula* φ*, a Kripke structure* $M = \langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_n \rangle$*, and a world* $s \in S$*:*

(i) $(M, s) \models \varphi$ *if* φ *is a fluent formula and* $M[\pi](s) \models \varphi$*;*

(ii) $(M, s) \models \mathbf{B}_i\varphi$ *if for each* t *such that* $(s, t) \in \mathcal{B}_i$*,* $(M, t) \models \varphi$*;*

(iii) $(M, s) \models \neg\varphi$ *if* $(M, s) \not\models \varphi$*;*

(iv) $(M, s) \models \varphi_1 \vee \varphi_2$ *if* $(M, s) \models \varphi_1$ *or* $(M, s) \models \varphi_2$*;*

(v) $(M, s) \models \varphi_1 \wedge \varphi_2$ *if* $(M, s) \models \varphi_1$ *and* $(M, s) \models \varphi_2$*.*

(vi) $(M, s) \models \mathbf{E}_\alpha\varphi$ *if* $(M, s) \models \mathbf{B}_i\varphi$ *for every* $i \in \alpha$*.*

(vii) $(M, s) \models \mathbf{C}_\alpha\varphi$ *if* $(M, s) \models \mathbf{E}^k_\alpha\varphi$ *for every* $k \geq 0$*, where*

[3] For simplicity of the presentation, we often omit the negative literals as well.

[4] (s, i, t) denotes the edge from node s to node t, labeled by i.

– $\mathbf{E}^0_\alpha\varphi = \varphi$ *and*
– $\mathbf{E}^{k+1}_\alpha = \mathbf{E}_\alpha(\mathbf{E}^k_\alpha\varphi)$.

For a Kripke structure M and a formula φ, $M \models \varphi$ denotes the fact that $(M, s) \models \varphi$ for each $s \in M[S]$.

$\models \varphi$ denotes the fact that $M \models \varphi$ for every Kripke structure M.

Example 2 (State)**.** *Let us consider a simplified version of Example 1 in which the agents are concerned only with the status of the coin. The three agents A, B, C do not know whether the coin has 'heads' or 'tails' up and this is a common belief. Let us assume that the coin is heads up. The beliefs of the agents about the world and about the beliefs of other agents can be captured by the state of Figure 1.*

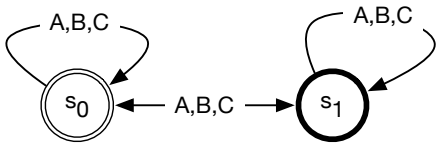

Figure 1: An example of a state

In the figure, a circle represents a world. The name of the world are written in the circle. Labeled edges between worlds denote the belief relations of the structure. A double circle identifies the real world. The interpretation of the world will be given whenever it is necessary. For example, we write $M[\pi](s_0) = \{head\}$ to denote that $head$ is true in the world s_0 and anything else is false.

We will occasionally be interested in Kripke structures that satisfy certain conditions. In particular, given a Kripke structure $M = \langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_n\rangle$ we identify the following properties:

- **K**: for each agent i and formulae φ, ψ, we have that $M \models (\mathbf{B}_i\varphi \wedge \mathbf{B}_i(\varphi \Rightarrow \psi)) \Rightarrow \mathbf{B}_i\psi$.
- **T**: for each agent i and formula ψ, we have that $M \models \mathbf{B}_i\psi \Rightarrow \psi$.
- **4**: for each agent i and formula ψ, we have that $M \models \mathbf{B}_i\psi \Rightarrow \mathbf{B}_i\mathbf{B}_i\psi$.
- **5**: for each agent i and formula ψ, we have that $M \models \neg\mathbf{B}_i\psi \Rightarrow \mathbf{B}_i\neg\mathbf{B}_i\psi$.
- **D**: for each agent i we have that $M \models \neg\mathbf{B}_i \bot$.

A Kripke structure is said to be a **T**-Kripke (**4**-Kripke, **K**-Kripke, **5**-Krikpe, **D**-Kripke, respectively) structure if it satisfies property **T** (**4**, **K**, **5**, **D**, respectively). A Kripke structure is said to be an **S5** structure if it satisfies the properties **K**, **T**, **4**, and **5**. The **S5** properties have been often used to capture the notion of knowledge. Consistency of a set of formulae is defined next.

Definition 3. *A set of belief formulae X is said to be* p-satisfiable *(or* p-consistent*) for $p \in \{\mathbf{S5}, \mathbf{K}, \mathbf{T}, \mathbf{4}, \mathbf{5}\}$ if there exists a p-Kripke structure M and a world $s \in M[S]$ such that $(M, s) \models \psi$ for every $\psi \in X$. In this case, (M, s) is referred to as a* p-model *of X.*

Finally, let us introduce a notion of equivalence between states.

Definition 4. *A state (M, s) is* equivalent *to a state (M', s') if $(M, s) \models \varphi$ iff $(M', s') \models \varphi$ for every formula $\varphi \in \mathcal{L}_{\mathcal{AG}}$.*

2.2. Update Models

The formalism of *update models* has been used to describe transformations of (pointed) Kripke structures according to a predetermined transformation pattern. An update model is structured similarly to a pointed Kripke structure and it describes how to transform a pointed Kripke structure using an update operator defined in (Baltag and Moss, 2004; van Benthem et al., 2006).

Let us start with some preliminary definitions. An $\mathcal{L}_{\mathcal{AG}}$-substitution is a set $\{p_1 \to \varphi_1, \ldots, p_k \to \varphi_k\}$, where each p_i is a distinct fluent in $\mathcal{F}$ and each $\varphi_i \in \mathcal{L}_{\mathcal{AG}}$. $SUB_{\mathcal{L}_{\mathcal{AG}}}$ denotes the set of all $\mathcal{L}_{\mathcal{AG}}$-substitutions.

Definition 5 (Update Model)**.** *Given a set $\mathcal{AG}$ of n agents, an* update model $\boldsymbol{\Sigma}$ *is a tuple $\langle \Sigma, R_1, \ldots, R_n, pre, sub \rangle$ where*

(i) Σ is a set, whose elements are called events*;*

(ii) each R_i is a binary relation on Σ;

(iii) $pre : \Sigma \to \mathcal{L}_{\mathcal{AG}}$ is a function mapping each event $e \in \Sigma$ to a formula in $\mathcal{L}_{\mathcal{AG}}$; and

(iv) $sub : \Sigma \to SUB_{\mathcal{L}_{\mathcal{AG}}}$ is a function mapping each event $e \in \Sigma$ to a substitution in $SUB_{\mathcal{L}_{\mathcal{AG}}}$.

An update instance *ω is a pair $(\boldsymbol{\Sigma}, e)$ where $\boldsymbol{\Sigma}$ is an update model $\langle \Sigma, R_1, \ldots, R_n, pre, sub \rangle$ and e, referred to as a* designated event*, is a member of Σ.*

Intuitively, an update model represents different views of an action occurrence which are associated with the observability of agents. Each view is represented by an event in Σ. The designated event is the one that agents who are aware of the action occurrence will observe. The relation R_i describes agent i's uncertainty on action execution—i.e., if $(\sigma, \tau) \in R_i$ and event σ is performed, then agent i may believe that event τ is executed instead. pre defines the action precondition and sub specifies the changes of fluent values after the execution of an action.

Definition 6 (Updates by an Update Model)**.** *Let M be a Kripke structure and $\boldsymbol{\Sigma} = \langle \Sigma, R_1, \ldots, R_n, pre, sub \rangle$ be an update model. The* update operator *induced by $\boldsymbol{\Sigma}$ defines a Kripke structure $M' = M \otimes \boldsymbol{\Sigma}$, where:*

*(*i*) $M'[S] = \{(s, \tau) \mid s \in M[S], \tau \in \Sigma, (M, s) \models pre(\tau)\}$;*

*(*ii*) $((s, \tau), (s', \tau')) \in M'[i]$ iff $(s, \tau), (s', \tau') \in M'[S]$, $(s, s') \in M[i]$ and $(\tau, \tau') \in R_i$;*

*(*iii*) For all $(s, \tau) \in M'[S]$ and $f \in \mathcal{F}$, $M'[\pi]((s, \tau)) \models f$ iff $f \to \varphi \in sub(\tau)$ and $(M, s) \models \varphi$.*

The structure M' is obtained from the component-wise cross-product of the old structure M and the update model $\boldsymbol{\Sigma}$, by **(i)** removing pairs (s, τ) such that (M, s) does not satisfy the action precondition (checking for satisfaction of action's precondition), and **(ii)** removing links of the form $((s, \tau), (s', \tau'))$ from the cross product of $M[i]$ and R_i if $(s, s') \notin M[i]$ or $(\tau, \tau') \notin R_i$ (ensuring that each agent's accessibility relation is updated according to the update model).

An *update template* is a pair $(\boldsymbol{\Sigma}, \Gamma)$, where $\boldsymbol{\Sigma}$ is an update model with the set of events Σ and $\Gamma \subseteq \Sigma$. The update of a state (M, s) given an update template $(\boldsymbol{\Sigma}, \Gamma)$ is a set of states, denoted by $(M, s) \otimes (\boldsymbol{\Sigma}, \Gamma)$, where

$$(M, s) \otimes (\boldsymbol{\Sigma}, \Gamma) = \{(M \otimes \boldsymbol{\Sigma}, (s, \tau)) \mid \tau \in \Gamma, (M, s) \models pre(\tau)\}$$

Remark 1. *In the following, we will often represent an update instance by a graph with rectangles, double rectangles, and labeled links between rectangles representing events, designated events, and the relation of agents, respectively, as in the graphical representation of a Kripke structure.*

2.3. Finitary **S5***-Theories*

A finitary **S5**-theory, introduced in (Son et al., 2014), is a collection of formula which has finitely many **S5**-models, up to a notion of equivalence (Definition 4). To define finitary **S5**-theories, we need the following notion. Given a set of propositions $\mathcal{F}$, a *complete clause* over $\mathcal{F}$ is a disjunction of the form $\bigvee_{p\in\mathcal{F}} p^*$ where p^* is either p or $\neg p$. We will consider formulae of the following forms:

$$\varphi \tag{1}$$
$$\mathbf{C}(\mathbf{B}_i\varphi) \tag{2}$$
$$\mathbf{C}(\mathbf{B}_i\varphi \vee \mathbf{B}_i\neg\varphi) \tag{3}$$
$$\mathbf{C}(\neg\mathbf{B}_i\varphi \wedge \neg\mathbf{B}_i\neg\varphi) \tag{4}$$

where φ is a fluent formula.

Definition 7. *A theory T is said to be* primitive finitary **S5** *if*

- *Each formula in T is of the form (1)-(4); and*
- *For each complete clause φ over $\mathcal{F}$ and each agent i, T contains either* **(i)** $\mathbf{C}(\mathbf{B}_i\varphi)$ *or* **(ii)** $\mathbf{C}(\mathbf{B}_i\varphi \vee \mathbf{B}_i\neg\varphi)$ *or* **(iii)** $\mathbf{C}(\neg\mathbf{B}_i\varphi \wedge \neg\mathbf{B}_i\neg\varphi)$.

A theory T is a finitary **S5**-theory *if $T \models H$ and H is a primitive finitary* **S5***-theory.*
T is pure *if T contains only formulae of the form* (1)-(4).

We say that a state (M, s) is *canonical* if for every pairs of worlds $u, v \in M[S]$ and $u \neq v$, $M[\pi](u) \not\equiv M[\pi](v)$ holds. We have that

Theorem 1 (From (Son et al., 2014))**.** *Every finitary* **S5***-theory T has finitely many finite canonical models, up to equivalence. If T is pure then these models are minimal and their structures are identical up to the name of the points.*

2.4. Why an Action Language?

As mentioned earlier, the Dagstuhl seminars (Agotnes et al., 2014; Baral et al., 2017) identified one of the main research topics in EPM: *the development of an adequate specification language for EPM*. This problem arises from the fact that EPM has been defined and investigated using a DEL based approach, in which actions are represented by update models (Subsection 2.2). This representation has been useful for the understanding of EPM and the study of its complexity, but comes with a significant drawback—practical and conceptual knowledge representation challenges—discussed by Herzig[5] at the Dagstuhl seminar Baral et al. (2017). Let us consider a slight modification

[5] http://materials.dagstuhl.de/files/17/17231/17231.AndreasHerzig.Slides.pdf

of Example 1, where the box is open, A has looked at the coin, while both B and C are distracted, and A can announce whether the coin lies heads up or tails up. However, only agents who are attentive (to A) could listen to what A says. Assume that A announces that the coin lies heads up. Intuitively, this action occurrence can have different effects on the beliefs of the other agents—depending on the context and the specific features of each of them, e.g., whether the agent is attentive to A. As a result, we need a variety of update models to represent this primitive action. Herzig refers to this problem as the *action type vs. action token* problem.

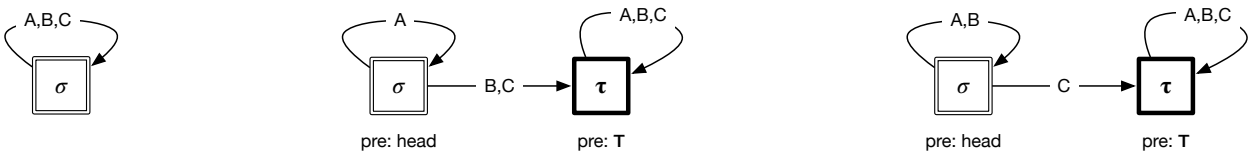

Figure 2: Update models for announcing *"the coin lies heads up"* by A in different situations

Fig. 2 shows three update models; they describe the occurrence of the announcement by A, stating that the coin lies heads up, assuming that the coin indeed lies heads up in the real state of the world. On the left is the update model when both B and C are attentive. The model in the middle depicts the situation when both B nor C are not attentive. The update model on the right captures the case of B being attentive and C being not attentive. In the figures, σ and τ are events and σ is a *designated event,* $head$ is a propositional variable denoting that the coin lies heads up.

Observe that these models are used only when the coin indeed lies heads up. The update models corresponding to the situation where $head$ is false (i.e., when A makes a false announcement) in the real state of the world are different from those in the figure and are omitted. It is easy to see that the number of update models needed to represent such simple announcement of *"the coin lies heads up"* by A is exponential in the number of agents. This is certainly an undesirable consequence of using update models and epistemic actions for representing and reasoning about effects of actions in multi-agent domains. Therefore, any specification language for representing and reasoning about the effects of actions in multi-agent domains should consider that the announcement of the coin lies heads up by A is simply a *primitive action*. In our view, the update models should be derived from the concrete state, which is a combination of the real state of the world and the state of beliefs of the agents, and not specified directly. A more detailed discussion on this issue can be found in the related work section.

3. The language m$\mathcal{A}^*$: Syntax

In this paper we consider multi-agent domains in which the agents are truthful and no false information may be announced or observed. Furthermore, the underlying assumptions guiding the semantics of our language are the rationality principle and the idea that beliefs of an agent are inertial. In other words, agents believe nothing which they are not forced to believe, and the beliefs of an agent remain the same unless something causes them to change. [1]

In this section and the next section, we introduce the language m$\mathcal{A}^*$ for describing actions and their effects in multi-agent environment. The language is built over a signature $\langle \mathcal{AG}, \mathcal{F}, \mathcal{A} \rangle$, where

[1] need to see if inertial really holds — or need to define what is inertial in MAD?

$\mathcal{AG}$ is a finite set of agent identifiers, $\mathcal{F}$ is a set of fluents, and $\mathcal{A}$ is a set of actions. Each action in $\mathcal{A}$ is an action the agents in the domain are capable of performing.

Similar to any action language developed for single-agent environments, m$\mathcal{A}^*$ consists of three components which will be used in describing the actions and their effects, the initial state, and the query language (see, e.g., (Gelfond and Lifschitz, 1998)). We will next present each of these components. Before we do so, let us denote the multi-agent domain in Example 1 by D_1. For this domain, we have that $\mathcal{AG} = \{A, B, C\}$. The set of fluents $\mathcal{F}$ for this domain consists of:

- $tail$: the coin lies tails up ($head$ is often used in place of $\neg tail$);
- $has\_key(x)$: agent x has the key of the box;
- $opened$: the box is open; and
- $looking(x)$: agent x is looking at the box.

The set of actions for D_1 consists of:

- $open$: an agent opens the box;
- $peek$: an agent peeks into the box;
- $signal(y)$: an agent signals agent y (to look at the box);
- $distract(y)$: an agent distracts agent y (so that y does not look at the box); and
- $shout\_tail$: an agent announces that the coin lies tail up.

where $x, y \in \{A, B, C\}$. We start with the description of actions and their effects.

3.1. Actions and effects

We envision three types of actions that an agent can perform: *world-altering actions* (also known as *ontic actions*), *sensing actions*, and *announcement actions*. Intuitively,

- A world-altering action is used to explicitly modify certain properties of the world—e.g., the agent A opens the box in Example 1, or the agent A distracts the agent B so that B does not look at the box (also in Example 1);
- A sensing action is used by an agent to refine its beliefs about the world, by making direct observations—e.g., an agent peeks into the box; the effect of the sensing action is to reduce the amount of uncertainty of the agent;
- An announcement action is used by an agent to affect the beliefs of the agents receiving the communication—we operate under the assumption that agents receiving an announcement always believe what is being announced.

For the sake of simplicity, we assume that each action $a \in \mathcal{A}$ falls in exactly one of the three categories.[6] In a multi-agent systems, we need to identify an action occurrence with the agents who execute it. For $a \in \mathcal{A}$ and $\alpha \subseteq \mathcal{AG}$, we write a to denote the joint-execution of a by the agents in α and call it an *action instance*. We will use $\mathcal{AI}$ to denote the set of possible action instances $\mathcal{A} \times \mathcal{AG}$. Elements of $\mathcal{AI}$ will be written in san-serif font. For simplicity of the presentation, we often use "an action" or "an action instance" interchangeably when it is clear from the context which term is appropriate. Furthermore, when α is a singleton set, $\{x\}$, we simplify $\langle\{x\}\rangle$ to $\langle x\rangle$.

In general, an action can be executed only under certain conditions, called its *executability conditions*. For example, the statement "to open the box, an agent must have its key" in Example 1 describes the executability condition of the action of opening a box. The first type of statements in m$\mathcal{A}^*$ is used to describe the executability conditions of action occurrences and is of the following form:

$$\textbf{executable } \mathsf{a} \textbf{ if } \psi \tag{5}$$

where $\mathsf{a} \in \mathcal{AI}$ and ψ is a belief formula. A statement of type (5) will be referred to as the *executability condition of the action occurrence* a. ψ is referred as the *precondition of* a. For simplicity of the presentation, we will assume that each action occurrence a is associated with exactly one executability condition. When $\psi = \top$, the statement will be omitted.

For an occurrence of a world-altering action a, such as the action of opening the box by some agent, we have statements of the following type that express the change that may be caused by such action:

$$\mathsf{a} \textbf{ causes } \ell \textbf{ if } \psi \tag{6}$$

where ℓ is a fluent literal, $x \subseteq \mathcal{AG}$, and ψ is a belief formula. Intuitively, if the real state of the world and of the beliefs match the condition described by ψ, then the real state of the world is affected by the change that makes the literal ℓ true after the execution of a. When $\psi = \top$, the part "**if** ψ" will be omitted from (6). We also use

$$\mathsf{a} \textbf{ causes } \phi \textbf{ if } \psi$$

where ϕ is a set of fluent literals as a shorthand for the set $\{\mathsf{a} \textbf{ causes } \ell \textbf{ if } \psi \mid \ell \in \phi\}$.

Sensing actions, such as the action of looking into the box, allow agents to learn about the value of a fluent in the real state of the world (e.g., learn whether the coin lies head or tail up). We use statements of the following kind to represent effects of sensing action occurrences:

$$\mathsf{a} \textbf{ determines } f \tag{7}$$

where f is a fluent and $\mathsf{a} \in \mathcal{AI}$ is a sensing action. Statements of type (7) encode the occurrence of a sensing action a which enables the agent(s) to learn the value of the fluent f. f is referred to as a *sensed fluent* of a.

For actions such as the action of an agent telling another agent that the coin lies head up, we have statements of the following kind, that express the change that may be caused by the occurrences of such actions:

[6] It is easy to relax this condition, but it would make the presentation more tedious.

$$\mathsf{a} \textbf{ announces } \varphi \tag{8}$$

where φ is a fluent formula and $\mathsf{a} \in \mathcal{AI}$. a is called *an announcement action*.

We will next illustrate the use of (5)-(8) in representing the actions of the domain D_1.

Example 3. The actions of domain D_1 can be specified by the following statements:

executable $open\langle x\rangle$ **if** $has\_key(x)$
executable $peek\langle x\rangle$ **if** $opened, looking(x)$
executable $shout\_tail\langle x\rangle$ **if** $\mathbf{B}_x(tail), tail$
executable $signal(y)\langle x\rangle$ **if** $looking(x), \neg looking(y)$
executable $distract(y)\langle x\rangle$ **if** $looking(x), looking(y)$

$open\langle x\rangle$ **causes** $opened$
$signal(y)\langle x\rangle$ **causes** $looking(y)$
$distract(y)\langle x\rangle$ **causes** $\neg looking(y)$
$peek\langle x\rangle$ **determines** $tail$
$shout\_tail\langle x\rangle$ **announces** $tail$

where x and y are different agents in $\{A, B, C\}$. The first five statements encode the executability conditions of the five actions in the domain. The next three statements describe the effects of the occurrence of three world-altering actions. $peek\langle x\rangle$ is an example of an occurrence of a sensing action. Finally, $shout\_tail\langle x\rangle$ is an example of an occurrence of an announcement action.

3.2. Observability: observers, partial observers, and others

One of the key differences between single-agent and multi-agent domains lies in how the execution of an action changes the beliefs of agents. This is because, in multi-agent domains, an agent might be oblivious about the occurrence of an action or unable to observe the effect of an action. For example, watching another agent opening the box would allow the agent to know that the box is open after the execution of the action; however, the agent would still believe that the box is closed if she is not aware of the action occurrence. On the other hand, watching another agent peeking into the box does not help the observer in learning whether the coin lies heads or tails up; the only thing she would learn is that the agent who is peeking into the box has knowledge of the status of the coin.

m$\mathcal{A}^*$ needs to have a component for representing the fact that not all the agents may be completely aware of the presence of actions being executed. Depending on the action and the current situation, we can categorize agents in three classes:

- *Full observers*,
- *Partial observers*, and
- *Oblivious* (or *others*).

action type	full observers	partial observers	oblivious/others
world-altering actions	$\surd$		$\surd$
sensing actions	$\surd$	$\surd$	$\surd$
announcement actions	$\surd$	$\surd$	$\surd$

Table 1: Action types and agent observability

This categorization is dynamic, as with the change in the state an agent's category may change. In this paper, we will consider the possible observabilities of agents for different action types detailed in Table 1.

The first row indicates that, for a world-altering action, an agent can either be a *full observer*, i.e., completely aware of the occurrence of that action, or oblivious of the occurrence of the action. In the second case, the observability of the agent is categorized as *other*. Note that we assume that the observer agents know about each others' status and they are also aware of the fact that the other agents are oblivious. The oblivious agents have no clue of anything.

For a sensing action, an agent can either be a *full observer*, i.e., it is aware of the occurrence of that action and of its results, it can be a *partial observer*, gaining knowledge that the full observers have performed a sensing action but without knowledge of the result of the observation, or it can be oblivious of the occurrence of the action (i.e., *other*). Once again, we assume that the observer agents know about each others' status and they also know about the agents partially observing the action and about the agents that are oblivious. The partially observing agents know about each others' status, and they also know about the observing agents and the agents that are oblivious. The oblivious agents have no clue of anything. The behavior is analogous for the case of announcement actions.

The dynamic nature of the agents observability can be manipulated and this is specified using agent observability statements of the following form:[7]

$$z \textbf{ observes } \mathsf{a} \textbf{ if } \varphi \tag{9}$$

$$z \textbf{ aware\_of } \mathsf{a} \textbf{ if } \psi \tag{10}$$

where $z \in \mathcal{AG}$, $\mathsf{a} \in \mathcal{AI}$, and φ and ψ are fluent formulae. (9) indicates that agent z is a full observer of a if φ holds. (10) states that agent z is a partial observer of a if ψ holds. z, a, and φ (resp. ψ) are referred to as the observed agent, the action instance, and the condition of (9) (resp. (10)). The next example illustrates the use of the above statements in specifying the agents observability of the domain D_1.

Example 4 (Observability in D_1)**.** *The actions of D_1 are described in Example 3. The observability*

[7] As discussed earlier, the " **if** $\top$" are omitted from the statements.

of agents in D_1 can be described by the set O_1 of statements

x **observes** $open\langle x\rangle$	x **observes** $peek\langle x\rangle$
y **observes** $open\langle x\rangle$ **if** $looking(y)$	y **aware_of** $peek\langle x\rangle$ **if** $looking(y)$
y **observes** $shout\_tail\langle x\rangle$	
x **observes** $distract(y)\langle x\rangle$	x **observes** $signal(y)\langle x\rangle$
y **observes** $distract(y)\langle x\rangle$	y **observes** $signal(y)\langle x\rangle$

where x and y denote different agents in $\{A, B, C\}$. The above statements say that agent x is a fully observant agent when $open\langle x\rangle$, $peek\langle x\rangle$, $distract(y)\langle x\rangle$, $signal(y)\langle x\rangle$, or $shout\_tail\langle x\rangle$ are executed; y is a fully observant agent if it is looking (at the box) when $open\langle x\rangle$ is executed. y is a partially observant agent if it is looking when $peek\langle x\rangle$ is executed. An agent different from x and y is oblivious in all cases.

In the following, we will assume that for every agent $z \in \mathcal{AG}$, if z is an observed agent in a statement of the form (9) then it is not an observed agent in a statement of the form (10) for the same occurrence a such that $\varphi \wedge \psi$ is consistent and vice versa.

Definition 8. *An m$\mathcal{A}^*$* domain *is a collection of statements of the forms (5)-(10).*

Similar to action domains in the language $\mathcal{A}$ introduced by Gelfond and Lifschitz (1993), an m$\mathcal{A}^*$ domain could contain two statements specifying contradictory effects of an action occurrence such as

$$\mathsf{a} \textbf{ causes } f \textbf{ if } \varphi \qquad \text{and} \qquad \mathsf{a} \textbf{ causes } \neg f \textbf{ if } \psi$$

where $\varphi \wedge \psi$ is a consistent formula, i.e., there exists some pointed Kripke structure (M, s) such that $(M, s) \models \varphi \wedge \psi$. Such a domain is not sensible and will be characterized as *inconsistent*. We define

Definition 9. *An m$\mathcal{A}^*$* domain D *is* consistent *if for every pointed Kripke structure (M, s) and two statements*

$$\mathsf{a} \textbf{ causes } f \textbf{ if } \varphi \qquad \text{and} \qquad \mathsf{a} \textbf{ causes } \neg f \textbf{ if } \psi$$

in D, $(M, s) \not\models \varphi \wedge \psi$.

From now on, whenever we say an m$\mathcal{A}^*$ domain D, we will assume that D is consistent.

3.3. Initial State

A domain specification encodes the actions and their effects and the observability of agents in each situation. The initial state, that encodes both the initial state of the world and the initial beliefs of the agents, is specified in m$\mathcal{A}^*$ using *initial statements* of the following form:

$$\textbf{initially } \varphi \tag{11}$$

where φ is a formula. Intuitively, this statement says that φ is true in the initial state. We will later discuss restrictions on the formula φ to ensure the computability of the Kripke structures describing the initial state.

Example 5 (Representing Initial State of D_1)**.** Let us reconsider Example 1. The initial state of D_1 can be expressed by the following statements:

$$\begin{array}{ll}
\textbf{initially } \mathbf{C}(has_key(A)) & \\
\textbf{initially } \mathbf{C}(\neg has_key(B)) & \\
\textbf{initially } \mathbf{C}(\neg has_key(C)) & \\
\textbf{initially } \mathbf{C}(\neg opened) & \\
\textbf{initially } \mathbf{C}(\neg \mathbf{B}_x tail \wedge \neg \mathbf{B}_x \neg tail) & \text{for } x \in \{A, B, C\} \\
\textbf{initially } \mathbf{C}(looking(x)) & \text{for } x \in \{A, B, C\}
\end{array}$$

These statements indicate that everyone knows that A has the key and B and C do not have the key, the box is closed, no one knows whether the coin lies head or tail up, and everyone is looking at the box.

The notion of an action theory in m$\mathcal{A}^*$ is defined next.

Definition 10 (Action Theory)**.** *An m$\mathcal{A}^*$-*action theory *is a pair (I, D) where D is an m$\mathcal{A}^*$ domain and I is a set of statements of the form* (11).

In Section 4, we will define the notion of entailment between action theories and queries, similar to the notion of entailment defined for action languages in single agent domains (e.g., (Gelfond and Lifschitz, 1998)). This relies on the following definition.

Definition 11 (Initial State/b-State)**.** *Let (I, D) be an action theory. An* initial state *of (I, D) is a state (M, s) such that for every statement*

$$\textbf{initially } \varphi$$

in I, $(M, s) \models \varphi$.

(M, s) is an initial **S5***-state if it is an initial state and M is a* **S5** *Kripke structure.*

The initial belief-state *(or* initial b-state*) of (I, D) is the collection of all initial states of (I, D).*

The initial **S5**-b-state *of (I, D) is the collection of all initial* **S5***-states of (I, D).*

Although Definition 11 is well-defined, it is easy to see that theoretically, there could be infinitely many initial states for an arbitrary m$\mathcal{A}^*$ theory. For example, given a state (M, s) and a set of formulae Σ such that $(M, s) \models \Sigma$, a new state (M', s) that also satisfies Σ can be constructed from M by adding a new world and keeping everything else unchanged. As such, identifying sensible classes of action theories whose initial belief states are finite, up to a notion of equivalence (Definition 4). Fortunately, the result on finitary **S5** theories[8] (Definition 7) allows us to identify a large class of action theories satisfying this property. We call them definite action theories and define them as follows.

[8] To keep the paper at a reasonable length, we do not discuss the details of finitary **S5** theories. Interested readers are referred to (Son et al., 2014) for details and proof of Theorem 1.

Definition 12 (Definite Action Theory)**.** *An action theory* (I, D) *is said to be* definite *if the theory* $\{\varphi \mid$ **initially** φ *belongs to* $I\}$ *is a finitary-***S5** *theory.*

Observe that Theorem 1 indicates that for definite action theories, the initial belief state is finite. An algorithm for computing the initial belief state is given in (Son et al., 2014). This, together with the definition of the transition function of m$\mathcal{A}^*$ domains in the next section, allows for the implementation of progression-based epistemic planning systems. A preliminary development can be found in (Le et al., 2018).

4. Update Model Based Semantics for m$\mathcal{A}^*$ Domains

An m$\mathcal{A}^*$ domain D specifies a transition system, whose nodes are "states" that encode the description of the state of the world and of the agents' beliefs. This transition system will be described by a transition function Φ_D, which maps pairs of action occurrences and states to states. For simplicity of the presentation, we assume that only one action occurrence happens at each point in time—it is relatively simple to extend it to cases where concurrent actions are present, and this is left as future work. As we have mentioned in Section 2, we will use pointed Kripke structures to represent states in m$\mathcal{A}^*$ action theories. A pointed Kripke structure encodes three components:

- The actual world;
- The state of beliefs of each agent about the real state of the world; and
- The state of beliefs of each agent about the beliefs of other agents.

These components are affected by the execution of actions. Observe that the notion of a state in m$\mathcal{A}^*$ action theories is more complex than the notion of state used in single-agent domains (i.e., a complete set of fluent literals).

Let $\mathcal{S}$ be the set of all possible pointed Kripke structures over $\mathcal{L}(\mathcal{F}, \mathcal{AG})$, the transition function Φ_D maps pairs of action instances and states into sets of states, i.e.,

$$\Phi_D : \mathcal{AI} \times \mathcal{S} \longrightarrow 2^{\mathcal{S}}$$

will be defined for each action type separately and in two steps. First, we define an update model representing the occurrence of a in a state (M, s). Second, we use the update model/instance defined in step one and an updated instance obtained from step one in defining Φ_D. We start by defining the notion of a frame of reference in order to define the function Φ_D.

4.1. Actions Visibility

Given a state (M, s) and an action occurrence a, let us define

$$\begin{aligned} F_D(\mathsf{a}, M, s) &= \{x \in \mathcal{AG} \mid [x \textbf{ observes } \mathsf{a} \textbf{ if } \varphi] \in D \text{ such that } (M, s) \models \varphi\} \\ P_D(\mathsf{a}, M, s) &= \{x \in \mathcal{AG} \mid [x \textbf{ aware\_of } \mathsf{a} \textbf{ if } \varphi] \in D \text{ such that } (M, s) \models \varphi\} \\ O_D(\mathsf{a}, M, s) &= \mathcal{AG} \setminus (F_D(\mathsf{a}, M, s) \cup P_D(\mathsf{a}, M, s)) \end{aligned}$$

We will refer to the tuple $(F_D(\mathsf{a}, M, s), P_D(\mathsf{a}, M, s), O_D(\mathsf{a}, M, s))$ as the *frame of reference* for the execution of a in (M, s). Intuitively, $F_D(\mathsf{a}, M, s)$ (resp. $P_D(\mathsf{a}, M, s)$ and $O_D(\mathsf{a}, M, s)$) are the agents that are fully observant (resp. partially observant and oblivious/other) of the execution of a in the state (M, s). As we assume that for each pair of an action occurrence a and a state (M, s), the sets $F_D(\mathsf{a}, M, s)$ and $P_D(\mathsf{a}, M, s)$ are disjoint, the domain specification D and the state (M, s) determine a unique frame of reference for each action occurrence.

The introduction of frames of reference allows us to elegantly model several types of actions that are aimed at modifying the frame of reference (referred to as *reference setting actions*). Some possibilities are illustrated in the following examples.

Example 6 (Reference Setting Actions)**.** *Example 4 shows two reference setting actions: signal$(y)\langle x\rangle$ and distract$(y)\langle x\rangle$. The action signal$(y)\langle x\rangle$ allows agent x to promote agent y into a higher level of observation for the effect of the action peek$\langle x\rangle$. On the other hand, the action distract$(y)\langle x\rangle$ allows agent x to demote agent y into a lower level of observation. The net effect of executing these actions is a change of frame.*

Let us consider the action signal$(y)\langle x\rangle$ and a state (M, s). Furthermore, let us assume that (M', s') is a state resulting from the execution of signal$(y)\langle x\rangle$ in (M, s). The frames of reference for the execution of the action $a = peek\langle x\rangle$ in these two states are detailed in Figure 3.

in (M, s)	$\xrightarrow{signal(y)\langle x\rangle}$	*in* (M', s')
$F_D(a, M, s)$		$F_D(a, M, s)$
$P_D(a, M, s)$		$P_D(a, M, s) \cup \{y\}$
$O_D(a, M, s)$		$O_D(a, M, s) \setminus \{y\}$

Figure 3: Frame of reference for $peek\langle x\rangle$ before (in (M, s)) and after $signal(y)\langle x\rangle$ (in (M', s'))

2 the above needs to be checked - the right column should be M',s' instead of M,s

2*Intuitively, after the execution of $signal(y)\langle x\rangle$, $looking(y)$ becomes true because of the statement*

$$signal(y)\langle x\rangle \textbf{ causes } looking(y)$$

in D_1. By definition, the statement

$$y \textbf{ aware\_of } peek\langle x\rangle \textbf{ if } looking(y)$$

indicates that y is partially observant.

Similar argument shows that $distract(y)\langle x\rangle$ demotes y to the lowest level of visibility, i.e., it will cause agent y to become oblivious of the successive peek$\langle x\rangle$ action. Again, for $a = peek\langle x\rangle$, we have the change of frame of reference summarized in Figure 4.

4.2. Update Model for Action Occurrences

Definition 13 (Update Model/Instance for World-Altering Actions)**.** *Given a world-altering action instance* a *with the precondition ψ and a frame of reference $\rho = (F, \emptyset, O)$, the update model for* a *and ρ, denoted by $\omega(\mathsf{a}, \rho)$, is defined by $\langle \Sigma, R_1, \ldots, R_n, pre, sub\rangle$ where*

- $\Sigma = \{\sigma, \epsilon\}$;

in (M, s)	$\xrightarrow{distract(y)\langle x\rangle}$	*in* (M', s')
$F_D(a, M, s)$		$F_D(a, M, s) \setminus \{y\}$
$P_D(a, M, s)$		$P_D(a, M, s) \setminus \{y\}$
$O_D(a, M, s)$		$O_D(a, M, s) \cup \{y\}$

Figure 4: Frame of reference for $peek\langle x\rangle$ before (in (M, s)) and after $distract(y)\langle x\rangle$ (in (M', s'))

- $R_i = \{(\sigma, \sigma), (\epsilon, \epsilon)\}$ *for* $i \in F$ *and* $R_i = \{(\sigma, \epsilon), (\epsilon, \epsilon)\}$ *for* $i \in O$*;*
- $pre(\sigma) = \psi$ *and* $pre(\epsilon) = \top$*; and*
- $sub(\epsilon) = \emptyset$ *and* $sub(\sigma) = \{p \to \Psi^+(p, \mathsf{a}) \vee (p \wedge \neg\Psi^-(p, \mathsf{a})) \mid p \in \mathcal{F}\}$*, where*

$$\Psi^+(p, \mathsf{a}) = \bigvee\{\varphi \mid [\mathsf{a} \textbf{ causes } p \textbf{ if } \varphi] \in D\}$$

and

$$\Psi^-(p, \mathsf{a}) = \bigvee\{\varphi \mid [\mathsf{a} \textbf{ causes } \neg p \textbf{ if } \varphi] \in D\}.$$

The update instance for the occurrence of a *and the frame of reference* ρ *is* $(\omega(\mathsf{a}, \rho), \{\sigma\})$.

Observe that the update model of the world-altering action occurrence a has at most two events. Each event corresponds to a group of agents. The links in the update model for each group of agents reflect the state of beliefs each group would have after the execution of the action. For example, fully observant agents (in F) will have no uncertainty. The next example illustrates this definition.

Example 7. *Going back to our original example, the action instance* $open\langle A\rangle$ *assumes that everyone is aware that* C *is not looking at the box while* B *and* A *are. Figure 5 (top right) depicts the state. For simplicity, in the worlds we report only the components of the interpretation related to the opened and tail fluents. The frame of reference for* $open(A)$ *in this situation is* $(\{A, B\}, \emptyset, \{C\})$*. The corresponding update instance for* $open\langle A\rangle$ *and the frame of reference* $(\{A, B\}, \emptyset, \{C\})$ *is given in Figure 5 (top left). The bottom part of Figure 5 shows the result of the application of the update instance to the state on the top right.*

In the next definition, we provide the update instance for a sensing action occurrence given a frame of reference. For simplicity of presentation, we will assume that the set of sensed fluents of the action is a singleton.

Definition 14 (Update Model/Instance for Sensing Actions)**.** *Let* a *be a sensing action instance with* $Sensed_D(\mathsf{a}) = \{f\}$*, let its precondition be* ψ*, and let* $\rho = (F, P, O)$ *be a frame of reference. The update model for* a *and* ρ*,* $\omega(\mathsf{a}, \rho)$*, is defined by* $\langle \Sigma, R_1, \ldots, R_n, pre, sub\rangle$ *where:*

- $\Sigma = \{\sigma, \tau, \epsilon\}$*;*
- R_i *is given by*

$$R_i = \begin{cases} \{(\sigma, \sigma), (\tau, \tau), (\epsilon, \epsilon)\} & \text{if } i \in F \\ \{(\sigma, \sigma), (\tau, \tau), (\epsilon, \epsilon), (\sigma, \tau), (\tau, \sigma)\} & \text{if } i \in P \\ \{(\sigma, \epsilon), (\tau, \epsilon), (\epsilon, \epsilon)\} & \text{if } i \in O \end{cases}$$

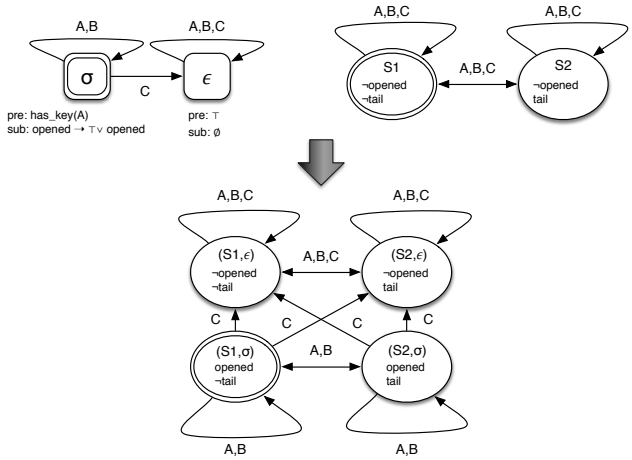

Figure 5: Update Instance $(\omega(\mathit{open}\langle A\rangle, (\{A, B\}, \emptyset, \{C\})), \{\sigma\})$ and its application

- *The preconditions* pre *are defined by*

$$pre(x) = \begin{cases} \psi \wedge f & \text{if } x = \sigma \\ \psi \wedge \neg f & \text{if } x = \tau \\ \top & \text{if } x = \epsilon \end{cases}$$

- $sub(x) = \emptyset$ *for each* $x \in \Sigma$.

The update instance for the sensing action occurrence a *and the frame of reference* ρ *is* $(\omega(\mathsf{a}, \rho), \{\sigma, \tau\})$.

Observe that an update instance of a sensing action occurrence has three events, each one corresponding to a group of agents. Each event is associated with a group of states in which the truth value of sensed fluent is either known to be true, known to be false, or unknown.

Example 8. *Let us consider the occurrence of* $peek\langle A\rangle$ *in the state described in Figure 6 (top right). The frame of reference for this occurrence of* $peek\langle A\rangle$ *is* $(\{A\}, \{B\}, \{C\})$. *The corresponding update instance is given in Figure 6 (top left).*

We will conclude the section with a discussion on the update model of for announcement actions.

Definition 15 (Update Model/Instance for Announcement Actions)**.** *Given an announcement action instance* a *that announces* φ *with the precondition* ψ *and a frame of reference* $\rho = (F, P, O)$*, the update model for* a *and* ρ*,* $\omega(\mathsf{a}, \rho)$*, is defined by* $\langle \Sigma, R_1, \ldots, R_n, pre, sub\rangle$ *where:*

- $\Sigma = \{\sigma, \tau, \epsilon\}$*;*
- R_i *is defined by*

$$R_i = \begin{cases} \{(\sigma,\sigma), (\tau,\tau), (\epsilon,\epsilon)\} & \text{if } i \in F \\ \{(\sigma,\sigma), (\tau,\tau), (\epsilon,\epsilon), (\sigma,\tau), (\tau,\sigma)\} & \text{if } i \in P \\ \{(\sigma,\epsilon), (\tau,\epsilon), (\epsilon,\epsilon)\} & \text{if } i \in O \end{cases}$$

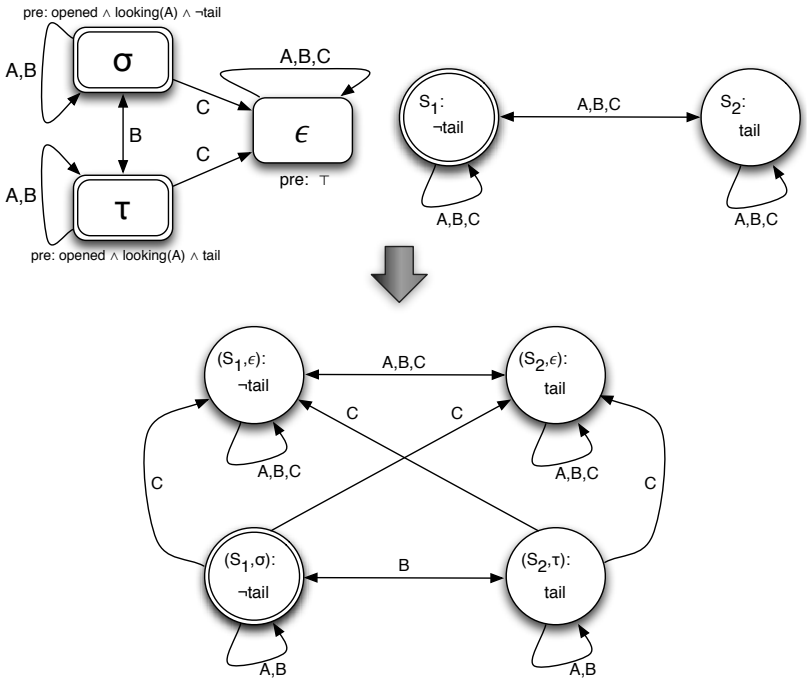

Figure 6: Update Instance $(\omega(peek\langle A\rangle, (\{A\},\{B\},\{C\})), \{\sigma,\tau\})$ and its application

- pre *is defined by*

$$pre(x) = \begin{cases} \psi \wedge \varphi & \textit{if } x = \sigma \\ \psi \wedge \neg\varphi & \textit{if } x = \tau \\ \top & \textit{if } x = \epsilon \end{cases}$$

- sub *is defined as* $sub(x) = \emptyset$ *for any* $x \in \Sigma$.

The update instance for the announcement action occurrence a *with respect to the frame of reference* ρ *is* $(\omega(\mathsf{a}, \rho), \{\sigma\})$.

As we can see, an update model for an announcement action and a frame of reference is structure-wise identical to the update model for a sensing action and a frame of reference. The main distinction lies in the set of designated events in the update instance for each type of actions. There is only one single designated event for announcement actions while there are two for sensing actions.

Example 9. *Let us assume that* A *and* B *have agreed to a scheme of informing each other if the coin lies heads up by raising an hand.* B *can only observe* A *if* B *is looking at the box (or looking at* A*).* C *is completely ignorant about the meaning of* A*'s raising her hand. This can be modeled by*

the following statements[9]*:*

executable $raising\_hand\langle A\rangle$ **if** $\mathbf{B}_x(\neg tail), \neg tail$
$raising\_hand\langle A\rangle$ **announces** $\neg tail$
A **observes** $raising\_hand\langle A\rangle$ **if** $\top$
B **observes** $raising\_hand\langle A\rangle$ **if** $looking(B)$

If A knows the coin lies heads up and raises her hand, B will be aware that the coin lies head up and C is completely ignorant about this.

Let us consider the action occurrence raising_hand$\langle A\rangle$ and the state in which B is looking at the box and thus both A and B are aware of it. We have that the frame of reference is $(\{A, B\}, \emptyset, \{C\})$ and thus the update instance for the occurrence of raising_hand$\langle A\rangle$ is shown in Figure 7.

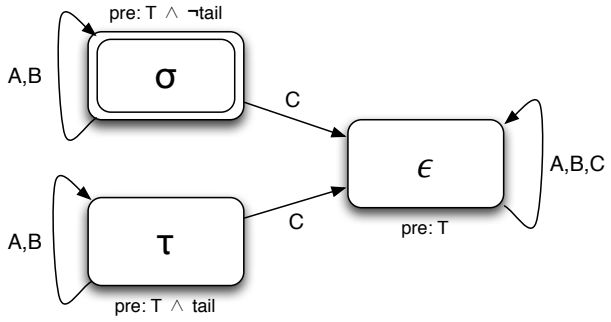

Figure 7: Update instance for the *raising_hand*$\langle A\rangle$ action and $\rho = (\{A, B\}, \emptyset, \{C\})$

4.3. Defining Φ_D

The epistemic actions representing action occurrences can be used in formalizing Φ_D similar to the proposals in Baral et al. (2012, 2013). *However, a main issue of the earlier definitions is that it does not deal with false beliefs*. To account for this and inspired by the suggestion in (van Eijck, 2017), we introduce some extra notations. For a pointed Kripke model (M, s), an agent $i \in \mathcal{AG}$, and a formula φ, we say that i has false belief about φ in (M, s) if

$$(M, s) \models \varphi \text{ and } (M, s) \models \mathbf{B}_i\neg\varphi.$$

For a set of agents S and a pointed Kripke model (M, s) and φ such that $(M, s) \models \varphi$, let $M[S, \varphi]$ be obtained from M by replacing $M[i]$ with $M[S, \varphi][i]$ where

- $M[S, \varphi][i] = (M[i] \setminus M[i]^s) \cup \{(s, s)\}$ for $i \in S$ and $(M, s) \models \mathbf{B}_i\neg\varphi$ where $M[i]^s = \{(s, u) \mid (s, u) \in M[i]\}$; and
- $M[S, \varphi][i] = M[i]$ for other agents, i.e., $i \in \mathcal{AG} \setminus S$ or $i \in S$ and $(M, s) \not\models \mathbf{B}_i\neg\varphi$.

[9] For simplicity, we ignore the effect that A's hand is raised when A raises her hand.

This process aims at correcting beliefs of agents with false beliefs. Intuitively, the links in $M[i]^s$ create the false belief of agent i. Therefore, to correct the false believe of i, we should replace them by the link (s, s). We now define Φ_D. Let ψ be precondition of a, (M, s) a state, and α a set of agents. We say that a is *executable* in (M, s) if $(M, s) \models \psi$. The result of executing a in (M, s) is a set of states, denoted by $\Phi_D(\mathsf{a}, (M, s))$, and is defined as follows.

- if a is not executable in (M, s) then $\Phi_D(\mathsf{a}, (M, s)) = \emptyset$
- if a is executable in (M, s) and $(\mathcal{E}, E_d)$ is the representation of the occurrence of a in (M, s) then
  - $\Phi_D(\mathsf{a}, (M, s)) = (M, s) \otimes (\mathcal{E}, E_d)$ if a is an ontic action instance;
  - $\Phi_D(\mathsf{a}, (M, s)) = M[F_D(\mathsf{a}, M, s), f] \otimes (\mathcal{E}, E_d)$ if a is a sensing action instance that senses f and $(M, s) \models f$;
  - $\Phi_D(\mathsf{a}, (M, s)) = M[F_D(\mathsf{a}, M, s), \neg f] \otimes (\mathcal{E}, E_d)$ if a is a sensing action instance that senses f and $(M, s) \models \neg f$;
  - $\Phi_D(\mathsf{a}, (M, s)) = M[F_D(\mathsf{a}, M, s), \varphi] \otimes (\mathcal{E}, E_d)$ if a is an announcement action instance that announces φ.

Finally, for a set of states $\mathcal{M}$,

- if a is not executable in some $(M, s) \in \mathcal{M}$ then $\Phi_D(\mathsf{a}, \mathcal{M}) = \emptyset$;
- if a is executable in for every $(M, s) \in \mathcal{M}$ then

$$\Phi_D(\mathsf{a}, \mathcal{M}) = \bigcup_{(M,s)\in\mathcal{M}} \Phi_D(\mathsf{a}, (M, s)).$$

4.4. Properties of Φ_D

The syntax and semantics of m$\mathcal{A}^*$ may be of interest in and of themselves, but of particular interest is the fact that m$\mathcal{A}^*$ satisfies certain useful properties—namely that it correctly captures certain intuitions concerning the effects of various types of actions. Specifically, if an agent is fully aware of the execution of an action instance then her belief will be updated with the effects of this action occurrence; an agent who is only partially aware of the action occurrence will believe that those who are fully aware of the action occurrence are certain about its effects; and an agent who is oblivious of the action occurrence is also ignorance of its effects. We will next present several theorems discussing these properties. To simplify the presentation, we will use the following notation throughout the theorems in this subsection.

- D denotes a consistent m$\mathcal{A}^*$ domain;
- (M, s) denotes a state; and

- a is an action instance, whose precondition is given by the statement

  executable a **if** ψ

  in D, and a is executable in (M, s).

- $\rho = (F, P, O)$ is the frame of reference of the execution of a in (M, s) where $F = F_D(\mathsf{a}, M, s)$, $P = P_D(\mathsf{a}, M, s)$, and $O = O_D(\mathsf{a}, M, s)$.

We begin with a theorem about the occurrence of an ontic-action instance.

Theorem 2. *Assume that* a *is an ontic-action instance. It holds that:*

1. *for every agent* $x \in F_D(\mathsf{a}, M, s)$ *and* $[\mathsf{a}$ **causes** ℓ **if** $\varphi]$ *belongs to* D*, if* $(M, s) \models \mathbf{B}_x\varphi$ *then* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_x\ell$*;*
2. *for every agent* $y \in O_D(\mathsf{a}, M, s)$ *and a belief formula* η*,* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_y\eta$ *iff* $(M, s) \models \mathbf{B}_y\eta$*;*
3. *for every pair of agents* $x \in F_D(\mathsf{a}, M, s)$ *and* $y \in O_D(\mathsf{a}, M, s)$ *and a belief formula* η*,* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_x\mathbf{B}_y\eta$ *if* $(M, s) \models \mathbf{B}_x\mathbf{B}_y\eta$*;*

In the above theorem, the first property discusses the changes in the beliefs of agents who are fully aware of the occurrence of an ontic-action instance. The second property shows that oblivious agents are still in the "old state," i.e., they believe nothing has happened. The third property indicates that fully aware agents are also aware that oblivious agents' beliefs do not change. This is particular useful in situations when an agent would like to create false beliefs about a fluent p for other agents: she only needs to secretly execute an action that changes the truth value of p.

Theorem 3. *Assume that* a *is a sensing action instance and* D *contains the statement* a **determines** f*. It holds that:*

1. $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{C}_{F_D(\mathsf{a},M,s)}f$ *if* $(M, s) \models f$*;*
2. $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{C}_{F_D(\mathsf{a},M,s)}\neg f$ *if* $(M, s) \models \neg f$*;*
3. $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{C}_{P_D(\mathsf{a},M,s)}(\mathbf{C}_{F_D(\mathsf{a},M,s)}f \vee \mathbf{C}_{F_D(\mathsf{a},M,s)}\neg f)$*;*
4. *for every agent* $y \in O_D(\mathsf{a}, M, s)$ *and a belief formula* η*,* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_y\eta$ *iff* $(M, s) \models \mathbf{B}_y\eta$*;*
5. *for every pair of agents* $x \in F_D(\mathsf{a}, M, s)$ *and* $y \in O_D(\mathsf{a}, M, s)$ *and a belief formula* η*,* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_x\mathbf{B}_y\eta$ *if* $(M, s) \models \mathbf{B}_x\mathbf{B}_y\eta$*;*

The first and second properties of the above theorem indicate that agents who are fully aware of the occurrence of the sensing action instance will be able to update their beliefs with the true truth value of the sensed fluent, thereby correcting any false beliefs that they might have before the execution of the action. The third property shows that agents who are partially aware of the action execution will know that agents who are fully aware of the action execution will have the correct beliefs about the sensed fluents. The fourth and fifth properties are about oblivious agents' beliefs.

Theorem 4. *Assume that* a *is an announcement action instance and* D *contains the statement* a **announces** φ*. If* $(M, s) \models \varphi$ *then it holds that*

1. $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{C}_{F_D(\mathsf{a},M,s)}\varphi$;
2. $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{C}_{P_D(\mathsf{a},M,s)}(\mathbf{C}_{F_D(\mathsf{a},M,s)}\varphi \vee \mathbf{C}_{F_D(\mathsf{a},M,s)}\neg\varphi)$;
3. *for every agent* $y \in O_D(\mathsf{a},M,s)$ *and a belief formula* η, $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{B}_y\eta$ *iff* $(M,s) \models \mathbf{B}_y\eta$;
4. *for every pair of agents* $x \in F_D(\mathsf{a},M,s)$ *and* $y \in O_D(\mathsf{a},M,s)$ *and a belief formula* η, $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{B}_x\mathbf{B}_y\eta$ *if* $(M,s) \models \mathbf{B}_x\mathbf{B}_y\eta$;

Similar to Theorem 3, the first property of the above theorem indicates that truthful announcement could help agents who are fully aware of the execution of the action will be able to correct their false beliefs. Likewise, partially aware agents will only know that fully aware agents know the truth value of the announced formula but they might not have the real value of this formula themselves. Furthermore, as in other types of actions, the beliefs of oblivious agents do not change.

4.5. Entailment in m$\mathcal{A}^*$ Action Theories

We are now ready to define the notion of entailment in m$\mathcal{A}^*$ action theories. It will be defined between m$\mathcal{A}^*$ action theories and queries of the following form:

$$\varphi \textbf{ after } \delta \tag{12}$$

where φ is a belief formula and δ is a *sequence of action instances* $a_1;\ldots;a_n$ $(n \geq 0)$—referred to as a *plan*. Let us observe that the entailment can be easily extended to consider more general forms of *conditional plans,* that include conditional statements (e.g., **if-then**) or even loops (e.g., **while**)—as discussed in (Levesque et al., 1997; Son and Baral, 2001). We leave these relatively simple extensions for future work.

The description of an evolution of a system will deal with *sets* of states. We refer to a set of states as a *belief state* (or a *b-state*). We need the following definitions. For a b-state B and an action occurrence a, we say that a is executable in B if $\Phi_D(a,(M,s)) \neq \emptyset$ for every state (M,s) in B. With a slight abuse of notation, we define

$$\Phi_D(a,B) = \begin{cases} \{\bot\} & \text{if } \Phi_D(a,(M,s)) = \emptyset \text{ in some state } (M,s) \text{ in } B \\ & \text{or } B = \{\bot\} \\ \bigcup_{(M,s)\in B} \Phi_D(a,(M,s)) & \text{otherwise} \end{cases}$$

where $\{\bot\}$ denotes that the occurrence of a in B fails. Note that we assume that no action is executable in $\bot$.

Let δ be a plan and B be a b-state. The set of b-states resulting from the execution of δ in B, denoted by $\Phi^*_D(\delta,B)$, is defined as follows:

- If δ is the empty plan $[\,]$ then $\Phi^*_D([\,],B) = B$;
- If δ is a plan of the form $a;\delta'$ (with $a \in \mathcal{A}$), then $\Phi^*_D(a;\delta',B) = \Phi^*_D(\delta',\Phi_D(a,B))$.

Intuitively, the execution of δ in B can go through several paths, each path might finish in a set of states. It is easy to see that if one of the states reached on a path during the execution of δ is $\bot$ (the failed state) then the final result of the execution of δ in B is $\{\bot\}$. $\Phi^*_D(\delta,B) = \{\bot\}$ indicates that the execution of δ in B fails.

We are now ready to define the notion of entailment.

Definition 16 (Entailment)**.** *An action theory* (I, D) *entails the query*

$$\varphi \textbf{ after } \delta,$$

denoted by $(I, D) \models \varphi$ **after** δ*, if*

(a) $\Phi_D^*(\delta, I_0) \neq \{\bot\}$ *and*

(b) $(M, s) \models \varphi$ *for each* $(M, s) \in \Phi_D^*(\delta, I_0)$

where I_0 *is the initial b-state of* (I, D)*.*

We say that (I, D) **S5***-entails the query* φ **after** δ*, denoted by* $(I, D) \models_{\mathbf{S5}} \varphi$ **after** δ*, if the two conditions (a)-(b) are satisfied with respect to* I_0 *being the initial* **S5***-b-state of* (I, D)*.*

4.6. Using m$\mathcal{A}^*$: An Illustration

The next example illustrates these definitions.

Example 10. *Let* D_1 *be the domain specification given in Examples 3 and 4 and* I_1 *be the set of initial statements given in Example 5. Furthermore, let* δ_A *be the sequence of actions:*

$$\delta_A = distract(C)\langle A\rangle; signal(B)\langle A\rangle; open\langle A\rangle; peek\langle A\rangle.$$

We can show that

$$(I_1, D_1) \models_{\mathbf{S5}} (\mathbf{B}_A tail \vee \mathbf{B}_A \neg tail) \wedge \mathbf{B}_A(\mathbf{B}_B(\mathbf{B}_A tail \vee \mathbf{B}_A \neg tail)) \textbf{ after } \delta_A$$
$$(I_1, D_1) \models_{\mathbf{S5}} \mathbf{B}_B(\mathbf{B}_A tail \vee \mathbf{B}_A \neg tail) \wedge (\neg \mathbf{B}_B tail \wedge \neg \mathbf{B}_B \neg tail) \textbf{ after } \delta_A$$
$$(I_1, D_1) \models_{\mathbf{S5}} \mathbf{B}_C[\textstyle\bigwedge_{i\in\{A,B,C\}}(\neg \mathbf{B}_i tail \wedge \neg \mathbf{B}_A \neg tail)] \textbf{ after } \delta_A$$

[3]

[3] This part needs to be rewritten

To see how the above conclusions hold, let us construct a **S5** *initial state for* (I_1, D_1) *(see also Figure 8). Since the truth values of all fluents but* $tail$ *are known to every agent, we have that* $M_0 = \langle\{s_0, s_1\}, \pi_0, \mathcal{B}_A^0, \mathcal{B}_B^0, \mathcal{B}_C^0\rangle$ *where*

$$\pi_0(s_0) = \left\{ \begin{array}{l} \neg opened, has\_key(A), \neg has\_key(B), \neg has\_key(C), \\ looking(A), looking(B), looking(C), \neg tail \end{array} \right\}$$

and

$$\pi_0(s_1) = \left\{ \begin{array}{l} \neg opened, has\_key(A), \neg has\_key(B), \neg has\_key(C), \\ looking(A), looking(B), looking(C), tail \end{array} \right\}$$

Furthermore, $\mathcal{B}_A^0 = \mathcal{B}_B^0 = \mathcal{B}_C^0 = \{(s_0, s_0), (s_0, s_1), (s_1, s_0), (s_1, s_1)\}$.

The execution of the action $distract(C)\langle A\rangle$ *in* (M_0, s_0) *results in the state* (M_1, s_2) *as shown in Figure 9 where* $M_1 = \langle\{s_0, s_1, s_2, s_3\}, \pi_1, \mathcal{B}_A^1, \mathcal{B}_B^1, \mathcal{B}_C^1\rangle$*. The top part of the new structure is a replica of* (M_0, s_0)*, encoding the beliefs of* B*, who is ignorant of the action occurrence. The bottom part encodes the beliefs of* A *and* C*, who are observers of the action occurrence. It includes two new worlds,* s_2 *and* s_3*, which represent the result of the execution of* $distract(A, C)$ *in* s_0 *and* s_1 *respectively, where* $\pi_1(s_0) = \pi_0(s_0)$*,* $\pi_1(s_1) = \pi_0(s_1)$*,*

$$\pi_1(s_2) = \pi_0(s_0) \setminus \{looking(C)\} \cup \{\neg looking(C)\}$$

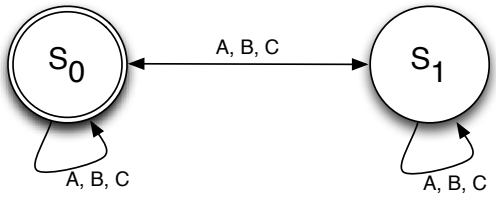

Figure 8: (M_0, s_0): an initial state of (I_1, D_1)

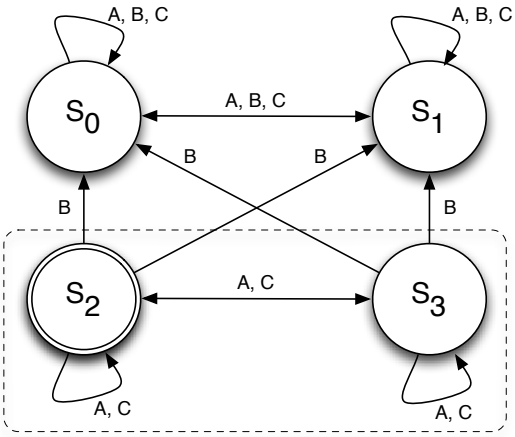

Figure 9: (M_1, s_2): result of execution of $distract(C)\langle A\rangle$ in (M_0, s_0)

and

$$\pi_1(s_3) = \pi_0(s_1) \setminus \{looking(C)\} \cup \{\neg looking(C)\}.$$

Finally, $\mathcal{B}^1_A = \mathcal{B}^0_A \cup \{(s_2, s_2), (s_2, s_3), (s_3, s_2), (s_3, s_3)\}$, $\mathcal{B}^1_C = \mathcal{B}^0_C \cup \{(s_2, s_2), (s_2, s_3), (s_3, s_2), (s_3, s_3)\}$, *and* $\mathcal{B}^1_B = \mathcal{B}^0_B \cup \{(s_2, s_0), (s_2, s_1), (s_3, s_0), (s_3, s_1)\}$.
The execution of $signal(A, B)$ *in* (M_1, s_2) *results in a new state* (M_2, s_6) *where*

$$M_2 = \langle \{s_0, \ldots, s_7\}, \pi_2, \mathcal{B}^2_A, \mathcal{B}^2_B, \mathcal{B}^2_C \rangle$$

where:

- *For* $i \in \{0, \ldots, 3\}$, *the state* s_{i+4} *is the result of executing* $signal(A, B)$ *in* s_i. *We have that* $\pi_2(s_{i+4}) = \pi_1(s_i)$ *because* B *is looking at the box already and thus the execution of this action does not change the state;*
- $\mathcal{B}^2_A = \mathcal{B}^1_A \cup \{(s_{i+4}, s_{j+4}) \mid 0 \leq i, j \leq 3 \text{ and } (s_i, s_j) \in \mathcal{B}^1_A\}$;
- $\mathcal{B}^2_B = \mathcal{B}^1_B \cup \{(s_{i+4}, s_{j+4}) \mid 0 \leq i, j \leq 3 \text{ and } (s_i, s_j) \in \mathcal{B}^1_B\}$; *and*
- $\mathcal{B}^2_C = \mathcal{B}^1_C \cup \{(s_{i+4}, s_j) \mid 0 \leq i, j \leq 3 \text{ and } (s_i, s_j) \in \mathcal{B}^1_C\}$.

The execution of $open(A)$ *in* (M_2, s_6) *will result in a new state* (M_3, s_{14}) *where*

$$M_3 = \langle \{s_0, \ldots, s_{15}\}, \pi_3, \mathcal{B}^3_A, \mathcal{B}^3_B, \mathcal{B}^3_C \rangle$$

where:

- *For* $i \in \{0, \ldots, 7\}$, *the state* s_{i+8} *is the result of executing* $open(A)$ *in* s_i. *We have that* $\pi_3(s_{i+8}) = \pi_2(s_i) \setminus \{\neg opened\} \cup \{opened\}$;

- $\mathcal{B}_A^3 = \mathcal{B}_A^2 \cup \{(s_{i+8}, s_{j+8}) \mid 0 \leq i, j \leq 7$ and $(s_i, s_j) \in \mathcal{B}_A^2\}$;
- $\mathcal{B}_B^3 = \mathcal{B}_B^2 \cup \{(s_{i+8}, s_{j+8}) \mid 0 \leq i, j \leq 7$ and $(s_i, s_j) \in \mathcal{B}_B^2\}$; *and*
- $\mathcal{B}_C^3 = \mathcal{B}_C^2 \cup \{(s_{i+8}, s_j) \mid 0 \leq i, j \leq 7$ and $(s_i, s_j) \in \mathcal{B}_C^2\}$.

Finally, the execution of $peek(A)$ *in* (M_3, s_{14}) *results in a new state* (M_4, s_{30}) *where*

$$M_4 = \langle \{s_i \mid 0 \leq i \leq 31\}, \pi_4, \mathcal{B}_A^4, \mathcal{B}_B^4, \mathcal{B}_C^4 \rangle$$

where:

- *For* $i \in \{0, \ldots, 15\}$*, the interpretation* $\pi_4(s_{i+15}) = \pi_3(s_i)$ *since the execution of a sensing action does not change the state of the world;*
- $\mathcal{B}_A^4 = \mathcal{B}_A^3 \cup \{(s_{i+16}, s_{j+16}) \mid 0 \leq i, j \leq 15$ and $(s_i, s_j) \in \mathcal{B}_A^3$ and $((\neg tail \in \pi_3(s_i) \cap \pi_3(s_j)) \vee (tail \in \pi_3(s_i) \cap \pi_3(s_j))\}$;
- $\mathcal{B}_B^4 = \mathcal{B}_B^3 \cup \{(s_{i+16}, s_{j+16}) \mid 0 \leq i, j \leq 15$ and $(s_i, s_j) \in \mathcal{B}_B^3\}$; *and*
- $\mathcal{B}_C^4 = \mathcal{B}_C^3 \cup \{(s_{i+16}, s_j) \mid 0 \leq i, j \leq 15$ and $(s_i, s_j) \in \mathcal{B}_C^3\}$.

We will finally show that $(M_4, s_{30}) \models \mathbf{B}_A tail \vee \mathbf{B}_A \neg tail$*. In fact, since* $(M_0, s_0) \models \neg tail$*, we will show that* $(M_4, s_{30}) \models \mathbf{B}_A \neg tail$*. To prove this, we need to show that* $(M_4, s_j) \models \neg tail$ *for every* $s_j \in M_4[S]$*,* $(s_{30}, s_j) \in \mathcal{B}_A^4$*. Since* $s_{30} \notin M_3[S]$*, we have that* $(s_{30}, s_j) \in \mathcal{B}_A^4$ *iff* $j \geq 16$*,* $(s_{14}, s_{j-16}) \in \mathcal{B}_A^3$*, and* $\neg tail \in \pi_3(s_{14}) \cap \pi_3(s_{j-16})$ *(because* $\neg tail \in \pi_3(s_{14})$*). This implies that* $(M_4, s_{30}) \models \mathbf{B}_A \neg tail$*. The proof of other conclusions is similar.*

We conclude the example with the observation that another initial state for (I_1, D_1) *is* (M_0, s_1) *and the execution of* δ_A *in this state results in a new state* (M', s') *with the property that* $(M', s') \models \mathbf{B}_A tail$*. Theoretically, this fact and the fact that* $(M_4, s_{30}) \models \mathbf{B}_A \neg tail$ *are insufficient for us to conclude that* $(I_1, D_1) \models_{\mathbf{S5}} \mathbf{B}_A tail \vee \mathbf{B}_A \neg tail$ *holds. This, however, holds under some additional assumption and thanks to Proposition* **??** *(see Section* **??***).*

5. Related Work and Discussion

In this section, we connect our work to related efforts in reasoning about actions and their effects in multi-agent domains. The background literature spans multiple areas. We will give a quick introduction and focus our attention on the most closely related works.

5.1. Related Logics

The research discussed in this paper relates to a broad variety of logics and languages used to deal with reasoning about actions and their effects in multi-agent domains—e.g., classical logic, non-monotonic logics, causal logics, high level action languages, modal logics, epistemic logics, dynamic logics and dynamic epistemic logics. Here, we provide a very brief overview of their relevance to reasoning about actions and their effects in multi-agent domains.

Classical logics, in particular, propositional and first-order logic, are often used to specify the physical state of the world. For example, the propositional formula $on\_table\_a \wedge on\_table\_b \wedge \neg on\_table\_c$ expresses that the blocks a and b are on the table and the block c is not of the table. Similarly, the first order formula $on\_table(a) \wedge on\_table(b) \wedge \neg on\_table(b) \wedge \forall X(ontable(X) \Rightarrow color(X, red))$, describes a situation where the blocks a and b are on the table, the block c is not on the table, and all blocks on the table are red. Propositional logic and first-order logic can be used to represent the effects of actions and to reason about them. However, straightforward encodings require a large number of axioms, especially to represent the inertia axioms—the properties of the world that do not change when a particular action is performed. Two approaches can be considered to address this problem: **(i)** By using *non-monotonic logics,* that can naturally express statements of the type *"Normally an action does not affect a property"* and can express exceptions to this statement; **(ii)** By an alternative approach, used for example in Reiter (2001), where the effects of actions on various properties of the world are expressed using a high-level logic, which is then translated, using sophisticated compilation techniques, into succinct encodings of inertia axioms in a classical logic.

While reasoning about the effect of actions, the relationship between some properties of the world may give rise to "qualification" and "ramification". Expressing this in classical logic leads to problems in many cases, especially when the relationship between the properties are causal in nature. For example consider the two statements:

(a) A person cannot be in two places at the same time and

(b) A person cannot be married to two persons at the same time.

In classical logic, their representations are very similar: $at(X) \wedge at(Y) \Rightarrow X = Y$ and $married\_to(X) \wedge married\_to(Y) \Rightarrow X = Y$. But let us assume that, initially, a person is at location p and he performs the action of moving to a location q different from p. The statement *(a)* causes a ramification—since we would like to infer that, after the execution of the action, the person is at p and not at q. On the other hand, let us assume that initially a person is married to p and he (tries to) perform the action of marrying q (in a courthouse with marriage records) different from p. The statement *(b)*, this time, introduces a qualification, because of which the person is unable to perform the action of marrying q.

Causal logic allows us to express *(a)* and *(b)* in a different way: $at(X) \wedge X \neq Y$ **causes** $\neg at(Y)$ and $married\_to(X) \wedge married\_to(Y) \wedge X \neq Y$ **causes** $FALSE$. Such causal relationships can be expressed using logic programming, which has a non-classical connective "$\leftarrow$." In logic programming *(a)* and *(b)* can be expressed as: $\neg at(Y) \leftarrow at(X), X \neq Y$ and $\leftarrow married\_to(X), married\_to(Y), X \neq Y$.

High-level action languages were introduced to **(a)** provide a English-like specification language to describe the effects of actions on properties of the world and the relationships between these properties, and to provide a semantics that uses simple set theoretical notations; and **(b)** provide a framework that can be used to prove correctness of encodings in various logics for reasoning about effect of actions. For example, specifications in the language $\mathcal{A}$ Gelfond and Lifschitz (1993) are of the forms: *(i)* a **causes** f **if** $p_1, \ldots, p_n$; and *(ii)* f **after** $a_1, \ldots a_n$.

Modal logics add modal operators to various logics and their semantics is often defined using Kripke structures. Two simple modal logics that are relevant to reasoning about actions are temporal logics and epistemic logics. One of the simplest temporal logic is the forward linear temporal logic; it includes the operators $\Box$, $\bigcirc$, $\Diamond$, and $\mathcal{U}$, meaning "always in the future", "next time step", "sometime in the future" and "until." Formulae in this logic are often used to express goals that specify how we want the world to evolve. Reasoning is commonly focused on action-plans, to verify if an action-plan satisfies a desired temporal specification, or to derive action-plans that satisfy a goal, given as a temporal specification. Epistemic logics use the modal operators $\mathbf{K}_i$, where $\mathbf{K}_i f$ means that agent i knows that f is true. In this paper, we used belief logics which are similar to epistemic logics, the main difference being that we use the modal operators $\mathbf{B}_i$, where $\mathbf{B}_i f$ means that the agent i believes that f is true.

The community working on *reasoning about actions and change* initially used only the sequencing constructs to build action-plans as sequence of actions and reason about such sequences. This has been extended Scherl and Levesque (2003); Son and Baral (2001) to allow "**if-then**" and other procedural features, that become necessary when sensing actions are needed to be part of plans. GOLOG Levesque et al. (1997), an acronym for "alGOl in LOGic", borrows programming constructs from procedural languages to express complex plans which can be "evaluated" to generate valid action sequences. The evaluation of GOLOG programs, as well as other action-plans, is realized on top of action theories that allow one to reason about a single action and its impact on the world—and, in the process, take into account the issues regarding inertia, qualification and ramification.

A related area of research is focused on the exploration of *Dynamic logics.* Dynamic logics were originally developed to reason about program correctness. In the programming domain, the basic actions are assignments of values to variables. Reasoning about the effects of such assignment actions and the associated inertia is straightforward, since all variables retain their old values except for the variables being assigned. In traditional procedural programming languages, one does not have to worry about qualification and ramification, since assigning a value to a variable does not have any implicit impact on other variables.[10] Any needed impact is explicitly written as new assignment statements. The original focus of dynamic logics is to reason about programs that are built by composing simple assignment statements. Such programs are built using constructs such as: *(i)* $\alpha \cup \beta$ (i.e., execute α or β), *(ii)* $\alpha; \beta$ (i.e., execute α followed by β), *(iii)* $\alpha*$ (i.e., iterate α a finite number of times), *(iv)* $p?$ (i.e., test whether α holds), and *(v)* λ (i.e., no-op).

To reason about programs in dynamic logics, programs are used as modal operators; various axioms and inference rules have been developed that allow one to reason about the programs. The modal constructs used are of the form: *(i)* $[a]p$, meaning that, after performing a, it is necessarily the case that p is true in the world, and *(ii)* $\langle a \rangle p$, meaning that, after performing a, it is possible that p is true in the world. These modal constructs allow one to specify effects of arbitrary actions. For example, to express that an action a makes the formula ϕ true, one can write $[a]\phi$. Similarly, to express that a makes the formula ϕ true only when executed in a state where ψ is true, one can write $[\lambda](\psi \Rightarrow [a]\phi)$. However, when we go beyond assignment actions, one needs to account for

[10] If we ignore issues of aliasing, e.g., through pointers.

inertia. Similarly, when one goes beyond the programming environment and variable assignments, one needs to worry about qualification and ramification. The inertia axioms can be expressed in dynamic logic as shown in Meyer (2000), which is similar to writing inertia axioms using classical logic—i.e., they are encoded using a large number of formulae, that list one by one what properties are not affected by an action. Such approach to account for ramification is tedious and involves the use of logic programming, pre-compilation and then generating dynamic logic formulae based on such pre-compilation. In Prendinger and Schurz (1996), where dynamic logic is used for reasoning about actions and change, inertia is avoided by referring to it as an undesirable over-commitment; however, the authors admit that their formulation cannot address qualification, and they indicate that "non-monotonic logics are clearly superior" in that regard. Overall, when reasoning about actions and change (beyond simple variable assignments) using dynamic logics, one still needs to worry about inertia and the frame problem, qualification and ramification issues.

Many early works about action and change, as well as dynamic logics, reason about the world and do not worry about agent's knowledge about the world. When sensing actions are considered, one has to distinguish between the world and a single agent's knowledge about the world. This leads to the use of epistemic logics and Kripke structures, and frame axioms need to be developed with respect to knowledge formulae. This has been achieved by having frame axioms for the accessibility relations used in the Kripke structures. High level languages that can express sensing actions and their effects have been developed and matched with logical encodings. This paper extends such approach to the case of multi-agent domains, where other knowledge actions (besides sensing) are considered. A new dimension that emerges is the observability of the various agents as part of an action; some may have full observability, some others may have partial observability, and the rest have no observability. The result of such actions and such varied observability is that different agents have different knowledge and beliefs about the world and about each other's knowledge and beliefs.

Several researchers have considered reasoning about actions in a multi-agent domain using the dynamic logics approach. These proposals are focused on extending dynamic logics to *Dynamic Epistemic Logics (DEL),* to reason about the agent's knowledge about the world and about other agents' knowledge in presence of multi-agent actions. The extensions are twofold. In particular, in the formula $[\alpha]p$: *(i)* In DEL, p is a formula in epistemic logic, while in dynamic logic p is a classical logic formula, and *(ii)* In DEL, α is a more general action than in dynamic logic. In van Ditmarsch et al. (2007), α has the usual dynamic logic constructs for complex actions, plus constructs such as: *(a)* $L_A?\varphi$ whose intuitive meaning is that the agents in the group A learn that φ is true; and *(b)* $(\alpha!\alpha')$ whose intuitive meaning is that between α and α', we choose α. The latter is often written as $(!\alpha \cup \alpha')$. Using these actions, the authors of van Ditmarsch et al. (2007) show that one can express sensing actions (refereed to as *"read"* actions) in a multi-agent setting. Specifically, the fact that we have two agents a and b, a senses the value of p as being true, and it is common knowledge between a and b that b observes a sensing, can be modeled as the complex action $L_{ab}(!L_a?p \cup L_a?\neg p)$. However, the language of update models Baltag and Moss (2004) is more general and is currently preferred by the DEL community to express such multi-agent actions. In the language of update models, the action discussed above can be expressed as in Figure 10.

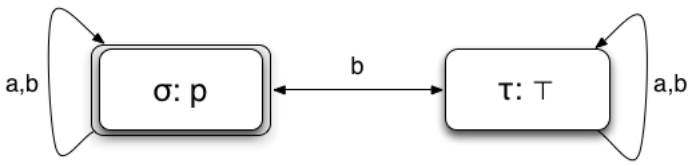

Figure 10: A program model for $L_{ab}(!L_a?p \cup L_a?\neg p)$

5.2. Relating m$\mathcal{A}^*$ and DEL with Update Models: Differences

The similarities between m$\mathcal{A}^*$ and the update models based approach has been discussed in detail in Section 4. We next detail the differences between the two approaches.

5.2.1. Multi-Agent Actions

A key difference between the formalism proposed in this paper and DEL with update models is with respect to the simplest of actions in presence of multiple agents.

Let us consider the simplified version of the coin in a box problem as presented in Example 2—with three agents A, B, and C, a box containing a coin, and initially it is common knowledge that none of the agents knows whether the coin lies heads up or tails up. Let us assume that agent A peeks into the box. In our formalism, we express this action as *peek(A)*. In DEL, the update model for the same action, as given in Figure 6, will also include additional information about all three agents A, B, and C encoding their "roles" or "perspectives" while A is peeking into the box. By roles or perspectives we mean information about what the agents are doing, in terms of who is watching whom and who knows about that.

It is evident that our representation of the action simply as *peek(A)* is much simpler than the DEL representation in Figure 6. But our representation does not include the information about what else the agents A, B, and C are doing while A is peeking into the box. In our formulation, such information is *part of the state,* and is expressed by using perspective fluents, such as *looking(B)*—that encodes the information that B is looking at the box—and *group_member(B,group(A))*—that encodes the information that B and A are together in the same group.

Thus, it appears that a critical difference between the m$\mathcal{A}^*$ approach to representing multi-agent actions and the approach used in DEL with update models lies in the way we encode the information about agents roles and perspectives—as part of the action in DEL with update models and as part of the state in m$\mathcal{A}^*$. At first glance, this difference may not appear far reaching. However, there are some far reaching implications of such difference, as discussed in the following subsections.

Narratives and Dynamic Evolution of Multi-agent Actions:. Let us consider a scenario with two agents A and B. Initially, agent B is looking at agent A. Agent A lifts a block and, after some time, agent A puts down the block. Some time later, agent B is distracted while agent A again lifts the block.

In our formulation, this narrative can be formalized by first describing the initial situation, and then describing the sequence of actions that occurred, which for this example is:

$$\textit{liftBlock}(A); \textit{putDown}(A); \textit{distract}(B); \textit{liftBlock}(A).$$

The description of this evolution of scenario in DEL is not as simple: each action occurrence will have to be described as an update model containing information about both agents A and B. In addition, such a description (in DEL) will be partly superfluous, as it will have to record information about B looking (or not looking) at A in the update model, when that information is already part of the state. Thus, the approach used in m$\mathcal{A}^*$ to describe this narrative is more natural than the representation in DEL.

Observe that, in our narrative, the action *liftBlock(*A*)* appears twice. However, due to the difference in the roles and perspectives over time, the two occurrences of *liftBlock(*A*)* correspond to two different update models. This shows how, using the m$\mathcal{A}^*$ formulation, we can support the dynamic evolution of update models, as result of changes in perspective fluents in the state. In DEL, the two update models are distinct and there is no direct connection between them and neither one does evolve from the other.

In order to further reinforce this point, let us consider another narrative example. Let us consider a scenario with three agents, A, B, and C. Initially, it is common knowledge that none of the agents knows whether the coin in the box is lying heads up or tails up. In addition, let us assume that initially A and B are looking at the box, while C is looking away. Let us consider the narrative where A peeks into the box; afterwards, A realizes that C is distracted and signals C to look at the box as well; finally A peeks into the box one more time. In m$\mathcal{A}^*$, this situation can be described again by a sequence of actions:

$$\textit{peek}(A); \textit{signal}(C); \textit{peek}(A)$$

The two occurrences of *peek(*A*)* correspond to two different update models; the second occurrence is an evolution of the first caused by the execution of *signal(*C*)*. In DEL, the relevance of the intermediate action *signal(*C*)*, and its impact on the second occurrence of *peek(*A*)*, is mostly lost—and this results in the use of two distinct update models for `peek(`A`)` with complete information about the whole action scenario.

The key aspect that allows a natural representation of narratives and evolution of update models in m$\mathcal{A}^*$ is the presence of the agents' perspectives and roles encoded as perspective fluents of a state, and their use to dynamically generate the update models of the actions. While DEL can include perspective fluents as part of the states as well, it does not have a way to take advantage of them in a similar way as m$\mathcal{A}^*$.

Separation of Specification of Actions and their Effects and the Observability of an Agent:. As alluded in the previous two sections, a core difference between m$\mathcal{A}^*$ and the DEL specification of actions in multi-agent domains, lies in that m$\mathcal{A}^*$ separates the specification of actions and action effects from the description of observability of the action occurrence by each agent. In both van Ditmarsch et al. (2007) and Baltag and Moss (2004), the observability of agents is hard-coded in the specification of each complex action. We discuss this difference in more detail using an example.

Let us reconsider the domain D_1. In order to describe the possible histories of the domain, we need to develop the update models for every action occurrence. Let us consider, for example, the action *peek(*A*)*; we need to have an update model for all of the following cases:

- Both B and C are looking;

- Either B or C is looking but not both; and
- Both B C are not looking.

In our approach, the above example is specified in a very different way: the action is about sensing $tail$. The agents who sense it, who observe the sensing take place, and who are oblivious can be specified directly or can be specified indirectly in terms of *conditions,* such as which agents are near the sensing, which ones are watching from far, and which ones are looking away, respectively. Actions can be planned and executed to change the observers and partial observers. In other words, in m$\mathcal{A}^*$, if we have a complex action $\alpha;\beta$, by executing α we may be able to change the world, in terms of who is fully observing, who is partially observing, and who is oblivious with respect to the next action β. This is not possible in DEL. Thus, while m$\mathcal{A}^*$ allows us to develop plans where an agent can manipulate the observability of other agents, such planning cannot be done in straightforward manner in DEL.

Simplicity by Design:. The formulation adopted in this paper is limited in expressivity to ensure simplicity. It is limited by the (perspective) fluents we have and how we use them. On the other hand, DEL is more complex and also more expressive.

One advantage of our simplicity is that it limits the number of possible plans of a particular length, contributing to the feasibility of multi-agent planning. In DEL, since update models are analogous to Kripke models, even in presence of a small number fluents, it is possible to generate an infinite number of update models. One can limit the number of update models by imposing restrictions on their structure. Nevertheless, as discussed earlier, an update model is much more complex than actions m$\mathcal{A}^*$, which are often[11] single units.

As an example,[12] Figure 11 displays an update model that cannot be represented in m$\mathcal{A}^*$. The intuition behind this update model is as follows. When A executes the action *peek(*A*)*, A believes that both A and B can see the outcome of sensing $tail$—i.e., A and B are fully observant. In reality, B is oblivious. This shows that, in multi-agent domains, an agent's observability could also be considered as beliefs, and as such affect the beliefs of an agent about other agents' beliefs after the execution of an action. The present m$\mathcal{A}^*$ language does not allow for such specification.

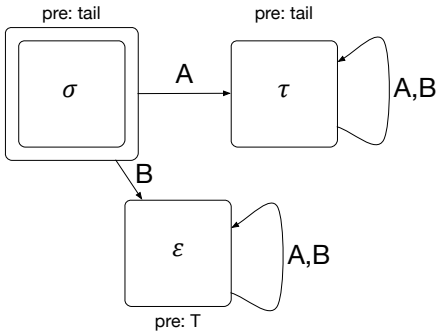

Figure 11: An update model of the *peek(*A*)* action without an equivalent m$\mathcal{A}^*$ representation

[11] We could allow parallel actions.

[12] We thank an anonymous reviewer of an earlier version of this paper who suggested a similar example.

The simplicity of our formulation is by design, and not an inherent flaw of our approach. Indeed, one could envision developing a complete encoding of the complex graph structure of an update model as part of state, using an extended collection of perspective fluents—but, at this time, we do not have a corresponding theory of change to guide us in using these more expressive perspective fluents to capture the full expressive power of update models in DEL. Hence, our current formalism is less expressive than DEL. However, the higher expressiveness of update models provides us with a target to expand m$\mathcal{A}^*$ and capture more general actions. Expanding m$\mathcal{A}^*$ to express actions as the one in Fig. 11 will be one of our immediate future goal.

On the other hand, as remarked earlier, the research on DEL with update models lacks at present an exploration of how update models can evolve as result of action executions.

Analogy with Belief Update:. Another approach to explore the differences between m$\mathcal{A}^*$ and DEL builds on the analogy to the corresponding differences between *belief updates* and the treatment of actions and change in early action languages Gelfond and Lifschitz (1998).

Papers on belief updates define and study the problem of updating a formula ϕ with a formula ψ. In contrast, in reasoning about actions and change, the focus is on defining the resulting state of the world after a particular action is performed in a particular world, given a description of *(i)* how the action may change the world, *(ii)* when the action can be executed; and *(iii)* how the fluents in the world may be (possibly causally) related to each other. In such a context, given a state s and an action a, it is possible to see the the determination of the resulting state as the update of s by a formula φ; But, what is important to consider is that the φ is not just the collection of effects of the action a, but incorporates several other components, that take into account the static causal laws as well as which conditions (part of the conditional effects of a) are true in s.

This situation is not dissimilar to the distinction between DEL update models and m$\mathcal{A}^*$. An update model can be encoded by an action formula, and the resulting state can be obtained by updating the starting state with such formula. In DEL, such action formula has to be given directly. Instead, our considerations in m$\mathcal{A}^*$ are in the spirit of the early research in reasoning about actions and change—where we focus on describing actions and their effects, their executability conditions, and where a resulting "state" is determined by applying these descriptions to the "state" where a particular action is performed. Thus, the action formula in this latter case is not explicitly given, but derived from the description of the actions, their effects, and executability conditions.

Taking the analogy further, an important application of reasoning about actions is to determine action sequences or plan structures that achieve a given goal. This is different from searching for a formula ψ which, if used to update a given initial state, will generate a goal state; the difference is partly due to the fact that the the space of formulae is infinite. Similarly, the space of update models is infinite, and it is not viable to look for an update model (or a sequence of update models) that will cause a transition from a given initial state to a goal state. Instead, m$\mathcal{A}^*$ supports the traditional way of planning by finding action sequences or plan structures that achieve a goal.

Executing Actions:. The notion of actions adopted in m$\mathcal{A}^*$ is designed to enable their executions by one or multiple agents. For example, the action *peek(*A*)* can be executed by agent A, by peeking into the box. On the other hand, an action modeled using update models in DEL is not executable, in the normal sense. For example, how does the action expressed in Figure 6 get executed? Who

does execute such action? What does executing the various edges of Figure 6 mean? The answers to these questions are not clear.

Furthermore, let us turn around such questions and focus on the perspective fluents: how does one execute the perspective fluents, such as *looking(B)*? The answer is that they are fluents, and they are not required or supposed to be executed. A more appropriate question would be: how do they become true? The answer is that, while our formulation could have some actions that make them true, when describing a state we need not worry about how exactly the facts in the state came about to be true. This is not the case when describing actions: when describing actions we need to describe something that can be executed. In summary, actions, or parts of actions, are supposed to be something that can be executed, while states, or parts of states, do not have such requirement.

Hence, our representation of actions is more appropriate, and follows the common meaning of an action,[13] than the representation of action in DEL.

Value of Update Models:. Having discussed the differences between m$\mathcal{A}^*$ and update models, we would like to point out that update models present a very good technical tool for the understanding of effects of actions in multi-agent domains. In fact, the transition function Φ for m$\mathcal{A}^*$ action theories can be effectively characterized using update models, as described in Section 4.

5.2.2. Specifying the Initial State

An important aspect of many algorithms for reasoning about actions and change (including planning) is to have a *finite* set of possible "initial states". Although most of the examples in DEL papers show a finite number of possible initial states (often a single Kripke structure), they do not focus on constraining the knowledge specified about the initial state to guarantee the finiteness of the set of possible initial states. This is an important concern of our paper, and we propose a restricted class of knowledge about the initial states that guarantees finiteness and yet is able to capture most of the examples in the literature. Observe that this condition identifies a class of epistemic planning problems as defined in van der Hoek and Wooldridge (2002); Löwe et al. (2011); **?** whose solutions can be computed using heuristic forward search.

We note that this problem is related to the *finite model property* in modal logics—which defines when a theory has (at least) one finite model Gabbay et al. (2003); van Benthem (2010). The problem addressed in Section **??** could be viewed as the problem of identifying a class of epistemic theories that has (up to equivalence) finitely many finite models and is, thus, a stronger problem than the finite model property problem. To the best of our knowledge, this more complex problem has not been addressed in multi-modal logics before. We believe that this is an important contribution of our development of m$\mathcal{A}^*$.

5.3. Previous Work by the Authors

[4]

[4] We might need to also refer to MA_STRIPS (Brafman and Domshlak, 2008) and perhaps the paper by Nebel and his students (Engesser et al., 2017) and the goal description language of Thielscher (Engesser et al., 2018) – see related work folder

[13] For example, the relevant dictionary meaning of "action is (1) something done or performed; act; deed. (2) an act that one consciously wills and that may be characterized by physical or mental activity.

Early attempts to adapt action languages to formalize multi-agent domains can be found in Baral et al. (2010b); Son et al. (2009); Son and Sakama (2009). In these works, the action languages $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ have been extended to formalize multi-agent domains.

The works in Son et al. (2009); Son and Sakama (2009) investigate the use of action language in multi-agent planning context and focus on the generation of decentralized plans for multiple agents, to either jointly achieve a goal or individual goals.

In Baral et al. (2010b), we show that several examples found in the literature—created to address certain aspect in multi-agent systems (e.g., Boella and van der Torre (2005); Gerbrandy (2006); van der Hoek et al. (2005); Herzig and Troquard (2006); Sauro et al. (2006); Spaan et al. (2006))—can be formalized using an extension of the action language $\mathcal{C}$. Yet, most of the extensions considered in Baral et al. (2010b); Son et al. (2009); Son and Sakama (2009) are inadequate for formalizing multi-agent domains in which reasoning about knowledge of other agents is critical. To address this shortcoming, we have developed and investigated several preliminary versions of m$\mathcal{A}^*$ Baral et al. (2010a); Pontelli et al. (2010); Baral and Gelfond (2010). We started with an attempt to formulate knowledge of multiple agents in Baral et al. (2010a); we successively extended this preliminary version of m$\mathcal{A}^*$ with the use of static observability specifications in Pontelli et al. (2010). The language developed in this paper subsumes that of Pontelli et al. (2010). In Baral and Gelfond (2010), we demonstrated the use of update models to describe the transition function for the action language m$\mathcal{A}^*$ of Pontelli et al. (2010).

5.4. m$\mathcal{A}^*$ and Action Languages for Single-Agent Domains

m$\mathcal{A}^*$ is a high-level action language for multi-agent domains. It is therefore instructive to discuss the connection between m$\mathcal{A}^*$ and action languages for single-agent domains. First, let us observe that m$\mathcal{A}^*$ has the following multi-agent domain specific features:

- it includes announcement actions; and
- it includes specification of the agents' observability of action occurrences.

As it turns out, if we remove all features that are specific to multi-agent domains from m$\mathcal{A}^*$, and consider the **S5**-entailment as its semantics, then the language is equivalent to the language $\mathcal{A}_K$ from Son and Baral (2001). Formally, let us consider a m$\mathcal{A}^*$ definite action theory (I, D) over the signature $\langle \mathcal{AG}, \mathcal{F}, \mathcal{A} \rangle$ such that $|\mathcal{AG}| = 1$ and D does not contain statements of the form (8) (announcement actions) and statements of the form (9)-(10). Let us define

$$I_{\mathcal{A}_K} = \{\varphi \mid \varphi \text{ appears in a statement of the form (1), (**??**), or (2) in } I\}.$$

Then, the following holds

$$(comp(I), D) \models_{\mathbf{S5}} \varphi \textbf{ after } \delta \quad \text{iff} \quad (I_{\mathcal{A}_K}, D) \models_{\mathcal{A}_K} \varphi \textbf{ after } \delta.$$

This shows that m$\mathcal{A}^*$ is indeed a generalization of action languages for single-agent domains to multi-agent domains. This also supports the claim that other elements that have been considered in action languages of single-agent domains, such as static causal laws, non-deterministic actions, or parallel actions could potentially be generalized to m$\mathcal{A}^*$. This is one of our goals for future work.

6. Conclusions and Future Works

In this paper, we developed an action language for representing and reasoning about effects of actions in multi-agent domains. The language considers world-altering actions, sensing actions, and announcement actions. It also allows the dynamic specification of agents' observability with respect to action occurrences, enabling varying degrees of visibility of action occurrences and action effects. The semantics of the language relies on the notion of states (pointed Kripke structures), used as representations of the states of the world and states of agents' knowledge and beliefs; the semantics builds on a transition function, which maps pairs of states and actions to sets of states.

We discussed several properties of the transition function and identified a class of theories (*definite action theories*) whose set of initial **S5**-states is finite, thus allowing for the development of algorithms for the **S5**-entailment relation that is critical in applications such as planning and temporal reasoning. We also relate the proposed language to the update model based approaches for representing and reasoning about effects of actions in multi-agent domains.

The development of m$\mathcal{A}^*$ is our first step towards our goal of developing automated reasoning and planning systems in multi-agent domains. This will be our focus in the near future. In addition, we plan to extend the language to deal with lying and/or misleading actions, refine the distinction between knowledge and beliefs of the agents, expand the language to include non-deterministic actions and static causal laws, and specify more general models of agents' observability, to capture some of the capabilities of update models that are missing from m$\mathcal{A}^*$.

Acknowledgments

References

Agotnes, T., Lakemeyer, G., Löwe, B., Nebel, B., 2014. Planning with epistemic goals. Tech. rep.

Allen, M., Zilberstein, S., 2009. Complexity of decentralized control: Special cases. In: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada. Curran Associates, Inc., pp. 19–27.

Aucher, G., Bolander, T., 2013. Undecidability in epistemic planning. In: Rossi, F. (Ed.), IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013. IJCAI/AAAI.

Baltag, A., Moss, L., 2004. Logics for epistemic programs. Synthese.

Baral, C., Bolander, T., McIlraith, S., Ditmarsch, H. V., 2017. Epistemic planning. Tech. rep. URL `http://www.dagstuhl.de/17231`

Baral, C., Gelfond, G., 2010. On representing actions in multi-agent domains. In: Proceedings of the Symposium on Constructive Mathematics.

Baral, C., Gelfond, G., Pontelli, E., Son, T., 2010a. Modeling multi-agent scenarios involving agents knowledge about other's knowledge using ASP. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, pp. 259–266.

Baral, C., Gelfond, G., Pontelli, E., Son, T. C., 2012. An action language for reasoning about beliefs in multi-agent domains. In: Proceedings of NMR.

Baral, C., Gelfond, G., Pontelli, E., Son, T. C., 2013. Reasoning about the beliefs of agents in multi-agent domains in the presence of state constraints: The action language mal. In: Leite, J., Son, T. C., Torroni, P., van deer Torre, L., Woltran, S. (Eds.), Proceedings of the 14th International Workshop, Computational Logic in Multi-Agent Systems, CLIMA VIX, Coruna, Spain, September 16-18, 2013. Vol. 8143 of Lecture Notes in Computer Science. Springer, pp. 290–306.

Baral, C., Son, T. C., Pontelli, E., 2010b. Reasoning about multi-agent domains using action language $\mathcal{C}$: A preliminary study. In: Dix, J., Fisher, M., Novák, P. (Eds.), Computational Logic in Multi-Agent Systems - 10th International Workshop, CLIMA X, Hamburg, Germany, September 9-10, 2009, Revised Selected and Invited Papers. Vol. 6214 of Lecture Notes in Computer Science. Springer, pp. 46–63.

Bernstein, D. S., Givan, R., Immerman, N., Zilberstein, S., 2002. The complexity of decentralized control of markov decision processes. Math. Oper. Res. 27 (4), 819–840.

Boella, G., van der Torre, L. W. N., 2005. Enforceable social laws. In: Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M. P., Wooldridge, M. (Eds.), 4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands. ACM, pp. 682–689.

Bolander, T., Andersen, M., 2011. Epistemic Planning for Single and Multi-Agent Systems. Journal of Applied Non-Classical Logics 21 (1).

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

Brafman, R. I., Domshlak, C., 2008. From one to many: Planning for loosely coupled multi-agent systems. In: Rintanen, J., Nebel, B., Beck, J. C., Hansen, E. A. (Eds.), Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008. AAAI, pp. 28–35.

Castellini, C., Giunchiglia, E., Tacchella, A., 2001. Improvements to sat-based conformant planning. In: Proceedings of 6th European Conference on Planning (ECP-01).

de Weerdt, M., Bos, A., Tonino, H., Witteveen, C., 2003. A resource logic for multi-agent plan merging. Ann. Math. Artif. Intell. 37 (1-2), 93–130.

de Weerdt, M., Clement, B., 2009. Introduction to planning in multiagent systems. Multiagent Grid Systems 5, 345–355.

del Val A., Shoham, Y., 1994. A unified view of belief revision and update. Journal of Logic and Computation, Special issue on Actions and processes, M. Georgeff (ed.).

Durfee, E., 1999. Distributed Problem Solving and Planning. In: Muliagent Systems (A Modern Approach to Distributed Artificial Intelligence). MIT Press, pp. 121–164.

Engesser, T., Bolander, T., Mattmüller, R., Nebel, B., 2017. Cooperative epistemic multi-agent planning for implicit coordination. In: Ghosh, S., Ramanujam, R. (Eds.), Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017. Vol. 243 of EPTCS. pp. 75–90.

Engesser, T., Mattmüller, R., Nebel, B., Thielscher, M., 2018. Game description language and dynamic epistemic logic compared. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. pp. 1795–1802.

Fagin, R., Halpern, J., Moses, Y., Vardi, M., 1995. Reasoning about Knowledge. MIT press.

Fikes, R., Nilson, N., 1971. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2 (3–4), 189–208.

Gabbay, D., Kurucz, A., Wolter, F., Zakharyaschev, M., 2003. Many-Dimensional Modal Logics: Theory and Application. Elsevier.

Gelfond, M., Lifschitz, V., 1993. Representing actions and change by logic programs. Journal of Logic Programming 17 (2,3,4), 301–323.

Gelfond, M., Lifschitz, V., 1998. Action Languages. Electronic Transactions on Artificial Intelligence 3 (6).

Gerbrandy, J., 2006. Logics of propositional control. In: Nakashima, H., Wellman, M. P., Weiss, G., Stone, P. (Eds.), 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006. ACM, pp. 193–200.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D., 1998. PDDL — the Planning Domain Definition Language. Version 1.2. Tech. Rep. CVC TR98003/DCS TR1165, Yale Center for Comp, Vis and Ctrl.

Goldman, C. V., Zilberstein, S., 2004. Decentralized control of cooperative systems: Categorization and complexity analysis. Journal of Artificial Intelligence Resesearch (JAIR) 22, 143–174.

Guestrin, C., Koller, D., Parr, R., 2001. Multiagent planning with factored mdps. In: Dietterich, T. G., Becker, S., Ghahramani, Z. (Eds.), Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]. MIT Press, pp. 1523–1530.

Herzig, A., Lang, J., Marquis, P., 2005. Action Progression and Revision in Multiagent Belief Structures. In: Sixth Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC).

Herzig, A., Troquard, N., 2006. Knowing how to play: uniform choices in logics of agency. In: Nakashima, H., Wellman, M. P., Weiss, G., Stone, P. (Eds.), 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006. pp. 209–216.

Huang, X., Fang, B., Wan, H., Liu, Y., 2017. A general multi-agent epistemic planner based on higher-order belief change. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17).

Katsuno, H., Mendelzon, A., 1992. On the difference between updating a knowledge base and revising it. In: Proceedings of KR 92. pp. 387–394.

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

Le, T., Fabiano, F., Son, T. C., Pontelli, E., 2018. EFP and PG-EFP: Epistemic Forward Search Planners in Multi-Agent Domains. In: International Confrence on Automated Planning and Scheduling (ICAPS). AAAI Press.

Levesque, H., Reiter, R., Lesperance, Y., Lin, F., Scherl, R., April-June 1997. GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming 31 (1-3), 59–84.

Lifschitz, V., 1987. On the semantics of STRIPS. In: Georgeff, M., Lansky, A. (Eds.), Reasoning about Actions and Plans. Morgan Kaufmann, San Mateo, CA., pp. 1–9.

Liu, Q., Liu, Y., 2018. Multi-agent epistemic planning with common knowledge. In: Lang, J. (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. ijcai.org, pp. 1912–1920.

Löwe, B., Pacuit, E., Witzel, A., 2011. Del planning and some tractable cases. In: van Ditmarsch, H., Lang, J., Ju, S. (Eds.), Logic, Rationality, and Interaction. Vol. 6953 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 179–192.

McCarthy, J., 1959. Programs with common sense. In: Proceedings of the Teddington Conference on the Mechanization of Thought Processes. Her Majesty's Stationery Office, London, pp. 75–91.

Meyer, J.-J., 2000. Dynamic logic for reasoning about actions and agents. In: Minker, J. (Ed.), Logic-Based Artificial Intelligence. Boston/Dordrecht: Kluwer, pp. 281–311, chapter 13.

Muise, C., Belle, V., Felli, P., McIlraith, S., Miller, T., Pearce, A. R., Sonenberg, L., 2015. Planning over multi-agent epistemic states: A classical planning approach. In: Proceedings of AAAI.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D. V., Marsella, S., 2003. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In: Gottlob, G., Walsh, T. (Eds.), IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003. Morgan Kaufmann, pp. 705–711.

Pednault, E., 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In: Brachman, R., Levesque, H., Reiter, R. (Eds.), Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, pp. 324–332.

Peshkin, L., Savova, V., 2002. Reinforcement learning for adaptive routing. In: Proceedings of the Int. Joint Conf. on Neural Networks.

Pontelli, E., Son, T., Baral, C., Gelfond, G., 2010. Logic programming for finding models in the logics of knowledge and its applications: A case study. Theory and Practice of Logic Programming 10 (4-6), 675–690.

Prendinger, H., Schurz, G., 1996. Reasoning about action and change - a dynamic logic approach. Journal of Logic, Language, and Information 5, 5–209.

Reiter, R., 2001. KNOWLEDGE IN ACTION: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press.

Sauro, L., Gerbrandy, J., van der Hoek, W., Wooldridge, M., 2006. Reasoning about action and cooperation. In: AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. ACM, New York, NY, USA, pp. 185–192.

Scherl, R., Levesque, H., 2003. Knowledge, action, and the frame problem. Artificial Intelligence 144 (1-2).

Son, T., Pontelli, E., Sakama, C., 2009. Logic Programming for Multiagent Planning with Negotiation. In: Hill, P. M., Warren, D. S. (Eds.), Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings. Vol. 5649 of Lecture Notes in Computer Science. Springer, pp. 99–114.

Son, T., Sakama, C., 2009. Reasoning and planning with cooperative actions for multiagents using answer set programming. In: Baldoni, M., Bentahar, J., Lloyd, J., van Riemsdijk, B. (Eds.), Declarative Agent Languages and Technologies VI, 6th International Workshop, DALT 2009, Budapest, Hungary, 2009, Revised Selected and Invited Papers. Vol. 5948. Springer, pp. 208–227.

Son, T. C., Baral, C., January 2001. Formalizing sensing actions - a transition function based approach. Artificial Intelligence 125 (1-2), 19–91.

Son, T. C., Pontelli, E., Baral, C., Gelfond, G., 2014. Finitary s5-theories. In: Fermé, E., Leite, J. (Eds.), Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings. Vol. 8761 of Lecture Notes in Computer Science. Springer, pp. 239–252.

Son, T. C., Pontelli, E., Baral, C., Gelfond, G., 2015. Exploring the KD45n Property of a Kripke Model after the Execution of an Action Sequence. In: AAAI.

Son, T. C., Tu, P. H., Gelfond, M., Morales, R., 2005. Conformant Planning for Domains with Constraints — A New Approach. In: Proceedings of the Twentieth National Conference on Artificial Intelligence. pp. 1211–1216.

Spaan, M. T. J., Gordon, G. J., Vlassis, N. A., 2006. Decentralized planning under uncertainty for teams of communicating agents. In: Nakashima, H., Wellman, M. P., Weiss, G., Stone, P. (Eds.), 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006. pp. 249–256.

van Benthem, J., 2007. Dynamic logic of belief revision. Journal of Applied Non-Classical Logics 17(2), 129–155.

van Benthem, J., 2010. Modal Logic for Open Minds. Center for the Study of Language and Information.

van Benthem, J., van Eijck, J., Kooi, B. P., 2006. Logics of communication and change. Inf. Comput. 204 (11), 1620–1662.

van der Hoek, W., Jamroga, W., Wooldridge, M., 2005. A logic for strategic reasoning. In: Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M. P., Wooldridge, M. (Eds.), 4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands. ACM, pp. 157–164.

van der Hoek, W., Wooldridge, M., 2002. Tractable multiagent planning for epistemic goals. In: The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings. ACM, pp. 1167–1174.

van Ditmarsch, H., van der Hoek, W., Kooi, B., 2007. Dynamic Epistemic Logic. Springer.

van Eijck, J., 2004. Dynamic epistemic modelling. Tech. rep.

van Eijck, J., 2017. Public announcements and public lies. Tech. rep., Lying Workshop.

Wan, H., Yang, R., Fang, L., Liu, Y., Xu, H., 2015. A complete epistemic planner without the epistemic closed world assumption. In: Yang, Q., Wooldridge, M. (Eds.), Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. AAAI Press, pp. 3257–3263.
URL http://ijcai.org/proceedings/2015

Appendix A: Proofs of Theorems

Recall that the following notations are used in the presentation of the theorems.

- D denotes a consistent m$\mathcal{A}^*$ domain;
- (M, s) denotes a state; and
- a is an action instance, whose precondition is given by the statement

$$\textbf{executable } \mathsf{a} \textbf{ if } \psi$$

  in D, and a is executable in (M, s).
- $\rho = (F, P, O)$ is the frame of reference of the execution of a in (M, s) where $F = F_D(\mathsf{a}, M, s)$, $P = P_D(\mathsf{a}, M, s)$, and $O = O_D(\mathsf{a}, M, s)$.

Theorem 2. *Assume that* a *is an ontic-action instance. It holds that:*

1. *for every agent* $x \in F_D(\mathsf{a}, M, s)$ *and* $[\mathsf{a}$ **causes** ℓ **if** $\varphi]$ *belongs to* D*, if* $(M, s) \models \mathbf{B}_x\varphi$ *then* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_x\ell$*;*
2. *for every agent* $y \in O_D(\mathsf{a}, M, s)$ *and a belief formula* η*,* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_y\eta$ *iff* $(M, s) \models \mathbf{B}_y\eta$*;*
3. *for every pair of agents* $x \in F_D(\mathsf{a}, M, s)$ *and* $y \in O_D(\mathsf{a}, M, s)$ *and a belief formula* η*, if* $(M, s) \models \mathbf{B}_x\mathbf{B}_y\eta$ *then* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{B}_x\mathbf{B}_y\eta$*.*

Proof. Since a is executable in (M, s), we have that $(M, s) \models \psi$. This means that

$$\Phi_D(\mathsf{a}, (M, s)) = (M, s) \otimes (\omega(\mathsf{a}, \rho), \{\sigma\})$$

where $(\omega(\mathsf{a}, \rho), \{\sigma\})$ is given in Definition 13. Assume that $(M', s') \in \Phi_D(\mathsf{a}, (M, s))$. By Definition 6, we have $s' = (s, \sigma)$. Assume that the fluent in ℓ is p, i.e., $\ell = p$ or $\ell = \neg p$.

1. Let $\Psi^+(p, \mathsf{a}) = \bigvee\{\varphi \mid [\mathsf{a}$ **causes** p **if** $\varphi] \in D\}$ and $\Psi^-(p, \mathsf{a}) = \bigvee\{\varphi \mid [\mathsf{a}$ **causes** $\neg p$ **if** $\varphi] \in D\}$ and $\theta = \Psi^+(p, \mathsf{a}) \vee (p \wedge \neg\Psi^-(p, \mathsf{a}))$. By Definition 13, $p \to \theta \in sub(\sigma)$. Furthermore, for every $u' \in M'[S]$ such that $(s', u') \in M'[x]$, it holds that $u' = (u, \sigma)$ for some $u \in M[S]$, $(M, u) \models \psi$, and $(s, u) \in M[x]$. Because $(M, s) \models \mathbf{B}_x\varphi$, we have that $(M, u) \models \varphi$. Consider two cases:
   - $\ell = p$. Then, $(M, u) \models \Psi^+(p, \mathsf{a})$, and hence, $(M, u) \models \theta$. So, $M'[\pi]((u, \sigma)) \models p$.
   - $\ell = \neg p$. Then, because $(M, u) \models \varphi$, the consistency of D implies that $(M, u) \not\models \theta$. Therefore, $M'[\pi]((u, \sigma)) \not\models p$, i.e., $M'[\pi]((u, \sigma)) \models \neg p$.

   Both cases imply that $M'[\pi]((u, \sigma)) \models \ell$. This holds for every $u' \in M'[S]$ such that $(s', u') \in M'[x]$, which implies $(M', s') \models \mathbf{B}_x\ell$.
2. By the construction of M', we have the following observations:
   - for every $u \in M[S]$ iff $(u, \epsilon) \in M'[S]$;
   - for every $z \in \mathcal{AG}$, $(u, v) \in M[z]$ iff $((u, \epsilon), (v, \epsilon)) \in M'[z]$; and
   - for every $u \in M[S]$ and $p \in \mathcal{F}$, $M'[\pi]((u, \epsilon)) \models p$ iff $(M', (u, \epsilon)) \models p$ because $sub(\epsilon) = \emptyset$.

   These observations allow us to conclude that for every formula η, $(M, u) \models \eta$ iff $(M', (u, \epsilon)) \models \eta$.
   Now, let us get back to the second item of the theorem. Consider $u' \in M'[S]$ such that $(s', u') \in M'[y]$. This holds iff there exists $u \in M[S]$, $(s, u) \in M[y]$, and $u' = (u, \epsilon)$.
   Since $(M, u) \models \eta$ iff $(M', (u, \epsilon)) \models \eta$ and this holds for every $u' \in M'[S]$ such that $(s', u') \in M'[y]$, we have that $(M, s) \models \mathbf{B}_y\eta$ iff $(M', s') \models \mathbf{B}_y\eta$.
3. Consider $u', v' \in M'[S]$ such that $(s', u') \in M'[x]$ and $(u', v') \in M'[y]$. This holds if there exist $u, v \in M[S]$, $(s, u) \in M[x]$ and $(u, v) \in M[y]$ such that $u' = (u, \sigma)$ and $v' = (v, \epsilon)$.
   Assume that $(M, s) \models \mathbf{B}_x\mathbf{B}_y\eta$. This implies that $(M, v) \models \eta$. The second item shows that $(M', (v, \epsilon)) \models \eta$, i.e., which implies $(M', s') \models \mathbf{B}_x\mathbf{B}_y\eta$. Since this holds for every $u', v' \in M'[S]$ such that $(s', u') \in M'[x]$ and $(u', v') \in M'[y]$, we have $(M', s') \models \mathbf{B}_x\mathbf{B}_y\ell$.

Since (M', s') is an arbitrary element in $\Phi_D(\mathsf{a}, (M, s))$, the theorem holds. □

Theorem 3. *Assume that* a *is a sensing action instance and* D *contains the statement* a **determines** f. *It holds that:*

1. *if* $(M, s) \models f$ *then* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{C}_{F_D(\mathsf{a}, M, s)}f$;
2. *if* $(M, s) \models \neg f$ *then* $\Phi_D(\mathsf{a}, (M, s)) \models \mathbf{C}_{F_D(\mathsf{a}, M, s)}\neg f$;

3. $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{C}_{P_D(\mathsf{a},M,s)}(\mathbf{C}_{F_D(\mathsf{a},M,s)}f \vee \mathbf{C}_{F_D(\mathsf{a},M,s)}\neg f)$;
4. $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{C}_{F_D(\mathsf{a},M,s)}(\mathbf{C}_{P_D(\mathsf{a},M,s)}(\mathbf{C}_{F_D(\mathsf{a},M,s)}f \vee \mathbf{C}_{F_D(\mathsf{a},M,s)}\neg f))$;
5. *for every agent* $y \in O_D(\mathsf{a},M,s)$ *and formula* η, $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{B}_y\eta$ *iff* $(M,s) \models \mathbf{B}_y\eta$;
6. *for every pair of agents* $x \in F_D(\mathsf{a},M,s)$ *and* $y \in O_D(\mathsf{a},M,s)$ *and a formula* η *if* $(M,s) \models \mathbf{B}_x\mathbf{B}_y\eta$ *then* $\Phi_D(\mathsf{a},(M,s)) \models \mathbf{B}_x\mathbf{B}_y\eta$.

Proof. We will prove the theorem for the case $(M,s) \models f$. The proof of the theorem when $(M,s) \models \neg f$ is similar and is omitted here. Since a is executable in (M,s), we have that $(M,s) \models \psi$. This means that

$$\Phi_D(\mathsf{a},(M,s)) = M[F_D(\mathsf{a},M,s),f] \otimes (\omega(\mathsf{a},\rho),\{\sigma,\tau\})$$

where $(\omega(\mathsf{a},\rho),\{\sigma,\tau\})$ is given in Definition 14. Let us denote $M[F_D(\mathsf{a},M,s),f]$ with M^*. Observe that by the definition of M^*, for each $x \in F$ there exists some $u \in M^*[x]$ such that $(M^*,u) \models f$.

We need to prove Items 1, 3, 4, 5, and 6.

Assume that $(M',s') \in \Phi_D(\mathsf{a},(M,s))$. By Definition 6, we have $s' = (s,\sigma)$.

1. *Proof of the first item of the theorem.*
   To prove $(M',s') \models \mathbf{C}_F f$, we need to show that

   $$(M',s') \models \mathbf{B}_{i_1}\mathbf{B}_{i_2}\ldots\mathbf{B}_{i_k}f$$

   for any sequence $i_1,\ldots,i_k$ of agents in F, i.e., $i_j \in F$ for $j = 1,\ldots,k$.
   Let $u'_1,\ldots,u'_{k+1} \in M'[S]$ such that $(s',u'_1) \in M'[i_1]$, $(u'_j,u'_{j+1}) \in M'[i_{j+1}]$ for $j = 1,\ldots,k$. Observe that for any $x \in F$ and $u' \in M'[S]$ such that $(s',u') \in M'[x]$, it holds that $u' = (u,\sigma)$ for some $u \in M^*[S]$, $(M^*,u) \models \psi \wedge f$, and $(s,u) \in M^*[x]$.
   This observation allows us to conclude that, for $u'_1,\ldots,u'_{k+1}$, there exist $u_1,\ldots,u_{k+1} \in M^*[S]$ such that $(s,u_1) \in M^*[i_1]$, $(u_j,u_{j+1}) \in M^*[i_{j+1}]$ for $j = 1,\ldots,k$, and for every $j = 1,\ldots,k+1$, $u'_j = (u_j,\sigma)$ and $u_i \models \psi \wedge f$. It is easy to see that this leads to $(M',s') \models \mathbf{C}_F f$.
2. *Proof of the third item of the theorem when* $(M,s) \models f$.
   To prove $(M',s') \models \mathbf{C}_P(\mathbf{C}_F f \vee \mathbf{C}_F\neg f)$, we need to show that

   $$(M',s') \models \mathbf{B}_{i_1}\mathbf{B}_{i_2}\ldots\mathbf{B}_{i_k}(\mathbf{C}_F f \vee \mathbf{C}_F\neg f)$$

   for any sequence $i_1,\ldots,i_k$ of agents in P, i.e., $i_j \in P$ for $j = 1,\ldots,k$.
   Let $u'_1,\ldots,u'_{k+1} \in M'[S]$ such that $(s',u'_1) \in M'[i_1]$, $(u'_j,u'_{j+1}) \in M'[i_{j+1}]$ for $j = 1,\ldots,k$. Similar to the argument in the previous item and Definitions 6 and 14 allows us to conclude that, for $u'_1,\ldots,u'_{k+1}$, there exist $u_1,\ldots,u_{k+1} \in M^*[S]$ such that $(s,u_1) \in M^*[i_1]$, $(u_j,u_{j+1}) \in M^*[i_{j+1}]$ for $j = 1,\ldots,k$, and for every $j = 1,\ldots,k+1$, either (*a*) $u'_j = (u_j,\sigma)$ and $u_i \models \psi \wedge f$ or (*b*) $u'_j = (u_j,\tau)$ and $u_i \models \psi \wedge \neg f$. This leads to two cases:
   (a) $u'_{k+1} = (u_{k+1},\sigma)$ and $u_{k+1} \models \psi \wedge f$. Then, similar to the proof in Item 1, we can show that $(M',u'_{k+1}) \models C_F f$.
   (b) $u'_{k+1} = (u_{k+1},\sigma)$ and $u_{k+1} \models \psi \wedge \neg f$. Again, similar to the proof in Item 1, we can show that $(M',u'_{k+1}) \models C_F\neg f$.

The two cases imply that $(M', s') \models \mathbf{C}_P(\mathbf{C}_F f \vee \mathbf{C}_F \neg f)$.

3. *Proof of the fourth item of the theorem when* $(M, s) \models f$.
   To prove $(M', s') \models \mathbf{C}_F(\mathbf{C}_P(\mathbf{C}_F f \vee \mathbf{C}_F \neg f))$, we need to show that

$$(M', s') \models \mathbf{B}_{i_1}\mathbf{B}_{i_2} \ldots \mathbf{B}_{i_k}(\mathbf{C}_P(\mathbf{C}_F f \vee \mathbf{C}_F \neg f))$$

   for any sequence $i_1, \ldots, i_k$ of agents in F, i.e., $i_j \in F$ for $j = 1, \ldots, k$. This holds because we can show that for each $u' = (u, \sigma)$ such that $u \in M^*[S]$ and $(M^*, u) \models \psi \wedge f$, $(M', u') \models \mathbf{C}_P(\mathbf{C}_F f \vee \mathbf{C}_F \neg f)$. The arguments for this conclusion are similar to the arguments used in the proof in Item 2.
4. The proof of the fifth and sixth items of this theorem is similar to the proof of the second and third item of Theorem 2, respectively.

Since (M', s') is an arbitrary element in $\Phi_D(\mathsf{a}, (M, s))$, the theorem holds. □

Appendix B: Examples of m$\mathcal{A}^*$ Domains

In this appendix, we illustrate the use of m$\mathcal{A}^*$ in representing multi-agent domains. The next example represents a domain in which the agent who executes an action might not be a full observer of the action occurrence.

Example 11. *Let us consider a domain with two agents $\mathcal{A} = \{A, B\}$. The two agents are operating in a room; agent A is blind while B is not. Both agents are aware that by flipping a switch it is possible to change the status of the light in the room, and both agents can perform such action. On the other hand, the effect of the execution of the action will be visible only to B. This action can be described by the following statements of m$\mathcal{A}^*$:*

$$\begin{array}{l} \textit{flip}\langle x\rangle \textbf{ causes } \textit{on} \textbf{ if } \neg\textit{on} \\ \textit{flip}\langle x\rangle \textbf{ causes } \neg\textit{on} \textbf{ if } \textit{on} \\ B \textbf{ observes } \textit{flip}\langle x\rangle \end{array}$$

We will next describe several examples that contain common actions that are typical to multi-agent domains in m$\mathcal{A}^*$. We refer to these actions as *reference setting actions*, such as the action of distracting another agent. This type of actions is interesting, because it is often necessary for agents to execute them in order to allow subsequent actions to achieve their intended effects (e.g., sharing a secret).

Example 12 (Distraction)**.** *Agent A wishes to access some files on C's computer without C's knowledge. Agent C is rather observant, therefore in order for A to succeed, she must first cause a distraction (such as pulling the fire alarm) in order to pull C's attention away from the computer. Once C is distracted, A may access the file on the computer. This can be described by the following statements:*

$$\begin{array}{l} \textit{distract}(C)\langle A\rangle \textbf{ causes } \textit{distracted}(C) \\ A \textbf{ observes } \textit{distract}(C)\langle A\rangle \\ C \textbf{ observes } \textit{distract}(C)\langle A\rangle \\ \textbf{executable } \textit{accessFile}(C)\langle A\rangle \textbf{ if } \textit{distracted}(C) \end{array}$$

The action that helps an agent to form or dissolve a group is also frequently needed in multi-agent domains. Groups enable, for example, the execution of communications that are only local to the group (e.g., a secret conversation).

Example 13 (Group Formation/Dissolution and Secret Communication)**.** *Continuing with Example 12, now that A has access to the file, she needs the assistance of agent B to learn its contents because the file is encrypted and the expertise of B is required for decryption. In order to read the file, she must first establish a connection with B—agent A must first open/invite a communications channel to B. Let $linked(x, y)$ denote that X and Y are connected and $distracted(Z)$ that Z is distracted. This action of A inviting B to connect via some communication channel can be represented using the action $openChannel(B)$ with the following following specification:*

$$
\begin{array}{l}
\textit{openChannel}(B)\langle A\rangle \textbf{ causes } \textit{linked}(A,B)\\
A \textbf{ observes } \textit{openChannel}(B)\langle A\rangle\\
B \textbf{ observes } \textit{openChannel}(B)\langle A\rangle\\
C \textbf{ observes } \textit{openChannel}(B)\langle A\rangle \textbf{ if } \neg\textit{distracted}(C)\\
\textbf{executable } \textit{openChannel}(B)\langle A\rangle \textbf{ if } \neg\textit{linked}(A,B)
\end{array}
$$

Once a channel has been opened, agents A and B are linked and they may together read the file. Once they have read the file, they disconnect the channel in order to leave no trace of their activity. This action can be represented using the action $closeChannel(B)$ with the following specification:

$$
\begin{array}{l}
\textit{closeChannel}(B)\langle A\rangle \textbf{ causes } \neg\textit{linked}(A,B)\\
A \textbf{ observes } \textit{closeChannel}(B)\langle A\rangle\\
B \textbf{ observes } \textit{closeChannel}(B)\langle A\rangle\\
C \textbf{ observes } \textit{closeChannel}(B)\langle A\rangle \textbf{ if } \neg\textit{distracted}(C)\\
\textbf{executable } \textit{closeChannel}(B)\langle A\rangle \textbf{ if } \textit{linked}(A,B)
\end{array}
$$

Reading the file allows A and B to understand its content. Let us assume that the file indicates whether tomorrow is the established date of a cyberattack against A's organization. This can be represented as follows.

$$
\begin{array}{l}
\textit{readFile}\langle A\rangle \textbf{ determines } \textit{attackDate}(\textit{tomorrow})\\
A \textbf{ observes } \textit{readFile}\langle A\rangle\\
B \textbf{ observes } \textit{readFile}\langle A\rangle \textbf{ if } \textit{linked}(A,B)\\
C \textbf{ aware\_of } \textit{readFile}\langle A\rangle \textbf{ if } \neg\textit{distracted}(C)\\
\textbf{executable } \textit{readFile}\langle A\rangle \textbf{ if } \textit{linked}(A,B)
\end{array}
$$

If a channel is open, it can be used to share the knowledge of an impeding attack. However, the communication is secure only if the third party is distracted. This action is an announcement action and can be represented using the following statements.

$$
\begin{array}{l}
\textit{warnOfAttack}(B)\langle A\rangle \textbf{ announces } \textit{attackDate}(\textit{tomorrow})\\
A \textbf{ observes } \textit{warnOfAttack}(B)\langle A\rangle\\
B \textbf{ observes } \textit{warnOfAttack}(B)\langle A\rangle\\
C \textbf{ observes } \textit{warnOfAttack}(B)\langle A\rangle \textbf{ if } \neg\textit{distracted}(C)\\
\textbf{executable } \textit{warnOfAttack}(B)\langle A\rangle \textbf{ if } \textit{linked}(A,B) \wedge \textit{attackDate}(\textit{tomorrow})
\end{array}
$$

A more general version of the actions of secretly creating/dissolving a group is given in the next example.

Example 14. *Consider an agent X joining an agent Y to gain visibility of everything that the agent X does; this could be modeled by the action join(X), where an agent joins X at the same level of visibility of X's actions:*

$$\begin{array}{l} \textit{join}(X)\langle Y\rangle \textbf{ causes } \textit{group\_member\_X}(Y) \\ X \textbf{ observes } \textit{join}(X)\langle Y\rangle \\ Y \textbf{ observes } \textit{join}(X)\langle Y\rangle \end{array}$$

This could be refined by adding the need to be invited to join X:

$$\textbf{executable } \textit{join}(Y)\langle A\rangle \textbf{ if } \textit{invited}(Y, X)$$

The effect of gaining visibility of the actions of X can be described by

$$Y \textbf{ observes } a\langle X\rangle \textbf{ if } \textit{group\_member\_X}(Y)$$

where $a\langle X\rangle$ is any action occurrence executed by agent X. The symmetrical operation is the operation of leaving the group, leading the agent Y to completely loose visibility of what agent X is doing:

$$\begin{array}{l} \textit{leave}(X)\langle Y\rangle \textbf{ causes } \neg\textit{group\_member\_X}(Y) \\ X \textbf{ observes } \textit{leave}(X)\langle Y\rangle \\ Y \textbf{ observes } \textit{leave}(X)\langle Y\rangle \end{array}$$

The agent Y may also decide to take the action of separating from the group, through the action separate(X), where the agent Y will observe X from "a distance", with the consequent loss of the intimate knowledge of X actions' effects:

$$\begin{array}{l} \textit{separate}(X)\langle Y\rangle \textbf{ causes } \neg\textit{group\_member\_X}(Y) \wedge \textit{group\_observer\_X}(Y) \\ X \textbf{ observes } \textit{separate}(X)\langle Y\rangle \\ Y \textbf{ observes } \textit{separate}(X)\langle Y\rangle \\ Y \textbf{ aware\_of } a\langle X\rangle \textbf{ if } \textit{group\_observer\_X}(Y) \end{array}$$

Distracting an agent is not only necessary for secure communication, it is also important for certain world-altering actions to achieve their intended effects, as in Example 12. Another example that highlights the importance of this type of actions can be seen next.

Example 15. *Agent D is a prisoner, having been captured by C. Agent A is charged with rescuing agent D. In order to do so, he must first distract C, and then trigger a release on C's computer. Once the release has been triggered, D may escape. The distract(A, C) action has already been presented in Example 12. Let us consider the other actions. Rescuing an agent means that the*

rescued agent is released. We use the following statements:

$$\begin{array}{l} \textit{rescue}(D)\langle A\rangle \textbf{ causes } \textit{released}(D) \\ A \textbf{ observes } \textit{rescue}(D)\langle A\rangle \\ D \textbf{ observes } \textit{rescue}(D)\langle A\rangle \\ C \textbf{ observes } \textit{rescue}(D)\langle A\rangle \textbf{ if } \neg\textit{distracted}(C) \\ \textbf{executable } \textit{rescue}(D)\langle A\rangle \textbf{ if } \textit{distracted}(C) \end{array}$$

The action of escaping can have different effects.

$$\begin{array}{l} \textit{escape}\langle D\rangle \textbf{ causes } \textit{dead}(D) \textbf{ if } \neg\textit{distracted}(C) \\ \textit{escape}\langle D\rangle \textbf{ causes } \textit{free}(D) \textbf{ if } \textit{distracted}(C) \\ A \textbf{ observes } \textit{escape}\langle D\rangle \\ D \textbf{ observes } \textit{escape}\langle D\rangle \\ C \textbf{ observes } \textit{escape}\langle D\rangle \textbf{ if } \neg\textit{distracted}(C) \end{array}$$

Reference setting actions can be used in modeling group formation activities.

Example 16. *Consider the join and leave actions from Example 14. The frame setting changes of these actions can be summarized as in Figure 12, where $a\langle X\rangle$ denotes the execution of action a by X and (M', s') denotes the state resulting from the execution of the corresponding action in (M, s).*

in (M, s)	$\xrightarrow{\textit{join}(X)\langle Y\rangle}$	*in* (M', s')
$F_D(a\langle X\rangle, M, s)$		$F_D(a\langle X\rangle, M, s) \cup \{Y\}$
$P_D(a\langle X\rangle, M, s)$		$P_D(a(\langle X\rangle, M, s) \setminus \{Y\}$
$O_D(a\langle X\rangle, M, s)$		$O_D(a\langle X\rangle, M, s) \setminus \{Y\}$
in (M, s)	$\xrightarrow{\textit{leave}(X)\langle Y\rangle}$	*in* (M', s')
$F_D(a\langle X\rangle, M, s)$		$F_D(a\langle X\rangle, M, s) \setminus \{Y\}$
$P_D(a\langle X\rangle, M, s)$		$P_D(a\langle X\rangle, M, s)$
$O_D(a\langle X\rangle, M, s)$		$O_D(a\langle X\rangle, M, s) \cup \{Y\}$
in (M, s)	$\xrightarrow{\textit{separate}(X)\langle Y\rangle}$	*in* (M', s')
$F_D(a\langle X\rangle, M, s)$		$F_D(a\langle X\rangle, M, s) \setminus \{Y\}$
$P_D(a\langle X\rangle, M, s)$		$P_D(a\langle X\rangle, M, s) \cup \{Y\}$
$O_D(a\langle X\rangle, M, s)$		$O_D(a\langle X\rangle, M, s)$

Figure 12: Frame of reference for $a\langle X\rangle$ before (in (M, s), left column) and after the execution of a joint/leave/separate action (in (M', s'), right column)

This article was processed using the comments style on December 11, 2019.
There remain 4 comments to be processed.