Representing and Reasoning about Dynamic Multi-Agent Domains:

An Action Language Approach

by

Gregory Gelfond

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2018 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Subbarao Kambhampati
Joohyung Lee
Larry Moss
Tran Cao Son

ARIZONA STATE UNIVERSITY

May 2018

ABSTRACT

Reasoning about actions forms the basis of many tasks such as prediction, planning, and diagnosis in a dynamic domain. Within the reasoning about actions community, a broad class of languages, called *action languages*, has been developed together with a methodology for their use in representing and reasoning about dynamic domains. With a few notable exceptions, the focus of these efforts has largely centered around single-agent systems. Agents rarely operate in a vacuum however, and almost in parallel, substantial work has been done within the dynamic epistemic logic community towards understanding how the actions of an agent may effect not just his own knowledge and/or beliefs, but those of his fellow agents as well. What is less understood by both communities is how to *represent and reason* about both the *direct and indirect effects* of both *ontic* and *epistemic* actions within a multi-agent setting. This dissertation presents ongoing research towards a framework for representing and reasoning about dynamic multi-agent domains involving both classes of actions.

The contributions of this work are as follows: the formulation of a precise mathematical model of a dynamic multi-agent domain based on the notion of a transition diagram; the development of the multi-agent action languages $m\mathcal{A}+$ and $m\mathcal{AL}$ based upon this model, as well as preliminary investigations of their properties and implementations via logic programming under the answer set semantics; precise formulations of the temporal projection, and planning problems within a multi-agent context; and an investigation of the application of the proposed approach to the representation of, and reasoning about, scenarios involving the modalities of knowledge and belief.

ACKNOWLEDGMENTS

I learned some time ago that one of the keys to happiness is having something that you are grateful for. During the course of this work I have been blessed by G-d with many people to (and for) whom my gratitude knows no bounds.

I would like to begin by thanking my advisor Dr. Chitta Baral for his patience, guidance, support, and freedom to pursue my research interests. I would also like to humbly thank my committee members, Dr. Subbarao Kambhampati, Dr. Joohyung Lee, Dr. Larry Moss, and Dr. Tran Cao Son for pushing me to explore certain questions more deeply, and for their generous feedback. I have learned a great deal from all of them, and working with them has been a pleasure.

I would also like to thank my parents and family. Without them, and their patience, love, and support, I would not have been able to persevere and reach this point. I have been truly blessed in this life with parents who have been my greatest source of inspiration and strength, and who have been among my greatest teachers and closest friends. I would also like to thank my sister Yulia and her husband Patrick. In addition to being a constant source of encouragement they have brought into this world my niece Alexandra and my nephew Jonathan who are a great source of joy in my life.

There are too many people to whom I am grateful to name here: members of my family, close friends, and even my students, all of whom have taught me much about both the world and myself. Suffice it to say that I am in your debt and eternally grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Reasoning about actions and change has been one of the cornerstones of artificial intelligence research ever since McCarthy's description of the "advice taker system" [35]. Since that time, a considerable body of work on a broad class of languages, called *action languages*, together with a methodology for their use has been developed [2, 26, 27]. A distinguishing characteristic of such languages is their simple syntax and semantics, based on a careful subset of natural language, which allow for concise and natural representations of huge transition systems, as well as elegant solutions to both the frame and ramification problems [5, 16, 26, 32, 34, 36, 37]. With a few notable exceptions, [16, 33], the focus of such languages has been on representing an agent's knowledge concerning *ontic actions* (i.e. those which primarily affect the physical environment). Agents rarely operate in isolation, oftentimes needing to exchange information, and consequently almost in parallel, substantial work has been done within the dynamic epistemic logic community towards understanding *epistemic actions* (i.e. those which primarily affect an agent's knowledge or beliefs) [4, 19, 44] and to a lesser extent *ontic actions* [43]. What is less understood by both communities is how to *represent and reason* about the *direct and indirect effects* of both classes of actions in a multi-agent setting. In the following sections we briefly present some of the questions that this dissertation seeks to begin answering in the context of the Classical Muddy Children Problem [19], and a multi-agent extension of the Lin's Briefcase Problem [12, 32] (more detailed examinations of these and other domains will be presented later in this dissertation).

1.1.1 The Classical Muddy Children Problem

The Classical Muddy Children Problem [19], presented in Example 1.1, is a canonical multi-agent domain illustrating how both the *individual* and *collective knowledge* of the agents in a domain changes as a consequence of communication.

Example 1.1 (The Classical Muddy Children Problem)**.** *N children are playing together outside and during their play, some of them get mud on their foreheads. Each child can see the mud on the foreheads of their siblings, but cannot tell whether or not they themselves are muddy. This fact is common knowledge among them. Their father comes along and announces that at least one of them is muddy. He then repeatedly asks "do you know whether or not you are muddy" until all of the children reply in the affirmative. Assuming that all of the children are intelligent, honest, and answer in unison, it can be shown that if* K *children are muddy, then after the* K^{th} *time the father asks his question, the children will all reply "yes". In this dissertation we concern ourselves with an instance of this problem where* $N = K = 3$. ◇

Within the dynamic epistemic logic community this particular domain is modeled by a sequence of graphs, called Kripke worlds, shown in Figures 1.1 to 1.4. In this representation, there are three children A, B, and C, and each node within a Kripke world defines a possible physical configuration of the world (node 111 denotes the possibility that all of the children are muddy while node 101 denotes the possibility that only A and C are). The edges of a Kripke world are labeled by agents[1], and an edge of the form $(\omega_1, \alpha, \omega_2)$ is read[2] as: "If α inhabits possible world ω_1, then he understands himself to be in possible world ω_2." In this

[1] Unlabeled edges are understood to be labeled by the set of all agents within the domain.

[2] The reading given here is taken from [19]; later in this work an alternative reading is proposed.

way, Kripke worlds represent both the *ontic* and *epistemic* properties of a *state of the domain*. Each interaction between the father and the children removes certain possible worlds from consideration, as shown in the aforementioned figures.

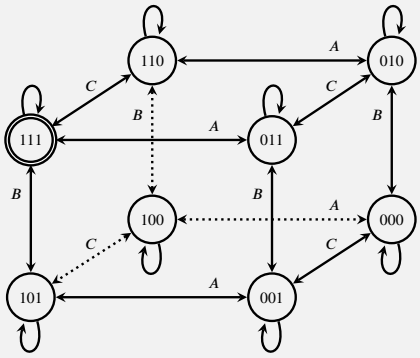

Figure 1.1: A Kripke world depicting the initial state of the Classical Muddy Children Problem.

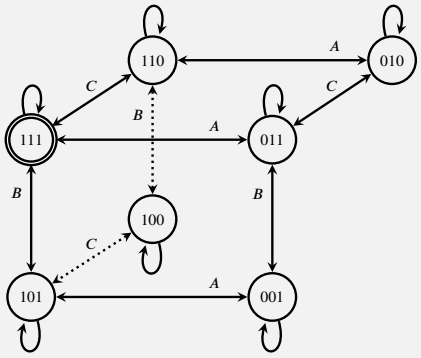

Figure 1.2: A Kripke world depicting the consequences of the father's announcement that at least one of the children is muddy.

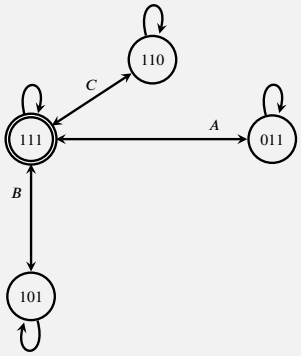

Figure 1.3: A Kripke world depicting the results of the children's first announcement that they do not know whether they're muddy or not.

Figure 1.4: A Kripke world depicting the results of the children's second announcement that they do not know whether they're muddy or not.

This approach seems to suggest that as is the case with single-agent domains, a multi-agent one may be described as a transition system, whose nodes represent states of the

domain, and whose edges represent actions [2, 26, 27]. Unlike the single-agent case however, in a multi-agent system, states of the domain are modeled by Kripke world, a more complex kind of object than a set of fluents. Another key distinction, is that actions such as a public announcement don't have what are called *ontic effects*, rather their effects are *epistemic* in quality, meaning they alter what the agents know/believe about the domain, rather than its physical properties.

From a knowledge representation standpoint, this raises a number of interesting questions:

- How can we define a language that will allow us to clearly, concisely, and in an elaboration tolerant manner, describe transition systems of this kind?
- Can our descriptions be written in such a way as to shorten the distance between the specification of a multi-agent system and its representation?
- In addition to public announcements, what other categories of actions are there that distinguish multi-agent domains from single-agent ones?
- Can we define the semantics of our language(s) in such a way as to be amenable to translation into a logic program, with the ultimate goal of automating various reasoning tasks such as *temporal projection*, *planning*, and *diagnostic reasoning*.

Each of these questions will be elaborated upon in subsequent chapters. Before we continue however, we present a similar discussion regarding a multi-agent extension of the Lin's Briefcase Domain of [32].

1.1.2 A Multi-Agent Lin's Briefcase Domain

Within the knowledge representation community, the Lin's Briefcase Domain of [32] was an important thought experiment capturing some of the difficulties in reasoning about

both the *direct* and *indirect* effects of actions. A multi-agent variant of this domain, first presented in [12], is shown in Example 1.2[3].

Example 1.2 (Multi-Agent Lin's Briefcase Domain)**.** *Three agents, A, B, and C, are together in a room with a locked briefcase which contains a coin. The briefcase is locked by two independent latches, each of which may be flipped open (or closed) by an agent. Once both latches are open, the briefcase is unlocked and an agent may peek inside to determine which face of the coin is showing.* ◇

This domain seems rather simple from the standpoint of a human reasoner, but poses some additional questions to those discussed in the prior section. Suppose that agent A flips the latch l_1 open. Intuitively, we ought to be able to conclude that l_1 is open, as a *direct effect* of this action, but beyond that, there are a number of possible *indirect effects*. For example:

- If l_2 was previously open, we ought to be able to conclude that the briefcase is now unlocked as well.
- If agents B and C observe agent A flipping l_1, they ought to be aware that l_1 is open, and potentially that the briefcase is unlocked.
- If agents B and C fail to observe agent A flipping l_1, their knowledge/beliefs regarding the world ought to remain as they are.
- Etc.

Prior work on modeling single-agent domains [26, 27, 32] led to the development of the action language $\mathcal{AL}$ which allows for the representation of both the *direct* and *indirect* effects of actions. In a multi-agent context, actions have indirect effects which not only affect

[3] A full treatment of this domain is left to Chapter 4.

a domain's ontic properties (e.g., indirectly causing the briefcase to become unlocked), but also may alter it's epistemic properties (e.g., altering what agents B and C know/believe about the briefcase). This naturally leads to the following additional research question: Can we define an action language which will allow us to model the direct and indirect effects of actions in a similar fashion to $\mathcal{AL}$ but in a multi-agent context?

1.2 Research Goals

The goal of this research has been to begin answering the questions posed in Sections 1.1.1 and 1.1.2 by combining the reasoning methodologies developed by both the action language and dynamic epistemic logic communities, leading to the development of the multi-agent action languages $m\mathcal{A}+$ [11] and $m\mathcal{AL}$ [12], which allow for representing and reasoning about dynamic multi-agent domains involving both ontic and epistemic actions. More specifically, the previous questions have been refined to the following:

- What *classes of actions* distinguish multi-agent domains from single-agent ones?
- Can we formalize a *high-level action language* for representing dynamic multi-agent domains such as the Classical Muddy Children Problem or Multi-Agent Lin's Briefcase Domain?
- Can we formalize the *temporal projection*, *planning*, and *diagnostic reasoning* problems for multi-agent domains in a similar fashion to that in [2, 26, 27]?
- Can logic programming under the answer set semantics be reasonably applied to representing and reasoning about dynamic multi-agent domains? In particular, can we automate the reasoning tasks of *temporal projection* and *planning* through the application of answer set programming techniques?

The rest of this dissertation is structured as follows: We begin by presenting the relevant background material, covering the basics of action languages [23], the language of update models [4, 43], and answer set programming [6, 21, 24, 25, 31]. Once this foundation has been established, we proceed with a discussion of the action language $m\mathcal{A}+$, a multi-agent extension of the action language $\mathcal{A}$ [27], which was first proposed in a primitive fashion in [7] and subsequently defined in [11]. We then present the language's application to representing and reasoning about multi-agent domains such as the Classical Muddy Children Problem. We next note that certain kinds of domains, such as the multi-agent variant of the classical Lin's Briefcase Domain [12, 32] are difficult to model in $m\mathcal{A}+$ due to the lack of a construct known as a *state constraint*. This leads us to a presentation of the action language $m\mathcal{AL}$, a multi-agent variant of the language $\mathcal{AL}$ [23, 27], followed by a detailed example of its application for representing and reasoning about multi-agent domains for which $m\mathcal{A}+$ is insufficiently expressive; Finally, we present a set of logic programs under the answer set semantics for the purpose of solving the *temporal projection* and *planning* problems in $m\mathcal{A}+$, and conclude with a discussion of future work.

Chapter 2

BACKGROUND

In this chapter we discuss a number of fundamental concepts from both the *reasoning about actions*, and *dynamic epistemic logic communities* which form the foundation of this work. Section 2.1 presents an overview of the action language $\mathcal{AL}$ and its application towards modeling dynamic single-agent domains. In Section 2.2 we discuss the dynamic epistemic logic approach towards modeling dynamic multi-agent domains via modal logic and the language of update models. Lastly, in Section 2.3 we cover logic programming under the answer set semantics, which provides the implementation framework for automating various reasoning tasks.

2.1 The Action Language $\mathcal{AL}$

The action language $\mathcal{AL}$ is one of many languages used for representing dynamic domains. It is based in part on the notion that such a domain may be modeled by a *transition diagram*, whose nodes correspond to possible worlds, and whose edges are labeled by actions. Such transition systems may be quite large, and one of the key challenges that has been met by the reasoning about actions community has been the creation of languages capable of representing such transition systems in a compact, intuitive, and elaboration tolerant manner. What follows is a brief introduction to the language taken in part from [23].

2.1.1 Syntax

In order to describe the syntax of $\mathcal{AL}$, we must first precisely define what is meant by a single-agent domain.

Definition 2.1 (Single-Agent Domain)**.** A *single-agent domain*, $\mathcal{D}$, defines a signature $\Sigma = (\mathcal{F}, \mathcal{A})$ where $\mathcal{F}$ and $\mathcal{A}$ are disjoint sets of symbols respectively defining the *ontic properties of the domain* (or *fluents*), and the *elementary actions* an agent may perform.[1] ◇

Example 2.2 (Lin's Briefcase Domain)**.** *Consider the following single-agent domain from [32]: Agent A is in a room containing a locked briefcase. The lock is governed by two independent latches, each of which may be flipped open (or closed). Once both latches are opened, the briefcase is unlocked and may be opened. This scenario may be modeled (in part) by a single-agent domain, $\mathcal{D}$, with the following signature:*

$$\mathcal{F} = \{open(l_1), open(l_2), locked\}$$

$$\mathcal{A} = \{flip(l_1), flip(l_2), open\}$$

◇

In the Lin's Briefcase Domain there are three fluents of interest, respectively denoting whether or not the two latches are open, and whether or not the briefcase is locked. Along the same lines, there are three actions which the agent may perform: flipping the respective latches open or closed, and opening the briefcase.

Generally speaking, actions may have both *direct* and *indirect* effects. The direct effects of actions are described by *dynamic causal laws* which are statements of the form:

$$a \textbf{ causes } \lambda \textbf{ if } \phi \tag{2.1}$$

[1] $\mathcal{AL}$ also supports so-called *compound actions* but for the purposes of this work, the discussion is limited to elementary ones.

where a is an action, λ is a fluent literal, and ϕ is a collection of fluent literals denoting the law's preconditions. Laws of this form are informally read as: "performing the action a in a state which satisfies ϕ causes λ to be true." If ϕ is empty, then we simply write the following:

$$a \textbf{ causes } \lambda \tag{2.2}$$

The indirect effects of actions, as well as dependencies between fluents, are described by *state constraints* (also known as *static causal laws*) which are statements of the form:

$$\lambda \textbf{ if } \phi \tag{2.3}$$

where λ is a fluent literal, and ϕ is a conjunction of fluent literals. Statements of this form are read as: "if ϕ is true in a given state, then λ must also be true in that state."

Finally, the actions an agent may perform are oftentimes constrained by his environment. Such constraints are represented by *executability conditions* which have the form:

$$\textbf{impossible } a \textbf{ if } \phi \tag{2.4}$$

where a is an elementary action and ϕ is a collection of fluent literals, are used to describe when an action may not be performed. Such statements are read as: "if ϕ is true in a state, then the action a may not be performed."

Definition 2.3 (Action Description of $\mathcal{AL}$)**.** An *action description of* $\mathcal{AL}$ is a collection of statements of the form (2.1) – (2.4). ◇

Example 2.4 (Lin's Briefcase Domain — Actions and their Effects)**.** *Recall the Lin's Briefcase Domain from Example 2.2: a single agent, A, is in a room containing a locked briefcase. The lock is governed by two independent latches, each of which may be flipped open (or closed). Once both latches are opened, the briefcase is unlocked and may be opened.*

Let us use the signature defined in Example 2.2, and let λ be a variable over the set $\{l_1, l_2\}$ representing the latches governing the briefcase. The direct effects of the action flip(λ) are represented via the following pair of dynamic causal laws:

$$flip(\lambda) \textbf{ causes } open(\lambda) \textbf{ if } \neg open(\lambda) \tag{2.5}$$

$$flip(\lambda) \textbf{ causes } \neg open(\lambda) \textbf{ if } open(\lambda) \tag{2.6}$$

Depending on the state in which the action flip(λ) occurs, the action may have the indirect effect of unlocking the briefcase. The following state constraint models the dependency between the fluent locked and the fluents open(l_1) and open(l_2):

$$\neg locked \textbf{ if } open(l_1) \wedge open(l_2) \tag{2.7}$$

Finally we represent the constraint that the agent may only open the briefcase if it is unlocked, by the following executability condition:

$$\textbf{impossible } open \textbf{ if } locked \tag{2.8}$$

$\diamond$

With the syntax of $\mathcal{AL}$ firmly in place, we now turn our attention towards its semantics.

2.1.2 Semantics

An action description of $\mathcal{AL}$ describes a transition diagram, whose nodes are complete consistent sets of fluent literals which satisfy all of the state constraints of the action description, and whose edges are labeled by actions. We begin our discussion of the semantics by introducing some notation.

Let Δ be an action description of $\mathcal{AL}$ over a signature Σ, and S be a set of fluent literals of Σ. By $\mathbf{Cn}_\Delta(S)$, we denote the smallest set of fluent literals of Σ that contains S, and

satisfies all of the state constraints of Δ. By $E(\sigma, a)$ we denote the set of all fluent literals, λ, such that Δ contains a dynamic causal law, *a* ***causes*** λ ***if*** ϕ, where $\phi \subseteq \sigma$. Intuitively, this set represents all of the direct effects of performing the action a in the state σ. With this notation in place, we may now define the transition system described by an action description of $\mathcal{AL}$.

Definition 2.5 (The Transition System)**.** Let Δ be an action description of $\mathcal{AL}$. The *transition system*, $\mathcal{T} = (\mathcal{S}, \mathcal{R})$, described by Δ is defined as follows:

- $\mathcal{S}$ is the set of all complete and consistent sets of fluent literals of Σ which satisfy the state constraints of Δ.
- $\mathcal{R}$ is the set of all triples of the form (σ, a, σ') such that Δ does not contain an executability condition, ***impossible*** *a* ***if*** ϕ, where $\phi \subseteq \sigma$ and:

$$\sigma' = \mathbf{Cn}_{\Delta}(E(\sigma, a) \cup (\sigma \cap \sigma')) \tag{2.9}$$

$\diamond$

Equation (2.9) is known as the McCain-Turner equation and is the product of considerable research on the nature of causality [34]. Intuitively, the arguments to $\mathbf{Cn}_{\Delta}$ combine the *direct effects* of the action, $E(\sigma, a)$, with those properties of the domain which carry over due to *inertia*, $(\sigma \cap \sigma')$. The consequence operator, $\mathbf{Cn}_{\Delta}$, extrapolates the *indirect effects* of the action in accordance with the state constraints of Δ.

Example 2.6 (Transition Diagram)**.** *The action description shown in Example 2.4 defines a transition diagram shown in part in Figure 2.1.* $\diamond$

Having finished our discussion of action languages, we now present the dynamic epistemic logic approach for modeling dynamic multi-agent domains.

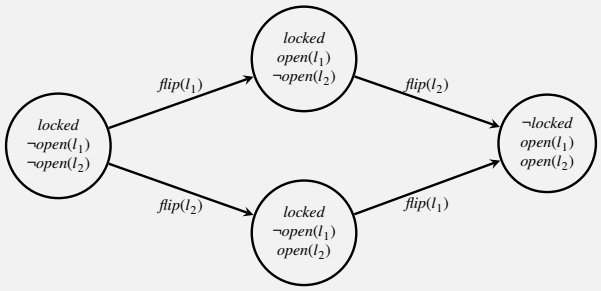

Figure 2.1: Partial transition diagram defined by the Lin's Briefcase Domain.

2.2 Dynamic Epistemic Logic

The action language approach for representing and reasoning about dynamic single agent domains largely focuses on modeling the *ontic properties* of a domain. Almost in parallel, the field of dynamic epistemic logic arose which was largely concerned with representing a dynamic domain's *epistemic properties*. Towards this end, a number of techniques of representing dynamic multi-agent domains have been developed, among them being the modal logics of knowledge and belief, and the language of event/update models. In this section we provide an overview of both of these in turn.

2.2.1 Modal Logic

In order to reason about a multi-agent domain, we need a language capable of expressing both its ontic and epistemic properties. Modal logic provides just such a language. Before we begin however, we must first define what we mean by a multi-agent domain.

Definition 2.7 (Multi-Agent Domain)**.** A *multi-agent domain*, $\mathcal{D}$, is defined over a signature $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$ where $\mathcal{AG}$, $\mathcal{F}$, and $\mathcal{A}$, are finite, disjoint, non-empty sets of symbols

respectively defining the *names of the agents within the domain*, the *properties of the domain* (or *fluents*), and the *elementary actions* which the agents may perform[2]. ◇

2.2.1.1 Syntax

The Concealed Coin Domain [4], a variation of which is presented in Example 2.8, is a canonical multi-agent domain used to illustrate how the knowledge/beliefs of the agents may change as a consequence of the actions they perform.

Example 2.8 (Concealed Coin Domain)**.** *Three agents, A, B, and C, are together in a room with a locked box which contains a coin. This fact, together with the fact that none of them knows which face of the coin is showing is common knowledge amongst them. An agent may unlock the box, as well as peek inside to determine which face of the coin is showing. This scenario describes the following multi-agent domain, $\mathcal{D}$, with the signature $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$, where:*

$$\mathcal{AG} = \{A, B, C\}$$

$$\mathcal{F} = \{locked, heads\}$$

$$\mathcal{A} = \{unlock(\alpha), peek(\alpha)\}$$

where α is variable over $\mathcal{AG}$. ◇

The Concealed Coin Domain describes both *ontic* and *epistemic* properties of the world. The ontic properties, such as whether or not the strongbox is locked, and which face of the coin is showing, are represented by the fluents *locked* and *heads* respectively. The various

[2]While unused in this chapter, the set of actions in a multi-agent domain plays an important role in subsequent chapters.

epistemic properties are described by *modal formulae*. The syntax of such formulae presented in this dissertation is taken from [19].

Definition 2.9 (Modal Formula)**.** Let $\mathcal{D}$ be a multi-agent domain with the signature $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$. We define a *modal formula* as follows:

- $f \in \mathcal{F}$ is a modal formula
- if φ is a modal formula, then $\neg\varphi$ is a modal formula
- if φ_1 and φ_2 are modal formulae, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, and $\varphi_1 \equiv \varphi_2$ are modal formulae
- if $\alpha \in \mathcal{AG}$ and φ is a modal formula, then $\Box_\alpha \varphi$ is a modal formula
- if $\gamma \subseteq \mathcal{AG}$ and φ is a modal formula, then $\mathcal{E}_\gamma \varphi$ and $C_\gamma \varphi$ are modal formulae
- nothing else is a modal formula

The set of all modal formulae that may be constructed given a particular signature, Σ, defines a language which we denote by $\mathcal{L}_\Sigma$. ◇

Depending on the *modality of interest*, modal formulae may be used to represent the individual (or collective) *knowledge* or *belief* in a given formula φ. If the modality of interest is that of *knowledge*, then formulae of the form $\Box_\alpha \varphi$, may be written as $\mathcal{K}_\alpha \varphi$, and are read as: "agent α knows φ to be true." Alternatively, if the modality is that of *belief*, then such formulae are written as $\mathcal{B}_\alpha \varphi$, and taken to mean: "agent α believes that φ is true."

Formulae of the form $\mathcal{E}_\gamma \varphi$ and $C_\gamma \varphi$ are used to represent information about the knowledge/beliefs of a group of agents, γ. Formulae of the form $\mathcal{E}_\gamma \varphi$ are read as: "Every agent in γ knows/believes φ to be true." The fact that some belief or knowledge is shared by every member of a group does not imply that every member of that group is aware of this. Group

knowledge that is both shared, and understood to be shared by the members of the group, is represented by formulae of the form $C_\gamma \varphi$, which are read as: "φ is a common knowledge (or a commonly held belief) by agents in γ." Unless otherwise noted, the modality of discourse assumed in this work is that of *belief*.

2.2.1.2 Semantics

With the syntax of modal logic in place, we now introduce the foundational notion of a *Kripke world* which is necessary for defining its semantics.

Definition 2.10 (Kripke World)**.** Let $\mathcal{D}$ be a multi-agent domain with signature, $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$, and let $\mathcal{AG} = \{\alpha_1, \ldots, \alpha_n\}$. A *Kripke model*, M, is a tuple of the form $(W, \pi, R_{\alpha_1}, \ldots, R_{\alpha_n})$ where:

- W is a nonempty set of *points*[3]
- π is an *interpretation function* which for each $\omega \in W$ gives an interpretation, $\pi(\omega) : \mathcal{F} \mapsto \{\top, \bot\}$
- each R_{α_i} is a binary relation on W called an *accessibility relation for agent* α_i

A *Kripke world* is a pair, (M, ω), where M is a Kripke model, and ω is a point in M designated the *reference point*. ◇

Points combined with their respective interpretation functions describe potential *physical configurations* of the domain, while the accessibility relations represent its various *epistemic properties*. Intuitively, the pair $(\omega_\sigma, \omega_\tau) \in R_{\alpha_i}$ represents the property that *from within point*

[3] In the literature these are often also referred to as *possible worlds*, or *states*. The term *points* is used to avoid overloading terminology.

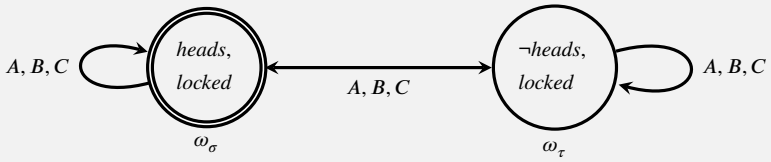

Figure 2.2: A *Kripke world* representing the initial state of The Concealed Coin Domain.

ω_σ, *agent* α_i ω_τ *to be possible*. In this work, we assume the existence of an impartial, external observer who knows which point in a Kripke structure corresponds to the *frame of reference* from which one can draw conclusions about the properties of the domain.

For convenience, we adopt a dot-notation for all record-like structures used in this work. For example, if M is a Kripke structure, then $M.W$ denotes the set of all points in M. Similarly, if $S = (M, \omega)$, is a Kripke world, then $S.M$ is the Kripke structure associated with S, and $S.W$ is the set of points in the Kripke structure associated with S.

Example 2.11 (Kripke World in the Concealed Coin Domain)**.** *Recall the Concealed Coin Domain from Example 2.8. A* Kripke world *describing this domain is given in Figure 2.2.*

◇

The Kripke world in Figure 2.2 has two points, ω_σ and ω_τ, whose associated interpretation functions model two distinct possible worlds: one in which the coin is facing heads, and in the other tails. The fact that the accessibility relations for the agents are symmetric, reflexive, transitive, and relate ω_σ to ω_τ and vice versa, reflects the fact that the agents have no means of distinguishing between the two. The double circle around ω_σ however marks it as the reference point.

Now that the notion of a Kripke world has been established, we can now present the entailment relation between Kripke worlds and modal formulae as defined in [19].

Definition 2.12 (Entailment Relation)**.** Let (M, ω_σ) be a Kripke world in a multi-agent domain, $\mathcal{D}$, with the signature $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$.

- $(M, \omega_\sigma) \models f$ where $f \in \mathcal{F}$ if and only if $M.\pi(\omega_\sigma)(f) = \top$
- $(M, \omega_\sigma) \models \neg\varphi$ if and only if $(M, \omega_\sigma) \not\models \varphi$
- $(M, \omega_\sigma) \models \varphi_1 \wedge \varphi_2$ if and only if $(M, \omega_\sigma) \models \varphi_1$ and $(M, \omega_\sigma) \models \varphi_2$
- $(M, \omega_\sigma) \models \varphi_1 \vee \varphi_2$ if and only if $(M, \omega_\sigma) \models \varphi_1$ or $(M, \omega_\sigma) \models \varphi_2$
- $(M, \omega_\sigma) \models \Box_\alpha\varphi$ if and only if $(M, \omega_\tau) \models \varphi$ for all ω_τ such that $(\omega_\sigma, \omega_\tau) \in M.R_\alpha$

Let $\mathcal{E}_\gamma^0\varphi$ be equivalent to φ, and let $\mathcal{E}_\gamma^{k+1}\varphi$ be equivalent to $\mathcal{E}_\gamma\mathcal{E}_\gamma^k\varphi$.

- $(M, \omega_\sigma) \models \mathcal{E}_\gamma\varphi$ if and only if $(M, \omega_\sigma) \models \Box_\alpha\varphi$ for each $\alpha \in \gamma$
- $(M, \omega_\sigma) \models \mathcal{C}_\gamma\varphi$ if and only if $(M, \omega_\sigma) \models \mathcal{E}_\gamma^k\varphi$ for $k = 1, 2, 3, \ldots$

◇

Example 2.13 (Entailment in the Concealed Coin Domain)**.** *Consider the Kripke world* (M, ω_σ) *shown in Figure 2.2. Among the formulae entailed by* (M, ω_σ) *are:*

$$\neg\Box_A h \wedge \neg\Box_A \neg h$$

$$\mathcal{C}_{\{A,B,C\}} l$$

$$\mathcal{C}_{\{A,B,C\}}(\neg\Box_A h \wedge \neg\Box_A \neg h)$$

◇

Having described the language used to represent the various properties of a multi-agent domain, we now turn our attention towards an approach taken by the dynamic epistemic logic community for describing actions and change within a dynamic multi-agent domain.

2.2.2 Update Models

An agent's actions may alter both the ontic and epistemic properties of a domain. Consider the act of unlocking the strongbox in the context of Example 2.8. In addition to the physical effect of changing the value of the fluent *locked*, such an action may change the beliefs of the other agents in the domain depending on what they know about the occurrence itself. For example, if agent B *observes* A unlocking the strongbox, we should conclude that B knows the strongbox is unlocked. Similarly, if C is *oblivious* of the action occurrence, he should continue to believe that the strongbox is locked.

Briefly stated, the approach presented in [4, 43] describes a transition system whose nodes represent *possible physical and epistemic configurations of the world*, and whose edges are labeled by *action occurrences* together with the *levels of awareness* that the agents of the domain have with respect to them. This model led to the development of the notion of an *event/update model and instances*, which may be thought of as akin to a Kripke structures/worlds describing action occurrences. What follows is a brief overview of the language of update models presented in [43].

We begin with the notion of a $\mathcal{L}$-substitution, which may be viewed as a particular class of function which models the *direct effects* of an action occurrence.

Definition 2.14 ($\mathcal{L}$-substitution)**.** *$\mathcal{L}$-substitutions* are functions of type $\mathcal{L} \mapsto \mathcal{L}$ that distributes over all language constructs, and that map all but a finite number of basic propositions to themselves. $\mathcal{L}$-substitutions can be represented as sets of bindings:

$$\{p_1 \mapsto \varphi_1, \ldots, p_n \mapsto \varphi_n\}$$

where all the p_i are different fluents. ϵ denotes the identity substitution, and $SUB_{\mathcal{L}}$ denotes the set of all $\mathcal{L}$-substitutions. ⋄

As was alluded to previously, an *update instance*, may be viewed as a Kripke world representing a particular action occurrence.

Definition 2.15 (Update Instance)**.** Let $\mathcal{D}$ be a multi-agent domain with the signature $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$, where $\mathcal{AG} = \{\alpha_1, \ldots, \alpha_n\}$. An *update model* over the language, $\mathcal{L}_\Sigma$, is a tuple of the form $(E, R_{\alpha_1}, \ldots, R_{\alpha_n}, pre, sub)$ where:

- E is a finite, non-empty set of *events*
- each R_{α_i} is a binary relation on E called an *accessibility relation* for agent α_i
- $pre : E \mapsto \mathcal{L}_\Sigma$ assigns a *precondition* to each event
- $sub : E \mapsto SUB_{\mathcal{L}_\Sigma}$ assigns a $\mathcal{L}$*-substitution* to each event representing its direct effects

An *update instance* is a pair, (U, ε), where U is an update model, and ε is an event in U. designated as the *reference event*. ◇

As with Kripke worlds, we assume the existence of an impartial, external observer who knows which event in an update model corresponds to the *frame of reference* from which one can draw conclusions about what actually transpired (as opposed to what the various agents understand to have transpired).

Example 2.16 (Update Instance in the Concealed Coin Domain)**.** *Recall the Concealed Coin Domain from Example 2.8, and consider an occurrence of the action unlock, where* A *is the actor,* B *is an observer, and* C *is oblivious of what has transpired. In other words, "agent* A *unlocks the box with agent* B *watching him and without the knowledge of agent* C*." The corresponding* update instance *is show in Figure 2.3.* ◇

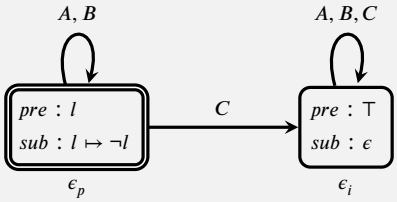

Figure 2.3: Update instance for an occurrence of the action *unlock* where A is the actor, B is an observer, and C is oblivious of what has transpired.

The update instance presented in Example 2.16 consist of two events: the *positive event*, ε_p, and the *inertial event*, ε_i. The positive event describes the actual event that transpired as part of the action occurrence, namely the strongbox being unlocked. The event has a precondition, $pre(\varepsilon_p) = locked$, stating that this is only possible if the strongbox is locked. The physical effect is represented by $sub(\varepsilon_p) = \{locked \mapsto \neg locked\}$, which has the informal reading of "unlocking the strongbox causes it to be unlocked". The inertial event is used to model the belief of oblivious agents that nothing has transpired. Consequently it has no preconditions and is associated with the identity substitution. What the agents believe about the action occurrence is represented by their accessibility relations. Note for example that ε_p is not accessible from the perspective of agent C. This reflects the fact that agent C is unaware that this event may have transpired. By that same token, ε_p is accessible from the perspectives of agents A and B, as they are both fully aware of what has happened.

As was mentioned previously, the transition system envisioned in [4, 43] is one whose nodes correspond to Kripke worlds and whose arcs are labeled by update instances representing individual action occurrences. The transitions themselves are defined through an operator known as the *update execution* given in Definition 2.17.

Definition 2.17 (Update Execution)**.** Given a Kripke world, (M, ω), and an update instance (U, ε), such that $(M, \omega) \models U.pre(\varepsilon)$, the successor world obtained by performing the ac-

tion occurrence represented by (U, ε), in (M, ω), is the Kripke world $(M, \omega) \otimes (U, \varepsilon) = (M', (\omega, \varepsilon))$ where:

- $M'.W = \{(\omega_\sigma, \varepsilon_\mu) \mid \omega_\sigma \in M.W, \varepsilon_\mu \in U.E, (M, \omega_\sigma) \models U.pre(\varepsilon_\mu)\}$
- $M'.R_i = \{((\omega_\sigma, \varepsilon_\mu), (\omega_\tau, \varepsilon_\nu)) \mid (\omega_\sigma, \omega_\tau) \in M.R_i \text{ and } (\varepsilon_\mu, \varepsilon_\nu) \in U.R_i\}$
- $M'.\pi((\omega, \varepsilon))(f) = U.sub(\varepsilon)(f)$

◇

Example 2.18 (Applying the Update Execution in the Concealed Coin Domain)**.** *Suppose that we are given the initial state of the Concealed Domain as in Example 2.11 defined by the Kripke world in Figure 2.2, and that unbeknownst to* **C***, agents* **A** *and* **B** *unlock the box together. This action occurrence is described by the update instance given by Figure 2.3. Application of the update execution yields the successor Kripke world shown in Figure 2.4.*

◇

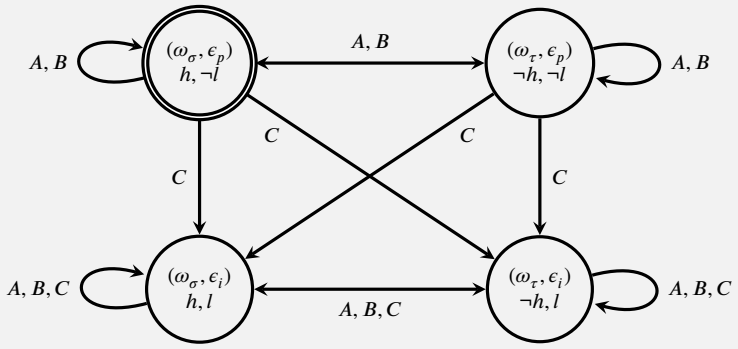

Figure 2.4: The *successor Kripke world* after an occurrence of the action *unlock* with *A* performing the action, *B* acting as an observer, and *C* oblivious of what has transpired.

2.3 Answer Set Programming

Answer set programming is a relatively new programming paradigm based on logic programming under the answer set semantics [6, 21, 24, 25, 31]. The paradigm has been successfully used in a broad range of applications ranging from modeling the reaction control system of the space shuttle [3, 42], to textual query answering [8], and high level planning and robotics [45]. What follows is a brief overview of the syntax and semantics of A-Prolog taken in part from [24] (with the authors' permission).

2.3.1 Syntax

Our discussion of the syntax and semantics of A-Prolog is done in the context of a given signature[4] $\Sigma = \langle \mathcal{O}, \mathcal{F}, \mathcal{P}, \mathcal{V} \rangle$, where the elements of $\mathcal{O}$, $\mathcal{F}$, and $\mathcal{P}$, are respectively referred to as *object*, *function* and *predicate constants*, and the elements of $\mathcal{V}$ are referred to as *variables*. For simplicity we assume that unless otherwise stated, the signatures of programs consist only of those symbols used in their rules. With the notion of a signature in place, we may introduce the notion of a *term*.

Definition 2.19 (A-Prolog Term)**.** *Terms* over a signature, Σ, are defined as follows:

- Variables and object constants are terms.
- If $t_1, \ldots, t_n$ are terms, and f is a function symbol of arity n, then $f(t_1, \ldots, t_n)$ is a term.

Terms which contain no symbols for arithmetic expressions and no variables are known as *ground terms*. ◇

[4]The reader should make note of the fact that the signature of an A-Prolog program is different in kind from that of a multi-agent domain.

Statements are built up from terms. An *atomic statement*, or simply an *atom*, is an expression of the form $p(t_1, \ldots, t_n)$, where p is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms. A *literal* is an either an atom, $p(t_1, \ldots, t_n)$ or its negation, $\neg p(t_1, \ldots, t_n)$. Ground atoms and their negations are referred to as *ground literals*. With these elementary constructs in place, we can proceed to define rules, and from there an A-Prolog program.

Definition 2.20 (A-Prolog Program)**.** An *A-Prolog program*, Π, consists of a signature, Σ, and a collection of *rules* of the form:

$$\lambda_0 \; or \; \ldots \; or \; \lambda_i \; \leftarrow \; \lambda_{i+1}, \ldots, \lambda_m, not \; \lambda_{m+1}, \ldots, not \; \lambda_n.$$

where each λ_i is a literal. ◇

The left-hand side of a rule is called its *head* and the right-hand side is called its *body*. Literals, possibly preceded by default negation, *not*, are often called *extended literals*. The body of the rule can be viewed as a set of extended literals (sometimes referred to as the *premises* of the rule).

The head or the body of a rule may be empty. A rule with an empty head is often referred to as a *constraint* and is written as:

$$\leftarrow \; \lambda_{i+1}, \ldots, \lambda_m, not \; \lambda_{m+1}, \ldots, not \; \lambda_n.$$

A rule with an empty body is known as a *fact* and is written as:

$$\lambda_0 \; or \; \ldots \; or \; \lambda_i.$$

The symbol *not* is a new logical connective called *default negation*, (or *negation as failure*). The extended literal *not* λ is often read as *"it is not believed that λ is true."* Note that this does not imply that λ is believed to be false. It is conceivable, in fact quite normal, for a rational reasoner to believe neither the statement p nor its negation $\neg p$. Clearly default

negation, *not* is different from the classical ¬. Whereas ¬*p* asserts that *p* is false, *not p* is a statement about *belief*.

Likewise, the disjunction *or* is also a new connective, sometimes referred to as *epistemic disjunction*. The statement λ_1 *or* λ_2 is read as *"*λ_1 *is believed to be true or* λ_2 *is believed to be true."* It is also different from its classical counterpart ∨. The statement $p \vee \neg p$ of propositional logic is a tautology; however the statement *p or not p* is not. The former states that the proposition *p* is either true or false, whereas the latter states that *p* is believed to be true or believed to be false. Once again, it is quite possible for a rational reasoner to have no beliefs regarding the truth or falsity of propositions.

Following the Prolog convention, non-numeric object, function, and predicate constants of Σ are denoted by identifiers beginning with lowercase letters; variables are identifiers beginning with capital letters. The variables of a program Π range over ground terms of Σ. A rule *r* with variables is viewed as a shorthand for the set of its *ground instances* – rules obtained from *r* by replacing its variables by ground terms of Σ and by evaluating arithmetic terms. The set of ground instances of the rules of a program Π is a called the *grounding* of Π; a program Π with variables can be viewed simply as a shorthand for its grounding. Lastly, a program Π which contains no default negations is known as a *basic program*.

With the syntax thus described, we now proceed with defining the semantics of an A-Prolog program.

2.3.2 Semantics

Before we proceed with defining the semantics of an A-Prolog program, we need to define what it means for a set of literals to satisfy a rule. We first define the notion for the parts that make up the rule, and then show the parts combine to define the satisfiability of

the rule.

Definition 2.21 (Satisfiability)**.** A set S of ground literals *satisfies*:

- a literal, λ, *if* $\lambda \in S$;
- an extended literal *not* λ, *if* $\lambda \notin S$;
- λ_0 *or* … *or* λ_n, *if* for some $1 \leq i \leq n$, $\lambda_i \in S$;
- a set X of ground extended literals, *if* S satisfies every element X;
- a rule r, *if* whenever S satisfies r's body, it satisfies r's head.

◇

Informally, a program Π can be viewed as the specification for *answer sets* – sets of beliefs that could be held by a rational reasoner associated with Π. Answer sets are represented by collections of ground literals. In forming such sets, the reasoner must be guided by the following informal principles:

1. Satisfy the rules of Π. In other words, believe in the head of a rule if you believe in its body.
2. Do not believe in contradictions.
3. Adhere to the *"Rationality Principle"* that says, "Believe nothing which you are not forced to believe."

These informal principles lead us to the following definition of an answer set, which is split into two parts: the first part defining the answer sets of a basic program; and the second defining the answer sets of an arbitrary logic program.

Definition 2.22 (Answer Set of a Basic Program)**.** Let Π be a basic program. An *answer set* of Π is a consistent set S of ground literals such that:

- S satisfies the rules of Π, and
- S is minimal (i.e., there is no proper subset of S that satisfies the rules of Π).

$\diamond$

Definition 2.23 (Answer Set of a Logic Program)**.** Let Π be an arbitrary logic program and S be a set of ground literals. By Π^S (known as the reduct of Π with respect to S) we denote the program obtained from Π by:

1. removing from Π all rules containing *not* λ such that $\lambda \in S$;
2. removing all other premises containing *not*

S is an answer set of Π if S is an answer set of Π^S. $\diamond$

The reader will note that at this point the syntax and semantics of A-Prolog have been presented on a mathematical level. From a programmatic perspective, both of these are realized and extended from their basic form by various *answer-set solvers*, and subsequent examples are presented in the dialect of the answer-set solver `clingo` [21].

Chapter 3

THE ACTION LANGUAGE $m\mathcal{A}+$

The preliminary ideas that were first presented in [7] were fleshed out and formalized in the action language $m\mathcal{A}+$ in [11], a multi-agent variant of the action language $\mathcal{A}$ [27]. The motivating goals behind the development of $m\mathcal{A}+$ were to formalize a subset of the English language to allow for the intuitive and elaboration tolerant [38] representation of multi-agent domains such as the Classical Muddy Children Problem [19] and the Concealed Coin Domains of [4]. A characteristic feature of such domains is there are no dependencies between fluents. As such, actions do not have indirect effects which may alter the *ontic properties* of the domain — instead, any indirect effects solely involve the domain's *epistemic properties*.

As a motivating example we consider the variant of the Concealed Coin Domain of [4] described in Example 3.1.

Example 3.1 (The Concealed Coin Domain)**.** *Three agents, **A**, **B**, and **C**, are together in a room with a box which contains a coin. Suppose that this fact, together with the fact that none of the agents knows which face of the coin is showing is a commonly held belief among them. Furthermore let us suppose that all of the agents are attentive and that this too is a commonly held belief. Lastly, let us assume that the coin is facing heads up and that all of the beliefs of the agents are true.*

An agent may peek inside to determine which face of the coin is showing. In addition, agent may signal or distract one of his fellows, thereby causing him to be respectively attentive or inattentive. Attentive agents are fully aware of what transpires around them. ⋄

3.1 Syntax

Theories of $m\mathcal{A}+$ are defined over a multi-agent domain $\mathcal{D}$ with a signature Σ. $m\mathcal{A}+$ supports two broad classes of actions: *ontic* (or *world-altering*) and *epistemic* actions. The former describe actions which affect the properties of the domain represented by fluents, while the latter describe actions which primarily affect the agents' beliefs. Epistemic actions are further broken into two categories: *sensing* and *communication*. Sensing actions represent actions which an agent may perform in order to learn the value of a fluent, while communication actions are used to represent actions which communicate information between agents.

Before we introduce the syntax of the language itself, it is useful to introduce two special classes of modal formulae: *fluent formulae* and *belief formulae*. The former are used to described the physical properties of a domain, while the latter are used to describe the beliefs of the agents regarding those properties and the beliefs of their fellow agents regarding them.

Definition 3.2 (Fluent Formula)**.** Let $\mathcal{D}$ be a multi-agent domain with the signature, $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$. A *fluent formula* is a formula built from the propositional variables in $\mathcal{F}$ and the traditional propositional operators $\wedge$, $\vee$, $\rightarrow$, $\neg$, etc. $\diamond$

Throughout this work we use the term, *fluent atom* to mean a fluent formula containing a single element $f \in \mathcal{F}$, and the term *fluent literal* to mean either a fluent atom, f, or its negation $\neg f$. As usual, $\top$ and $\bot$ denote *true* and *false*, respectively.

Definition 3.3 (Belief Formula)**.** A *belief formula* is defined as follows:

- if φ is a fluent formula, then φ is a belief formula
- if $\alpha \in \mathcal{AG}$ and φ is a belief formula, then $\mathcal{B}_\alpha \varphi$ is a belief formula

- if φ_1 and φ_2 are belief formulae, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, and $\varphi_1 \rightarrow \varphi_2$ are belief formula
- if $\gamma \subseteq \mathcal{AG}$ and φ is a belief formula, then $\mathcal{E}_\gamma \varphi$ and $\mathcal{C}_\gamma \varphi$ are belief formulae
- nothing else is a belief formula

$\diamond$

Domains such as the one presented in Example 3.1 describes several distinct *kinds* of information: ontic and epistemic properties which are *initially true*, and *causal relationships* between actions and their effects. Each of these distinct kinds is represented by a corresponding syntactic construct: *initial state axioms*, *dynamic causal laws*, as well *sensing* and *announcement axioms*. Initial state axioms are statements of the form:

$$\textbf{\textit{initially}}\ \varphi \tag{3.1}$$

where φ is a belief formula, and have the informal reading of: "φ is initially true."

The causal relationship between an *ontic action* and its direct effects are represented by *dynamic causal laws* which are statements of the form:

$$a\ \textbf{\textit{causes}}\ \lambda\ \textbf{\textit{if}}\ \phi \tag{3.2}$$

where a is an action, λ is a fluent literal, and ϕ is a belief formula. Laws of this form are read as: "performing the action a in a state which satisfies ϕ causes λ to be true." If ϕ is a tautology ($\phi = \top$), then we simply write the following:

$$a\ \textbf{\textit{causes}}\ \lambda \tag{3.3}$$

The direct effects of *epistemic actions* are described by *sensing* and *announcement/communication axioms*. Sensing axioms are statements of the form:

$$a\ \textbf{\textit{determines}}\ f \tag{3.4}$$

where a is the name of an action, and f is a fluent. Statements of this form are understood to mean: "if an agent performs the action a, he will learn the value of the fluent f." Communication axioms have a similar syntax:

$$a \textbf{ announces } \varphi \tag{3.5}$$

where a is the name of an action, and φ is a modal formula. In $m\mathcal{A}+$ only truthful announcements are allowed.

The actions that an agent may perform in any given situation are often limited by various properties of the domain. To represent this kind of information $m\mathcal{A}+$ includes a construct known as an *executability condition* which is a statement of the form:

$$\textbf{executable } a \textbf{ if } \varphi \tag{3.6}$$

where a is the name of an action, and φ is a belief formula. Statements of this form are read as follows: "action a is executable in a state which satisfies φ." If $\varphi = \top$, the statement is omitted.

Thus far, statements (3.2) – (3.5) describe the causal relationships between actions and their *direct effects*. Within a multi-agent context however, an action occurrence may *indirectly effect* the beliefs of the agents in the domain depending on the kind of knowledge the agents have regarding it. In general, for any given action occurrence we divide the agents of the domain into three groups known as *frames of reference* (or *levels of awareness*):

- those which have first-hand knowledge of the action occurrence
- those which have second-hand knowledge of the action occurrence
- those which have no knowledge of the action occurrence

Agents which have first-hand knowledge of an action occurrence are known as *full observers* (or *fully aware agents*). Such agents have full knowledge of both the action occurrence

and its effects. Agents which have second-hand knowledge are known as *partial observers* (or *partially aware agents*). Agents are said to have second-hand knowledge of an action occurrence if they observe an action occurrence *from a distance*. Such agents know that the action took place and who the participants were, but do not know its full consequences. Lastly, agents which have no knowledge of an action occurrence are called *oblivious agents*.

The frames of reference of the agents are dynamic in nature, and depend on the properties of the state in which the action occurs. In this work, we consider the possible frames of reference of the agents for different kinds of actions to be as shown in Table 3.1.

action type	**full observers**	**partial observers**	**oblivious**
ontic	✓		✓
sensing	✓	✓	✓
announcements	✓	✓	✓

Table 3.1: Action classes and frames of reference in $m\mathcal{A}+$.

With respect to ontic actions, the agents may either have first-hand knowledge of an action occurrence or be oblivious. For sensing and announcement actions however, an agent may have any of the allowable frames of reference. For example, consider the act of peeking into the box as described in the Concealed Coin Domain of [4]. The agents who peek into the box are said to have first-hand knowledge of that specific action occurrence. Consequently, they will all learn the contents of the box and this fact will be a common belief among them. Agents who are not peeking into the box but merely observe their fellows perform such an action are said to have second-hand knowledge of the action occurrence. These agents do not learn the contents of the box. However, it does become a common belief among them that those agents who peeked know the box's contents. Lastly, all other agents in the domain would be considered to be oblivious of that action occurrence.

Frames of reference are described by *perspective axioms* (or *observability axioms*) which

are statements of the form:

$$X \textbf{ observes } a \textbf{ if } \phi \tag{3.7}$$

$$X \textbf{ aware of } a \textbf{ if } \phi \tag{3.8}$$

where X is a set of agent names, a is an action, and ϕ is a fluent formula. Perspective axioms of the first form (called *observation axioms*) define the set of agents who are fully aware of both the action occurrence and its effects. Those of the second form (called *awareness axioms*) define the set of agents who are aware of the occurrence, but only partially of its effects. By default, we assume that all other agents within the domain are oblivious. As with dynamic causal laws, if ϕ is a tautology, we adopt the following shorthand:

$$X \textbf{ observes } a \tag{3.9}$$

$$X \textbf{ aware of } a \tag{3.10}$$

Definition 3.4 (Initial State Description)**.** An *initial state description* is a set of statements of the form (3.1). ◇

Definition 3.5 (Action Description of $m\mathcal{A}+$)**.** An *action description*, Δ, in $m\mathcal{A}+$ is a collection of statements of the form (3.2) – (3.10). Δ is *consistent* if for every pair of dynamic causal laws [a **causes** f **if** ϕ_1] and [a **causes** $\neg f$ **if** ϕ_2] in Δ, $\phi_1 \wedge \phi_2$ is inconsistent. ◇

Example 3.6 (Representing the Concealed Coin Domain)**.** *We begin our axiomatization by specifying the domain signature. A close reading of Example 3.1 suggests the following:*

$$\mathcal{AG} = \{A, B, C\}$$

$$\mathcal{F} = \{heads, attentive(\alpha)\}$$

$$\mathcal{A} = \{peek(\alpha), signal(\alpha_1, \alpha_2), distract(\alpha_1, \alpha_2)\}$$

where α, α_1, and α_2, are variables over $\mathcal{AG}$.

With the signature firmly in place, we can now give the initial state description, and the causal relationships between the actions and their effects. Example 3.1 states that initially:

- *it is a common belief amongst the agents that the box contains a coin*
- *it is a common belief amongst the agents that none of them know which face of the coin is showing*
- *it is a common belief amongst them that all of the agents are attentive*
- *all of the agents are in fact attentive*
- *the coin is actually facing heads up*
- *all of the beliefs of the agents are in fact true*

This information is captured by the following initial state description

$$\textbf{\textit{initially}}\ \mathit{heads} \tag{3.11}$$

$$\textbf{\textit{initially}}\ C(\neg \mathcal{B}_A \mathit{heads} \wedge \neg \mathcal{B}_A \neg \mathit{heads}) \tag{3.12}$$

$$\textbf{\textit{initially}}\ C(\neg \mathcal{B}_B \mathit{heads} \wedge \neg \mathcal{B}_B \neg \mathit{heads}) \tag{3.13}$$

$$\textbf{\textit{initially}}\ C(\neg \mathcal{B}_C \mathit{heads} \wedge \neg \mathcal{B}_C \neg \mathit{heads}) \tag{3.14}$$

$$\textbf{\textit{initially}}\ \mathit{attentive}(A) \wedge \mathit{attentive}(B) \wedge \mathit{attentive}(C) \tag{3.15}$$

$$\textbf{\textit{initially}}\ C(\mathit{attentive}(A) \wedge \mathit{attentive}(B) \wedge \mathit{attentive}(C)) \tag{3.16}$$

Having described the initial state, we move on to the actions and their effects. The action, $\mathit{peek}(\alpha)$*, is an* epistemic action *— in particular, it is a sensing action. Consequently its direct effects are represented by the following sensing axiom:*

$$\mathit{peek}(\alpha)\ \textbf{\textit{determines}}\ \mathit{heads} \tag{3.17}$$

Only the agent who is peeking is fully aware of the occurrence and its full effects. Attentive agents are only partially aware of the action's effects. This is represented by the following pair of perspective axioms:

$$\{\alpha\}\ \textbf{\textit{observes}}\ peek(\alpha) \tag{3.18}$$

$$\{\alpha_2\}\ \textbf{\textit{aware of}}\ peek(\alpha_1)\ \textbf{\textit{if}}\ attentive(\alpha_2) \tag{3.19}$$

The direct effects of the ontic actions $signal(\alpha_1, \alpha_2)$ *and* $distract(\alpha_1, \alpha_2)$ *are represented by the following dynamic causal laws:*

$$signal(\alpha_1, \alpha_2)\ \textbf{\textit{causes}}\ attentive(\alpha_2) \tag{3.20}$$

$$distract(\alpha_1, \alpha_2)\ \textbf{\textit{causes}}\ \neg attentive(\alpha_2) \tag{3.21}$$

As they are ontic actions, the agents who are directly involved in an occurrence of such an action, as well as any attentive agents are fully aware of the occurrence and its effects, with all other agents being considered to be oblivious. This is represented by the following perspective axioms:

$$\{\alpha_1, \alpha_2\}\ \textbf{\textit{observes}}\ signal(\alpha_1, \alpha_2) \tag{3.22}$$

$$\{\alpha\}\ \textbf{\textit{observes}}\ signal(\alpha_1, \alpha_2)\ \textbf{\textit{if}}\ attentive(\alpha) \tag{3.23}$$

$$\{\alpha_1, \alpha_2\}\ \textbf{\textit{observes}}\ distract(\alpha_1, \alpha_2) \tag{3.24}$$

$$\{\alpha\}\ \textbf{\textit{observes}}\ distract(\alpha_1, \alpha_2)\ \textbf{\textit{if}}\ attentive(\alpha) \tag{3.25}$$

◇

3.2 Semantics

As is the case with other action languages, an action description of $m\mathcal{A}+$ defines a transition diagram whose nodes correspond to *states of the domain* — which we model as

Kripke worlds — and whose arcs are labeled by *actions*. We begin our discussion of the semantics by taking a look at the meaning of initial state descriptions.

3.2.1 Deriving the Initial State

As a consequence of the Kripke semantics of modal formulae, there are potentially infinitely many Kripke worlds which may be models of the modal theory described by an initial state description. This presents a problem if we wish to be able to automate reasoning about the initial state of the domain, and also increases the complexity that a system designer must grapple with. In order to enable the use of modern computational technologies (e.g., answer set solvers, and certain kinds of planning systems), we introduce the notions of *restricted formulae* and *definite initial state descriptions*.

Definition 3.7 (Restricted Formula)**.** Let $\mathcal{D}$ be a multi-agent domain with signature, $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$. A *restricted formula* is defined as follows:

1. if φ is a fluent formula, then φ is a restricted formula
2. if $\alpha \in \mathcal{AG}$ and φ is a fluent formula, then $\mathcal{B}_\alpha\varphi$, $\mathcal{B}_\alpha\varphi \vee \mathcal{B}_\alpha\neg\varphi$, and $\neg\mathcal{B}_\alpha\varphi \wedge \neg\mathcal{B}_\alpha\neg\varphi$ is a restricted formula
3. if $\alpha \in \mathcal{AG}$ and φ is a restricted formula of the form (1) or (2), then $\mathcal{C}\varphi$ is a restricted formula
4. no other formula is a restricted formula

$\diamond$

Definition 3.8 (Definite Initial State Description)**.** Let $\mathcal{I}$ be an initial state description, and let:

$$\mathcal{T}_{\mathcal{I}} = \{\varphi \mid [\textbf{\textit{initially}}\ C\varphi] \in \mathcal{I}\} \cup$$
$$\{\varphi \mid [\textbf{\textit{initially}}\ \varphi] \in \mathcal{I} \wedge \varphi \text{ is a fluent formula}\}$$

$\mathcal{I}$ is said to be *definite* if:

- $\mathcal{T}_{\mathcal{I}}$ is a set of restricted formulae
- $\mathcal{T}_{\mathcal{I}} \nvDash \mathcal{B}_{\alpha}\varphi_1$ holds for every fluent formula φ_1 such that there exists no φ_2 satisfying $C(\mathcal{B}_{\alpha}\varphi_2) \in \mathcal{T}_{\mathcal{I}}$ and $\top \vDash \varphi_2 \rightarrow \varphi_1$
- $\mathcal{T}_{\mathcal{I}} \nvDash C(\mathcal{B}_{\alpha}\varphi_1 \vee \mathcal{B}_{\alpha}\neg\varphi_1)$ holds for every fluent formula φ_1 such that there exists no φ_2 satisfying $C(\mathcal{B}_{\alpha}\varphi_2 \vee \mathcal{B}_{\alpha}\neg\varphi_2) \in \mathcal{T}_{\mathcal{I}}$ and $\top \vDash \varphi_2 \rightarrow \varphi_1$

In addition, $\mathcal{I}$ is said to be *complete* if for each fluent $f \in \mathcal{F}$ either $[\textbf{\textit{initially}}\ f] \in \mathcal{I}$ or $[\textbf{\textit{initially}}\ \neg f] \in \mathcal{I}$. Lastly, $\mathcal{I}$ is said to be *consistent* if $\mathcal{T}_{\mathcal{I}}$ is consistent. ◇

By restricting ourselves to definite initial state descriptions, we can limit the kinds of Kripke worlds that are models of the corresponding modal theories to those which satisfy the **S5** axioms. In fact, it can be shown that all initial states of a complete and consistent definite initial state description which satisfy the **S5** axioms are equivalent to each other [13], allowing us to make use of the notion of a canonical initial state.

Theorem 3.1. *Let $\mathcal{D}$ be a multi-agent domain with the signature, $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$, $\mathcal{I}$ be a consistent and definite initial state description, and Δ be an action description of $m\mathcal{A}+$. There exists a unique Kripke world, (M, σ), where $|M.W| \leq 2^{|\mathcal{F}|}$, which satisfies the* **S5** *axioms, such that every Kripke model of $(\mathcal{I}, \Delta)$ is equivalent to (M, σ).*

Example 3.9 (The Concealed Coin Domain — Initial State)**.** *The initial state description presented in Example 3.6 is a complete and consistent definite initial state description and corresponds to the canonical initial state,* $\sigma_0 = (M_0, \omega_1)$ *shown in Figure 3.1.* σ_0 *consists of two points,* ω_1 *and* ω_2*, where:*

- $M_0.\pi(\omega_1) = \{heads, attentive(A), attentive(B), attentive(C)\}$
- $M_0.\pi(\omega_2) = \{\neg heads, attentive(A), attentive(B), attentive(C)\}$

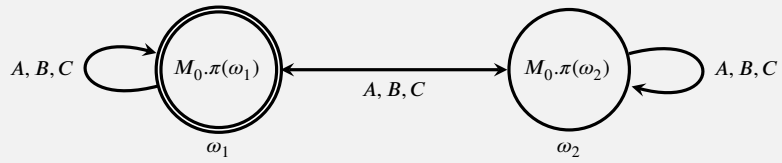

Figure 3.1: The initial state, σ_0, of the Concealed Coin Domain.

◇

3.2.2 The Transition Function

Before we define the transition function of $m\mathcal{A}+$, we must introduce some useful notation/terminology, namely, what it means for an action to be *executable* in a given state, and the frames of reference the agents have with respect to an action occurrence in a state.

Definition 3.10 (Executable Action)**.** Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an action for which there exists an executability condition [***executable*** a ***if*** ϕ] $\in \Delta$. The action a is *executable in* (M, ω) if $(M, \omega) \models \phi$. ◇

Now that we have defined the notion of executability, we may proceed with defining the frames of reference of the agents.

Definition 3.11 (Frames of Reference)**.** Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an action which occurs in σ. The various *frames of reference* of the agents are defined as follows:

- the set of agents who have *first hand knowledge* (or are *fully aware*) of a, denoted by $f(a, \sigma)$ is $\{\alpha \in \mathcal{AG} \mid [\alpha \textbf{\textit{ observes }} a \textbf{\textit{ if }} \phi] \in \Delta \wedge (M, \omega) \models \phi\}$
- the set of agents who have *second hand knowledge* (or are *partially aware*) of a, denoted by $p(a, \sigma)$ is $\{\alpha \in \mathcal{AG} \mid [\alpha \textbf{\textit{ aware of }} a \textbf{\textit{ if }} \phi] \in \Delta \wedge (M, \omega) \models \phi\}$
- the set of agents who have *no knowledge* (or are *oblivious*) of a, denoted by $o(a, \sigma)$ is $\mathcal{AG} \setminus (f(a, \sigma) \cup p(a, \sigma))$

$\diamond$

Note, that in $m\mathcal{A}+$, we make the following assumptions about the agents' frames of reference: members of $f(a, \sigma)$ know who their fellows are and the members of both $p(a, \sigma)$ and $o(a, \sigma)$; members of $p(a, \sigma)$ know who their fellows are as well as the members of $o(a, \sigma)$; and lastly, members of $o(a, \sigma)$ know who their fellow agents are.

The semantics of $m\mathcal{A}+$ is defined by a transition function which is based on the following intuition: for each action a which is executable in a state $\sigma = (M, \omega)$, we construct the corresponding update instance $U(\sigma, a)$, and then apply the update execution operator from [43] to obtain the successor state. Before we define the transition function itself, we first define how we construct the appropriate update instances. In the following definitions, we assume that: Δ is a consistent action description of $m\mathcal{A}+$; $\sigma = (M, \omega)$ is a state of the transition diagram defined by Δ; a is an action governed by the executability condition

[***executable*** *a* ***if*** ϕ] $\in \Delta$, and a is executable in σ. Graphical representations of the update instances are presented in Figures 3.2 – 3.4.

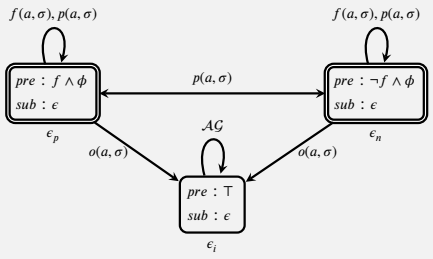

Figure 3.2: General update instance for sensing actions in $m\mathcal{A}+$.

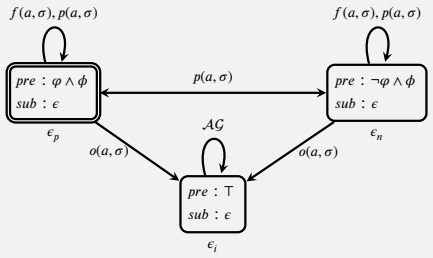

Figure 3.3: General update instance for announcement actions in $m\mathcal{A}+$.

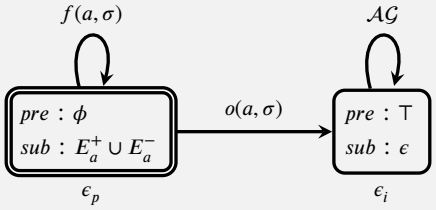

Figure 3.4: General update instance for ontic actions in $m\mathcal{A}+$.

Definition 3.12 (The Sensing Model $U_s(\sigma, a)$)**.** Let a be a sensing action governed by the sensing axiom [a ***determines*** f] which is executable in σ. *The update instance corresponding to an occurrence of a in* σ, $U_s(\sigma, a) = (U, \Gamma)$, where:

- $U.E = \{\varepsilon_p, \varepsilon_n, \varepsilon_i\}$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i)\}$ for each $\alpha \in f(a, \sigma)$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i), (\varepsilon_p, \varepsilon_n), (\varepsilon_n, \varepsilon_p)\}$ for each $\alpha \in p(a, \sigma)$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_i), (\varepsilon_n, \varepsilon_i), (\varepsilon_i, \varepsilon_i)\}$ for each $\alpha \in o(a, \sigma)$

- $U.pre(\varepsilon_p) = f \wedge \phi$
- $U.pre(\varepsilon_n) = \neg f \wedge \phi$
- $U.pre(\varepsilon_i) = \top$
- $U.sub(\varepsilon_p) = U.sub(\varepsilon_n) = U.sub(\varepsilon_i) = \epsilon$

and $\Gamma = \{\varepsilon_p, \varepsilon_n\}$. ◇

Definition 3.13 (The Announcement Model $U_a(\sigma, a)$)**.** Let a be an announcement described by the axiom [a ***announces*** φ] which is executable in σ. *The update instance corresponding to an occurrence of a in* σ, $U_a(\sigma, a) = (U, \Gamma)$, where:

- $U.E = \{\varepsilon_p, \varepsilon_n, \varepsilon_i\}$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i)\}$ for each $\alpha \in f(a, \sigma)$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i), (\varepsilon_p, \varepsilon_n), (\varepsilon_n, \varepsilon_p)\}$ for each $\alpha \in p(a, \sigma)$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_i), (\varepsilon_n, \varepsilon_i), (\varepsilon_i, \varepsilon_i)\}$ for each $\alpha \in o(a, \sigma)$
- $U.pre(\varepsilon_p) = \varphi \wedge \phi$
- $U.pre(\varepsilon_n) = \neg\varphi \wedge \phi$
- $U.pre(\varepsilon_i) = \top$
- $U.sub(\varepsilon_p) = U.sub(\varepsilon_n) = U.sub(\varepsilon_i) = \epsilon$

and $\Gamma = \{\varepsilon_p\}$. ◇

Definition 3.14 (The Ontic Model $U_o(\sigma, a)$)**.** Let a be an ontic action described by the dynamic causal law [a ***causes*** f ***if*** ϕ] which is executable in σ. *The update instance corresponding to an occurrence of a in* σ, $U_o(\sigma, a) = (U, \Gamma)$, where:

- $U.E = \{\varepsilon_p, \varepsilon_i\}$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_i, \varepsilon_i)\}$ for each $\alpha \in f(a, \sigma)$
- $U.R_\alpha = \{(\varepsilon_p, \varepsilon_i), (\varepsilon_i, \varepsilon_i)\}$ for each $\alpha \in o(a, \sigma)$
- $U.pre(\varepsilon_p) = \phi$
- $U.pre(\varepsilon_i) = \top$
- $U.sub(\varepsilon_p) = E_a^+ \cup E_a^-$ where:
  - $E_a^+ = \{f \rightarrow \phi \vee f \mid [a \textbf{ causes } f \textbf{ if } \phi] \in \Delta\}$
  - $E_a^- = \{f \rightarrow \neg\phi \wedge f \mid [a \textbf{ causes } \neg f \textbf{ if } \phi] \in \Delta\}$
- $U.sub(\varepsilon_i) = \epsilon$

and $\Gamma = \{\varepsilon_p\}$. ◇

Now that we have defined the means by which we obtain the appropriate update instances, we may proceed with defining the transition function.

Definition 3.15 (The Transition Function)**.** Let Δ be an action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state of the transition diagram defined by Δ, and a be an action. The successor state(s) obtained by performing the action a in the state σ are defined as follows:

$$\Phi_\Delta(\sigma, a) = \begin{cases} \sigma \otimes U_o(\sigma, a) & \text{ontic action} \\ \sigma \otimes U_s(\sigma, a) & \text{sensing action} \\ \sigma \otimes U_a(\sigma, a) & \text{otherwise} \end{cases}$$

◇

It can be shown that the semantics has a number of desirable properties, among which are the following:

Theorem 3.2. *Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be a sensing action that is executable in σ. Let $(M', \omega') \in \sigma \otimes U_s(\sigma, a)$. It holds that:*

- $\forall f \in \{f \mid [a \textbf{ determines } f] \in \Delta\}$, $(M', \omega') \models C_{f(a,\sigma)}\lambda$ *iff* $(M, \omega) \models \lambda$ *where* $\lambda \in \{f, \neg f\}$
- $\forall f \in \{f \mid [a \textbf{ determines } f] \in \Delta\}$, $(M', \omega') \models C_{p(a,\sigma)}(C_{f(a,\sigma)}f \vee C_{f(a,\sigma)}\neg f)$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *iff* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Intuitively, Theorem 3.2 states that agents who have first hand knowledge of the action occurrence will all know/believe the truth values of the sensed fluent. Those agents who have indirect knowledge of the action occurrence will know/believe that those agents who were involved in the action occurrence have learned the value of the sensed fluent. Lastly, the beliefs of oblivious agents carry over from one state to the next due to inertia.

Theorem 3.3. *Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an announcement action that is executable in σ that is governed by the announcement axiom $[a \textbf{ announces } \varphi] \in \Delta$. Let $(M', \omega') \in \sigma \otimes U_a(\sigma, a)$. It holds that:*

- $(M', \omega') \models C_{f(a,\sigma)}\varphi$
- $(M', \omega') \models C_{p(a,\sigma)}(C_{f(a,\sigma)}\varphi \vee C_{f(a,\sigma)}\neg\varphi)$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *iff* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Theorem 3.3 states that agents who have first hand knowledge of the action occurrence will all know/believe the truth values of the announced formula. Those agents who have indirect knowledge of the action occurrence will know/believe that those agents who were involved in the action occurrence have learned the value of the respective formula. Lastly, the beliefs of oblivious agents carry over from one state to the next due to inertia.

Theorem 3.4. *Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an ontic action that is executable in σ. Let $(M', \omega') \in \sigma \otimes U_s(\sigma, a)$. It holds that:*

- $\forall \alpha \in f(a, \sigma)$*, dynamic causal law* $[a$ ***causes*** λ ***if*** $\phi]$ $\in \Delta$*, and* $\omega_1, \omega_2 \in M.W$*:*
  - $(\varepsilon_p, \omega_1) \models \lambda$ *if* $(M, \omega_1) \models \phi$
  - $(\omega_1, \omega_2) \in M.R_\alpha$ *iff* $((\varepsilon_p, \omega_1), (\varepsilon_p, \omega_2)) \in M'.R_\alpha$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *iff* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Theorem 3.4 keeps the same pattern, stating that those agents which have first hand knowledge of the action occurrence know/believe in all of its effects, while the knowledge/beliefs of oblivious agents remain unchanged.

3.3 Representing Dynamic Domains with $m\mathcal{A}+$

Having defined the syntax and semantics of $m\mathcal{A}+$ in the previous section, we now shift our focus to presenting the a couple of in-depth examples of its application in modeling the Classical Muddy Children Problem [19] and a new multi-agent domain known as the Escapee Domain [15].

3.3.1 The Classical Muddy Children Domain

To recap Example 1.1, the Classical Muddy Children Problem goes as follows: N children are playing together outside and during their play, some of them get mud on their foreheads. Each child can see the mud on the foreheads of their siblings, but cannot tell whether or not they themselves are muddy. This fact is common knowledge among them. Their father comes along and announces that at least one of them is muddy. He then repeatedly asks "do you know whether or not you are muddy" until all of the children reply in the affirmative. Assuming that all of the children are intelligent, honest, and answer in unison, it can be shown that if K children are muddy, then after the K^{th} time the father asks his question, the children will all reply "yes". As stated previously, here we concern ourselves with an instance of this problem where $N = K = 3$.

The solution presented in [19] takes the form of a dialogue where the initial state of the domain is presented as a given, as is the trajectory that unfolds as a consequence of a sequence of *public announcements*. In this section we present a complete formalization of the domain in the language of $m\mathcal{A}+$ and show how the semantics of the language enables us to obtain the precise trajectory presented in [19] and Example 1.1.

Example 3.16 (Representing the Classical Muddy Children Problem)**.** *We begin our representation by defining the domain signature. A close reading of the domain suggests the following:*

$$\mathcal{AG} = \{A, B, C\}$$

$$\mathcal{F} = \{muddy(\alpha)\}$$

$$\mathcal{A} = \{declare, respond\}$$

where α is a variable over $\mathcal{AG}$. In this axiomatization, the three children are named A,

B, and C; fluents of the form muddy(α) are read as "α is muddy"; the action, declare, represents the father's initial announcement that at least one child is muddy; and finally the action, respond, represents the children's responses of "no" to their father's questions.

With the signature firmly in place, we can now present the initial state axioms. Turning once again to the domain description we know that the following are common knowledge/-beliefs among the children:

- *Each child can see the mud on the foreheads of their siblings.*
- *Each child cannot tell whether or not they themselves are muddy.*

The former can be represented by the following collection of initial state axioms:

$$\textbf{\textit{initially}}\ C(\mathcal{K}_A muddy(B) \vee \mathcal{K}_A \neg muddy(B)) \tag{3.26}$$

$$\textbf{\textit{initially}}\ C(\mathcal{K}_A muddy(C) \vee \mathcal{K}_A \neg muddy(C)) \tag{3.27}$$

$$\textbf{\textit{initially}}\ C(\mathcal{K}_B muddy(A) \vee \mathcal{K}_B \neg muddy(A)) \tag{3.28}$$

$$\textbf{\textit{initially}}\ C(\mathcal{K}_B muddy(C) \vee \mathcal{K}_B \neg muddy(C)) \tag{3.29}$$

$$\textbf{\textit{initially}}\ C(\mathcal{K}_C muddy(A) \vee \mathcal{K}_C \neg muddy(A)) \tag{3.30}$$

$$\textbf{\textit{initially}}\ C(\mathcal{K}_C muddy(B) \vee \mathcal{K}_C \neg muddy(B)) \tag{3.31}$$

while the latter is captured by the following:

$$\textbf{\textit{initially}}\ C\neg\mathcal{K}_A muddy(A) \tag{3.32}$$

$$\textbf{\textit{initially}}\ C\neg\mathcal{K}_A \neg muddy(A) \tag{3.33}$$

$$\textbf{\textit{initially}}\ C\neg\mathcal{K}_B muddy(B) \tag{3.34}$$

$$\textbf{\textit{initially}}\ C\neg\mathcal{K}_B \neg muddy(B) \tag{3.35}$$

$$\textbf{\textit{initially}}\ C\neg\mathcal{K}_B muddy(C) \tag{3.36}$$

$$\textbf{\textit{initially}}\ C\neg\mathcal{K}_B \neg muddy(C) \tag{3.37}$$

If we label each point of the minimal model for axioms (3.26) *through* (3.37) *by the agents who are muddy (as a shorthand for the positive and negative forms of the fluents muddy*α*), we obtain the initial state shown in Figure 3.5.*

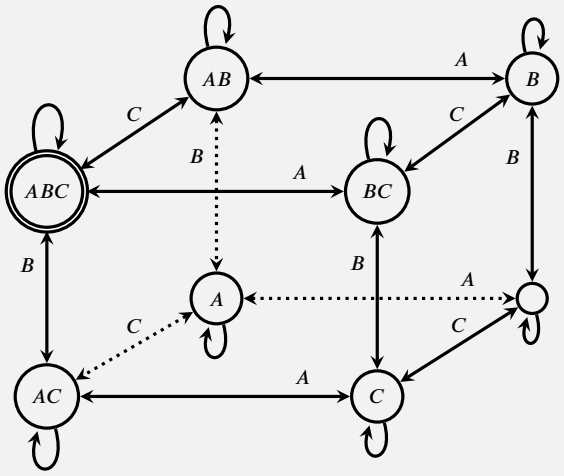

Figure 3.5: Minimal model of the initial state axioms for the Muddy Children Problem.

As described by the domain signature, there are two actions in this particular representation: declare, and respond. Each of these however are specific instances of a kind of action known as a public announcement*. Public announcements are a kind of communication action which are in essence observed (or participated in) jointly by all of the agents operating within the domain. As we shall see, these can be represented in a rather straightforward manner in m*$\mathcal{A}$*+.*

The action declare corresponds a public announcement of the fact that "at least one of the children is muddy." The action itself is represented by the following announcement axiom*:*

$$declare \ \textbf{\textit{announces}} \ muddy(A) \vee muddy(B) \vee muddy(C) \tag{3.38}$$

while its public nature is modeled by the general observability axiom:

$$\{A, B, C\} \textbf{ observes } declare \tag{3.39}$$

In the context of this domain, axiom (3.39) *can be read as: "every agent of the domain is fully aware of any occurrence of the action declare." It is worth noting that this axiom states that the observability of the action is* independent *of the state in which it occurs, and hence the semantics of the occurrence is given by the update instance shown in Figure 3.6, with* $\varphi = muddy(A) \vee muddy(B) \vee muddy(C)$.

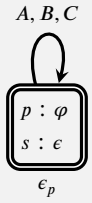

Figure 3.6: Update instance for the father's declaration in the Muddy Children Problem.

At this point it is worth noting that application of the transition function defines the following trajectory for the father's announcement in the initial state (in keeping with our intuition from Chapter 1.1.1):

As was the case with the action declare, the action respond is also a public announcement, this time made by the children. Here, the formula being announced corresponds to the statement that "none of the children know whether or not they are muddy." As with declare, the action may be modeled by the following axioms:

$$respond \textbf{ announces } \neg\mathcal{K}_\alpha muddy(\alpha) \wedge \neg\mathcal{K}_\alpha \neg muddy(\alpha) \tag{3.40}$$

$$\{A, B, C\} \textbf{ observes } respond \tag{3.41}$$

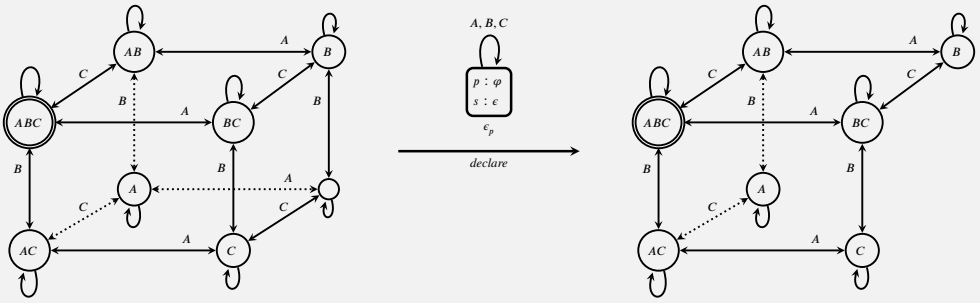

Figure 3.7: Trajectory defined by the father's declaration in the Muddy Children Problem.

The reader will note the use of the variable α *in axioms* (3.40) *and* (3.41) *is merely a convenient shorthand notation. A careful application of the transition function shows that the axiomatization presented here yields the same trajectory as the canonical solution presented in Chapter 1.1.1 and [19].* $\diamond$

A careful study of axioms (3.26) through (3.41) brings out two of the virtues of the action language approach generally, and of $m\mathcal{A}+$ more specifically. Firstly, the textual nature of the action description makes it unnecessary for a system designer to manually specify all of the possible update instances for every potential occurrence of an action. It is sufficient to write general axioms which govern an an action as an *action category*, and leave it to the semantics to define the instances for specific action occurrences. Secondly, the separation of the *observability of an occurrence* from the definition of an *action* allows us to represent action categories more generally, and allowing the public or private nature of an action occurrence to now be dependent on the state in which it occurs. This second virtue will be more readily apparent in our discussion of the Escapee Domain in the next section.

3.3.2 The Escapee Domain

The Escapee Domain, which was first presented in [15], is an important multi-agent domain as it illustrates the importance of being able to *model actions which alter the observability of subsequent actions* performed by the agents in the domain. In addition, it shows how *certain actions can be modeled through sequences of more primitive, elementary actions*. The domain goes as follows: Suppose that agent A is held captive by a hostile agent B. In order to escape, A must open his cell without B's knowledge. Fortunately, agent C is a double agent in B's organization and may release A from his cell. C does not want to break his cover however, so he may release A only if B is not watching. Once A has been released, he must work together with C to subdue agent B, and then make his escape. A will only work with C if he believes that C is an ally.

Example 3.17 (Representing the Escapee Domain)**.** *As before, we begin our representation by presenting the domain signature, as well as the initial state description. A close reading of the problem description suggests the following signature:*

$$\mathcal{AG} = \{A, B, C\}$$

$$\mathcal{F} = \{free(\alpha), bound(\alpha), captor(\alpha_1, \alpha_2), attentive(\alpha), allies(\alpha_1, \alpha_2), united(\alpha_1, \alpha_2)\}$$

$$\mathcal{A} = \{escape(\alpha), release(\alpha_1, \alpha_2), subdue(\alpha_1, \alpha_2, \alpha_3), unite(\alpha_1, \alpha_2),$$
$$signal(\alpha_1, \alpha_2), distract(\alpha_1, \alpha_2), tell(\alpha_1, \alpha_2, \varphi)\}$$

where α, and α_i are variables over $\mathcal{AG}$, and φ is a formula of the form $allies(\alpha_1, \alpha_2)$.

Unlike the Classical Muddy Children Problem, it is useful to identify two broader categories of fluents: ontic fluents *and* perspective fluents*. Ontic fluents are used to describe actual physical properties of the domain, while perspective fluents (while fluents), are additionally used to define the observability of action occurrences as a function of the state*

in which they transpire. In this particular domain, fluents of the form attentive(α) *are* perspective fluents. *In addition, fluents of the form united*(α_1, α_2) *are also a kind of perspective fluent which is used in modeling collaborative actions. These fluents may be manipulated directly by the agents via the perspective altering actions signal/distract and unite/disband, respectively.*

With the signature firmly in place, we can now present the initial state axioms. Initially, it is common knowledge that each child knows the status of his fellows. This is represented fairly readily by the following axioms:

$$\textbf{\textit{initially}}\ \mathcal{C}_{\{A,B,C\}}(\mathit{attentive}(\alpha)) \wedge \mathit{bound}(A) \tag{3.42}$$

$$\textbf{\textit{initially}}\ \mathcal{C}_{\{A,B,C\}}(\neg\mathcal{B}_A \mathit{allies}(A,C) \wedge \neg\mathcal{B}_B \mathit{allies}(B,C)) \tag{3.43}$$

$$\textbf{\textit{initially}}\ \mathcal{B}_C \mathit{allies}(A,C) \tag{3.44}$$

Axiom (3.42) *states that initially it is a commonly held belief amongst all of the agents that* A *is bound, and that all of them are attentive to their surroundings. Axiom* (3.43) *states that is also a commonly held belief that neither agents* A *nor* B *believe that* C *is allied with agent* A*. Finally, axiom* (3.44) *states that agent* C *does believe himself to be a double agent allied with* A*.*

Unlike our previous example, the Escapee Domain has a large number of elementary actions. Another avenue of departure is that the domain presents an opportunity to explore how sequences of elementary actions allow for the modeling of collaborative, and also private actions on the part of the agents. In the interest of clarity, we will divide the actions into two categories: perspective altering *and* general *actions. We begin our representation by focusing on the perspective altering actions signal/distract and unite/disband.*

The actions signal and distract in a straightforward manner as before, with their observability limited to those agents directly involved in the action occurrences, and attentive

agents:

$$\mathit{signal}(\alpha_1, \alpha_2) \textbf{ causes } \mathit{attentive}(\alpha_2) \tag{3.45}$$

$$\mathit{distract}(\alpha_1, \alpha_2) \textbf{ causes } \neg\mathit{attentive}(\alpha_2) \tag{3.46}$$

$$\{\alpha_1, \alpha_2\} \textbf{ observes } \mathit{signal}(\alpha_1, \alpha_2) \tag{3.47}$$

$$\{\alpha\} \textbf{ observes } \mathit{signal}(\alpha_1, \alpha_2) \textbf{ if } \mathit{attentive}(\alpha) \tag{3.48}$$

$$\{\alpha_1, \alpha_2\} \textbf{ observes } \mathit{distract}(\alpha_1, \alpha_2) \tag{3.49}$$

$$\{\alpha\} \textbf{ observes } \mathit{distract}(\alpha_1, \alpha_2) \textbf{ if } \mathit{attentive}(\alpha) \tag{3.50}$$

In general, agents may unite in order to act together. In the context of the Escapee Domain, an agent must be ¬bound before he may unite with another agent to collaboratively perform some action. In addition, an agent will only unite with someone whom he believes is an ally. Once they are done collaborating, they may disband. This behavior is defined by the following axioms:

$$\mathit{unite}(\alpha_1, \alpha_2) \textbf{ causes } \mathit{united}(\alpha_1, \alpha_2) \tag{3.51}$$

$$\textbf{executable } \mathit{unite}(\alpha_1, \alpha_2) \textbf{ if } \neg\mathit{bound}(\alpha_1) \wedge \neg\mathit{bound}(\alpha_2) \wedge \mathcal{B}_{\alpha_1}\mathit{allies}(\alpha_1, \alpha_2) \tag{3.52}$$

$$\mathit{disband}(\alpha_1, \alpha_2) \textbf{ causes } \neg\mathit{united}(\alpha_1, \alpha_2) \tag{3.53}$$

The observation axioms governing the frames of reference of the agents with respect to occurrences of the actions unite and disband follow the same pattern as those for the actions signal and distract:

$$\{\alpha_1, \alpha_2\} \textbf{ observes } \mathit{unite}(\alpha_1, \alpha_2) \tag{3.54}$$

$$\{\alpha\} \textbf{ observes } \mathit{unite}(\alpha_1, \alpha_2) \textbf{ if } \mathit{attentive}(\alpha) \tag{3.55}$$

$$\{\alpha_1, \alpha_2\} \textbf{ observes } \mathit{disband}(\alpha_1, \alpha_2) \tag{3.56}$$

$$\{\alpha\} \textbf{ observes } \mathit{disband}(\alpha_1, \alpha_2) \textbf{ if } \mathit{attentive}(\alpha) \tag{3.57}$$

Now that we have finished defining the behaviors of the perspective altering actions, we turn our attention to the remaining actions. A single agent may release another agent causing him to no longer be bound. A pair of agents working together may subdue an agent, causing him to be bound.

$$release(\alpha_1, \alpha_2) \textbf{ causes } \neg bound(\alpha_2) \tag{3.58}$$

$$subdue(\alpha_1, \alpha_2, \alpha_3) \textbf{ causes } bound(\alpha_3) \tag{3.59}$$

$$\textbf{executable } subdue(\alpha_1, \alpha_2, \alpha_3) \textbf{ if } united(\alpha_1, \alpha_2) \vee united(\alpha_2, \alpha_1) \tag{3.60}$$

The observation axioms for the actions release and subdue also follow directly from our intuition:

$$\{\alpha_1, \alpha_2\} \textbf{ observes } release(\alpha_1, \alpha_2) \tag{3.61}$$

$$\{\alpha\} \textbf{ observes } release(\alpha_1, \alpha_2) \textbf{ if } attentive(\alpha) \tag{3.62}$$

$$\{\alpha_1, \alpha_2, \alpha_3\} \textbf{ observes } subdue(\alpha_1, \alpha_2, \alpha_3) \tag{3.63}$$

The representation of the action escape is fairly straightforward as well. Once an agent has escaped, he is free. From the domain description, we know that an agent (in this case A), may only escape once his captor has been subdued (i.e. bound). The relevant observation axioms follow a now familiar pattern.

$$escape(\alpha) \textbf{ causes } free(\alpha) \tag{3.64}$$

$$\textbf{executable } escape(\alpha_1) \textbf{ if } captor(\alpha_2, \alpha_1) \wedge bound(\alpha_2) \wedge \tag{3.65}$$

$$(\neg united(\alpha_1, \alpha_3) \vee \neg united(\alpha_3, \alpha_1)) \tag{3.66}$$

$$\{\alpha\} \textbf{ observes } escape(\alpha) \tag{3.67}$$

$$\{\alpha_2\} \textbf{ observes } escape(\alpha_1) \textbf{ if } attentive(\alpha_2) \tag{3.68}$$

Lastly, an agent may tell another agent some fact about the domain. The action tell is a communication action, and is represented by the corresponding axiom:

$$tell(\alpha_1, \alpha_2, \varphi) \textbf{ communicates } \varphi \tag{3.69}$$

where φ is of the form allies(α_1, α_2). Agents may eavesdrop however, and therefore in the Escapee Domain, communication must be done with caution. For this domain, we assume that attentive agents are fully aware of what is said between their fellows. This assumption is encoded by the following observation axioms:

$$\{\alpha_1, \alpha_2\} \textbf{ observes } tell(\alpha_1, \alpha_2, \varphi) \tag{3.70}$$

$$\{\alpha_3\} \textbf{ observes } tell(\alpha_1, \alpha_2, \varphi) \textbf{ if } attentive(\alpha_3) \tag{3.71}$$

$\diamond$

A careful study of Example 3.17 shows that private actions (both ontic and involving communication) can be defined and reasoned about in terms of *sequences of elementary actions*. Example 3.18 emphasizes this important trait of $m\mathcal{A}+$, which distinguishes it from the approach of [4, 43] which would require the definition of potentially complex update models to model such actions. This high level view has a benefit from a knowledge representation standpoint in that it allows a system designer to focus on modeling an orthogonal set of primitive actions, and that the resulting action description is also transparent to a reader. This also allows for the development of automated systems to solve the *temporal projection* and *planning* problems which will be discussed in greater detail in Chapter 5.

Example 3.18 (Private Communication in the Escapee Domain)**.** *In order for agent A to eventually escape, agent C must first inform A that they are allies. This communication must be done privately however so that C may maintain his cover. In the language of update models, this would be achieved by defining an update model for the following action:*

Agent C *tells* A *that* $allies(A, C)$ *is true with* B *oblivious of what has transpired. Using our axiomatization however we can model this as the result of the following action sequence:*

$$[distract(C, B), tell(C, A, allies(A, C))]$$

Axiom (3.46) *states that the consequence of* $distract(C, B)$ *will be to cause* $\neg attentive(B)$ *to be come true in the subsequent state of the domain. This in turn will cause the occurrence of the action* $tell(C, A, allies(A, C))$ *to be done without the knowledge of agent* B *due to axioms* (3.70) *and* (3.71). ◇

Chapter 4

THE ACTION LANGUAGE $m\mathcal{AL}$

The action language $m\mathcal{A}+$ described in Chapter 3 is a first step towards the development of a multi-agent action language. The language's structure closely mirrors that of the action language $\mathcal{A}$ and consequently, it inherits some of $\mathcal{A}$'s deficiencies, mainly the lack of *state constraints* (also known as *static causal laws*), the importance of which was illustrated by the Lin's Briefcase Domain in [32]. The action language $\mathcal{AL}$ was developed partly in response to this discovery [27], and adding such constructs is a natural evolution of the language $m\mathcal{A}+$. As a motivating example, consider the multi-agent variant of the Lin's Briefcase Domain of [32] presented in Example 4.1.

Example 4.1 (A Multi-Agent Lin's Briefcase Domain)**.** *Three agents, A, B, and C, are together in a room with a locked briefcase which contains a coin. The briefcase is locked by two independent latches, each of which may be flipped open (or closed) by an agent. Once both latches are open, the briefcase is unlocked and an agent may peek inside to determine which face of the coin is showing. In addition, agent may signal or distract one of his fellows, thereby causing him to be respectively attentive or inattentive. Attentive agents are fully aware of what transpires around them. Suppose that the briefcase is locked, and that this fact, together with the fact that none of the agents knows which face of the coin is showing is a commonly held belief among them. Furthermore let us suppose that all of the agents are attentive and that this too is a commonly held belief. Finally, let us assume that the coin is actually facing heads up.* ◇

The domain presented in Example 4.1 is an interesting elaboration of the Concealed Coin Domain of [4] due to the dependency between the state of the lock and its governing

latches. $m\mathcal{A}+$ lacks a mechanism for describing complex dependencies between fluents, hence the addition of state constraints, giving us the multi-agent action language $m\mathcal{AL}$.

4.1 Syntax

Theories of $m\mathcal{AL}$ are defined over a multi-agent domain $\mathcal{D}$ with a signature Σ. Like its predecessor, $m\mathcal{AL}$ supports two broad classes of actions: *ontic* and *epistemic* actions, and as with $m\mathcal{A}+$, epistemic actions are divided into *sensing* and *communication* actions. The language also inherits the notion of initial state axioms and initial state descriptions from $m\mathcal{A}+$, and hence we refer the reader to Chapter 3.1.

The direct effects of ontic actions are described by *dynamic causal laws* which are statements of the form:

$$a \textbf{ causes } \lambda \textbf{ if } \phi \tag{4.1}$$

where a is an action, λ is a fluent literal, and ϕ is a conjunction of fluent literals. Laws of this form are read as: "performing the action a in a state which satisfies ϕ causes λ to be true." If ϕ is a tautology, then we simply write the following:

$$a \textbf{ causes } \lambda \tag{4.2}$$

Sensing actions are described by *sensing axioms* which have the form:

$$a \textbf{ determines } f \tag{4.3}$$

where a is the name of an action, and f is a fluent. Statements of this form are understood to mean: "if an agent performs the action a, he will learn the value of the fluent f." Communication actions are described by *communication axioms* which have the form:

$$a \textbf{ communicates } \varphi \tag{4.4}$$

where a is the name of an action, and φ is a modal formula. In $m\mathcal{AL}$ only truthful announcements are allowed.

The constructs (4.1) – (4.4) only describe the *direct effects* of their respective actions. In general, an agent's actions may indirectly affect the knowledge/beliefs of his fellows, as well as the values of various fluents. As in $m\mathcal{A}+$, for any given action occurrence we divide the agents of the domain into three groups known as *frames of reference* (or *levels of awareness*):

- those which have first-hand knowledge of the action occurrence
- those which have second-hand knowledge of the action occurrence
- those which have no knowledge of the action occurrence

Agents which have first-hand knowledge of an action occurrence are known as *full observers* (or *fully aware agents*). Such agents have full knowledge of both the action occurrence and its effects. Agents which have second-hand knowledge are known as *partial observers* (or *partially aware agents*). Agents are said to have second-hand knowledge of an action occurrence if they observe an action occurrence *from a distance*. Such agents know that the action took place and who the participants were, but do not know its full consequences. Lastly, agents which have no knowledge of an action occurrence are called *oblivious agents*.

The frames of reference of the agents are dynamic in nature, and depend on the properties of the state in which the action occurs. In this work, we consider the possible frames of reference of the agents for different kinds of actions to be as shown in Table 4.1.

action type	**full observers**	**partial observers**	**oblivious**
ontic	✓		✓
sensing	✓	✓	✓
announcements	✓	✓	✓

Table 4.1: Action classes and frames of reference in $m\mathcal{AL}$.

Frames of reference are described by *perspective axioms* which are statements of the form:

$$X \textbf{ observes } a \textbf{ if } \phi \tag{4.5}$$

$$X \textbf{ aware of } a \textbf{ if } \phi \tag{4.6}$$

where X is a set of agent names, a is an action, and ϕ is a modal formula. Perspective axioms of the first form (called *observation axioms*) define the set of agents who are fully aware of both the action occurrence and its effects. Those of the second form (called *awareness axioms*) define the set of agents who are aware of the occurrence, but only partially of its effects. By default, we assume that all other agents within the domain are oblivious. As with dynamic causal laws, if ϕ is a tautology, we adopt the following shorthand:

$$X \textbf{ observes } a \tag{4.7}$$

$$X \textbf{ aware of } a \tag{4.8}$$

The inclusion of observation axioms allows us to make explicit the assumption that agents are aware of the actions they perform. In $m\mathcal{AL}$, the only assumptions made regarding the frames of reference of the agents are that those who are fully aware of an action occurrence and its effects, as well as those who are aware only of the occurrence, know the frames of reference of all of the agents within the domain.

Unlike $m\mathcal{A}+$, $m\mathcal{AL}$ also allows us to represent indirect effects of the second form by including *state constraints* which are statements of the form:

$$\lambda \textbf{ if } \phi \tag{4.9}$$

where λ is a fluent literal and ϕ is a conjunction of fluent literals. Statements of this form are read as: "if ϕ is true in a state, then λ must also be true in that state."

Lastly, *executability conditions*, which are statements of the form:

$$\textbf{\textit{impossible}}\ a\ \textbf{\textit{if}}\ \phi \tag{4.10}$$

where a is an action and ϕ is a modal formula, are used to describe when actions may not be performed.

Definition 4.2 (Action Description of $m\mathcal{AL}$)**.** An *action description*, Δ, in $m\mathcal{AL}$ is a collection of statements of the form (4.1)–(4.10). ◇

Example 4.3 (Representing the Multi-Agent Lin's Briefcase Domain)**.** *As before, we begin our axiomatization by specifying the domain signature. A close reading of Example 4.1 suggests the following:*

$$\alpha = \{A, B, C\}$$

$$\mathcal{F} = \{open(\lambda), locked, heads, attentive(\alpha)\}$$

$$\mathcal{A} = \{flip(\alpha, \lambda), peek(\alpha), signal(\alpha_1, \alpha_2), distract(\alpha_1, \alpha_2)\}$$

and α, α_1, and α_2, are variables over $\mathcal{AG}$, and λ is a variable ranging over the set $\{l_1, l_2\}$.

With the signature now in place, we can now give the initial state description, and the causal relationships between the actions and their effects. Example 4.1 states that initially:

- *it is a common belief amongst the agents that the briefcase contains a coin*
- *it is a common belief amongst the agents that none of them know which face of the coin is showing*
- *it is a common belief amongst them that all of the agents are attentive*
- *it is a common belief amongst them that the briefcase is locked (and hence each of the two latches is closed)*

- *all of the agents are in fact attentive*
- *the coin is actually facing heads up*
- *the briefcase is locked (and hence each of the two latches is closed)*
- *all of the beliefs of the agents are in fact true*

This information may be represented by the following initial state description:

$$\textbf{\textit{initially}}\ locked \tag{4.11}$$

$$\textbf{\textit{initially}}\ C(locked) \tag{4.12}$$

$$\textbf{\textit{initially}}\ \neg open(l_1) \wedge \neg open(l_2) \tag{4.13}$$

$$\textbf{\textit{initially}}\ heads \tag{4.14}$$

$$\textbf{\textit{initially}}\ C(\neg \mathcal{B}_A heads \wedge \neg \mathcal{B}_A \neg heads) \tag{4.15}$$

$$\textbf{\textit{initially}}\ C(\neg \mathcal{B}_B heads \wedge \neg \mathcal{B}_B \neg heads) \tag{4.16}$$

$$\textbf{\textit{initially}}\ C(\neg \mathcal{B}_C heads \wedge \neg \mathcal{B}_C \neg heads) \tag{4.17}$$

$$\textbf{\textit{initially}}\ attentive(A) \wedge attentive(B) \wedge attentive(C) \tag{4.18}$$

$$\textbf{\textit{initially}}\ C(attentive(A) \wedge attentive(B) \wedge attentive(C)) \tag{4.19}$$

Now that we have described the initial state, we now shift our attention towards the actions and their effects. The direct effects of the action flip(α, λ) *are represented via the following pair of dynamic causal laws:*

$$flip(\alpha, \lambda)\ \textbf{\textit{causes}}\ open(\lambda)\ \textbf{\textit{if}}\ \neg open(\lambda) \tag{4.20}$$

$$flip(\alpha, \lambda)\ \textbf{\textit{causes}}\ \neg open(\lambda)\ \textbf{\textit{if}}\ open(\lambda) \tag{4.21}$$

The following state constraint models the indirect effects of the action, flip(α, λ)*, namely that the briefcase is unlocked once both latches are open.*

$$\neg locked\ \textbf{\textit{if}}\ open(l_1) \wedge open(l_2) \tag{4.22}$$

The agent directly performing the action flip(α, λ)*, as well as any attentive agents are considered to be fully aware of the action occurrence and of its full effects. This information may be encoded by the following pair of perspective axioms:*

$$\{\alpha\}\ \textbf{\textit{observes}}\ \textit{flip}(\alpha, \lambda) \tag{4.23}$$

$$\{\alpha_2\}\ \textbf{\textit{observes}}\ \textit{flip}(\alpha_1, \lambda)\ \textbf{\textit{if}}\ \textit{attentive}(\alpha_2) \tag{4.24}$$

The action, peek(α)*, is an epistemic action — in particular, it is a sensing action. Consequently its direct effects are represented by the following sensing axiom:*

$$\textit{peek}(\alpha)\ \textbf{\textit{determines}}\ \textit{heads} \tag{4.25}$$

The fact that an agent may not peek into a locked briefcase is represented by the following executability condition:

$$\textbf{\textit{impossible}}\ \textit{peek}(\alpha)\ \textbf{\textit{if}}\ \textit{locked} \tag{4.26}$$

Unlike the action flip(α, λ)*, only the agent who is peeking is fully aware of the occurrence and its full effects. Agents who are attentive, are only partially aware of the action's effects. This is modeled by the following perspective axioms:*

$$\{\alpha\}\ \textbf{\textit{observes}}\ \textit{peek}(\alpha) \tag{4.27}$$

$$\{\alpha_2\}\ \textbf{\textit{aware of}}\ \textit{peek}(\alpha_1)\ \textbf{\textit{if}}\ \textit{attentive}(\alpha_2) \tag{4.28}$$

Lastly, the actions signal(α_1, α_2) *and distract*(α_1, α_2) *are described in a similar fashion:*

$$\textit{signal}(\alpha_1, \alpha_2)\ \textbf{\textit{causes}}\ \textit{attentive}(\alpha_2) \tag{4.29}$$

$$\{\alpha_1, \alpha_2\}\ \textbf{\textit{observes}}\ \textit{signal}(\alpha_1, \alpha_2) \tag{4.30}$$

$$\{\alpha\}\ \textbf{\textit{observes}}\ \textit{signal}(\alpha_1, \alpha_2)\ \textbf{\textit{if}}\ \textit{attentive}(\alpha) \tag{4.31}$$

$$\textit{distract}(\alpha_1, \alpha_2)\ \textbf{\textit{causes}}\ \neg\textit{attentive}(\alpha_2) \tag{4.32}$$

$$\{\alpha_1, \alpha_2\}\ \textbf{\textit{observes}}\ \textit{distract}(\alpha_1, \alpha_2) \tag{4.33}$$

$$\{\alpha\}\ \textbf{\textit{observes}}\ \textit{distract}(\alpha_1, \alpha_2)\ \textbf{\textit{if}}\ \textit{attentive}(\alpha) \tag{4.34}$$

◇

4.2 Semantics

As is the case with other action languages, an action description of $m\mathcal{AL}$ defines a transition diagram whose nodes correspond to *states of the domain* — which we model as Kripke worlds — and whose arcs are labeled by *actions*. Within a particular state, the points comprising the underlying Kripke world correspond to complete consistent sets of fluent literals closed under the state constraints of the action description.

4.2.1 Deriving the Initial State

As was mentioned previously, $m\mathcal{AL}$ inherits the notions of initial state axiom, and initial state description from its predecessor. Likewise, in $m\mathcal{AL}$ we make use of the notion of a definite initial state description from Section 3.2.1 and their associated semantics.

Example 4.4 (Multi-Agent Lin's Briefcase Domain — Initial State)**.** *The initial state description presented in Example 4.3 is a complete and consistent definite initial state description, and corresponds to the canonical initial state,* $\sigma_0 = (M_0, \omega_1)$ *shown in Figure 4.1.* σ_0 *consists of two points,* ω_1 *and* ω_2*, where:*

- $M_0.\pi(\omega_1) = \{heads, attentive(A), attentive(B), attentive(C), \neg open(l_1), \neg open(l_2), locked\}$
- $M_0.\pi(\omega_2) = \{\neg heads, attentive(A), attentive(B), attentive(C), \neg open(l_1), \neg open(l_2), locked\}$

◇

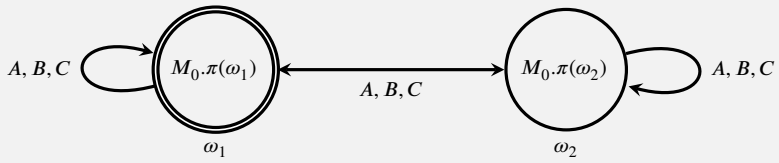

Figure 4.1: The initial state, σ_0, of the Multi-Agent Lin's Briefcase Domain.

4.2.2 The Transition Function

As was the case with $m\mathcal{A}+$, before we define the transition function of $m\mathcal{AL}$, we must define what it means for an action to be *executable* in a given state, and the frames of reference the agents have with respect to an action occurrence in a state.

Definition 4.5 (Executable Action)**.** Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$, be a state in the transition diagram defined by Δ, and a be an action for which there exists an impossibility condition [***impossible*** a ***if*** ϕ] $\in \Delta$. The action a is *executable in* (M, ω) if $(M, \omega) \nvDash \phi$. $\diamond$

Now that we have defined the notion of executability, we proceed with defining the frames of reference of the agents.

Definition 4.6 (Frames of Reference)**.** Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an action which occurs in σ. The various *frames of reference* of the agents are defined as follows:

- the set of agents who have *first hand knowledge* (or are *fully aware*) of a, denoted by $f(a, \sigma)$ is $\{\alpha \in \mathcal{AG} \mid [\alpha$ ***observes*** a ***if*** $\phi] \in \Delta \wedge (M, \omega) \vDash \phi\}$
- the set of agents who have *second hand knowledge* (or are *partially aware*) of a, denoted by $p(a, \sigma)$ is $\{\alpha \in \mathcal{AG} \mid [\alpha$ ***aware of*** a ***if*** $\phi] \in \Delta \wedge (M, \omega) \vDash \phi\}$

- the set of agents who have *no knowledge* (or are *oblivious*) of a, denoted by $o(a, \sigma)$ is $\mathcal{AG} \setminus (f(a, \sigma) \cup p(a, \sigma))$

$\diamond$

As was the case in $m\mathcal{A}+$, in $m\mathcal{AL}$ we make the following assumptions about the agents' frames of reference: members of $f(a, \sigma)$ know who their fellows are and the members of both $p(a, \sigma)$ and $o(a, \sigma)$; members of $p(a, \sigma)$ know who their fellows are as well as the members of $o(a, \sigma)$; and lastly, members of $o(a, \sigma)$ know who their fellow agents are.

The semantics of $m\mathcal{AL}$ is defined in terms of a transition function. The inclusion of state constraints however make the approach used in defining the semantics of $m\mathcal{A}+$ unsuitable. The approach taken with $m\mathcal{AL}$ is a *hybrid approach* making use of the McCain-Turner equation [34] used in defining the semantics of $\mathcal{AL}$ together with the event models of [4]. The key intuition behind our semantics is that reasoning about the effects of an action is a two step process:

1. The agent first reasons about how his fellows may perceive his action — this is done by constructing the event model representing the action occurrence and using what we call an *epistemic update* to obtain a pointed frame describing the general structure of the successor state.

2. The agent then reasons about how his actions may actually play out in the domain — this is done by expanding the points of the pointed frame obtained in step (1) by using the McCain-Turner equation to obtain the resulting Kripke world describing the successor state.

Before we continue describing the semantics of $m\mathcal{AL}$, we define the notions of *frame/-pointed frame*, and *event model/pointed event model*.

Definition 4.7 (Pointed Frame)**.** Let $\mathcal{D}$ be a multi-agent domain with signature, $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$, where $\mathcal{AG} = \{\alpha_1, \ldots, \alpha_n\}$. A *frame*, F, is a tuple of the form $(W, R_{\alpha_1}, \ldots, R_{\alpha_n})$ where:

- W is a nonempty set of *points*
- each R_{α_i} is a binary relation on W called an *accessibility relation for agent* α_i

A *pointed frame* is a pair, (F, ω), where M is a frame, and ω is a point in F. ⋄

Definition 4.8 (Pointed Event Model)**.** Let $\mathcal{D}$ be a multi-agent domain with the signature $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$, where $\mathcal{AG} = \{\alpha_1, \ldots, \alpha_n\}$. An *event model* over a language, $\mathcal{L}_\Sigma$, $\mathcal{E}$, is a tuple of the form $(E, R_{\alpha_1}, \ldots, R_{\alpha_n}, pre)$ where:

- E is a finite, non-empty set of *events*
- each R_{α_i} is a binary relation on E called an *accessibility relation* for agent α_i
- $pre : E \mapsto \mathcal{L}_\Sigma$ assigns a *precondition* to each event

A *pointed event model* is a pair, $(\mathcal{E}, \varepsilon)$, where $\mathcal{E}$ is an event model, and ε is an event of $\mathcal{E}$. ⋄

4.2.2.1 The Epistemic Update

When reasoning about the effects of an action, an agent first establishes what a *pointed frame* describing the general configuration of the successor state. This is done by constructing a *pointed event model* describing how the agents of the domain perceive the action occurrence. In the context of $m\mathcal{AL}$, we define three particular *pointed event models*, for ontic, sensing, and communication actions respectively: $\mathcal{E}_o(\sigma, a)$, $\mathcal{E}_s(\sigma, a)$, and $\mathcal{E}_c(\sigma, a)$. Graphical representations of the update instances are presented in Figures 4.2 to 4.4.

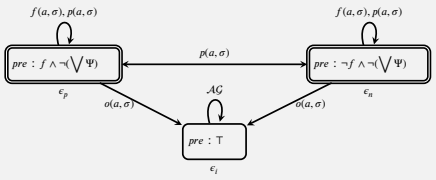

Figure 4.2: General update instance for sensing actions in $m\mathcal{AL}$.

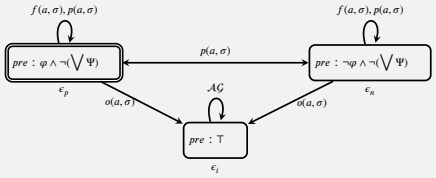

Figure 4.3: General update instance for communication actions in $m\mathcal{AL}$.

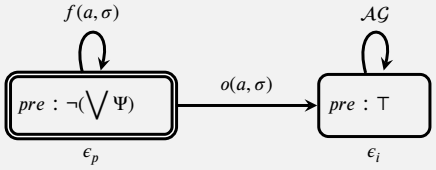

Figure 4.4: General update instance for ontic actions in $m\mathcal{AL}$.

The intuition behind $\mathcal{E}_o(\sigma, a)$ is relatively straightforward: the agents are either aware of the action occurrence or are oblivious. In addition, we make the assumption that those agents who are aware of the action occurrences know which agents are oblivious.

Definition 4.9 (The Ontic Model $\mathcal{E}_o(\sigma, a)$)**.** The function $\mathcal{E}_o(\sigma, a)$ yields the set of pointed event models represented by the pair $(\mathcal{E}, \Gamma)$ where $\mathcal{E}$ is defined as follows:

- $\mathcal{E}.E = \{\varepsilon_p, \varepsilon_i\}$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_i, \varepsilon_i)\}$ for each agent in $f(\sigma, a)$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_i), (\varepsilon_i, \varepsilon_i)\}$ for each agent in $o(\sigma, a)$

Let $\Psi = \{\phi \mid [\textbf{\textit{impossible a if}}\ \phi] \in \Delta\}$.

- $\mathcal{E}.pre(\varepsilon_p) = \neg(\bigvee \Psi)$
- $\mathcal{E}.pre(\varepsilon_i) = \top$

and $\Gamma = \{\varepsilon_p\}$. $\diamond$

$\mathcal{E}_s(\sigma, a)$ is based on the following intuition: the real value of f is revealed to those agents who are performing the action, causing it to become a commonly held belief among them; agents who observe the action learn that the value of f has been revealed to those agents who were directly involved in it; and the beliefs of oblivious agents remain unchanged.

Definition 4.10 (The Sensing Model $\mathcal{E}_s(\sigma, a)$)**.** The function $\mathcal{E}_s(\sigma, a)$ yields the set of pointed event models represented by the pair $(\mathcal{E}, \Gamma)$ where $\mathcal{E}$ is defined as follows:

- $\mathcal{E}.E = \{\varepsilon_p, \varepsilon_n, \varepsilon_i\}$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i)\}$ for each agent in $f(\sigma, a)$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i), (\varepsilon_p, \varepsilon_n), (\varepsilon_n, \varepsilon_p)\}$ for each agent in $p(\sigma, a)$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_i), (\varepsilon_n, \varepsilon_i), (\varepsilon_i, \varepsilon_i)\}$ for each agent in $o(\sigma, a)$

Let f be the fluent determined by the sensing axiom for the action a, and let $\Psi = \{\phi \mid [\textbf{\textit{impossible}}\ a\ \textbf{\textit{if}}\ \phi] \in \Delta\}$.

- $\mathcal{E}.pre(\varepsilon_p) = f \wedge \neg(\bigvee \Psi)$
- $\mathcal{E}.pre(\varepsilon_n) = \neg f \wedge \neg(\bigvee \Psi)$
- $\mathcal{E}.pre(\varepsilon_i) = \top$

and $\Gamma = \{\varepsilon_p, \varepsilon_n\}$. $\diamond$

The intuition behind $\mathcal{E}_c(\sigma, a)$ is similar to that of sensing actions: φ becomes a commonly held belief among those agents who receive/hear the message; agents who observe the action learn that the value of φ has been revealed to those agents who heard it (they are however unaware of the truth of φ); and lastly, the beliefs of oblivious agents are unchanged.

Definition 4.11 (The Communication Model $\mathcal{E}_c(\sigma, a)$)**.** The function $\mathcal{E}_c(\sigma, a)$ yields the set of pointed event models by the pair $(\mathcal{E}, \Gamma)$ where $\mathcal{E}$ is defined as follows:

- $\mathcal{E}.E = \{\varepsilon_p, \varepsilon_n, \varepsilon_i\}$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i)\}$ for each agent in $f(\sigma, a)$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_p), (\varepsilon_n, \varepsilon_n), (\varepsilon_i, \varepsilon_i), (\varepsilon_p, \varepsilon_n), (\varepsilon_n, \varepsilon_p)\}$ for each agent in $p(\sigma, a)$
- $\mathcal{E}.R_\alpha = \{(\varepsilon_p, \varepsilon_i), (\varepsilon_n, \varepsilon_i), (\varepsilon_i, \varepsilon_i)\}$ for each agent in $o(\sigma, a)$

Let φ be the formula specified by the communication axiom for the action a, and let $\Psi = \{\phi \mid [\textbf{\textit{impossible } a \textit{ if }} \phi] \in \Delta\}$.

- $\mathcal{E}.pre(\varepsilon_p) = \varphi \wedge \neg(\bigvee \Psi)$
- $\mathcal{E}.pre(\varepsilon_n) = \neg\varphi \wedge \neg(\bigvee \Psi)$
- $\mathcal{E}.pre(\varepsilon_i) = \top$

and $\Gamma = \{\varepsilon_p\}$. ◇

In order to obtain the pointed frame describing the successor state, we apply an operation that we call the *epistemic update*, which when applied to a state and a pointed event model, yields a *pointed frame* capturing the general structure of the successor state.

Definition 4.12 (Epistemic Update)**.** Given a state, $\sigma = (M, \omega)$, and an pointed event model $\mathcal{U} = (\mathcal{E}, \varepsilon)$, such that $\sigma \vDash \mathcal{E}.pre(\varepsilon)$, $\mathbf{Eu}(\sigma, \mathcal{U})$ defines the pointed frame $(F, (\omega, \varepsilon))$ where:

- $F.W = \{(\omega_j, \varepsilon_j) \mid \omega_j \in M.W, \varepsilon_j \in \mathcal{E}.E, (M, \omega_j) \vDash \mathcal{E}.pre(\varepsilon_j)\}$
- $F.R_\alpha = \{((\omega_j, \varepsilon_j), (\omega_k, \varepsilon_k)) \mid (\omega_j, \omega_k) \in M.R_\alpha, (\varepsilon_j, \varepsilon_k) \in \mathcal{E}.R_\alpha\}$

◇

Example 4.13 (Applying the Epistemic Update)**.** *Recall the initial state of the Multi-Agent Lin's Briefcase Domain shown in Figure 4.1. Suppose that A distracts C. The action, distract*(A, C)*, is an* ontic action *which directly affects the fluent attentive*(C) *as specified by the dynamic causal law* (4.32)*:*

$$distract(\alpha_1, \alpha_2) \textbf{ causes } \neg attentive(\alpha_2)$$

The perspective axioms (4.33) *and* (4.34)*:*

$$\{\alpha_1, \alpha_2\} \textbf{ observes } distract(\alpha_1, \alpha_2)$$

$$\{\alpha\} \textbf{ observes } distract(\alpha_1, \alpha_2) \textbf{ if } attentive(\alpha)$$

together with the fact that the agents are initially attentive, give $f(\sigma_0, distract(A, C)) = \{A, B, C\}$ *and* $o(\sigma_0, distract(A, C)) = \emptyset$ *as the agents' frames of reference. The pointed frame describing the successor state resulting from the occurrence of the action distract*(A, C) *is given by* $\mathbf{Eu}(\sigma_0, \mathcal{E}_o(\sigma, distract(A, C))))$*, and is shown in Figure 4.5.* ◇

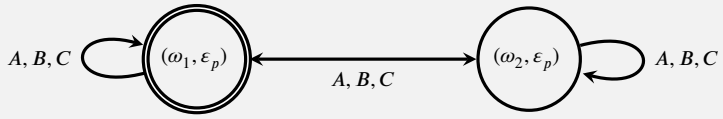

Figure 4.5: Pointed frame resulting from applying the epistemic update.

4.2.2.2 The Ontic Update

The epistemic update only describes how an agent reasons about how his actions are perceived by his fellows. In order to obtain the full successor state, he must then reason about how his actions may actually play out. This is accomplished by abstracting away the presence of other agents, turning the problem into one concerning the effects of an action in a

single-agent domain. This is done by applying what we term an *ontic update* operation to the pointed frame. Prior to defining the ontic update, we must first describe how to relate pointed frame to the framework for reasoning about the effects of an action from the perspective of $\mathcal{AL}$.

Definition 4.14 ($\mathcal{AL}$ State)**.** Let Δ be an action description of $m\mathcal{AL}$ and σ be a state of the transition diagram defined by Δ. Each point, ω of σ corresponds to a complete, consistent set of fluent literals, $\mathcal{AL}(\sigma, \omega)$, defined as follows:

$$\{f \mid \sigma.\pi(\omega)(f) = \top\} \cup \{\neg f \mid \sigma.\pi(\omega)(f) = \bot\}$$

$\diamond$

Intuitively, a pointed frame describes the basic structure of the successor state. Let (ω, ε) be a point in a pointed frame obtained by the application of the epistemic update. Such a point is interpreted in a similar fashion to to an atom of the form $do(\varepsilon, \omega)$ in the situation calculus[1] [39] (i.e., ε occurs in the possible world represented by the point ω). Using the originating state, σ and the pointed frame, we obtain the corresponding expansions of the point ω by applying the McCain-Turner equation [34] to the possible worlds defined by $\mathcal{AL}(\sigma, \omega)$.

Definition 4.15 (Scenario Expansion)**.** Let $\sigma = (M, \omega)$ be a state of the transition diagram defined by Δ; $\mathcal{U} = (\mathcal{E}, \varepsilon)$ be a pointed event model corresponding to the occurrence of an action, a, in σ; $\mathcal{S} = \mathbf{Eu}(\sigma, \mathcal{U})$ be the pointed frame describing the structure of the successor state; and $w = (\omega, \varepsilon)$ be a point in $\mathcal{S}$. The *expansion of the point w consistent with σ*, (denoted by $C(\sigma, w)$), is defined as follows:

- if $\varepsilon = \varepsilon_i$, then $C(\sigma, w) = \{\mathcal{AL}(\sigma, \omega)\}$

[1] With some admitted abuse of notation.

- $C(\sigma, w) = \{\tau(w) \mid \tau(w) = \mathbf{Cn}_\Delta(E(\mathcal{AL}(\sigma, \omega), a) \cup (\mathcal{AL}(\sigma, \omega) \cap \tau(w)))\}$ otherwise

$\diamond$

It should be of no surprise that Equation (4.15) is the McCain-Turner equation, which we first saw in Section 2.1.2. As before, the arguments to $\mathbf{Cn}_\Delta$ combine the *direct effects* of the action, $E(\mathcal{AL}(\sigma, \omega), a)$, with those properties of the domain which carry over due to *inertia*, $(\mathcal{AL}(\sigma, \omega) \cap \tau(w))$. The consequence operator, $\mathbf{Cn}_\Delta$, extrapolates the *indirect effects* of the action in accordance with the state constraints of Δ.

Having defined the basic framework, we may now define the *ontic update* operation.

Definition 4.16 (Ontic Update)**.** Let Δ be an action description of $m\mathcal{AL}$, σ be a state of the transition diagram defined by Δ, and $\mathcal{S} = (F, \omega)$ be the pointed frame describing the structure of the successor state. The *ontic update*, $\mathbf{Ou}_\Delta(\sigma, \mathcal{S})$ defines a set of Kripke worlds (M', Γ) where:

- $M'.W$ is the set of new points of the form $\omega_{\tau_i(w)}$ for each $\tau_i(w) \in C(\sigma, \mathcal{S})$
- $M'.\pi(\omega_{\tau_i(w)})(f) = \top$ if $f \in \tau_i(w)$
- $M'.\pi(\omega_{\tau_i(w)})(f) = \bot$ if $\neg f \in \tau_i(w)$
- $M'.R_\alpha = \{(\omega_{\tau_i(w_1)}, \omega_{\tau_j(w_2)}) \mid \omega_{\tau_i(w_1)}, \omega_{\tau_j(w_2)} \in M'.W,$ and $(w_1, w_2) \in \mathcal{S}.R_\alpha\}$
- $\Gamma = \{\omega_{\tau_i(w)} \mid \tau_i(w) \in C(\sigma, \omega)\}$

$\diamond$

Example 4.17 (Applying the Ontic Update)**.** *Let $\mathcal{S}_0$ denote the pointed frame from Example 4.13. Application of the ontic update, $\mathbf{Ou}_\Delta(\sigma_0, \mathcal{S}_0)$, gives the set of Kripke world depicted in Figure 4.6. The pointed Kripke structure, M_1, shown in Figure 4.6 consists of two points, $\omega_3 = \omega_{\tau_1}(\omega_1, \varepsilon_p)$, and $\omega_4 = \omega_{\tau_1}(\omega_2, \varepsilon_p)$, where:*

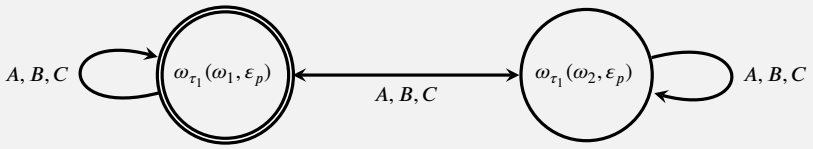

Figure 4.6: Successor state obtained by applying the ontic update.

- $M_1.\pi(\omega_3) = \{heads, locked, \neg open(l_1), \neg open(l_2), attentive(A), attentive(B), \neg attentive(C)\}$
- $M_1.\pi(\omega_4) = \{\neg heads, locked, \neg open(l_1), \neg open(l_2), attentive(A), attentive(B), \neg attentive(C)\}$

$\diamond$

4.2.2.3 The Full Transition Function

As was mentioned previously, the transition function is based on the following intuition: an agent first reasons about how his action is perceived, and then reasons about how it may actually play out. This intuition is realized in the definition of our transition function, $\Phi_\Delta(\sigma, a)$.

Definition 4.18 (The Transition Function)**.** Let Δ be an action description of $m\mathcal{AL}$, σ be a state of the transition diagram defined by Δ, and a be an action. The successor state(s) obtained by performing the action a in the state σ are defined as follows:

$$\Phi_\Delta(\sigma, a) = \begin{cases} \mathbf{Ou}_\Delta(\sigma, \mathbf{Eu}(\sigma, \mathcal{E}_o(\sigma, a))) & \text{ontic action} \\ \mathbf{Ou}_\Delta(\sigma, \mathbf{Eu}(\sigma, \mathcal{E}_s(\sigma, a))) & \text{sensing action} \\ \mathbf{Ou}_\Delta(\sigma, \mathbf{Eu}(\sigma, \mathcal{E}_c(\sigma, a))) & \text{otherwise} \end{cases}$$

$\diamond$

As with $m\mathcal{A}+$, it can be shown that the semantics has a number of desirable properties, among which are the following:

Theorem 4.1. *Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an ontic action that is executable in σ. Let $(M', \omega') \in \Phi_\Delta(\sigma, a)$. It holds that:*

- $\forall \alpha \in f(a, \sigma)$*, and dynamic causal law* $[a$ ***causes*** λ ***if*** $\phi] \in \Delta$*, if* $(M, \omega) \models \mathcal{B}_\alpha \phi$*, then* $(M', \omega') \models \mathcal{B}_\alpha \lambda$
- $\forall \alpha \in f(a, \sigma)$*, and state constraint* $[\lambda$ ***if*** $\phi] \in \Delta$*,* $(M, \omega) \models \mathcal{B}_\alpha(\phi \implies \lambda)$
- $\forall \alpha \in o(a, \sigma)$*,* $\forall \lambda$*,* $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *if and only if* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Theorem 4.1 in essence states that the direct effects of an ontic action are a commonly held belief among those agents who have first hand knowledge of the action occurrence. In addition, it tells us that every state of the transition diagram satisfies the state constraints of the program. Lastly, it states that the beliefs of oblivious agents are unchanged from one state to the next.

Theorem 4.2. *Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be a sensing action that is executable in σ and described by the sensing axiom* $[a$ ***determines*** $f] \in \Delta$*. Let* $(M', \omega') \in \Phi_\Delta(\sigma, a)$*. It holds that:*

- $(M', \omega') \models \mathcal{C}_{f(a,\sigma)} \lambda$ *if and only if* $(M, \omega) \models \lambda$ *where* $\lambda \in \{f, \neg f\}$
- $(M', \omega') \models \mathcal{C}_{p(a,\sigma)}(\mathcal{C}_{f(a,\sigma)} f \vee \mathcal{C}_{f(a,\sigma)} \neg f)$
- $\forall \alpha \in o(a, \sigma)$*,* $\forall \lambda$*,* $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *if and only if* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Theorem 4.2 informs us that the value of the fluent f is revealed to all of the agents who are participants in the occurrence of the sensing action related to f. Furthermore, it states that those agents who merely observe the action occurrence learn that the value of the fluent has been revealed. Lastly, it too states that the beliefs of oblivious remain unchanged due to inertia.

Theorem 4.3. *Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be a communication action that is executable in σ and described by the sensing axiom* $[a$ ***communicates*** $\varphi] \in \Delta$. *Let $(M', \omega') \in \Phi_{\Delta}(\sigma, a)$. It holds that:*

- $(M', \omega') \models C_{f(a,\sigma)}\varphi$
- $(M', \omega') \models C_{p(a,\sigma)}(C_{f(a,\sigma)}\varphi \vee C_{f(a,\sigma)}\neg\varphi)$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \models \mathcal{B}_{\alpha}\lambda$ *if and only if* $(M, \omega) \models \mathcal{B}_{\alpha}\lambda$

Finally, Theorem 4.3 captures the same intuition as Theorem 4.2, except that it now applies to the modal formula being announced, as opposed to the value of a single fluent.

4.3 Representing Dynamic Domains with $m\mathcal{AL}$

Having defined the syntax and semantics of $m\mathcal{AL}$ in the context of a multi-agent variation of the Lin's Briefcase Domain [12, 32], we now shift our focus to presenting an in-depth example of its application to modeling a multi-agent variation of the Burner Ignition Domain of [24].

4.3.1 The Burner Ignition Domain

Two students, A and B, are in a chemistry class and must use a gas powered burner in their experiment. The burner is connected to a gas tank by a pipeline. The tank is on the left-most end of the pipeline, while the burner is on the right-most end. The pipeline is comprised of three sections connected together by valves as shown in Figure 4.7. Each section can be pressurized by the tank or depressurized. Opening a valve causes the section to its right to be pressurized if the section on its left is as well. Moreover, for safety reasons, a valve can only be opened if the next valve in the pipeline is closed. Closing a valve causes the pipe section on its right to become depressurized. The burner may be ignited (and therefore lit and usable) if the entire pipeline is pressurized. Furthermore, A and B have different personalities, and while A is studious and attentive, B is easily distracted. For safety reasons, A and B decide to not touch the valves unless both of them are paying attention, and may signal each other to ensure that they're on task.

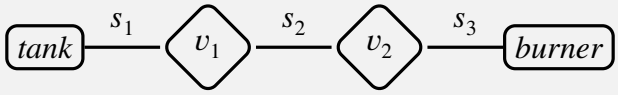

Figure 4.7: Pipeline configuration in the Burner Ignition Domain.

Example 4.19 (Representing the Burner Ignition Domain)**.** *We begin our representation as before by presenting the domain signature, as well as the initial state description. A close reading of the problem description suggests the following:*

$$\mathcal{AG} = \{A, B\}$$

$$\mathcal{F} = \{lit, open(v), attentive(\alpha), pressurized(\sigma)\}$$

$$\mathcal{A} = \{signal(\alpha_1, \alpha_2), open(\alpha, v), close(\alpha, v), ignite(\alpha)\}$$

where α is a variable over $\mathcal{AG}$; v is a variable over $\{v_1, v_2\}$ representing the individual valves; and σ is a variable over $\{s_1, s_2\}$ representing the sections of the pipeline. Atoms of the form attentive(α) have the same reading as before, while the fluents which are specific to this domain have capture the following intuitions:

- *lit - denotes whether or not the burner is lit.*
- *open(v) - denotes whether or not a particular valve is in the open position.*
- *pressurized(σ) - denotes whether or not a particular section of the pipeline has been pressurized.*

In addition to the object constants representing the valves and sections of the pipeline, we also introduce a number of auxiliary relations to help define the structure of the pipeline. It should be noted that these relations do not represent fluents *(as the structure of the pipeline is static in this domain). For this example, we represent the pipeline configuration from Figure 4.7 with the following set of facts:*

$$\{connected(t, s_1), connected(s_1, v_1, s_2), connected(s_2, v_2, s_3), connected(s_3, b)\}$$

where t and b denote the tank and burner respectively.

With the signature firmly in place, we can now present the initial state axioms. Let us suppose that the initial state of the domain is by the following commonly held beliefs/knowledge between *A* and *B:*

- *Both students are behaving in accordance with their personalities initially (i.e. A is paying attention, while B is not).*
- *The burner is initially off.*
- *None of the pipeline sections are pressurized.*

- *All of the valves are closed.*

and are represented by the following initial state axioms*:*

$$\textbf{\textit{initially}}\ C\mathcal{K}_A attentive(A) \tag{4.35}$$

$$\textbf{\textit{initially}}\ C\mathcal{K}_A \neg attentive(B) \tag{4.36}$$

$$\textbf{\textit{initially}}\ C\neg lit \tag{4.37}$$

$$\textbf{\textit{initially}}\ C\neg pressurized(s_1) \tag{4.38}$$

$$\textbf{\textit{initially}}\ C\neg pressurized(s_2) \tag{4.39}$$

$$\textbf{\textit{initially}}\ C\neg pressurized(s_3) \tag{4.40}$$

$$\textbf{\textit{initially}}\ C\neg open(v_1) \tag{4.41}$$

$$\textbf{\textit{initially}}\ C\neg open(v_2) \tag{4.42}$$

For simplicity of the presentation a number of additional axioms are omitted, such as those detailing the actual structure of the pipeline, just as we omitted the specification of the auxiliary relations and object constants from the domain signature.

Taking a look at the domain signature we can see the following actions: signal(α_1, α_2)*, open*(α, v)*, close*(α, v)*, and ignite. All of these are* ontic actions*, but what distinguishes them in character from the kinds of ontic actions presented in Chapter 3 is that most of them have indirect effects that are not limited to the epistemic properties of the domain. We begin our presentation with the axioms defining the behavior of the action signal:*

$$signal(\alpha_1, \alpha_2)\ \textbf{\textit{causes}}\ attentive(\alpha_2) \tag{4.43}$$

$$\{\alpha_1, \alpha_2\}\ \textbf{\textit{observes}}\ signal(\alpha_1, \alpha_2) \tag{4.44}$$

$$\{\alpha\}\ \textbf{\textit{observes}}\ signal(\alpha_1, \alpha_2)\ \textbf{\textit{if}}\ attentive(\alpha) \tag{4.45}$$

As was the case with the other multi-agent domains presented in this text, the action signal is of interest due to its ability to alter the perspectives of the agents for any subsequent actions.

The actions open and close are a symmetric pair of actions which enable an agent to manipulate valves along the pipeline. The direct effects of the action open are fairly straightforward: opening a valve causes it to become open. *Similarly, for the action close.*

$$open(\alpha, \upsilon) \textbf{ causes } open(\upsilon) \tag{4.46}$$

$$\alpha \textbf{ observes } open(\alpha, \upsilon) \tag{4.47}$$

$$\alpha_2 \textbf{ observes } open(\alpha_1, \upsilon) \textbf{ if } attentive(\alpha_2) \tag{4.48}$$

$$close(\alpha, \upsilon) \textbf{ causes } \neg open(\upsilon) \tag{4.49}$$

$$\alpha \textbf{ observes } close(\alpha, \upsilon) \tag{4.50}$$

$$\alpha_2 \textbf{ observes } close(\alpha_1, \upsilon) \textbf{ if } attentive(\alpha_2) \tag{4.51}$$

In addition to the standard epistemic indirect effects, opening a valve may also indirectly cause the relevant adjacent sections to become pressurized. We also take as a given that the section of the pipeline directly connected to the tank is always pressurized:

$$pressurized(\sigma) \textbf{ if } connected(t, \sigma) \tag{4.52}$$

$$pressurized(\sigma_2) \textbf{ if } open(\upsilon) \wedge connected(\sigma_1, \upsilon, \sigma_2) \wedge pressurized(\sigma_1) \tag{4.53}$$

Lastly, with regards to open and close, one cannot open a valve that's already been opened, and one cannot close a valve if it has already been closed. Furthermore, for safety reasons, a valve may only be opened if the next valve in the sequence is closed.

$$\textbf{impossible } open(\alpha, \upsilon) \textbf{ if } open(\upsilon) \tag{4.54}$$

$$\textbf{impossible } close(\alpha, \upsilon) \textbf{ if } \neg open(\upsilon) \tag{4.55}$$

$$\textbf{impossible } open(\alpha, \upsilon_1) \textbf{ if } connected(\sigma_1, \upsilon_1, \sigma_2) \wedge connected(\sigma_2, \upsilon_2, \sigma_3) \wedge open(\upsilon_2) \tag{4.56}$$

Furthermore, the safety protocol that **A** *and* **B** *have agreed upon says that the valves are not to be touch if either one of them is not paying attention. This is encoded by the following*

executability conditions:

$$\textbf{\textit{impossible}}\ open(\alpha, v)\ \textbf{\textit{if}}\ \neg attentive(\alpha) \tag{4.57}$$

$$\textbf{\textit{impossible}}\ open(\alpha_1, v)\ \textbf{\textit{if}}\ \neg attentive(\alpha_2) \tag{4.58}$$

$$\textbf{\textit{impossible}}\ close(\alpha, v)\ \textbf{\textit{if}}\ \neg attentive(\alpha) \tag{4.59}$$

$$\textbf{\textit{impossible}}\ close(\alpha_1, v)\ \textbf{\textit{if}}\ \neg attentive(\alpha_2) \tag{4.60}$$

The final action to define is ignite. For safety reasons, the burner may only be ignited if the section of pipeline connected to the burner is pressurized.

$$ignite(\alpha)\ \textbf{\textit{causes}}\ lit \tag{4.61}$$

$$\textbf{\textit{impossible}}\ ignite(\alpha)\ \textbf{\textit{if}}\ \neg pressurized(\sigma) \wedge connected(\sigma, b) \tag{4.62}$$

$$\neg lit\ \textbf{\textit{if}}\ \neg pressurized(\sigma) \wedge connected(\sigma, b) \tag{4.63}$$

As before, we also have the executability conditions corresponding to the safety protocol:

$$\textbf{\textit{impossible}}\ ignite(\alpha)\ \textbf{\textit{if}}\ \neg attentive(\alpha) \tag{4.64}$$

$$\textbf{\textit{impossible}}\ ignite(\alpha_1)\ \textbf{\textit{if}}\ \neg attentive(\alpha_2) \tag{4.65}$$

◇

Example 4.20 (Burner Ignition Domain — Lighting the Burner)**.** *Suppose that the initial state of the domain is the one given by axioms* (4.35) *through* (4.42)*. How might A and B work together to light the burner while following the necessary precautions? A careful reading of axioms* (4.43) *through* (4.65) *suggests the following sequence of actions:*

$$signal(A, B), open(A, v_1), open(B, v_2), ignite(A)$$

The first action, makes it possible for both agents to follow the protocol by making B now focus on his task. Once both agents are attentive, the valves may be opened in sequence, and the burner may be ignited. ◇

Chapter 5

MULTI-AGENT REASONING VIA ASP

In this chapter we explore how answer set programming [21, 24] may be applied to automate various reasoning tasks within a multi-agent context. We begin exploring a representation of the Classical Muddy Children Problem, followed by a discussion of the answer-set prolog representation of the semantics of $m\mathcal{A}+$. We then conclude by presenting solutions to the *temporal projection* and *planning* problems in the context of $m\mathcal{A}+$.

5.1 The Classical Muddy Children Problem in ASP

In this section we take an in depth look at the use of answer set programming [21, 24] to represent an instance of the Classical Muddy Children Problem from [19]. This work was originally presented in less detail in [10]. For a detailed presentation of the domain itself we refer the reader to either Chapters 1.1.1 and 3.3.1, or [19].

5.1.1 Representing the Domain Signature

The definition of the domain's signature deviates slightly from that presented in Example 3.16:

$$\mathcal{AG} = \{A, B, C\}$$

$$\mathcal{F} = \{m_A, m_B, m_C\}$$

$$\mathcal{A} = \{declare(\varphi)\}$$

```
% A, B, and C are agents.
agent(a). agent(b). agent(c).

% if AG is an agent then m(AG) is a fluent corresponding to m_AG
fluent(m(AG)) :- agent(AG).

% declare(m_A ∨ m_B ∨ m_C) is an action
action(declare(or(m(a),m(b),m(c)))).

% if AG is an agent then declare(K_AG(m_AG)) is an action
action(declare(neg(k(AG,m(AG))))) :- agent(AG).

% if AG is an agent then declare(¬K_AG(¬m_AG)) is an action
action(declare(neg(k(AG,neg(m(AG)))))) :- agent(AG).

% ⊤ and ⊥ are defined as primitive values
value(top). value(bot).
```

Listing 5.1: ASP representation of the domain signature for the Muddy Children Domain.

where a fluent of the form m_α is read as "agent α is muddy", and φ is a modal formula[1]. This is represented by the collection of rules in Listing 5.1.

5.1.2 Representing Modal Formulae

The solution to the Classical Muddy Children Problems involves the use of modal logic. Definition 2.9 is recursive however, and not all of the possible formulae that can be defined over the signature specified in Section 5.1.1 are relevant, so we define only a subset of them in our representation. In particular, we need to define the formulae necessary to capture the public announcements made by both the father and the children, as well as those needed to formulate what is initially commonly known by the children. This is accomplished through

[1]The modal formulae allowed are limited to those pertaining to the domain directly.

the definitions of the relations `literal/1` and `formula/1`.

We begin by defining the notion of a fluent literal, which is a fluent or its negation. The fluents themselves are defined as part of the domain signature, however, the notion of a fluent literal is a syntactic construct that we use to construct modal formulae, so we treat them separately in our program. Literals are defined by the predicate `literal/1`, and atoms of the form `literal(F)` are read as: "F is a literal." With the notion of a fluent literal firmly in place, we proceed to define the modal formulae of interest in this domain. Formulae are defined by the predicate `formula/1`, and atoms of the form `formula(F)` are read as: "F is a formula." For our purposes, if F is fluent literal, then F is a formula, as is the disjunction of three fluent literals. Listing 5.2 shows how the definitions of these *fluent formulae* are realized in answer-set prolog.

If A is an agent and F is a fluent literal, then `k(A,F)` and `neg(k(A,F))` are formulae corresponding to modal formulae of the form $\mathcal{K}_A F$ and $\neg\mathcal{K}_A F$. The disjunction of such formulae is also considered to be a formula. In addition, we can apply the operator C to any of the previously defined formulae and still have a modal formula. Listing 5.3 shows the corresponding answer-set prolog definitions.

5.1.2.1 Representing States of the Domain

Recall that in our model of a dynamic multi-agent domain, states are understood as Kripke worlds. As detailed in Definition 2.10, a Kripke world is a pair, (M, ω) where M is a Kripke model and ω is a particular point in M describing the actual physical configuration of the domain. Kripke worlds themselves are complex, record-like structures, and we represent them as a collection of atoms for each field of the record:

- `step/1` - Kripke worlds are named in accordance with their step/position along a

```
% A literal is defined as a fluent F or its negation ¬F.
literal(F) :- fluent(F).

literal(neg(F)) :- fluent(F).

% The fluent literals F and ¬F are complementary.
complementary(F, neg(F)) :- fluent(F).

complementary(neg(F), F) :- fluent(F).

% If F is fluent literal then F is a formula.
formula(F) :- literal(F).

% The disjunction of three distinct, non-complementary fluent literals is a
% formula.
formula(or(F1,F2,F3)) :-
    literal(F1), literal(F2), literal(F3),
    F1 != F2, F2 != F3, F1 != F3,
    not complementary(F1, F2),
    not complementary(F2, F3),
    not complementary(F1, F3).
```

Listing 5.2: ASP definition of fluent formulae in the Muddy Children Domain.

trajectory in the domain we are modeling. Steps are enumerated starting from 0, and atoms of the form `step(X)` are read as: "X is a step/state of the domain."

- `point/2` - defines the set of points for a given Kripke world. Atoms of the form `point(P,T)` are read as: "P is a point in the Kripke world defined for step T."

- `value_of/4` - defines the interpretation functions for a given Kripke world. Atoms of the form `value_of(F,V,P,T)` are read as: "fluent F has the value V in the point P of the Kripke world for step T." Another way of reading atoms of this form using the notation given in Chapter 2.2.1.2 is: $T.\pi(P)(F) = V$.

- `k_reachable/4` - defines the accessibility relations for the agents in a given Kripke

```
% If A is an agent and F is a literal, then K_A F is a formula.
formula(k(A,F)) :- agent(A), literal(F).

% If A is an agent and F is a literal, then ¬K_A F is a formula.
formula(neg(k(A,F))) :- agent(A), literal(F).

% K_{A_1} F_1 ∨ K_{A_2} F_2 is also a formula if the disjuncts are distinct.
formula(or(k(A1,F1),k(A2,F2))) :-
    formula(k(A1,F1)), formula(k(A2,F2)),
    k(A1,F1) < k(A2,F2).

% If K_A F is a formula then C(K_A F) is also a formula.
formula(c(k(A,F))) :- formula(k(A,F)).

% If ¬K_A F is a formula then C(¬K_A F) is also a formula.
formula(c(neg(k(A,F)))) :- formula(neg(k(A,F))).

% If φ is a defined disjunctive formula then Cφ is one as well.
formula(c(or(F1,F2,F3))) :- formula(or(F1,F2,F3)).

formula(c(or(k(A1,F1),k(A2,F2)))) :- formula(or(k(A1,F1),k(A2,F2))).
```

Listing 5.3: ASP definition of modal formulae in the Muddy Children Domain.

world. Atoms of the form `k_reachable(P1,P2,A,T)` are read as: "the pair (ω_1, ω_2) belongs to the accessibility relation for agent A in the Kripke world defined at step T."

- `reference_point/2` - defines the reference point for a given Kripke world. Atoms of the form `reference_point(P,T)` are read as: "point P is the reference point associated with the Kripke world for step T."

5.1.2.2 Representing the Entailment Relation

Now that we have a basic notation for both modal formulae and the Kripke worlds defining the states of our domain, we can focus on representing the entailment relation between them

```
% If F is a fluent then (T, P) ⊨ F if and only if T.P(F) = ⊤
entailed_by(F, P, T) :- value_of(F, top, P, T),
    fluent(F), point(P, T).

% If F is a fluent then (T, P) ⊨ ¬F if and only if T.P(F) = ⊥
entailed_by(neg(F), P, T) :- value_of(F, bot, P, T),
    fluent(F), point(P, T).

% (T, P) ⊨ F1 ∨ F2 if and only if (T, P) ⊨ F1 or (T, P) ⊨ F2
entailed_by(or(F1,F2), P, T) :- entailed_by(F1, P, T),
    formula(or(F1,F2)), point(P, T).

entailed_by(or(F1,F2), P, T) :- entailed_by(F2, P, T),
    formula(or(F1,F2)), point(P, T).

% (T, P) ⊨ F1 ∨ F2 ∨ F3 if and only if (T, P) ⊨ F1 or (T, P) ⊨ F2 or (T, P) ⊨ F3
entailed_by(or(F1,F2,F3), P, T) :- entailed_by(F1, P, T),
    formula(or(F1,F2,F3)), point(P, T).

entailed_by(or(F1,F2,F3), P, T) :- entailed_by(F2, P, T),
    formula(or(F1,F2,F3)), point(P, T).

entailed_by(or(F1,F2,F3), P, T) :- entailed_by(F3, P, T),
    formula(or(F1,F2,F3)), point(P, T).
```

Listing 5.4: ASP definition of the entailment relation for fluent formulae in the Muddy Children Domain.

in answer-set prolog. This relation is described by atoms of the form `entailed_by(F,P,T)` which are read as $(T, P) \models F$. As was the case in defining the syntax of modal formulae, it is important to notice that the representation of the entailment relation follows Definition 2.12, and is restricted to those formulae of interest in the domain of discourse. Listing 5.4 shows the definition of the relation `entailed_by/3` for fluent formulae.

Having represented the entailment relation for the fluent formulae of interest in this particular domain, we turn our attention towards entailment relation for the modal formulae involving the modal operators $\mathcal{K}$ and C. It is important to note that here we deviate from

Definition 2.12, and instead make use of the property of G-reachability outlined in [19]. Listing 5.5 gives continues the definition of the relation `entailed_by/3` for such formulae.

5.1.3 Representing the Initial State

As has been mentioned in previously, we view a multi-agent domain as a transition system whose nodes correspond to states of the domain (modeled as Kripke worlds) and whose arcs are labeled by actions. The trajectory is comprised of several states (or steps), which we simply enumerate from $0 \ldots 3$. This is done in a straightforward fashion by the set of facts: `step(0..3)`. These steps are then associated as names for the individual Kripke worlds defining the states of our transition system.

The names of the points from which we construct the Kripke worlds at each step of the trajectory are all taken from the same set, which is defined by the relation `symbol/1`. Facts of the form `symbol(X)` are read as "X is a symbol." The set of points in the initial state of the Classical Muddy Children is comprised of all eight state symbols. The remaining components of the initial state are defined explicitly (with the exception of the accessibility relations) using the vocabulary established in the previous sections.

The accessibility relations of our Kripke worlds on the other hand are generated by a choice rule, along with rules which describe the properties that the accessibility relations must have in order to satisfy the **S5** axioms (i.e., they must be symmetric, reflexive and transitive). Listing 5.6 presents the bulk of the answer-set prolog representation of the initial state, including the generation of candidate accessibility relations.

Up until this point, we have only specified some general properties of the initial Kripke world. What's left for us to specify is the formula that must be entailed by it in order for it to correctly represent the initial state of this instance of Classical Muddy Children Problem:

```
% (T, P_1) ⊨ K_A F if and only if (T, P_2) ⊨ F for all P_2 such that (P_1, P_2) ∈ T.R_A
entailed_by(k(A,F), P, T) :- entailed_by_all_k_reachable(F, A, P, T),
    formula(k(A,F)), point(P, T), agent(A).

% (T, P) ⊨ ¬K_A F if and only if (T, P) ⊭ K_A F
entailed_by(neg(k(A,F)), P, T) :- not entailed_by(k(A,F), P, T),
    formula(neg(k(A,F))), point(P, T), agent(A).

entailed_by_all_k_reachable(F, A, P, T) :-
    not -entailed_by_all_k_reachable(F, A, P, T),
    formula(F), point(P, T), agent(A).

-entailed_by_all_k_reachable(F, A, P1, T) :-
    k_reachable(P1, P2, A, T), not entailed_by(F, P2, T),
    formula(F), point(P1, T), point(P2, T), agent(A).

% (T, P_1) ⊨ C F if and only if for all P_2 such that P_2 is g-reachable from
% P_1, (T, P_2) ⊨ F.
entailed_by(c(F), P, T) :- entailed_by_all_g_reachable(F, P, T),
    formula(c(F)), point(P, T).

entailed_by_all_g_reachable(F, P, T) :-
    not -entailed_by_all_g_reachable(F, P, T),
    formula(F), point(P, T).

-entailed_by_all_g_reachable(F, P1, T) :-
    g_reachable(P1, P2, T), not entailed_by(F, P2, T),
    formula(F), point(P1, T), point(P2, T).

g_reachable(P1, P2, T) :- k_reachable(P1, P2, A, T),
    point(P1, T), point(P2, T), agent(A).

g_reachable(P1, P2, T) :- g_reachable(P1, P3, T), g_reachable(P3, P2, T),
    point(P1, T), point(P2, T), point(P3, T).
```

Listing 5.5: ASP definition of the entailment relation for modal formulae in the Muddy Children Domain.

```
symbol(w1). symbol(w2). symbol(w3). symbol(w4).
symbol(w5). symbol(w6). symbol(w7). symbol(w8).

point(P, 0) :- symbol(P).

reference_point(w1, 0).

value_of(m(a), top, w1, 0). value_of(m(b), top, w1, 0).
value_of(m(c), top, w1, 0). value_of(m(a), top, w2, 0).
value_of(m(b), top, w2, 0). value_of(m(c), bot, w2, 0).
value_of(m(a), top, w3, 0). value_of(m(b), bot, w3, 0).
value_of(m(c), top, w3, 0). value_of(m(a), top, w4, 0).
value_of(m(b), bot, w4, 0). value_of(m(c), bot, w4, 0).
value_of(m(a), bot, w5, 0). value_of(m(b), top, w5, 0).
value_of(m(c), top, w5, 0). value_of(m(a), bot, w6, 0).
value_of(m(b), top, w6, 0). value_of(m(c), bot, w6, 0).
value_of(m(a), bot, w7, 0). value_of(m(b), bot, w7, 0).
value_of(m(c), top, w7, 0). value_of(m(a), bot, w8, 0).
value_of(m(b), bot, w8, 0). value_of(m(c), bot, w8, 0).

{k_reachable(P1, P2, A, 0) : point(P1, 0), point(P2, 0)} :- agent(A).

k_reachable(P1, P2, A, 0) :- k_reachable(P2, P1, A, 0),
    point(P1, 0), point(P2, 0), agent(A).

k_reachable(P1, P2, A, 0) :-
    k_reachable(P1, P3, A, 0), k_reachable(P3, P2, A, 0),
    point(P1, 0), point(P2, 0), point(P3, 0), agent(A).

k_reachable(P, P, A, 0) :- point(P, 0), agent(A).
```

Listing 5.6: ASP definition of the initial state in the Muddy Children Domain.

namely that it is common knowledge amongst the children that none of them knows whether or not they are muddy, and that each of them knows whether or not their fellows are.

We begin by defining the relation `holds/2` which we use to specify the formula that hold in a particular state of the domain. Atoms of the form `holds(F,T)` are read: "formula F holds in the step/state T." On top of the relation `holds/2`, we use the relation `initially/1` to represent the initial state axioms defining the formulae which must be satisfied by the initial state. Atoms of the form `initially(F)` are taken to mean "formula F is initially true." With these relations in place, we constrain our allowed accessibility relations to those which ensure that our initial state entails the correct formulae. The final part of our initial state description is given in Listing 5.7.

Before continuing the with the answer-set prolog representation of the Classical Muddy Children Domain, it bears repeating that at no point have the accessibility relations for the agents been explicitly specified. Instead, they are derived programmatically using a *generate and test* strategy. The rules in Listing 5.6 simply specify the *general properties* that the accessibility relations must have under the **S5** axiom system, while those in Listing 5.7 limit the relations themselves to those which make the resulting Kripke world entail the set of formulae specified by the relation `initially/1`. Using an answer set solver such as `clingo` [21], we may compute the relevant accessibility relations (and thereby the minimal model of the modal theory specified by the initial state axioms), as shown in Example 5.1. This basic principle has been expanded upon in [9] to show the broader applicability of logic programming to *finding the models of various modal theories*.

Example 5.1 (Generating the Initial State of the Domain)**.** *Let us suppose that program Listings 5.1 through 5.7 have been collected into the following logic programming modules:* `domain-signature.lp`*;* `del-definitions.lp`*; and* `initial-state.lp`*. Using the answer-set solver* `clingo`*, (and restricting the output to only those atoms formed by*

```
holds(F, T) :- entailed_by(F, P, T), reference_point(P, T),
    formula(F), point(P, T), step(T).

% It is common knowledge among the children that each of them knows
% whether or not their fellows are muddy
initially(c(or(k(a,m(b)),k(a,neg(m(b)))))).
initially(c(or(k(a,m(c)),k(a,neg(m(c)))))).
initially(c(or(k(b,m(a)),k(b,neg(m(a)))))).
initially(c(or(k(b,m(c)),k(b,neg(m(c)))))).
initially(c(or(k(c,m(a)),k(c,neg(m(a)))))).
initially(c(or(k(c,m(b)),k(c,neg(m(b)))))).

% It is also common knowledge among the children that none of them know
% whether or not they themselves are muddy
initially(c(neg(k(a,m(a))))).
initially(c(neg(k(a,neg(m(a)))))).
initially(c(neg(k(b,m(b))))).
initially(c(neg(k(b,neg(m(b)))))).
initially(c(neg(k(c,m(c))))).
initially(c(neg(k(c,neg(m(c)))))).

:- initially(F), not holds(F, 0).
```

Listing 5.7: ASP definition of the relations `holds/2` and initial state axioms in the Muddy Children Domain.

the predicate symbol `k_reachable/4`*), we can obtain the accessibility relations shown in Listing 5.8*[2]*. Rendering the results graphically gives us precisely the Kripke world shown in Figures 1.1 and 3.5.*

◇

[2]The output of `clingo` has been formatted for presentation purposes.

```
$ clingo -n 0 domain-signature.lp initial-state.lp del-definitions.lp
k_reachable(w1,w1,a,0) k_reachable(w1,w1,b,0) k_reachable(w1,w1,c,0)
k_reachable(w2,w2,a,0) k_reachable(w2,w2,b,0) k_reachable(w2,w2,c,0)
k_reachable(w3,w3,a,0) k_reachable(w3,w3,b,0) k_reachable(w3,w3,c,0)
k_reachable(w4,w4,a,0) k_reachable(w4,w4,b,0) k_reachable(w4,w4,c,0)
k_reachable(w5,w5,a,0) k_reachable(w5,w5,b,0) k_reachable(w5,w5,c,0)
k_reachable(w6,w6,a,0) k_reachable(w6,w6,b,0) k_reachable(w6,w6,c,0)
k_reachable(w7,w7,a,0) k_reachable(w7,w7,b,0) k_reachable(w7,w7,c,0)
k_reachable(w8,w8,a,0) k_reachable(w8,w8,b,0) k_reachable(w8,w8,c,0)
k_reachable(w1,w2,c,0) k_reachable(w1,w3,b,0) k_reachable(w1,w5,a,0)
k_reachable(w2,w1,c,0) k_reachable(w2,w4,b,0) k_reachable(w2,w6,a,0)
k_reachable(w3,w1,b,0) k_reachable(w3,w4,c,0) k_reachable(w3,w7,a,0)
k_reachable(w4,w2,b,0) k_reachable(w4,w3,c,0) k_reachable(w4,w8,a,0)
k_reachable(w5,w1,a,0) k_reachable(w5,w6,c,0) k_reachable(w5,w7,b,0)
k_reachable(w6,w2,a,0) k_reachable(w6,w5,c,0) k_reachable(w6,w8,b,0)
k_reachable(w7,w3,a,0) k_reachable(w7,w5,b,0) k_reachable(w7,w8,c,0)
k_reachable(w8,w4,a,0) k_reachable(w8,w6,b,0) k_reachable(w8,w7,c,0)

SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.702s (Solving: 0.03s 1st Model: 0.00s Unsat: 0.03s)
CPU Time     : 0.621s
```

Listing 5.8: Accessibility relations derived for the initial state of the Classical Muddy Children Domain

5.1.4 Representing the Effects of Actions

Having specified the initial state of the trajectory, we turn our attention towards specifying the effects of the action *declare*(φ). As has been discussed previously, such actions are *public announcements*, and a close analysis of the update model semantics of $m\mathcal{A}+$ from Chapter 3.2 and [4, 43] shows that the effects of such actions are to remove those points points from the Kripke world which fail to satisfy φ, and to modify the agents' accessibility relations accordingly. All other points carry over to the successor state. This is shown in Listing 5.9,

```
-point(P, T) :-
    occurs(declare(F), T1), not entailed_by(F, P, T - 1),
    point(P, T - 1), action(declare(F)), step(T), step(T - 1).

point(P, T) :-
    point(P, T - 1), not -point(P, T),
    step(T), step(T - 1).

k_reachable(P1, P2, A, T) :-
    k_reachable(P1, P2, A, T - 1),
    point(P1, T), point(P1, T - 1),
    point(P2, T), point(P2, T - 1),
    agent(A), step(T), step(T - 1).

value_of(F, V, P, T) :-
    value_of(F, V, P, T - 1),
    point(P, T), point(P, T - 1),
    step(T), step(T - 1).

reference_point(P, T) :-
    reference_point(P, T - 1),
    point(P, T), point(P, T - 1),
    step(T), step(T - 1).
```

Listing 5.9: ASP definition of the effects of the action *declare*(φ) in the Muddy Children Domain.

and bears a close reading.

The first rule captures the idea that occurrences of the action *declare*(F) remove from the successor state those points which do not entail F. Next is a *default* which states that if P was a point in the state $T-1$, it will remain a point state T, unless otherwise specified. Finally, we define a quasi-inertial property on the part of our accessibility relations: namely that the accessibility relations pertaining to those points which remain in the successor Kripke world are unchanged. Similarly, their interpretations carry over into the successor state, as does the reference point.

```
occurs(declare(or(m(a),m(b),m(c))), 0).

occurs(declare(neg(k(A,m(A)))), 1) :- agent(A).

occurs(declare(neg(k(A,neg(m(A))))), 1) :- agent(A).

occurs(declare(neg(k(A,m(A)))), 2) :- agent(A).

occurs(declare(neg(k(A,neg(m(A))))), 2) :- agent(A).
```

Listing 5.10: ASP definition of the domain history in the Muddy Children Domain.

5.1.5 Representing the Trajectory

We conclude our representation by specifying the trajectory described in the Classical Muddy Children Problem. This is done through the relation `occurs/2`, where atoms of the form `occurs(A,S)` are read as: "action a occurs in the state S." According to the problem description, the first action which occurs is the father's declaration that at least one of the children is muddy. This is represented in our vocabulary by an occurrence of the action $declare(m_A \vee m_B \vee m_C)$ in the initial state.

The remaining actions that we must model are the children's repeated answers to their father's queries as to whether or not they know if they are muddy. The children answer in the negative twice following their father's initial declaration, which we specify (for simplicity) as occurrences of two parallel declarations: that they do not know that they are muddy; and that they do not know that they are not muddy. The entire trajectory is given by the domain history described in Listing 5.10.

Let Π_{mc} denote the logic program developed in this chapter. Π_{mc} has a number of interesting properties, one of which is described in Theorem 5.1, and presented in Example 5.2.

Theorem 5.1. *Let a_1, a_2, and a_3 be defined as follows:*

$$a_1 = declare(m_A \vee m_B \vee m_C)$$

$$a_2 = declare(\neg \mathcal{K}_\alpha m_\alpha \wedge \neg \mathcal{K}_\alpha \neg m_\alpha)$$

$$a_3 = declare(\neg \mathcal{K}_\alpha m_\alpha \wedge \neg \mathcal{K}_\alpha \neg m_\alpha)$$

Π_{mc} has a single answer set corresponding to the solution of the the Classical Muddy Children Problem for $N = K = 3$ described by the trajectory $(\sigma_0, a_1, \sigma_1)$, $(\sigma_1, a_2, \sigma_2)$, $(\sigma_2, a_3, \sigma_3)$.

Example 5.2 (Computing the Full Trajectory in the Muddy Children Domain)**.** *Let us suppose that program Listings 5.1 through 5.7 have been collected into the logic programming modules:* `domain-signature.lp`*;* `del-definitions.lp`*; and* `initial-state.lp`*. In addition, let Listings 5.9 and 5.10 be defined in the modules* `action-definitions.lp` *and* `domain-history.lp` *respectively. Using the answer-set solver* `clingo`*, (and restricting the output to only those atoms formed by the predicate symbol* `point/2`*), we can see the evolution of the set of points comprising the Kripke world which represents the state of the domain over time in Listing 5.11.*

◇

The presentation of Π_{mc} lays an important foundation for the subsequent section on *temporal projection* as a careful reader will notice that a very specific instance of the temporal projection problem dealing with the consequences of the action *declare* is dealt with in Π_{mc}. In the next section we will present a *generalization* of the ideas discussed here to provide an answer-set prolog representation of the semantics of $m\mathcal{A}+$ and show its application to the temporal projection problem.

```
$ clingo -n 0 domain-signature.lp initial-state.lp del-definitions.lp
         action-definitions.lp domain-history.lp
point(w1,0) point(w2,0) point(w3,0) point(w4,0)
point(w5,0) point(w6,0) point(w7,0) point(w8,0)

point(w7,1) point(w6,1) point(w5,1) point(w4,1)
point(w3,1) point(w2,1) point(w1,1)

point(w1,2) point(w2,2) point(w3,2) point(w5,2)

point(w1,3)

SATISFIABLE

Models       : 1
Calls        : 1
Time         : 3.752s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 3.645s
```

Listing 5.11: Derived trajectory in the Classical Muddy Children Domain.

5.2 An ASP Semantics for $m\mathcal{A}+$

The previous sections detailed a domain specific answer-set prolog representation in the context of the Classical Muddy Children Problem. In this section, we generalize the representation and present an answer-set prolog realization of the update model semantics of $m\mathcal{A}+$ as described in Chapter 3.2, from the perspective of automating the *temporal projection problem*. Briefly stated, temporal projection involves answering the following question: "Given a state σ and an action a, what is the successor state(s) obtained by performing a in σ?"

Recall from Definition 3.15, that the transition function of $m\mathcal{A}+$ is defined as follows:

$$\Phi_{\Delta}(\sigma, a) = \begin{cases} \sigma \otimes U_o(\sigma, a) & \text{ontic action} \\ \sigma \otimes U_s(\sigma, a) & \text{sensing action} \\ \sigma \otimes U_a(\sigma, a) & \text{otherwise} \end{cases}$$

where Δ is an action description, σ is a state of the domain, and a is an action. In the prior sections of this chapter, we have already presented a general notation for the notion of a *state of the domain* as described by a Kripke world, along with a translation of the *initial state axioms* into answer-set prolog. What remains is: the definition of a general representation of the various causal laws and axioms of $m\mathcal{A}+$ for the purposes of defining actions and their effects; a set of lp-functions [17] corresponding to the functions U_o, U_s, and U_a used by the transition function for obtaining the update instance for an action occurrence; and the representation of the update execution operator $\otimes$.

5.2.1 Representing Actions and their Effects

When it comes to representing actions and their effects, it is important to keep in mind a clear separation between *action occurrences as specified in the domain history* and the semantic objects defined by *their corresponding update instances*. The domain history, which serves as a record of the action occurrences at each step of a trajectory, is defined via the predicate `occurs/2` as detailed in the previous sections. In this section, we present a notation for actions and their effects based on the notion of named records whose fields on some level correspond to various components of their respective causal laws/axioms in an $m\mathcal{A}+$ action description.

In general, actions are defined as record-like structures whose fields are defined by the

following relations:

- `action(A)` - atoms of this form are define the *names* of individual actions and are read as: "A is an action."

- `type(A,T)` - atoms of this form define the *type* of the given action. As we've seen, actions in $m\mathcal{A}+$ are categorized into three broad groups: *sensing*, *communication*, and *ontic/physical*. Atoms of this form are read as: "action A is of the type T."

The remaining fields are conditionally included as part of the record based on the action's type:

- `announces(A,F)` - atoms of this form define the formula that is being announcement by a communication action, and are read as: "action A announces the formula F." The reader should notice the similarity between atoms of this type and *announcement axioms* of $m\mathcal{A}+$ as shown in (3.5).

- `determines(A,F)` - atoms of this form define the fluent revealed by a sensing action, and are read as: "action A determines the value of the fluent F." As with the previous predicate, the reader should not that this directly reflects the intent conveyed by sensing axioms of $m\mathcal{A}+$ defined in (3.4).

- `effect(A,F)` - atoms of this form define the effect of an ontic or physical action, and are read as: "action A makes the fluent literal F true." As before, this conveys part of the intent behind dynamic causal laws of $m\mathcal{A}+$ seen in (3.3).

The final field of our record, `precondition/3`, is included for ontic actions and defines the preconditions associated with the action's effects. Atoms of the form `precondition(A,F,P)`

are read as: "action A's effect F has the precondition P." For simplicity there is an underlying assumption that only one predicate of this type is defined for each action/effect pair[3].

Now that we've defined the vocabulary for representing *actions* generally, we recall that individual *action occurrences* are described in a domain history as collection of facts of the form `occurs(A,T)` and are read as: "action A occurs at time T." This is in keeping with our prior established notation and the general representation style from [2]. This is also in line with our intuition that an action occurrence is a specific instance of an action that transpires at a particular time.

As an intermediate step to defining our lp-functions, we also give definitions for the predicates which define the *frames of reference* that the agents of the domain have with respect to an individual action occurrence. In keeping with the intuition and various perspective axioms outlined in Chapter 3.1, these are:

- `observes(AG,A,T)` - defines the set of agents which have first-hand knowledge of the action occurrence (i.e. the full observers). Atoms of this form are read as: "agent Ag is a full observer of the occurrence of the action A at time T."

- `aware(AG,A,T)` - defines the set of agents which have second-hand knowledge of the action occurrence (i.e. the partial observers). Atoms of this form are read as: "agent Ag is aware of the occurrence of action A at time T."

- `oblivious(AG,A,T)` - defines the set of agents who are oblivious of the action occurrence. Atoms of this final form are read as: "agent AG is oblivious of the occurrence of the action A at time T."

The notation that we have established up to this point for defining actions and their effects additionally defines the input predicates for our lp-functions for U_o, U_s, and U_a. We now

[3]This assumption is in place due to practical programming considerations, not mathematical ones.

present our desired output predicates. Intuitively, each lp-function will take as its input the relevant portions of the domain description, a description of a state, and an action occurrence from the domain history, and will give as output a set of facts defining the corresponding update instance as specified by Definitions 3.14, 3.12, and 3.13.

Recall from Definition 2.15 that an update model representing an action occurrence is a tuple of the form $(E, R_{\alpha_1}, \ldots, R_{\alpha_n}, pre, sub)$, where:

- E is a finite, non-empty set of *events*
- each R_{α_i} is a binary relation on E called an *accessibility relation* for agent α_i
- $pre : E \mapsto \mathcal{L}_\Sigma$ assigns a *precondition* to each event
- $sub : E \mapsto SUB_{\mathcal{L}_\Sigma}$ assigns a $\mathcal{L}$*-substitution* to each event representing its direct effects

In keeping with this definition, the update model for a particular action occurrence is represented as a named record whose fields reflect the relevant fields of the tuple. In this particular, we assume that only one action occurs at any given time, and hence we can associate an update model with the action occurrence through their time step in the domain history. Update models are defined by the following predicates:

- `event(E,T)` - atoms of this from are read as: "E is an event associated with the update model from time T."
- `reference_event(E,T)` - atoms of this form are read as: "E is the reference event associated with the update model for time T."
- `e_reachable(E1,E2,A,T)` - atoms of this form are read as: "the pair (E_1, E_2) belongs to the accessibility relation for agent A in the update model defined for time T."

- `pre(F,E,T)` - atoms of this form define the preconditions associated with individual events in the update model and are read as: "formula F is the precondition for event E in the update model defined at time T."

- `sub(F1,F2,E,T)` - atoms of this define the $\mathcal{L}$-substitution associated with a particular event and are read as: "$F_1 \mapsto F_2$ is the $\mathcal{L}$-substitution associated with the event E in the update model defined for time T."

In the subsections that follow, we'll examine how this vocabulary is used to translate actions of each distinct category into their corresponding update instances.

What remains is to complete the definitions of the various lp-functions from *action occurrences* to *update instances* in the language of $m\mathcal{A}+$. For reference, we keep in mind the generalized update instance for such actions as presented in Figures 3.2 to 3.4. In this section we present the lp-function for *deterministic ontic actions*, leaving those for sensing and communication actions to the appendices[4].

Every deterministic ontic action, a, has an event associated with its direct effects, Φ, in addition to the so-called *inertial event*, ϵ_i. The relevant rules are given in Listing 5.12.

The preconditions associated with the individual events are almost directly pulled from the corresponding input predicate `precondition/3` in a fairly straightforward fashion, while the precondition for the inertial event is defined to be $\top$ as per Definition 3.14. Listing 5.13 shows the corresponding definitions.

Having defined both the events themselves and their respective preconditions, we now shift our focus to defining the requisite substitutions. Taking into account Definition 3.14, if according to our action description $[a \textbf{\textit{ causes }} \varphi]$, we define the requisite substitution $\{\neg\varphi \mapsto \varphi, \varphi \mapsto \varphi\}$. Similarly, if $[a \textbf{\textit{ causes }} \neg\varphi]$ we define the substitution $\{\varphi \mapsto \neg\varphi, \neg\varphi \mapsto$

[4]The reader should note that the definitions of the lp-functions for both sensing and communication actions follow in the exact same style however.

```
% Define the event representing the action's effects taking hold.
event(e(Phi,T), T) :-
    occurs(A, T), type(A, ontic),
    effect(A, Phi), action(A).

% Define the inertial event.
event(e(T), T) :-
    occurs(A, T), type(A, ontic),
    action(A).

% Mark e(Phi,T) as the reference event.
reference_event(e(Phi,T), T) :-
    occurs(A, T), type(A, ontic),
    event(e(Phi,T), T), action(A).
```

Listing 5.12: ASP definition of the events for the update instance corresponding to an occurrence of a deterministic ontic action.

```
% Define the precondition for the reference event.
pre(Psi, e(Phi,T), T) :-
    precondition(A, Phi, Psi),
    occurs(A, T), type(A, ontic),
    effect(A, Phi), event(e(Phi,T), T),
    action(A).

% Define the precondition for the inertial event.
pre(top, e(T), T) :-
    occurs(A, T), type(A, ontic),
    event(e(T), T), action(A).
```

Listing 5.13: ASP definition of the preconditions associated with the events in an update instance corresponding to an occurrence of a deterministic ontic action.

```
% If the action causes a fluent to be made true, then the corresponding
% substitution has the form ¬F ↦ F
sub(neg(Phi), Phi, e(Phi,T), T) :-
    occurs(A, T), type(A, ontic),
    effect(A, Phi), fluent(Phi),
    event(e(Phi,T), T), action(A).

% If the action causes a fluent to be made false, then the corresponding
% substitution has the form F ↦ ¬F
sub(Phi, neg(Phi), e(neg(Phi),T), T) :-
    occurs(A, T), type(A, ontic),
    effect(A, neg(Phi)), fluent(Phi),
    event(e(neg(Phi),T), T), action(A).

% Assign the identity substitution to all other fluents in the domain.
sub(Psi, Psi, e(Phi,T), T) :-
    occurs(A, T), type(A, ontic), fluent(Psi),
    not effect(A, Psi), not effect(A, neg(Psi)),
    event(e(Phi,T), T), action(A),
    Phi != Psi.

sub(Psi, Psi, e(neg(Phi),T), T) :-
    occurs(A, T), type(A, ontic), fluent(Psi),
    not effect(A, Psi), not effect(A, neg(Psi)),
    event(e(neg(Phi),T), T), action(A),
    Phi != Psi.
```

Listing 5.14: ASP definition of the $\mathcal{L}$-substitutions for events modeling the effects of an action occurrence.

$\neg\varphi\}$. Lastly, we must associate the identity substitution with the inertial event, and all remaining formulae are unaffected by the action. The corresponding definitions are given in Listings 5.14 and 5.15.

The final set of rules given in Listing 5.16 defines the accessibility relations for the agents in accordance with Definition 3.14.

```
% Define the substitutions for the inertial event. In this semantics,
% the identity substitution is assigned for every fluent.
sub(Phi, Phi, e(T), T) :-
    occurs(A, T), type(A, ontic),
    event(e(T), T), action(A),
    fluent(Phi).
```

Listing 5.15: ASP definition of the $\mathcal{L}$-substitutions associated with the inertial event.

```
% Define the accessibility relations for the update model.
e_reachable(e(Phi,T), e(Phi,T), AG, T) :-
    occurs(A, T), type(A, ontic),
    observes(AG, A, T), event(e(Phi,T), T),
    agent(AG), action(A).

e_reachable(e(Phi,T), e(T), AG, T) :-
    occurs(A, T), type(A, ontic),
    oblivious(AG, A, T), event(e(Phi,T), T),
    event(e(T), T), agent(AG), action(A).

e_reachable(e(T), e(T), AG, T) :-
    occurs(A, T), type(A, ontic),
    event(e(T), T), agent(AG), action(A).
```

Listing 5.16: ASP definition of the accessibility relations associated the update instance for an occurrence of a deterministic ontic action.

5.2.2 Representing the Update Execution Operator

Recall from Definition 2.17 that the *update execution* operator which is at the heart of the transition function in Definition 3.15 is essentially the product of two graphs: the Kripke world corresponding to the current state of the domain; and the update instance representing an action. The representation of the operator in Listing 5.17 directly reflects its definition.

The first part of our representation involves defining the set of potential points which

```
% Define points in the successor state
point(w(P,E), T) :-
    point(P, T - 1), event(E, T - 1),
    pre(Phi, E, T - 1), entailed_by(Phi, P, T - 1),
    action_occurs(T - 1).

% Define the reference point for the successor state
reference_point(w(P,E), T) :-
    reference_point(P, T - 1), reference_event(E, T - 1),
    point(w(P,E), T), action_occurs(T - 1).

% Define the accessibility relations for the successor state.
k_reachable(w(P1,E1), w(P2,E2), AG, T) :-
   k_reachable(P1, P2, AG, T - 1), e_reachable(E1, E2, AG, T - 1),
   point(w(P1,E1), T), point(w(P2,E2), T),
   point(P1, T - 1), point(P2, T - 1),
   event(E1, T - 1), event(E2, T - 1),
   agent(AG), action_occurs(T - 1).

% Define the interpretation functions due to identity substitutions.
value_of(Phi, top, w(P,E), T) :-
   value_of(Phi, top, P, T - 1), sub(Phi, Phi, E, T - 1),
   point(w(P,E), T), point(P, T - 1), event(E, T - 1),
   fluent(Phi), action_occurs(T - 1).

value_of(Phi, bot, w(P,E), T) :-
   value_of(Phi, bot, P, T - 1), sub(Phi, Phi, E, T - 1),
   point(w(P,E), T), point(P, T - 1), event(E, T - 1),
   fluent(Phi), action_occurs(T - 1).

% Define the interpretation functions for substitutions of the form F ↦ ¬F
value_of(Phi, bot, w(P,E), T) :-
   value_of(Phi, top, P, T - 1), sub(Phi, neg(Phi), E, T - 1),
   point(w(P,E), T), point(P, T - 1), event(E, T - 1),
   fluent(Phi), action_occurs(T - 1).

% Define the interpretation functions for substitutions of the form ¬F ↦ F
value_of(Phi, top, w(P,E), T) :-
   value_of(Phi, bot, P, T - 1), sub(neg(Phi), Phi, E, T - 1),
   point(w(P,E), T), point(P, T - 1), event(E, T - 1),
   fluent(Phi), action_occurs(T - 1).
```

Listing 5.17: ASP definition of the update execution operator.

comprise the Kripke world corresponding to the successor state. This reflects the first part of the operator's definition:

$$M'.W = \{(\omega_\sigma, \varepsilon_\mu) \mid \omega_\sigma \in M.W, \varepsilon_\mu \in U.E, (M, \omega_\sigma) \models U.pre(\varepsilon_\mu)\}$$

The next part of our representation reflects the part of the definition defining the accessibility relations for each agent in the Kripke world defining the successor state:

$$M'.R_i = \{((\omega_\sigma, \varepsilon_\mu), (\omega_\tau, \varepsilon_\nu)) \mid (\omega_\sigma, \omega_\tau) \in M.R_i \text{ and } (\varepsilon_\mu, \varepsilon_\nu) \in U.R_i\}$$

Finally, the third set of rules concerns the interpretation functions associated with each point in the Kripke world defining the successor state.

$$M'.\pi((\omega, \varepsilon))(f) = U.sub(\varepsilon)(f)$$

5.3 Temporal Projection for $m\mathcal{A}+$ via ASP

Now that the answer-set prolog representations of the notions of: states of the domain; modal formulae and the entailment relation; actions and the domain history; the lp-functions from action occurrences to update instances; and the update execution operator; in this section we illustrate how the individual modules can be put together to give an answer-set programming semantics for $m\mathcal{A}+$, and be applied to the task of temporal projection. Recall that the semantics of $m\mathcal{A}+$ is given through the transition function in Definition 3.15:

$$\Phi_\Delta(\sigma, a) = \begin{cases} \sigma \otimes U_o(\sigma, a) & \text{ontic action} \\ \sigma \otimes U_s(\sigma, a) & \text{sensing action} \\ \sigma \otimes U_a(\sigma, a) & \text{otherwise} \end{cases}$$

As was mentioned previously, the temporal projection problem involves answering the following question: "Given a state σ and an action a, what is the successor state(s)

obtained by performing a in σ?" It is clear that the definition of $\Phi_{\Delta}(\sigma, a)$ already provides a mathematical formulation of the problem's solution. What remains is for us to discuss a means of automating this reasoning task through the use of logic programming.

Taking a step back, we can see that each individual component of the transition function corresponds to a a logic program (or fragment of a program) developed in the previous sections. For reference, we associate the following names with each individual *answer-set prolog module*:

- Π_{Σ} - represents the module containing the definition of the *domain signature*, Σ, modulo the definitions of actions as shown in Section 5.1.1.

- Π_{del} - represents the syntax and semantics of the subset of modal logic that is relevant to the domain in question as shown in Sections 5.1.2 and 5.1.2.2.

- Π_{σ} - represents a particular state of the domain according to the vocabulary described in Section 5.1.2.1. Typically this module will define the initial state of the domain (if the focus is planning), or the initial state of a trajectory of interest (for temporal projection).

- $\Pi_{\mathcal{A}}$ - represents the module defining the actions of the domain and their respective effects as shown in Section 5.2.1.

- Π_{ontic} - represent the module defining the lp-function from *action occurrences* of *ontic* actions to their corresponding update instances. The lp-function was presented in Section 5.2.1. Modules for *sensing*, Π_{sns}, and announcement actions Π_{ann} are also present.

- Π_{h} - represents the module defining the domain history (i.e. a record of the action occurrences which took place from a particular state/step of a trajectory in the broader

diagram).

- $\Pi_{\otimes}$ - represents the module defining the update execution operator as shown in Section 5.2.2.

Theorem 5.2. *Let Δ be an action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state of the transition diagram defined by Δ, and a be an action. The successor state(s) obtained by performing the action a in the state σ are given as part of the answer-set(s) of the logic program $\Pi_{m\mathcal{A}+}$ given below:*

$$\Pi_{\Delta} \cup \Pi_{\sigma} \cup \Pi_{\otimes} \cup \Pi_{h} \cup \begin{cases} \Pi_{ontic} & \textit{ontic action} \\ \Pi_{sns} & \textit{sensing action} \\ \Pi_{ann} & \textit{announcement action} \end{cases} \tag{5.1}$$

where $\Pi_{\Delta} = \Pi_{\Sigma} \cup \Pi_{\mathcal{A}} \cup \Pi_{del}$.

Theorem 5.2 establishes the equivalence between $\Phi_{\Delta}(\sigma, a)$ and the union of a set of logic programs, and consequently provides a means for us to automate the task of temporal projection.

Example 5.3 (Temporal Projection in the Concealed Coin Domain: Single Action)**.** *Recall the Concealed Coin Domain from Example 5.3: Three agents, **A**, **B**, and **C**, are together in a room with a box which contains a coin. Suppose that this fact, together with the fact that none of the agents knows which face of the coin is showing is a commonly held belief among them. Furthermore let us suppose that all of the agents are attentive and that this too is a commonly held belief. Lastly, let us assume that the coin is facing heads up and that all of the beliefs of the agents are true.*

An agent may peek inside to determine which face of the coin is showing. In addition, agent may signal or distract one of his fellows, thereby causing him to be respectively attentive

```
step(0..1).

agent(a; b; c).

fluent(locked).

fluent(heads).

fluent(attentive(AG)) :- agent(AG).

value(top). value(bot).
```

Listing 5.18: ASP definition of the domain signature for the Concealed Coin Domain.

or inattentive. Attentive agents are fully aware of what transpires around them. What is the successor state obtained after agent A distracts agent C?

In keeping with our past examples, we begin by representing the domain signature and initial state of the domain as shown in Listings 5.18 and 5.19. These modules correspond to Π_Σ *and* Π_σ *and are inherently* domain dependent. Π_{del} *while mathematically may be considered* domain independent*, from a practical consideration is also treated as a* domain dependent *module, but the listing has been omitted from the example in the interest of clarity of presentation.* Π_h *which defines the domain history contains the single fact* `occurs(distract(a,c), 0).` *The remaining modules are unchanged from their prior descriptions in this chapter.*

◇

Intuitively, the consequences of *A* distracting *C* ought to be that where it was initially common knowledge that *C* was attentive, it now becomes common knowledge that he is not. Computing the answer sets of the program:

$$\Pi_\Delta \cup \Pi_\sigma \cup \Pi_\otimes \cup \Pi_h \cup \Pi_{ontic}$$

```
point(w1, 0). point(w2, 0).

reference_point(w1, 0).

value_of(locked, top, w1, 0).
value_of(heads, top, w1, 0).
value_of(attentive(a), top, w1, 0).
value_of(attentive(b), top, w1, 0).
value_of(attentive(c), top, w1, 0).

value_of(locked, top, w2, 0).
value_of(heads, bot, w2, 0).
value_of(attentive(a), top, w2, 0).
value_of(attentive(b), top, w2, 0).
value_of(attentive(c), top, w2, 0).

k_reachable(w1, w1, a, 0). k_reachable(w1, w2, a, 0).
k_reachable(w2, w1, a, 0). k_reachable(w2, w2, a, 0).

k_reachable(w1, w1, b, 0). k_reachable(w1, w2, b, 0).
k_reachable(w2, w1, b, 0). k_reachable(w2, w2, b, 0).

k_reachable(w1, w1, c, 0). k_reachable(w1, w2, c, 0).
k_reachable(w2, w1, c, 0). k_reachable(w2, w2, c, 0).

holds(F, T) :-
    entailed_by(F, P, T), reference_point(P, T),
    modal_formula(F), point(P, T), step(T).
```

Listing 5.19: ASP definition of the initial state for the Concealed Coin Domain.

```
$ clingo -n 0 domain-signature.lp initial-state.lp del-definitions.lp
             action-definitions.lp domain-history.lp lp-ontic.lp
holds(c(attentive(c)),0)

holds(c(neg(attentive(c))),1)

SATISFIABLE

Models       : 1
Calls        : 1
Time         : 4.052s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 4.031s
```

Listing 5.20: Partial output of temporal projection for $distract(A, C)$ in the Concealed Coin Domain.

as shown in Listing 5.20 shows that the answer set matches our intuition[5].

Example 5.4 (Temporal Projection in the Concealed Coin Domain: Multiple Actions)**.** *In this example, we examine temporal projection over a sequence of elementary actions in a slight elaboration of the Concealed Coin Domain. Let us assume that the box is initially locked, and may be unlocked via the action unlock. This action is modeled rather simply by introducing a new ontic action into the domain, and representing its effects by adding a dynamic causal law and corresponding perspective axioms. Suppose that agent* A *wishes to peek into the box and therefore learn whether the coin is facing heads or tails, but that he wishes to do so without the knowledge of agent* C*. One possible plan to achieve this goal would be:*

$$[distract(A, C), unlock(A), peek(A)]$$

Temporal projection would enable us to verify the plan, and to see what other potential consequences of the action sequence might be. If the coin is initially facing heads up, then

[5] As before, the output of `clingo` has been formatted and trimmed for presentation purposes.

```
$ clingo -n 0 domain-signature.lp initial-state.lp del-definitions.lp
             action-definitions.lp domain-history.lp lp-ontic.lp
             lp-sensing.lp
holds(op(a,heads),3)
holds(op(c,locked),3)
holds(op(c,op(a,locked)),3)
holds(neg(op(c,op(a,heads))),3)
holds(neg(op(c,op(a,neg(heads)))),3)

SATISFIABLE

Models       : 1
Calls        : 1
Time         : 26.709s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.01s)
CPU Time     : 26.292s
```

Listing 5.21: Partial output of temporal projection for the action sequence from Example 5.4.

at a minimum, A will become aware of this fact, and C will be oblivious given that he has been distracted with other matters. Computing the answer sets of the program:

$$\Pi_{\Delta} \cup \Pi_{\sigma} \cup \Pi_{\otimes} \cup \Pi_{h} \cup \Pi_{ontic} \cup \Pi_{sns}$$

using the new domain history and updated action description gives the output[6] *shown in Listing 5.21. Examining the answer sets shows once again that the formulae captured by the relation* `holds/2` *comports with our intuition.* ◇

5.4 Multi-Agent Planning for $m\mathcal{A}+$ via ASP

Temporal projection as discussed in the previous section forms the basis of a more general reasoning task known as *planning*. The planning problem centers on answering the

[6]Once again, he output of `clingo` has been formatted for presentation purposes.

following question: "Let $\mathcal{D}$ be a dynamic domain defined by an action description Δ; σ_0 be a state in the transition diagram defined by Δ; and σ_G be a *goal state*. Is there a sequence of actions which labels a path from σ_0 to σ_G in the transition diagram defined by Δ?"

The approach taken in [2, 24] (and within the action language community more broadly) for solving the planning problem follows a *generate and test strategy*. Candidate plans are generated via a logic program known as a *planning module*, and temporal projection is used in conjunction with a *goal specification module*, to evaluate whether or not the candidate plan defines an appropriate trajectory. In this section we present both modules in a general fashion similar to the way temporal projection was presented in the previous section, followed by an in-depth presentation of planning in the context of the Concealed Coin Domain from Example 3.6, and the Escapee Domain of Example 3.17.

5.4.1 Goal Specification

Goal states are defined in terms of the set of properties that they must satisfy, which in a single agent context are described by consistent sets of fluent literals. In a multi-agent context goals may be specified in terms of both *ontic* and *epistemic properties* of the domain. The relevant properties of a goal state are defined through a new pair of relations, `goal` and `goal/1`, which are understood as follows:

- `goal` - this relation represents the notion of the goal having been met. The atom `goal` is read as: "the desired goal has been reached at some step/state along the trajectory being investigated."

- `goal/1` - this relation represents the notion of an individual step/state satisfying the desired goal. Atoms of the form `goal(T)` are read as: "the desired goal has been reached at step T of the current trajectory."

```
% The goal has been reached if there exists some step of the trajectory
% T within the time horizon N which satisfies the goal.
goal :- goal(T),
    T <= N,
    step(T),
    horizon(N).

% Step T of a trajectory satisfies the goal if formulae F_1,...,F_n
% hold in that state.
goal(T) :- holds(F1, T), ..., holds(Fn, T),
    step(T).

% The goal must be eventually satisfied.
:- not goal.
```

Listing 5.22: Generalized ASP representation of the domain specific part of a goal specification.

The relation `goal/1` is defined in terms of the relation `holds/2` from Section 5.1.3. Individual properties are specified by modal formulae, and the goal itself is given in terms of their conjunction: $\varphi_1 \wedge \ldots \wedge \varphi_n$. Listing 5.22 shows a *domain independent* generalization of the goal specification module. It is important to note that in general, we limit our search for plans/paths whose length is constrained to within a certain time horizon.

5.4.2 Action Generation

Action generation is a *domain independent* module which serves as the *generate* component of the *generate and test strategy* for planning. Candidate domain histories are generated by a *choice rule*, and various other heuristic rules may be included to constrain/refine the kinds of potential plans which are explored. Temporal projection is then applied in combination with the goal module to determine whether or not the trajectories label a path from the initial state to a goal state. For the purposes of this dissertation, we limit the kinds of

```
% Choose exactly one action A for each step of the trajectory if the
% goal has not been achieved.
1{occurs(A,T) : action(A)}1 :-
    not goal(T), step(T),
    horizon(N), T <= N.
```

Listing 5.23: Elementary ASP action generator in the language of `clingo`.

plans under consideration to those in which *only a single action occurs at any particular step/state along the trajectory*. Listing 5.23 shows an elementary action generator (in the language of the `clingo` answer-set programming system) which selects exactly one action occurrence for each step of the candidate trajectory. Listing 5.24 shows the same module expressed in the language of the `DLV` system. It should be noted that while equivalent, the input language of the `DLV` system is more closely inline with the definition of the answer-set prolog language given in Chapter 2.

Example 5.5 (Multi-Agent Planning in the Concealed Coin Domain)**.** *In the context of the Concealed Coin Domain, suppose that agent A wishes to determine the value of the fluent heads, but to do so without the knowledge of agent C. Example 5.4 presented one potential plan to accomplish this task, but how might we derive the trajectory? In keeping with the discussion on planning, we begin by specifying the goal as shown in Listing 5.25. If we extend our planner by adding a restriction which limits the actions considered to only those performed by agent A, and limit the class of modal formulae to only those which are relevant for evaluating whether the goal has been reached or not, the answer sets given by the program (where* Π_g *and* Π_{pl} *denote the goal and planning modules):*

$$\Pi_{\Delta} \cup \Pi_{\sigma} \cup \Pi_{\otimes} \cup \Pi_{ontic} \cup \Pi_{sns} \cup \Pi_{g} \cup \Pi_{pl}$$

are shown in Listing 5.26. ◇

```
% If the goal has not been reached at a given step within our horizon,
% every action either occurs or does not.
occurs(A, T) | -occurs(A, T) :-
    not goal(T), step(T),
    horizon(T), T <= N.

% It is not possible for more than one action to occur at a given step
% of the candidate trajectory.
:- occurs(A1, T), occurs(A2, T),
   action(A1), action(A2),
   A1 != A2.

% At each step along the candidate trajectory at least one action must
% transpire (i.e., we cannot have a state in which no action is taken).
:- goal(T1), not goal(T1 - 1),
   step(T1), step(T2), T2 < T1,
   not action_taken(T2).

action_taken(T) :- occurs(A, T).
```

Listing 5.24: Elementary ASP action generator in the language of DLV.

```
goal(T) :-
    holds(neg(locked), T), holds(op(a,heads), T),
    holds(neg(op(c,op(a,heads))), T), step(T).

goal(T) :-
    holds(neg(locked), T), holds(op(a,neg(heads)), T),
    holds(neg(op(c,op(a,neg(heads)))), T), step(T).
```

Listing 5.25: Goal specification for Example 5.5.

5.5 Comparison with other Methodologies

At this point it is worth emphasizing that both action languages and answer-set prolog are *knowledge representation languages*, and *not planning languages*. In fact, of those logic

```
$ clingo -n 0 domain-signature.lp initial-state.lp del-definitions.lp
             action-definitions.lp lp-ontic.lp lp-sensing.lp goal.lp plan.lp
Solving...
Answer: 1
occurs(distract(a,c),0) occurs(unlock(a),1) occurs(peek(a),2)
Answer: 2
occurs(unlock(a),0) occurs(distract(a,c),1) occurs(peek(a),2)
SATISFIABLE

Models       : 2
Calls        : 1
Time         : 79.602s (Solving: 3.23s 1st Model: 0.54s Unsat: 2.59s)
CPU Time     : 78.773s
```

Listing 5.26: Partial output of planning from Example 5.5.

programs presented in the previous sections, only those in Sections 5.4.1 and 5.4.2 pertain to the planning problem specifically. This is in marked contrast with other related work such as the classical planning approaches of [40, 41] and DEC-POMDP approaches outlined in [30], and bears some discussion.

As has been stated previously, both $m\mathcal{A}+$ and $m\mathcal{AL}$ are primarily to be understood as *knowledge representation languages*, and hence a general means of describing dynamic multi-agent domains. What this means is that they are intended to provide a concise, elaboration tolerant, and intuitive means of describing potentially huge and complex transition systems characterizing such domains. Once such a system has been described, various different kinds of reasoning tasks can then be cast in terms of these diagrams. As has been mentioned before, *planning* is reducible to finding a path in the diagram from some initial state to a goal state. *Temporal projection* is reducible to finding the successor state(s) in accordance with the diagram's transition function. *Query answering* is reducible to determining various properties of a particular state of the diagram.

Before we compare this approach to classical or DEC-POMDP based planning, some

elaboration of the fundamental assumptions with regards to the nature of both the action descriptions and respective transition diagrams of $m\mathcal{A}+$ and $m\mathcal{AL}$ is warranted. Specifically, the approach outlined in this dissertation makes two assumptions:

1. The knowledge/beliefs of the agents with regards to the *initial state* are *shared* and *complete*.

2. The knowledge of the agents with regards to *actions and their effects* is also *shared* and *complete*.

Both of these assumptions are inherited from the approach taken by the languages $\mathcal{A}$ and $\mathcal{AL}$, and are what make the framework capable of modeling both *cooperative* and *adversarial* planning. Before discussing this in some more detail, we also note that there is a distinction between the tasks of *planning* and *execution*, both of which are discrete steps within what is called an agent-loop [2].

Consider the "Escapee Domain" as outlined in Chapter 3.3.2 and also in [15]. The scenario is an example of an adversarial planning situation in which agents A and C must work together against agent B. When finding a plan to achieve the goal of A's escape it is tempting to ask: "From *whose perspective* is the plan being generated?" or "Who is the *active reasoner*?" Assumptions (1) and (2) render this question moot, as the transition diagram itself is global in nature. This mirrors a certain intuition about the reasoning process — namely that a dispassionate reasoner "steps outside himself" and reasons about how his actions (and those of his fellows) may play out, in the third person. Once the plan is generated, the execution phase of the agent loop comes into play, and it is here that the actual mechanics of coordination (i.e., the *question of how* individual components of the plan are relayed to other agents) is addressed. Such coordination is considered in the context of this work to be done on a "metal-level."

5.6 Classical Planning Approaches

Thematically, the work done in [40, 41] centers on the application of *classical planning techniques* in a multi-agent context. The work presented in this dissertation however seeks to answer a more general question, namely "How can we *represent and reason about dynamic multi-agent domains*?" This focus on knowledge representation and a view of reasoning which includes tasks other than planning (such as temporal projection, query answering, and diagnostic reasoning), leads to some significant differences.

With regards to [40], in keeping with the classical planning tradition, the authors' choice of representation language may be adequately described as a multi-agent extension to STRIPS, achieved by adding to the language what are called *restricted modal literals*, which are defined by the following grammar:

$$\varphi \Rightarrow p \mid \mathcal{B}_{\alpha}\varphi \mid \neg\varphi$$

where p is a proposition, and α is the name of an agent in the domain. Modal literals of the form $\mathcal{B}_{\alpha}\varphi$ are read in the standard way as "agent α *believes* φ." For a number of reasons, the depth of literals of this form is restricted[7]. In the action languages $m\mathcal{A}+$ and $m\mathcal{AL}$, no such restrictions are imposed. Furthermore, $m\mathcal{A}+$ and $m\mathcal{AL}$ are multi-modal, including operators to describe more complex constructs such as *common knowledge/belief* and allow for the use of *modal formulae* which are more expressive than *restricted modal literals*.

Another consequence of the use of restricted modal literals and classical planning systems is that the definition of a *state of the domain* differs signifcantly from that used in both $m\mathcal{A}+$ and $m\mathcal{AL}$. Rather than modeling a state as a *Kripke world*, a state is seen as a *consistent collection of restricted modal literals*. By keeping the depth of such literals bound to a positive integer N, these sets are finite, and the satisfaction of various axioms of belief

[7] In the paper the authors specifically deal with literals whose depth is ≤ 3.

(specifically the **KD45** axioms) is made explicit by encoding them directly as part of an action's effects. In both $m\mathcal{A}+$ and $m\mathcal{AL}$, these axioms are characterized by the accessibility relations of the agents and are an inherent property of the semantics. It should be noted however that this approach is worth considering as an *approximation semantics* of $m\mathcal{A}+$ with the aim of *improving the computational aspects* of various reasoning tasks (such as the approach taken for planning outlined in this chapter).

A final consequence that bears mentioning is tha by extending STRIPS, the formalism of [40] lacks the notion of a state constraint in the vein of $\mathcal{AL}$ and $m\mathcal{AL}$. Specifically, they represent "state constraints as ancillary conditional effects of actions." This precludes them from being able to model domains with recursive interdependencies between fluents [33] and impacts the *elaboration tolerance* [38] of their formalizations as well.

The work presented in [41] addresses the application of classical planning techniques to *cooperative multi-agent planning* domains in which the agents have both *private* and *public* knowledge about the domain. More specifically, it describes a particular *distributed search algorithm* which may be used to find plans for such problem domains. In part, due to its emphasis on cooperative multi-agent planning and overall approach, the work appears to share a similarity with that of [22]. Many of the aforementioned comparisons however are still applicable, and an additional one is that both $m\mathcal{A}+$ and $m\mathcal{AL}$ model the domain from the perspective of a system designer (who is outside of the domain itself), and not the perspective of the individual agents operating within the domain as in. This means that there is no notion of *private* or *public* fluents as in [41], and renders $m\mathcal{A}+$ and $m\mathcal{AL}$ incapable of representing *privacy preserving planning* as envisioned in [41] not directly representable.

5.7 DEC-POMDP Approaches

The DEC-POMDP approach outlined in [30] while interesting in its own right, is again, not intended as a knowledge representation formalism, and consequently shares some of the same distinctions that have already been outlined. Some additional distinctions are worth commenting on however, in particular the way in which *uncertainty* is addressed. The work presented in [30] describes several classes of uncertainty: regarding *action outcomes*; the accuracy of *sensor information*; and uncertainty about *the information of other agents*. Sensor data is not something considered in this work, so the discussion will center around the other two categories.

One significant difference is how uncertainty is understood in each representation. DEC-POMDPS understand/model uncertainty using probabilistic information (viewing probability as a measure of degrees of belief). An important part of the definition of a DEC-POMDP is the *transition probability function* which defines the probability of a transition from a state σ to σ' after an occurrence of an action a. This is a very different way of describing actions and their effects (even nondeterministic effects) from the action language approach. Another distinction is that uncertainty about beliefs of the agents is described in this work through the use of disjunction and modal operators such as $\mathcal{B}$, not via probability. As an example, the notion of an agent α being uncertain about the value of a fluent f would be modeled in either $m\mathcal{A}+$ or $m\mathcal{AL}$ by the modal formula $\mathcal{B}_A f \vee \mathcal{B}_A \neg f$. Such operators may be nested, allowing for the description of uncertainty about the beliefs of other agents present in the domain as well, but in a courser way than with probability (which is a natural direction for future research as outlined in Chapter 6.1.2).

DEC-POMDPS also model reasoning tasks such as planning in a profoundly different way from the approach taken in this work. As was mentioned previously, the action language

approach reduces planning to finding a path in the transition diagram from some initial state to a goal state, using a general purpose computation framework in the form of answer set programming to accomplish this task. Each DEC-POMDP on the other hand is associated with a *reward function*, and when combined with the probabilistic transition function, reduces the task of planning to *the optimization of an objective function*. These two distinct models differ substantially in terms of their underlying kinds, leading to divergent computational models.

One final point worth mentioning is that in keeping with the action language tradition, both $m\mathcal{A}+$ and $m\mathcal{AL}$ are languages which at their core are used to represent the transition diagram at the heart of dynamic multi-agent domain. The approach outlined in [30] is a *computational approach*, in that the mathematical object of a DEC-POMDP is defined, but there does not appear to be a *representation language* by which a system designer may in concise, intuitive, and elaboration tolerant manner describe such an object. This is an important distinction that bears emphasis - $m\mathcal{A}+$ and $m\mathcal{AL}$ are *knowledge representation languages* first, and the answer-set prolog semantics of $m\mathcal{A}+$, while important and having a computational aspect, is not the primary feature of the language.

Chapter 6

CONCLUSIONS AND FUTURE WORK

To recap, this dissertation presents work done on combining the formalisms developed within both the action language and dynamic epistemic logic communities, for representing and reasoning about dynamic multi-agent domains. The action languages $m\mathcal{AL}$ and $m\mathcal{A}+$ provide a high level representation framework which distinguishes itself from the update model approach of [4, 43] making explicit the distinction between an *action* (an object which is described as part of a domain description), and an *action occurrence* (an instance of an action which is recorded as part of the domain history). This distinction is made possible by the fact that in both of these languages, the frames of reference of the agents are *state specific attributes*, as opposed to information that distinguishes one action from another as is the case with the update model approach.

My own future work in this area will center on attempting to answer three broad questions:

- What is the distinction between knowledge and belief and can we capture this distinction in high level formalism that is amenable to computation?
- How can we incorporate the idea of probability as a measure of an agent's degree of belief into a compatible formalism for multi-agent reasoning?
- How can we more efficiently automate various multi-agent reasoning tasks?

6.1 Regarding Knowledge and Belief

Recall from Definition 2.9, that the syntax of modal formulae presents us with only a single class of modal operator, $\Box_\alpha$, which may be by convention interpreted to denote either

the modality of knowledge or belief. Furthermore, as part of the semantics of such formulae, Definition 2.10 tells us that the points of a particular Kripke world correspond on some level to *possible worlds*, and that the accessibility relations describe a what the agents perceive about them. From [19], if a pair $(\omega_\sigma, \omega_\tau)$ belongs to the accessibility relation of an agent α, it is understood to mean something along the following lines: "from the possible world ω_σ, agent α consider ω_τ to be possible." This means that neither modality is embedded directly within the notion of a Kripke model/world itself. Rather, depending on the modality of interest, only certain *kinds of models* are held up for consideration. If the modality of interest is that of knowledge, then models which satisfy the **S5** axioms are typically considered, whereas the modality of belief is associated with other axiom systems. These axiom systems constrain the kinds of accessibility relations that are part of the Kripke model for a particular theory.

This means that neither the syntax, nor the semantics of modal formulae is expressive enough to allow for the accommodation of both modalities within the same formalism. Intuitively however, what an agent *knows*, and what he may *believe*, while related, are two distinct pieces of information that ought to be representable within a multi-agent framework.

On a related note, the particular view of answer-set prolog that has been shown thus far in this dissertation has been through a programmatic lens. In other words, answer-set prolog has been used in this work not as a knowledge representation tool in its own right, but as a programming language with high level features suitable for solving problems of considerable combinatorial complexity. While a valid use of the language, there is another way to understand the meaning of an answer-set prolog program that is more apropos to the current discussion. Namely, rather than viewing an answer-set prolog program as a tool for solving combinatorial problems, we can adopt and exploit an *epistemic view*, in that programs are *descriptions of the beliefs of an agent about a particular domain.* This view

was elaborated upon, leading to an extension of the base language to *epistemic specifications* [29], a logic programming language which allows us to specify on some level both the modalities of knowledge and of belief.

6.1.1 Epistemic Specifications

Epistemic specifications may be viewed as an extension of answer-set prolog by the addition of a new class of literal, called a *subjective literal*, by applying the modal operators $\mathcal{K}$ or $\mathcal{M}$ to literals. Intuitively, a literal of the form $\mathcal{K}f$ may be understood as denoting that "f *is known*", while a literal for the form $\mathcal{M}f$ may be taken to mean that "f *may be believed*." The reader should already note that the language contains within it both the modalities of knowledge and belief, although the intuitions behind their meanings differ somewhat from those which inform dynamic epistemic logic. Informally, and without getting into the specifics, the semantics is based on the idea of a *world view*, which may be seen as a set of answer sets of the base program, where $\mathcal{M}f$ is satisfied if f is satisfied by at least one of the answer sets comprising a world view, and $\mathcal{K}f$ is satisfied if f is true in all of them. This parallels Definition 2.12 regarding the entailment relation for modal formulae of the form $\Box_\alpha\varphi$, suggesting a relationship between between world views and certain kinds of Kripke models, and also hints at a potential means of dealing with both modalities in a single framework. Example 6.1 presents some preliminary thoughts on this point.

Example 6.1 (World Views and Kripke Models)**.** *Consider the following disjunctive logic program,* Π_A*, describing the beliefs of an agent A:*

$$p \vee \neg p \leftarrow q.$$

$$q.$$

Π_A *has a world view comprised of its two answer sets:* $\{p, q\}$ *and* $\{\neg p, q\}$. *One way that this world view may be interpreted is shown by the* **S5***-model in Figure 6.1.*

The epistemic specification captured by Π_A *satisfies the formula* $\mathcal{K}q$ *and it should be noted that the Kripke model entails the formula* $\Box_A q$*, suggesting a relationship between the epistemic operator* $\mathcal{K}$ *and the modal operator* $\Box_A$. *A more important thing to notice however is that* Π_A *entails both* $\mathcal{M}p$ *and* $\mathcal{M}\neg p$*, while the Kripke model entails no analogous formula. This leads to the following as a potential alternative reading of the epistemic operators* $\mathcal{M}$ *and* $\mathcal{K}$*:*

- $\mathcal{M}f$ - *the agent* believes f to be possible.
- $\mathcal{K}f$ - *the agent* believes that f is true

Under this interpretation, the epistemic operator $\mathcal{K}$ *is analogous to the modal operator* $\mathcal{B}$*, coupled with the* **S5** *axiom system. If we introduce a means of representing* external information*, by for example marking which of the answer sets comprising a world view corresponds to the true state of the domain, it becomes clear that it is possible to introduce a third operator corresponding to knowledge under the interpretation of* knowledge as true belief[1]. ◇

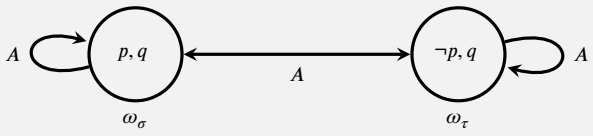

Figure 6.1: A Kripke model satisfying the **S5** axiom system corresponding to the answer sets/world view of Π_A.

[1] In fact, it is even possible to model a view of *knowledge as justified true belief* if we use the semantics of epistemic specifications as an inherent justification.

In addition of having a potential theoretical benefit, the language of epistemic specifications may also have a programmatic benefit when it comes to automating various reasoning tasks. Recent work building off of [29] has seen the development of preliminary solvers for programs written in this language, and it may be the case that the programs described in Chapter 5 may be simplified considerably by shifting to an alternative formalism. As one example of a potential simplification, it is quite possible that the modules described in Chapter 5.1.2 through 5.1.2.2 may be eliminated entirely given that the language of epistemic specifications already contains within its semantics analogous constructs to those which are emulated in pure answer-set prolog.

6.1.2 Probability as Degree of Belief

Another area of new research is in the integration of probability into our understanding of belief. Continuing the discussion from Example 6.1, we can notice that all of the world views of Π_A, and hence all of the points of the corresponding Kripke model are held as *equally possible* (i.e., all possible worlds are equally likely). What if this is not the case? In other words, how can we incorporate an understanding of probability as *a degree (or measure of the degree) of belief* into our formalism and representations? Work done in [18] extended the foundations of answer-set prolog to include probabilistic information, and a similar extension of the language of epistemic specifications seems to be a natural investigation to pursue. In conjunction with this, an extension of the Kripke semantics of modal logic by probability seems to also be a worthwhile endeavor. Example 6.2 presents some basic thoughts on this question.

Example 6.2 (Incorporating Probability as Degree of Belief)**.** *Recall the logic program,* Π_A*,*

presented in Example 6.1:

$$p \vee \neg p \leftarrow q.$$

$$q.$$

In this program, given q, agent A believes that either p or its negation are equally likely. What if this is not the case? What if our agent knows the following:

$$P(p \mid q) = 0.75$$

$$P(\neg p \mid q) = 0.25$$

Here we have conditional probabilites for both p and its negation given that q is true. This information would then weigh *the agents beliefs, inclining him to believe in p over ¬p. If we use a similar interpretation to that in Example 6.1, we could easily envision an extension of the Kripke semantics to give us a* probabilistic Kripke model *as shown in Figure 6.2, along with a new definition of the entailment relation which incorporates this notion of* weight *or* degree of certainty. ◇

Figure 6.2: A probabilistic Kripke model satisfying the **S5** axiom system corresponding to the answer sets/world view of Example 6.2.

6.2 Optimizing Multi-Agent Reasoning

The final area of research that I intend to pursue focuses on program optimization and answer-set solving algorithms in order to make automated reasoning in the style presented in this dissertation more efficient. In general, results from [1] states that for the *logic of public announcements*, the satisfiability and model checking problems are respectively *PSPACE*-complete and in P, while for dynamic epistemic logic with event models (which are used to define the semantics of $m\mathcal{A}+$), the model checking problem is *PSPACE*-complete while the satisfiability problem is *NEXPTIME*-complete. This seems to suggest that in general, an approximation semantics similar to the approach taken in [40] seems to be a promising avenue of optimization.

Outside of an approxmation semantics, when it comes to automating *planning*, one potential source of inefficiency is in the fact that the ground instance of the program becomes very large due to several factors:

- The recursive (even in limited form) definition of modal formulae in Chapter 5.1.2. The more agents and fluents in the domain, the larger the collection of modal formulae that make up the ground instance of the program.

- Together with the number of modal formulae of interest in the domain, the necessary definiton of the corresponding entailment relation from Chapter 5.1.2.2 also greatly expands the ground instance of the program within a single time step.

- When attempting to do temporal projection or planning, the transition function, which depends on a complex definition of a state from Chapter 5.1.2.1, individual update models for action occurrences in Chapter 5.2.1, and the update execution operator in Chapter 5.2.2, compounds issue even more, making automated planning for even

small domains difficult.

Each of the issues presented above has to do with the *size of the resulting ground program*. As part of my future research, I hope to investigate two areas to work around this. The first, is in automatic program optimization through program rewriting, and the second is in solvers which do not require a priori grounding.

With program rewriting it may be possible to discover ways of transforming or simplifying the abstract syntax tree of the program while preserving equivalence but reducing the size of the resulting ground program. Recent work has been done in this area [20, 28] and bears further investigation. In addition, the kinds of programs used in this work may serve as benchmark test cases. Another avenue of research has been partly touched on by the use of the Prolog programming language in [9], but may be extended into solving through multiple logic programming paradigms or in moving to a new approach for answer-set solving in general. Current answer set solvers such as `clingo` and `dlv` operate on a ground instance of the input program, which as is the case with the programs presented in Chapter 5 grow to be very large, making computation difficult. It may be possible that an alternative algorithm which does not require an a priori grounding could reduce the computational cost associated with these programs.

REFERENCES

[1] Guillaume Aucher and François Schwarzentruber. On the Complexity of Dynamic Epistemic Logic. *CoRR*, abs/1310.6406, 2013.

[2] Marcello Balduccini and Michael Gelfond. The AAA Architecture: An Overview. In *AAAI Spring Symposium 2008 on Architectures for Intelligent Theory-Based Agents*, 2008.

[3] Marcello Balduccini, Michael Gelfond, Richard Watson, and Monica Nogueira. The USA-Advisor: A Case Study in Answer Set Planning. In Thomas Eiter, Wolfgang Faber, and Miroslaw Truszczyński, editors, *Logic Programming and Nonmotonic Reasoning*, volume 2173 of *Lecture Notes in Computer Science*, pages 439–442. Springer Berlin Heidelberg, 2001.

[4] Alexandru Baltag and Lawrence S. Moss. Logics for Epistemic Programs. *Synthese*, 139(2):165–224, 2004.

[5] Chitta Baral. Reasoning About Actions: Non-deterministic Effects, Constraints, and Qualification. In *Proceedings of the 14th International Joint Conferences on Artificial Intelligence 95*, IJCAI '95, pages 2017–2023. Morgan Kaufmann, 1995.

[6] Chitta Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, 2003.

[7] Chitta Baral and Gregory Gelfond. On Representing Actions in Multi-Agent Domains. In Marcello Balduccini and Tran Cao Son, editors, *Proceedings of the Symposium on Constructive Mathematics*, pages 213–232. Springer, 2011.

[8] Chitta Baral, Gregory Gelfond, Michael Gelfond, and Richard Scherl. Textual Inference by Combining Multiple Logic Programming Paradigms. In *AAAI'05 Workshop on Inference for Textual Question Answering*, AAAI '05, 2005.

[9] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. 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, 2010.

[10] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. Using Answer Set Programming to Model Multi-Agent Scenarios Involving Agents' Knowledge About Other's Knowledge. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '10, pages 259–266, 2010.

[11] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. An Action Language for Reasoning about Beliefs in Multi-Agent Domains. In *Proceedings of the 14th International Workshop on Non-Monotonic Reasoning*, 2012.

[12] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. Reasoning about the Beliefs of Agents in Multi-Agent Domains in the Presence of State Constraints: The Action Language mAL. In João Leite, Tran Cao Son, Paolo Torroni, Leon Torre, and Stefan Woltran, editors, *Computational Logic in Multi-Agent Systems*, volume 8143 of *Lecture Notes in Computer Science*, pages 290–306. Springer Berlin Heidelberg, 2013.

[13] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. Finitary S5-Theories. In Eduardo Fermé and João Leite, editors, *Logics in Artificial Intelligence*, volume 8761 of *Lecture Notes in Computer Science*, pages 239–252. Springer International Publishing, 2014.

[14] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. An Action Language for Multi-Agent Domains: Foundations. *CoRR*, abs/1511.01960, 2015.

[15] Chitta Baral, Gregory Gelfond, Enrico Pontelli, and Tran Cao Son. Multi-Agent Action Modeling through Action Sequences and Perspective Fluents. In *Proceedings of the AAAI Spring Symposium on Common Sense Reasoning 2015*, AAAI '15, 2015.

[16] Chitta Baral and Michael Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 31:85–117, 1997.

[17] Chitta Baral, Michael Gelfond, and Olga Kosheleva. Approximating General Logic Programs. In *Proceedings of the 1993 international symposium on Logic Programming*, pages 181–198, 1993.

[18] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic Reasoning With Answer Sets. In Vladimir Lifschitz and Ilkka Niemelä, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 21–33, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[19] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[20] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Challenges in answer set solving. In Marcello Balduccini and Tran Cao Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, pages 74–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[21] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012.

[22] Gregory Gelfond. A Declarative Framework for Modeling Multi-Agent Systems. Master's thesis, Texas Tech University, 2007.

[23] Michael Gelfond. *Handbook of Knowledge Representation – Chapter 7: Answer Sets*. Elsevier, 2007.

[24] Michael Gelfond and Yulia Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents*. Cambridge University Press, 2014.

[25] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press, 1988.

[26] Michael Gelfond and Vladimir Lifschitz. Representing Action and Change by Logic Programs. *Journal of Logic Programming*, 17:301–322, 1993.

[27] Michael Gelfond and Vladimir Lifschitz. Action Languages. *Electronic Transactions on AI*, 3, 1998.

[28] Amelia Harrison and Yuliya Lierler. First-Order Modular Logic Programs and their Conservative Extensions. *Theory and Practice of Logic programming, 32nd Int'l. Conference on Logic Programming (ICLP) Special Issue*, 2016.

[29] Patrick Kahl. *Refining the Semantics for Epistemic Logic Programs*. PhD thesis, Texas Tech University, 2014.

[30] Mykel J. Kochenderfer, Christopher Amato, Girish Chowdhary, Jonathan P. How, Hayley J. Davison Reynolds, Jason R. Thornton, Pedro A. Torres-Carrasquillo, N. Kemal Üre, and John Vian. *Decision Making Under Uncertainty: Theory and Application*. The MIT Press, 1st edition, 2015.

[31] Vladimir Lifschitz. What is Answer Set Programming? In *AAAI*, pages 1594–1597, 2008.

[32] Fangzhen Lin. Embracing Causality in Specifying the Indirect Effects of Actions. In *Proceedings of the 14th International Joint Conferences on Artificial Intelligence*, IJCAI '95. Morgan Kaufmann, 1995.

[33] Fangzhen Lin and Yoav Shoham. Provably correct theories of action. *Journal of the ACM*, 42(2):293–320, 1995.

[34] Norman McCain and Hudson Turner. A Causal Theory of Ramifications and Qualifications. In *Proceedings of the 14th International Joint Conferences on Artificial Intelligence*, IJCAI '95. Morgan Kaufmann, 1995.

[35] John McCarthy. Programs with common sense. In *Semantic Information Processing*, pages 403–418. MIT Press, 1959.

[36] John McCarthy. Mathematical Logic in Artificial Intelligence. *Daedalus*, 117(1):297–311, 1988.

[37] John McCarthy. *Formalizing Common Sense – Papers by John McCarthy*. Ablex Publishing Corporation, 1990.

[38] John McCarthy. Elaboration Tolerance. In *Common Sense*, volume 98, 1998.

[39] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[40] Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian Pearce, and Liz Sonenberg. Planning Over Multi-Agent Epistemic States: A Classical Planning Approach. In *In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI '15, 2015.

[41] Raz Nissim and Ronen Brafman. Distributed Heuristic Forward Search for Multi-Agent Planning. *Journal of Artificial Intelligence Research*, 51(1):293–332, 2014.

[42] Monica Nogueira, Marcello Balduccini, Michael Gelfond, Richard Watson, and Matthew Barry. An A-Prolog Decision Support System for the Space Shuttle. In I. V. Ramakrishnan, editor, *Practical Aspects of Declarative Languages*, volume 1990 of *Lecture Notes in Computer Science*, pages 169–183. Springer Berlin Heidelberg, 2001.

[43] Johan van Benthem, Jan van Eijck, and Barteld Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

[44] Han van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Springer, 2008.

[45] Shiqi Zhang, Mohan Sridharan, Michael Gelfond, and Jeremy Wyatt. Towards an Architecture for Knowledge Representation and Reasoning in Robotics. In Michael Beetz, Benjamin Johnston, and Mary-Anne Williams, editors, *Social Robotics*, volume 8755 of *Lecture Notes in Computer Science*, pages 400–410. Springer International Publishing, 2014.

APPENDIX A

THEOREMS AND PROOFS

A.1 Proofs for Theorems in Chapter 3

Theorem 3.1. *Let $\mathcal{D}$ be a multi-agent domain with the signature, $\Sigma = (\mathcal{AG}, \mathcal{F}, \mathcal{A})$, $\mathcal{I}$ be a consistent and definite initial state description, and Δ be an action description of $m\mathcal{A}+$. There exists a unique Kripke world, (M, σ), where $|M.W| \leq 2^{|\mathcal{F}|}$, which satisfies the* **S5** *axioms, such that every Kripke model of $(\mathcal{I}, \Delta)$ is equivalent to (M, σ).*

Proof. Let us consider two arbitrary **S5**-states (M_0, ω_0) and (M_1, ω_1) satisfying $\mathcal{I}$. From Lemma A.6 we can assume that both M_0 and M_1 are in reduced form; this guarantees that the number of points in M_0 and M_1 is bound by the number of interpretations which is $2^{|\mathcal{F}|}$. Furthermore, we can assume that M_0 and M_1 have the same points (since each interpretation in M_0 appears in M_1 and vice versa, and no interpretation can be associated with two distinct points as the states are in reduced form).

For the sake of simplicity, let us assume that $M_0.W = M_1.W$; let us also assume by contradiction, that there are two points $\mu, \nu \in M_0.W$ and $(\mu, \nu) \in M_0.R_\alpha$ and $(\mu, \nu) \notin M_1.R_\alpha$. Let $\varphi = \neg state(\nu)$. Because of the construction, we can see that $(M_1, \mu) \models \varphi$. Since the two states are initial states for $\mathcal{I}$, this means that they are models of one of the following formulae: $\mathcal{CB}_\alpha\varphi$ or $\mathcal{CB}_\alpha\varphi \vee \mathcal{B}_\alpha\neg\varphi$. On the other hand, since $(\mu, \nu) \in M_0.R_\alpha$, it is easy to see that $(M_0, \mu) \not\models \mathcal{B}_\alpha\varphi$ and $(M_0, \mu) \not\models \neg\varphi$. This contradicts the fact that both (M_0, ω_0) and (M_1, ω_1) are models of $\mathcal{I}$. □

Theorem 3.2. *Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be a sensing action that is executable in σ. Let $(M', \omega') \in \sigma \otimes U_s(\sigma, a)$. It holds that:*

- $\forall f \in \{f \mid [a \textbf{ determines } f] \in \Delta\}$, $(M', \omega') \models C_{f(a,\sigma)}\lambda$ *iff* $(M, \omega) \models \lambda$ *where* $\lambda \in \{f, \neg f\}$

- $\forall f \in \{f \mid [a \textbf{ determines } f] \in \Delta\}$, $(M', \omega') \vDash C_{p(a,\sigma)}(C_{f(a,\sigma)}f \vee C_{f(a,\sigma)}\neg f)$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \vDash \mathcal{B}_\alpha \lambda$ *iff* $(M, \omega) \vDash \mathcal{B}_\alpha \lambda$

Proof. The proof is broken up into three parts, one for each assertion made by the theorem.

1. $\forall f \in \{f \mid [a \textbf{ determines } f] \in \Delta\}$, $(M', \omega') \vDash C_{f(a,\sigma)}\lambda$ iff $(M, \omega) \vDash \lambda$ where $\lambda \in \{f, \neg f\}$
   a) *Left-to-right*: $(M', \omega') \vDash C_{f(a,\sigma)}\lambda \implies (M, \omega) \vDash \lambda$ where $\lambda \in \{f, \neg f\}$
      i. Let $G = f(a, \sigma)$, and $(M', \omega') \vDash C_{f(a,\sigma)}\lambda$. It immediately follows that $(M', \sigma) \vDash \lambda$ for every σ that is G-reachable from ω'.
      ii. From Definitions (3.12) and (3.15), this may only be true if and only if for every τ such that τ is G-reachable from ω, $(M, \tau) \vDash \lambda$.
      iii. It necessarily follows that $(M, \omega) \vDash \lambda$, and therefore the implication holds.
   b) *Right-to-left*: $(M, \omega) \vDash \lambda \implies (M', \omega') \vDash C_{f(a,\sigma)}\lambda$ where $\lambda \in \{f, \neg f\}$
      i. Let $G = f(a, \sigma)$ and $(M, \omega) \vDash \lambda$.
      ii. From Definitions (3.12) and (3.15), it must be the case that $(M', \omega') \vDash \lambda$. Furthermore, it also follows that for every τ that is G-reachable from ω' in M', $(M', tau) \vDash \lambda$.
      iii. Hence, $(M', \omega') \vDash C_{f(a,\sigma)}\lambda$, and the implication holds.
2. $\forall f \in \{f \mid [a \textbf{ determines } f] \in \Delta\}$, $(M', \omega') \vDash C_{p(a,\sigma)}(C_{f(a,\sigma)}f \vee C_{f(a,\sigma)}\neg f)$
   a) We are given that $(M', \omega') \in \sigma \otimes U_s(\sigma, a)$. By definition, the following must hold (via Definitions (3.12) and (3.15):
      i. ω' has the form (ω, ε_p) or (ω, ε_n), depending on the interpreted value of f.

ii. The only points reachable from ω' by agents in $p(a,\sigma)$, have the form (σ,ε_p), or (τ,ε_n), where σ and τ are points in M where $\sigma.\pi(f)=\top$ and $\tau.\pi(f)=\bot$.

iii. (ω,ε_p) is reachable from itself by agents in $p(a,\sigma)$ or $f(a,\sigma)$. Similarly for (ω,ε_n).

iv. Lastly, (ω,ε_n) is reachable from (ω,ε_p) by agents in $p(a,\sigma)$ and vice versa.

b) Let $G_1 = p(a,\sigma)$ and $G_2 = f(a,\sigma)$.

c) A consequence of (2(a)ii) and (2(a)iii) is that (ω,ε_p) and (ω,ε_n) are G_1-reachable from (ω,ε_p).

d) Every point that is G_2-reachable from (ω,ε_p) must have the form (τ,ε_p) where τ is some point in M reachable from ω by agents in G_2. Similarly, Every point that is G_2-reachable from (ω,ε_n) must have the form (τ,ε_n) where τ is some point in M reachable from ω by agents in G_2.

e) From (2c), (2d), and (A.2), every point that is G_2-reachable from ω' must either satisfy f or $\neg f$, and hence satisfy $(C_{f(a,\sigma)}f \vee C_{f(a,\sigma)}\neg f)$.

f) From (2e), (2c) and (A.2) it must be the case that every point that is G_1-reachable from ω' satisfies $C_{f(a,\sigma)}\neg f)$, and hence it must be the case that $(M',\omega') \models C_{p(a,\sigma)}(C_{f(a,\sigma)}f \vee C_{f(a,\sigma)}\neg f)$.

3. $\forall\alpha \in o(a,\sigma), \forall\lambda, (M',\omega') \models \mathcal{B}_\alpha\lambda$ iff $(M,\omega) \models \mathcal{B}_\alpha\lambda$

a) *Left-to-right*: $(M',\omega') \models \mathcal{B}_\alpha\lambda \implies (M,\omega) \models \mathcal{B}_\alpha\lambda$

i. Let $\alpha \in o(a,\sigma)$, and $(M',\omega') \models \mathcal{B}_\alpha\lambda$. It immediately follows that $(M',\sigma) \models \lambda$ for every point σ that is reachable from ω' in M' by agent α's accessibility relation.

ii. From Definitions (3.12) and (3.15), it must be the case that:

A. $\omega' = (\omega, \varepsilon_i)$ where ω is the reference point in M.

B. Every point σ reachable from ω' must have the form (ω, ε_i) where ω is a point reachable from the reference point in M by agent α's accessibility relation.

iii. Consequently, every point ω reachable from the reference point in in M by agent α's accessibility relation must satisfy λ.

iv. Hence by definition, $(M, \omega) \models \mathcal{B}_\alpha \lambda$.

b) *Right-to-left*: $(M, \omega) \models \mathcal{B}_\alpha \lambda \implies (M', \omega') \models \mathcal{B}_\alpha \lambda$

i. Let $\alpha \in o(a, \sigma)$, and $(M, \omega) \models \mathcal{B}_\alpha \lambda$. It follows by definition that every point σ reachable from ω in M by agent α's accessibility relation must satisfy λ.

ii. Now consider any point τ in M' that is reachable from ω' by agent α's accessibility relation. From Definitions (3.12) and (3.15), τ must have the form (σ, ε_i) for some corresponding point σ in M. Furthermore, σ is reachable from the reference point in M by agent α's accessibility relation.

iii. Furthermore, Definitions (3.12) and (3.15) tell us that τ must satisfy λ as well.

iv. As a consequence, it must be the case that every point τ reachable from ω' in M' by agent α's accessibility relation must satisfy λ.

v. Hence, by definition of the entailment relation, $M', \omega') \models \mathcal{B}_\alpha \lambda$.

□

Theorem 3.3. *Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an announcement action that is executable in σ that*

is governed by the announcement axiom $[a$ ***announces*** $\varphi] \in \Delta$. *Let* $(M', \omega') \in \sigma \otimes U_a(\sigma, a)$. *It holds that:*

- $(M', \omega') \vDash C_{f(a,\sigma)}\varphi$
- $(M', \omega') \vDash C_{p(a,\sigma)}(C_{f(a,\sigma)}\varphi \vee C_{f(a,\sigma)}\neg\varphi)$
- $\forall\alpha \in o(a, \sigma)$, $\forall\lambda$, $(M', \omega') \vDash \mathcal{B}_\alpha\lambda$ *iff* $(M, \omega) \vDash \mathcal{B}_\alpha\lambda$

Proof. The proof is broken up into three parts, one for each assertion made by the theorem.

1. $(M', \omega') \vDash C_{f(a,\sigma)}\varphi$
    a) We are given that $(M', \omega') \in \sigma \otimes U_a(\sigma, a)$, and that a is executable in σ, and is governed by an announcement axiom of the form $[a$ ***announces*** $\varphi]$. From definitions (3.13) and (3.15), the following must hold:
        i. $\sigma \vDash \varphi$
        ii. $(M', \omega') \vDash \varphi$
        iii. Every point in M' that is reachable from ω') by members of $f(a, \sigma)$ must satisfy φ.
    b) Consequently, by definition of the entailment relation and (A.2), $(M', \omega') \vDash C_{f(a,\sigma)}\varphi$.
2. $(M', \omega') \vDash C_{p(a,\sigma)}(C_{f(a,\sigma)}\varphi \vee C_{f(a,\sigma)}\neg\varphi)$
    a) We are given that $(M', \omega') \in \sigma \otimes U_a(\sigma, a)$. By definition, the following must hold (via Definitions (3.13) and (3.15):
        i. ω' has the form (ω, ε_p) or (ω, ε_n), depending on the interpreted value of φ.

ii. The only points reachable from ω' by agents in $p(a,\sigma)$, have the form (σ,ε_p), or (τ,ε_n), where σ and τ are points in M where $\sigma.\pi(\varphi)=\top$ and $\tau.\pi(\varphi)=\bot$.

iii. (ω,ε_p) is reachable from itself by agents in $p(a,\sigma)$ or $f(a,\sigma)$. Similarly for (ω,ε_n).

iv. Lastly, (ω,ε_n) is reachable from (ω,ε_p) by agents in $p(a,\sigma)$ and vice versa.

b) Let $G_1 = p(a,\sigma)$ and $G_2 = f(a,\sigma)$.

c) A consequence of (2(a)ii) and (2(a)iii) is that (ω,ε_p) and (ω,ε_n) are G_1-reachable from (ω,ε_p).

d) Every point that is G_2-reachable from (ω,ε_p) must have the form (τ,ε_p) where τ is some point in M reachable from ω by agents in G_2. Similarly, Every point that is G_2-reachable from (ω,ε_n) must have the form (τ,ε_n) where τ is some point in M reachable from ω by agents in G_2.

e) From (2c), (2d), and (A.2), every point that is G_2-reachable from ω' must either satisfy f or $\neg f$, and hence satisfy $(\mathcal{C}_{f(a,\sigma)}f \vee \mathcal{C}_{f(a,\sigma)}\neg f)$.

f) From (2e), (2c) and (A.2) it must be the case that every point that is G_1-reachable from ω' satisfies $\mathcal{C}_{f(a,\sigma)}\neg f)$, and hence it must be the case that $(M',\omega') \vDash \mathcal{C}_{p(a,\sigma)}(\mathcal{C}_{f(a,\sigma)}f \vee \mathcal{C}_{f(a,\sigma)}\neg f)$.

3. $\forall\alpha \in o(a,\sigma), \forall\lambda, (M',\omega') \vDash \mathcal{B}_\alpha\lambda$ iff $(M,\omega) \vDash \mathcal{B}_\alpha\lambda$

a) *Left-to-right*: $(M',\omega') \vDash \mathcal{B}_\alpha\lambda \implies (M,\omega) \vDash \mathcal{B}_\alpha\lambda$

i. Let $\alpha \in o(a,\sigma)$, and $(M',\omega') \vDash \mathcal{B}_\alpha\lambda$. It immediately follows that $(M',\sigma) \vDash \lambda$ for every point σ that is reachable from ω' in M' by agent α's accessibility relation.

ii. From Definitions (3.13) and (3.15), it must be the case that:

A. $\omega' = (\omega, \varepsilon_i)$ where ω is the reference point in M.

B. Every point σ reachable from ω' must have the form (ω, ε_i) where ω is a point reachable from the reference point in M by agent α's accessibility relation.

iii. Consequently, every point ω reachable from the reference point in in M by agent α's accessibility relation must satisfy λ.

iv. Hence by definition, $(M, \omega) \models \mathcal{B}_\alpha \lambda$.

b) *Right-to-left*: $(M, \omega) \models \mathcal{B}_\alpha \lambda \implies (M', \omega') \models \mathcal{B}_\alpha \lambda$

i. Let $\alpha \in o(a, \sigma)$, and $(M, \omega) \models \mathcal{B}_\alpha \lambda$. It follows by definition that every point σ reachable from ω in M by agent α's accessibility relation must satisfy λ.

ii. Now consider any point τ in M' that is reachable from ω' by agent α's accessibility relation. From Definitions (3.13) and (3.15), τ must have the form (σ, ε_i) for some corresponding point σ in M. Furthermore, σ is reachable from the reference point in M by agent α's accessibility relation.

iii. Furthermore, Definitions (3.13) and (3.15) tell us that τ must satisfy λ as well.

iv. As a consequence, it must be the case that every point τ reachable from ω' in M' by agent α's accessibility relation must satisfy λ.

v. Hence, by definition of the entailment relation, $M', \omega') \models \mathcal{B}_\alpha \lambda$.

□

Theorem 3.4. *Let Δ be a consistent action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an ontic action that is executable in σ. Let $(M', \omega') \in \sigma \otimes U_s(\sigma, a)$. It holds that:*

- $\forall \alpha \in f(a, \sigma)$, *dynamic causal law* $[a$ ***causes*** λ ***if*** $\phi] \in \Delta$, *and* $\omega_1, \omega_2 \in M.W$*:*
  - $(\varepsilon_p, \omega_1) \vDash \lambda$ *if* $(M, \omega_1) \vDash \phi$
  - $(\omega_1, \omega_2) \in M.R_\alpha$ *iff* $((\varepsilon_p, \omega_1), (\varepsilon_p, \omega_2)) \in M'.R_\alpha$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \vDash \mathcal{B}_\alpha \lambda$ *iff* $(M, \omega) \vDash \mathcal{B}_\alpha \lambda$

Proof. The proof is broken up into parts, one for each assertion made by the theorem:

1. $\forall \alpha \in f(a, \sigma)$, dynamic causal law $[a$ ***causes*** λ ***if*** $\phi] \in \Delta$, and $\omega_1, \omega_2 \in M.W$:
   - $(\varepsilon_p, \omega_1) \vDash \lambda$ if $(M, \omega_1) \vDash \phi$
     a) We are given that $(\varepsilon_p, \omega_1)$ is a point in $\sigma \otimes U_o(\sigma, a)$.
     b) From (1a) and Definitions (3.14) and (3.15), $(\varepsilon_p, \omega_1) \vDash \lambda$ may only be true if a substitution of the form $\neg\lambda \rightarrow \lambda$ was applied as part of the application of the $\otimes$ operator.
     c) (1b) may only hold if $(M, \omega_1) \vDash \phi$
     d) Hence, $(\varepsilon_p, \omega_1) \vDash \lambda$ if $(M, \omega_1) \vDash \phi$
   - $(\omega_1, \omega_2) \in M.R_\alpha$ iff $((\varepsilon_p, \omega_1), (\varepsilon_p, \omega_2)) \in M'.R_\alpha$
     - This assertion follows immediately from Definitions (3.14) and (3.15).
2. $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \vDash \mathcal{B}_\alpha \lambda$ iff $(M, \omega) \vDash \mathcal{B}_\alpha \lambda$
   a) *Left-to-right*: $(M', \omega') \vDash \mathcal{B}_\alpha \lambda \implies (M, \omega) \vDash \mathcal{B}_\alpha \lambda$
      i. Let $\alpha \in o(a, \sigma)$, and $(M', \omega') \vDash \mathcal{B}_\alpha \lambda$. It immediately follows that $(M', \sigma) \vDash \lambda$ for every point σ that is reachable from ω' in M' by agent α's accessibility relation.
      ii. From Definitions (3.14) and (3.15), it must be the case that:

A. $\omega' = (\omega, \varepsilon_i)$ where ω is the reference point in M.

B. Every point σ reachable from ω' must have the form (ω, ε_i) where ω is a point reachable from the reference point in M by agent α's accessibility relation.

iii. Consequently, every point ω reachable from the reference point in in M by agent α's accessibility relation must satisfy λ.

iv. Hence by definition, $(M, \omega) \vDash \mathcal{B}_\alpha \lambda$.

b) *Right-to-left*: $(M, \omega) \vDash \mathcal{B}_\alpha \lambda \implies (M', \omega') \vDash \mathcal{B}_\alpha \lambda$

i. Let $\alpha \in o(a, \sigma)$, and $(M, \omega) \vDash \mathcal{B}_\alpha \lambda$. It follows by definition that every point σ reachable from ω in M by agent α's accessibility relation must satisfy λ.

ii. Now consider any point τ in M' that is reachable from ω' by agent α's accessibility relation. From Definitions (3.14) and (3.15), τ must have the form (σ, ε_i) for some corresponding point σ in M. Furthermore, σ is reachable from the reference point in M by agent α's accessibility relation.

iii. Furthermore, Definitions (3.14) and (3.15) tell us that τ must satisfy λ as well.

iv. As a consequence, it must be the case that every point τ reachable from ω' in M' by agent α's accessibility relation must satisfy λ.

v. Hence, by definition of the entailment relation, $M', \omega') \vDash \mathcal{B}_\alpha \lambda$.

□

A.2 Proofs for Theorems in Chapter 4

Theorem 4.1. *Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be an ontic action that is executable in σ. Let $(M', \omega') \in \Phi_\Delta(\sigma, a)$. It holds that:*

- $\forall \alpha \in f(a, \sigma)$*, and dynamic causal law* $[a$ ***causes*** λ ***if*** $\phi] \in \Delta$*, if* $(M, \omega) \models \mathcal{B}_\alpha \phi$*, then* $(M', \omega') \models \mathcal{B}_\alpha \lambda$
- $\forall \alpha \in f(a, \sigma)$*, and state constraint* $[\lambda$ ***if*** $\phi] \in \Delta$*,* $(M, \omega) \models \mathcal{B}_\alpha(\phi \implies \lambda)$
- $\forall \alpha \in o(a, \sigma)$*,* $\forall \lambda$*,* $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *if and only if* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Proof. The proof is broken up into parts, one for each assertion made by the theorem:

1. $\forall \alpha \in f(a, \sigma)$, and dynamic causal law $[a$ ***causes*** λ ***if*** $\phi] \in \Delta$, if $(M, \omega) \models \mathcal{B}_\alpha \phi$, then $(M', \omega') \models \mathcal{B}_\alpha \lambda$.
   a) We are given that $\alpha \in f(a, \sigma)$ and that $(M, \omega) \models \mathcal{B}_\alpha \phi$.
   b) From (1a) and Definition 2.12, it must be the case that every point reachable from ω in M according to the accessibility relation for agent α satisfies ϕ.
   c) From (1b) and Definitions 4.12 and 4.9, it must be the case that from every point in the pointed frame $(F, (\omega, \varepsilon_p))$ reachable from (ω, ε_p), must satisfy ϕ.
   d) From (1c), and Definitions 4.15 and 4.16, it must be the case that every point reachable from ω' in M' by agent α's accessibility relation satisfies λ.
   e) Consequently, from (1d) and Definition 2.12, $(M', \omega') \models \mathcal{B}_\alpha \lambda$.
   f) Hence, if $(M, \omega) \models \mathcal{B}_\alpha \phi$, then $(M', \omega') \models \mathcal{B}_\alpha \lambda$.
2. $\forall \alpha \in f(a, \sigma)$, and state constraint $[\lambda$ ***if*** $\phi] \in \Delta$, $(M, \omega) \models \mathcal{B}_\alpha(\phi \implies \lambda)$.

a) We are given that $\sigma = (M, \omega)$ is a state in the transition diagram defined by Δ.

b) Suppose that $(M, \omega) \models \mathcal{B}_\alpha \neg\phi$. In this case, the implication $\phi \rightarrow \lambda$ is vacuously true.

c) Suppose now that $(M, \omega) \models \mathcal{B}_\alpha \phi$.

d) From (2c) and Definition 2.12, it must be the case that every point reachable from ω in M according to agent α's accessibility relation satisfies ϕ.

e) From (2a) and the definition of a state in Δ, it must be the case that every point of M correspond to complete consistent sets of fluent literals closed under the state constraints of Δ.

f) As a consequence of (2e), it must be the case every point reachable from ω in M according to agent α's accessibility relation satisfies λ.

g) Hence, from (2f), the implication $\phi \rightarrow \lambda$ is satisfied, and consequently by virtue of Definition 2.12, it must be the case that $(M, \omega) \models \mathcal{B}_\alpha(\phi \rightarrow \lambda)$.

3. $\forall \alpha \in o(a, \sigma), \forall \lambda, (M', \omega') \models \mathcal{B}_\alpha \lambda$ if and only if $(M, \omega) \models \mathcal{B}_\alpha \lambda$.

a) *Left-to-right*: $(M', \omega') \models \mathcal{B}_\alpha \lambda \implies (M, \omega) \models \mathcal{B}_\alpha \lambda$.

i. Let $\alpha \in o(a, \sigma)$, and $(M', \omega') \models \mathcal{B}_\alpha \lambda$. It immediately follows that $(M', \tau) \models \lambda$ for every point τ that is reachable from ω' in M' by agent α's accessibility relation.

ii. From (3(a)i) and Definition 4.18 it must be the case that every such point τ is obtained by the application of the *scenario expansion* of a point (ρ, ε_i), where ρ is a point reachable from ω in M according to agent α's accessibility relation.

iii. From Definitions 4.15 and 4.14, it is clear that the interpration functions of every point τ match those of their corresponding points ρ in M. Con-

sequently, every point reachable from ω in M by agent α's accessibility relation must satisfy λ.

iv. Hence, from (3(a)iii) and Definition 2.12, it must be the case that $(M, \omega) \models \mathcal{B}_\alpha \lambda$.

b) *Right-to-left*: $(M, \omega) \models \mathcal{B}_\alpha \lambda \implies (M', \omega') \models \mathcal{B}_\alpha \lambda$.

i. Let $\alpha \in o(a, \sigma)$, and $(M, \omega) \models \mathcal{B}_\alpha \lambda$. It follows by definition that every point σ reachable from ω in M by agent α's accessibility relation must satisfy λ.

ii. Consequently, by Definitions 4.14, 4.15 and 4.18, it must be the case that every point reachable from ω' in M' satisfy λ, by virtue of those points being derived from the scenario expansion of points of the form (ρ, ε_i), where ρ is a point reachable from ω in M according to agent α's accessibility relation.

iii. Hence from (3(b)ii) it must be the case by Definition 2.12, that $(M', \omega') \models \mathcal{B}_\alpha \lambda$.

□

Theorem 4.2. *Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be a sensing action that is executable in σ and described by the sensing axiom* [*a* ***determines*** *f*] $\in \Delta$. *Let $(M', \omega') \in \Phi_\Delta(\sigma, a)$. It holds that:*

- $(M', \omega') \models C_{f(a,\sigma)} \lambda$ *if and only if* $(M, \omega) \models \lambda$ *where* $\lambda \in \{f, \neg f\}$
- $(M', \omega') \models C_{p(a,\sigma)}(C_{f(a,\sigma)} f \vee C_{f(a,\sigma)} \neg f)$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *if and only if* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Proof. The proof is broken up into parts, one for each assertion made by the theorem:

1. $(M', \omega') \vDash C_{f(a,\sigma)}\lambda$ if and only if $(M, \omega) \vDash \lambda$ where $\lambda \in \{f, \neg f\}$:

   a) We are given that $(M', \omega') \vDash C_{f(a,\sigma)}\lambda$.

   b) It immediately follows that for every point τ reachable from ω' in M' by agent α's accessibility relation must satisfy λ.

   c) From (1b) and Definitions 4.10, 4.12, and 4.18, it must be the case that every such point τ was derived by the scenario expansion of some point from $\mathbf{Eu}(\sigma, \mathcal{E}_s(\sigma, a))$.

   d) From the same set of definitions as in (1c), we know that a point may only belong to $\mathbf{Eu}(\sigma, \mathcal{E}_s(\sigma, a))$, if it has the form (ρ, ε_p) or (ρ, ε_n) depending on the value of λ.

   e) Via Definition 4.12, points of the form (ρ, ε_p) may only belong to $\mathbf{Eu}(\sigma, \mathcal{E}_s(\sigma, a))$ if the $M.\pi(\rho)(\lambda) = \top$, and similarly for the negative case. .

   f) Consequently, it must be the case $(M, \omega) \vDash \lambda$.

2. $(M', \omega') \vDash C_{p(a,\sigma)}(C_{f(a,\sigma)}f \vee C_{f(a,\sigma)}\neg f)$:

   a) We are given via Definition 4.18 that (M', ω') belongs to $\mathbf{Ou}_{\Delta}(\sigma, \mathbf{Eu}(\sigma, \mathcal{E}_s(\sigma, a)))$. Consequently, the following must be true:

      i. ω' has the form (ω, ε_p) or (ω, ε_n), depending on the interpreted value of f.

      ii. The only points reachable from ω' by agents in $p(a, \sigma)$, were obtained by scenario expansions of those having the form (σ, ε_p), or (τ, ε_n), where σ and τ are points in M where $\sigma.\pi(f) = \top$ and $\tau.\pi(f) = \bot$.

      iii. Those obtained from (ω, ε_p) are reachable from themselves by agents in $p(a, \sigma)$ or $f(a, \sigma)$. Similarly for those corresponding to (ω, ε_n).

iv. Lastly, those obtained by expanding (ω, ε_n) are reachable from those corresponding to (ω, ε_p) by agents in $p(a, \sigma)$ and vice versa.

b) Let $G_1 = p(a, \sigma)$ and $G_2 = f(a, \sigma)$.

c) A consequence of (2(a)ii) and (2(a)iii) is that (ω, ε_p) and (ω, ε_n) are G_1-reachable from (ω, ε_p).

d) Every point that is G_2-reachable from (ω, ε_p) must have the form (τ, ε_p) where τ is some point in M reachable from ω by agents in G_2. Similarly, Every point that is G_2-reachable from (ω, ε_n) must have the form (τ, ε_n) where τ is some point in M reachable from ω by agents in G_2.

e) From (2c), (2d), and (A.2), every point that is G_2-reachable from ω' must either satisfy f or $\neg f$, and hence satisfy $(C_{f(a,\sigma)} f \vee C_{f(a,\sigma)} \neg f)$.

f) From (2e), (2c) and (A.2) it must be the case that every point that is G_1-reachable from ω' satisfies $C_{f(a,\sigma)} \neg f)$, and hence it must be the case that $(M', \omega') \models C_{p(a,\sigma)}(C_{f(a,\sigma)} f \vee C_{f(a,\sigma)} \neg f)$.

3. $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \models \mathcal{B}_\alpha \lambda$ if and only if $(M, \omega) \models \mathcal{B}_\alpha \lambda$:

a) The proof for this asssertion directly mirrors that of (3) for Theorem 4.1.

□

Theorem 4.3. *Let Δ be a consistent action description of $m\mathcal{AL}$, $\sigma = (M, \omega)$ be a state in the transition diagram defined by Δ, and a be a communication action that is executable in σ and described by the sensing axiom* [*a* ***communicates*** *φ*] $\in \Delta$. *Let $(M', \omega') \in \Phi_\Delta(\sigma, a)$. It holds that:*

- $(M', \omega') \models C_{f(a,\sigma)} \varphi$

- $(M', \omega') \models C_{p(a,\sigma)}(C_{f(a,\sigma)}\varphi \vee C_{f(a,\sigma)}\neg\varphi)$
- $\forall \alpha \in o(a, \sigma)$, $\forall \lambda$, $(M', \omega') \models \mathcal{B}_\alpha \lambda$ *if and only if* $(M, \omega) \models \mathcal{B}_\alpha \lambda$

Proof. The proof of this theorem directly mirrors that of the proof for Theorem 4.2, changing only the references from the sensing model to the communication model. □

A.3 Proofs for Theorems in Chapter 5

Theorem 5.1. *Let a_1, a_2, and a_3 be defined as follows:*

$$a_1 = declare(m_A \vee m_B \vee m_C)$$

$$a_2 = declare(\neg\mathcal{K}_\alpha m_\alpha \wedge \neg\mathcal{K}_\alpha \neg m_\alpha)$$

$$a_3 = declare(\neg\mathcal{K}_\alpha m_\alpha \wedge \neg\mathcal{K}_\alpha \neg m_\alpha)$$

Π_{mc} has a single answer set corresponding to the solution of the the Classical Muddy Children Problem for $N = K = 3$ described by the trajectory $(\sigma_0, a_1, \sigma_1)$, $(\sigma_1, a_2, \sigma_2)$, $(\sigma_2, a_3, \sigma_3)$.

Before presenting the proof, we introduce some preliminary notation. As detailed in Chapter 5, Π_{mc} is comprised of several components detailed in Sections 5.1.1 through 5.1.5. Rather than referring to these components by the names of their *program texts*, we will use the following notation:

- Π_Σ - denotes the logic program presented in Listing 5.1 which defines the domain signature.
- Π_φ - denotes the logic program presented in Listings 5.2 and 5.3 which defines the set of modal formulae of interest used in the Classical Muddy Children domain.
- $\Pi_\models$ - denotes the logic program presented in Listings 5.4 and 5.5 which defines the entailment relation between Kripke worlds and the formulae defined in Π_φ.

- Π_{σ_0} - denotes the logic program presented in Listings 5.6 and 5.7 which defines the initial state of the Classical Muddy Children domain.

- Π_a - denotes the logic program presented in Listing 5.9 which defines the effects of the actions *declare*.

- Π_H - denotes the logic program presented in Listing 5.10 which defines the history:

$$a_1 = declare(m_A \vee m_B \vee m_C)$$
$$a_2 = declare(\neg \mathcal{K}_\alpha m_\alpha \wedge \neg \mathcal{K}_\alpha \neg m_\alpha)$$
$$a_3 = declare(\neg \mathcal{K}_\alpha m_\alpha \wedge \neg \mathcal{K}_\alpha \neg m_\alpha)$$

  $\langle occurs(a_1, 0), occurs(a_2, 1), occurs(a_3, 2)\rangle$.

Lemma A.1 (Correct Generation of the Initial State)**.** *Let* Π_0 *be the union of* Π_Σ, Π_φ, $\Pi_\models$, *and* Π_{σ_0}. Π_0 *has a single answer set which corresponds to the initial state of the Classical Muddy Children Problem as show in Figure 3.5.*

Proof. This lemma follows from two claims:

1. The answer set of Π_0 correctly defines the points of the Kripke world defining the initial state of the Classicaly Muddy Children Problem *and* their respective interpretation functions.

2. The accessibility relations of the Kripke world defined by the answer set of Π_0 correspond to those shown in Figure 3.5, and consequently entail the formulae specified in Π_{σ_0}.

It is clear from Listing 5.6 that Π_{σ_0}, and consequently Π_0 must satisfy (1), as both the points of the resultant Kripke world and their interpretation functions are given as facts, and hence must belong to the answer set of Π_0.

Regarding (2), we begin by assuming that the module $\Pi_{\models}$ correctly captures the entailment relation for those modal formulae specified in Π_{φ}, as specified by Definition 2.12. Under this assumption, due to the presence of the constraint:

$$\leftarrow \textit{initially}(\varphi) \wedge \textit{not holds}(\varphi, 0).$$

it must be case that the Kripke world defined by the answer set of Π_0 entails all of the formulae outlined in Π_{σ_0} by the collection of facts of the form $\textit{initially}(\varphi)$. Furthermore, we know that the accessibility relations must be *reflexive*, *symmetric*, and *transitive*, by virtue of the rules in Listing 5.6 defining the relation *k_reachable* in Π_{σ_0}.

What remains for us to show is that $\Pi_{\models}$ correctly captures the entailment relation for those modal formulae specified in Π_{φ}. This is accomplished in two steps:

3. First, we note that the definition of the relation *holds* given in Π_{σ_0} is essentially a verbatim transciption of the definition of the entailment relation between a Kripke world and an arbitrary modal formula as given in [19]. Of particular importance is that we note that atoms of the form $\textit{holds}(\varphi, T)$ may only belong to an answer of the program if φ is entailed by the Kripke world defined by that answer set. This follows immediately from the definition of satisfaction.

4. We next note that the rules which comprise $\Pi_{\models}$, correspond to transcriptions of each component of Definition 2.12, with the exception of those rules defining the entailment relation pertaining to formulae containing the modal operator $\mathcal{C}$. Those particular rules themselves are answer-set prolog transcriptions of Lemma A.2. It is clear from the definition of satisfaction that it must be the case that atoms of the form $\textit{entailed\_by}(\varphi, \boldsymbol{P}, \boldsymbol{T})$ may only belong to the answer set of Π_{σ_0} if the entailment relation is satisfied.

It necessarily follows from (1) through (4) that the answer set of Π_0 must correspond to the Kripke world corresponding to the initial state of the Classical Muddy Children Domain as

shown in Figure 3.5. □

Proof. It immediately follows from Lemma A.1 that the for step 0, the answer set of Π_{mc} correctly defines the Kripke world for the initial state of the Classical Muddy Children Domain. What remains is to show that at subsequent step of the trajectory, the Kripke world defined by the atoms of the form $point(P, T)$ and $k\_reachable(P_1, P_2, A, T)$ define the correct matches those of Figures 1.2 through 1.4 (modulo the differences in notation). To do this, we turn our attention towards Π_a and Π_H.

Let us consider the first step of the trajectory: $(\sigma_0, a_1, \sigma_1)$. We must first show that the Kripke world defining σ_1 belongs to the answer set of Π_{mc}. This is an immediate consequence of the satisfaction relation and the rules of:

$$\Pi_a \cup \{occurs(declare(declare(m_A \vee m_B \vee m_C), 0)\}$$

From the satisfaction relation it is clear that the symbol w_8 does not carry over as a point from step 0 to step 1. Similarly for those accessibility relations pertaining to it. Had it carried over, there would be a contradiction due to the introduction of the fact $\neg point(w_8, 1)$ by the satisfaction of the rules of Π_a. Similarly for the remaining steps of the trajectory. □

Theorem 5.2. *Let Δ be an action description of $m\mathcal{A}+$, $\sigma = (M, \omega)$ be a state of the transition diagram defined by Δ, and a be an action. The successor state(s) obtained by performing the action a in the state σ are given as part of the answer-set(s) of the logic program $\Pi_{m\mathcal{A}+}$ given below:*

$$\Pi_\Delta \cup \Pi_\sigma \cup \Pi_\otimes \cup \Pi_h \cup \begin{cases} \Pi_{ontic} & \textit{ontic action} \\ \Pi_{sns} & \textit{sensing action} \\ \Pi_{ann} & \textit{announcement action} \end{cases} \tag{5.1}$$

where $\Pi_\Delta = \Pi_\Sigma \cup \Pi_\mathcal{A} \cup \Pi_{del}$.

Proof. We begin by assuming that Π_h describes the occurence of a *single elementary action from* $\Pi_{\mathcal{A}}$. Let σ' denote a successor state defined by the answer-set(s) of $\Pi_{m\mathcal{A}+}$, and let $U(\Pi_h)$ be the update model corresponding to the action occurrence defined by $\Pi_h \cup \Pi_{ontic} \cup \Pi_{sns} \cup \Pi_{ann}$.

The proof is based on showing the following:

1. A point ω' belongs to the Kripke world which defines σ' if and only if ω' belongs to $(M, \omega) \otimes U(\Pi_h)$.

2. The pair (ω'_1, ω'_2) belongs to the accessibility relation for an agent α in σ' if and only if that pair belongs to the accessibility relation for α in $(M, \omega) \otimes U(\Pi_h)$.

3. A fluent f is true in the interpretation function of ω' if and only if it is true in the interpretation function of $(M, \omega) \otimes U(\Pi_h)$.

We examine each of these points in turn:

1. A point ω' belongs to the Kripke world which defines σ' if and only if ω' belongs to $(M, \omega) \otimes U(\Pi_h)$.

    a) Left-to-right (by contradiction): A point ω' belongs to the Kripke world which defines σ' if ω' belongs to $(M, \omega) \otimes U(\Pi_h)$.

        i. Let us suppose that ω' belongs to the Kripke world which defines σ', but does *not belong* to $(M, \omega) \otimes U(\Pi_h)$.

        ii. ω' may only belong to the Kripke world which defines σ', if it belongs to the answer-set of $\Pi_{m\mathcal{A}+}$.

        iii. 1(a)ii may only be true however if an atom of the form $point(P, T)$ belongs to the answer-set in question, where $P = \omega'$ and T is the step of the diagram at which σ' is defined.

iv. 1(a)iii may only be true if the following rule from Listing 5.17:

$$point(w(P,E),T) \leftarrow point(P,T-1),\ event(E,T-1),$$
$$pre(\Phi,E,T-1),\ entailed\_by(\Phi,P,T-1),$$
$$action\_occurs(T-1).$$

is satisfied by $\Pi_{m\mathcal{A}+}$.

v. However, from Definitions 2.17 and 2.21, the rule in 1(a)iv is only satisfied, if ω' belongs to $(M,\omega) \otimes U(\Pi_h)$. Hence, we have a contradiction, and therefore it must be the case that if ω' belongs to the Kripke world which defines σ', it must also belong to $(M,\omega) \otimes U(\Pi_h)$.

b) Right-to-left (by contradiction): ω' belongs to $(M,\omega) \otimes U(\Pi_h)$ if ω' belongs to the Kripke world which defines σ'.

i. Let us suppose that ω' belongs to $(M,\omega) \otimes U(\Pi_h)$ but does *not belong* to to the Kripke world which defines σ'.

ii. By Definition 2.17, this may only be true if: ω' has the form $(\omega_i, \varepsilon_i)$ where ω_i is a point in M and ε_i is an event in $U(\Pi_h)$, and ω_i satisfies the precondition of ε_i.

iii. Given 1(b)ii however, it must be the case that the body of the rule from 1(a)iv must be satisfied. Hence, an atom of the form $point(P,T)$ belongs to the answer-set of $\Pi_{m\mathcal{A}+}$.

iv. From 1(b)iii however it must be the case that ω' belongs to the Kripke world which defines σ', which contradicts our assumption. Consequently it must be the case that if ω' belongs to $(M,\omega) \otimes U(\Pi_h)$ then ω' belongs to the Kripke world which defines σ'.

c) From 1a and 1b it must be the case that (1) holds.

2. The proof for 2 follows the same pattern as for 1, with the rule whose head is $k\_reachable(w(P_1, E_1), w(P_2, E_2), AG, T)$ in Listing 5.17.

3. Finally, as with the prior two points, the proof for 3 follows the same argument, but using the remaining rules of Listing 5.17.

□

A.4 General Lemmas and Theorems

In this section we present the general lemmas and theorems that play a role in the proofs of the theorems presented in this dissertation, and are of interest in and of themselves as well.

Lemma A.2 (G-reachability and Entailment)**.** *Let M be a Kripke model over a multi-agent domain $\mathcal{D}$.*

- $(M, \omega) \models \mathcal{E}^k_\gamma \varphi$ *if and only if* $(M, \omega) \models \varphi$ *for all τ that are γ-reachable from ω in k steps.*

- $(M, \omega) \models \mathcal{C}_\gamma \varphi$ *if and only if* $(M, \omega) \models \varphi$ *for all τ that are γ-reachable from ω.*

Proof. The proof of this lemma is given in [19] and we refer the reader to that text. □

The following lemmas and their respective proofs first appeared in [14] and are presented below in full:

Lemma A.3. *Every* **S5***-state (M, ω) is equivalent to a* **S5***-state (M', ω) such that for every point $\sigma \in M'.W$, we have that σ is reachable from ω.*

By Construction. The result derives from the fact that, if there is a world σ which is not reachable from ω, then for each formula φ we have that $(M, \omega) \models \varphi$ if and only if $(M', \sigma) \models \varphi$, where M' is defined as follows:

1. $M'.W = M.W \setminus \{\sigma\}$ (i.e., we remove the unreachable world σ);
2. $M'.\pi(\tau) = M.\pi(\tau)$ for every $\tau \in M'.W$ (i.e., all interpretations associated with the remaining points are the same)
3. $M'.R_\alpha = M.R_\alpha \setminus \{(\mu, \nu) \mid (\mu, \nu) \in M.R_\alpha, \mu \notin M'.W \vee \nu \notin M'.W\}$, (i.e., we maintain the same accessibility relations except for removing all cases related to the world σ)

□

Lemma A.4. *Let (M, ω) be a* **S5***-state such that every point in $M.W$ is reachable from ω, and let φ be a formula. $(M, \omega) \models C\varphi$ if and only if $M.\pi(\omega) \models \varphi$ for every point in $M.W$.*

Proof. Because we know that (M, ω) is an **S5**-state, we know that all of the accessibility relations must be reflexive, transitive, and symmetric. Furthermore, from the definition of the entailment relation for the operator C, we know that $(M, \omega) \models C\varphi$ implies that $(M, \omega) \models \varphi$ in every point τ reachable from ω, which may only be true if $M.\pi(\tau) \models \varphi$. The converse is obvious from the fact that each state is reachable from ω. □

Lemma A.5. *Let (M, ω) be a* **S5***-state such that every point in $M.W$ is reachable from ω. Furthermore, let $(\tilde{M}, \tilde{\omega})$ be the reduced state of (M, ω). It holds that:*

1. $(M, \omega) \models \varphi$ *iff* $(\tilde{M}\tilde{\omega}) \models \varphi$
2. $(M, \omega) \models C\varphi$ *iff* $(\tilde{M}\tilde{\omega}) \models C\varphi$
3. $(M, \omega) \models CB_\alpha\varphi$ *iff* $(\tilde{M}\tilde{\omega}) \models CB_\alpha\varphi$

4. $(M, \omega) \models C\mathcal{B}_\alpha\varphi \vee \mathcal{B}_\alpha\neg\varphi$ *iff* $(\tilde{M}, \tilde{\omega}) \models C\mathcal{B}_\alpha\varphi \vee \mathcal{B}_\alpha\neg\varphi$

5. $(M, \omega) \models C\neg\mathcal{B}_\alpha\varphi \vee \neg\mathcal{B}_\alpha\neg\varphi$ *iff* $(\tilde{M}, \tilde{\omega}) \models C\neg\mathcal{B}_\alpha\varphi \vee \neg\mathcal{B}_\alpha\neg\varphi$

Proof. This lemma is an immediate consequence of Lemma A.4 and the construction of $(\tilde{M}, \tilde{\omega})$. □

Lemma A.6. *Let $\mathcal{I}$ be a set of initial state axioms of a* definite action theory. *Every* **S5***-state satisfying $\mathcal{I}$ is equivalent to an* **S5***-state (M', ω') such that $|M'.W| \leq 2^{|\mathcal{F}|}$.*

Proof. By Lemma A.3 we can assume that every point in M is reachable from ω. Furthermore, Lemma A.5 shows that $(\tilde{M}, \tilde{\omega})$ is also a **S5**-state satisfying $\mathcal{I}$. It is clear that $\left|\tilde{M}.W\right| \leq 2^{|\mathcal{F}|}$ as each interpretation function may be modeled as a subset of $\mathcal{F}$ and $\left|\tilde{M}.W\right|$ is bound by the number of of distinct interpretations. We will now show that the reduced state $(\tilde{M}, \tilde{\omega})$ of (M, ω) satisfies the lemma. To complete the proof, we must show that for any arbitrary formula φ and point $\omega \in M.W$, the following holds:

$$(M, \omega) \models \varphi \text{ iff } (\tilde{M}, \tilde{\omega}) \models \varphi \tag{A.1}$$

This will be done by induction on the depth of φ:

1. Suppose that $depth(\varphi) = 0$. In this case, φ is a fluent formula and (A.1) trivially holds from the construction of $(\tilde{M}, \tilde{\omega})$.

2. Suppose that for any φ where $depth(\varphi) \leq k$, (A.1) holds.

3. Consider a formula φ where $depth(\varphi) = k + 1$. Here we have four distinct case:

   a) $\varphi = \mathcal{B}_\alpha\psi$. In this case, $(M, \omega) \models \varphi$ iff for every point, τ, reachable from ω by agent α's accessibility relation, $(M, \tau) \models \psi$, which by construction of $(\tilde{M}, \tilde{\omega})$ and the inductive hypothesis, must also hold for every point reachable from $\tilde{\omega}$ in $\tilde{M}$ by α's accessibility relation. Hence, $(\tilde{M}, \tilde{\omega}) \models \varphi$.

b) The proof for the cases involving the connectives: $\neg$, $\vee$, and $\wedge$, follows similarly to (3a).

Furthermore, from (3a), (A.1) holds for formulae involving the modal operators C and $\mathcal{E}$, and hence the lemma holds true.

□