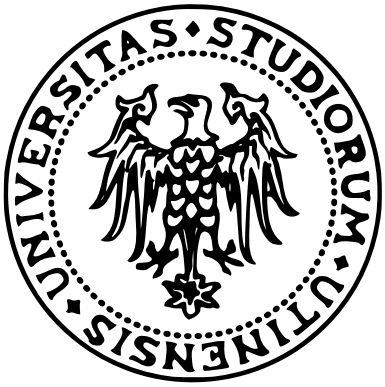
Planning while Believing to Know



University of Udine
Department of Mathematics, Computer Science and Physics

Candidate:
Francesco Fabiano

Advisor:
Prof. Agostino Dovier

Co-Advisors:
Prof. Enrico Pontelli
Prof. Alessandro Dal Palù

Thesis submitted for the degree of
Doctor of Philosophy in Computer Science, Mathematics and Physics

Cicle: XXXIV

Years: 2018–2021

Ai miei Genitori

To my Parents

Abstract

Over the last few years, the concept of *Artificial Intelligence* (AI) has become essential in our daily life and in several working scenarios. Among the various branches of AI, *automated planning* and the study of *multi-agent systems* are central research fields. This thesis focuses on a combination of these two areas: that is, a specialized kind of planning known as *Multi-agent Epistemic Planning*. This field of research is concentrated on all those scenarios where agents, reasoning in the space of knowledge/beliefs, try to find a plan to reach a desirable state from a starting one. This requires agents able to reason about her/his and others' knowledge/beliefs and, therefore, capable of performing *epistemic reasoning*. Being aware of the information flows and the others' states of mind is, in fact, a key aspect in several planning situations. That is why developing autonomous agents, that can reason considering the perspectives of their peers, is paramount to model a variety of real-world domains.

The objective of our work is to formalize an environment where a complete characterization of the agents' knowledge/beliefs interactions and updates are possible. In particular, we achieved such a goal by defining a new action-based language for Multi-agent Epistemic Planning and implementing epistemic solvers based on it. These planners, flexible enough to reason about various domains and different nuances of knowledge/belief update, can provide a solid base for further research on epistemic reasoning or real-base applications. This is true, especially considering that one of the proposed approaches formally verifies that the obtained plan is correct with respect to semantics on which is based.

This dissertation also proposes the design of a more general epistemic planning framework. This architecture, following famous cognitive theories, tries to emulate some characteristics of the human decision-making process. In particular, we envisioned a system composed of several solving processes, each one with its own trade-off between efficiency and correctness, which are arbitrated by a *meta-cognitive* module.

Contents

Appendices

List of Figures

List of Tables

Thesis Organization

Here we will provide a high-level overview of each chapter of the thesis. Alongside the chapters' content, we will also provide references to published scientific articles—produced during the Ph.D. period—that constitute the backbone of this work. These articles have been validated and enriched by the peer-review process, providing a solid foundation for this dissertation.

1. The first chapter serves as an introduction to the whole thesis. It starts by providing the motivation for our work, explaining why we decided to direct our research efforts to *Multi-agent Epistemic Planning*. It then illustrates some basic notions related: *(i)* to the planning area; *(ii)* to the epistemology world; and *(iii)* to their connections. While this chapter does not present any new contributions, it helps in setting the foundations necessary to understand the actual contributions. Given the introductory nature of this chapter, we decided to take inspiration from more experienced authors. In particular, the prominent reference for the planning introduction was provided by Russell and Norvig [2010] while, for the epistemic preamble we referred to Fagin et al. [1995], Bolander and Andersen [2011], Baral et al. [2015], van Ditmarsch et al. [2015].

2. The second chapter introduces the first contribution of our research. In particular, we present a variation of an epistemic action language, *i.e.*, a language used to define Multi-agent Epistemic Planning problems. This new language is based on non-well-founded data structures, called *possibilities*—initially theorized by Gerbrandy and Groeneveld [1997]—rather than the more classical Kripke structures. The chapter illustrates the components of the language highlighting its advantages with respect to its predecessor, concluding with some remarks on the correctness of the new specification. To provide enough information about the newly adopted possibilities, we mostly referred to works by their original authors, *i.e.*, Gerbrandy and Groeneveld [1997], Gerbrandy [1999]. On the other hand, the formal definition of the new language was firstly presented in our work Fabiano et al. [2019] and later improved in Fabiano et al. [2020].

3. Chapter three further analyzes the concept of epistemic action languages. In particular, it explores how to enrich the aforementioned language with features

that would allow it to represent more realistic scenarios. We accomplished that by formalizing the idea of *trust* between agents. This extension has been devised for both the possibilities-based and the Kripke structure-based languages. Finally, we provide some fundamental properties to ensure that the communications with trust behave as expected. This chapter derives from the work Fabiano [2020] where we initially tackled the idea of formalizing trust in our epistemic action language.

4. As a final advancement in our formalization of a general epistemic language specification we present, in chapter four, the idea of *agents' attitudes*. These attitudes are used to associate each agent with a particular set of biases about the information received by others. With these extra characterizations, we can formalize domains with a wider spectrum of interactions; allowing for epistemic planning domains to become even more realistic. As for the previous chapters, we captured some fundamental properties of this new addition ensuring its correctness. This idea was firstly explored and formalized in Fabiano et al. [2021a].

5. In the fifth chapter we present the implementation of a *general and comprehensive epistemic solver*. This C++ planner, called EFP, integrates all the previous theoretical advancements and constitutes a tool that we hope will be adopted by the community as the basis for future research. EFP is able to plan while considering belief relations and concepts such as lies, misconceptions, trust, and so on. While generality is our primary concern, EFP shows state-of-the-art performances in reasoning on complete epistemic states as we can see from the various experimental evaluations presented in the chapter. Finally, we also present PLATO, a version of the planner in Answer Set Programming. This planner and its use of the declarative approach are then compared with EFP and its more classical imperative paradigm. Thanks to its declarative nature PLATO allows us to formally validate its behaviour, with respect to the underlying semantics, and therefore the computed plans. Furthermore, this permits to empirically confirm the results obtained by the versions of EFP that implement the underlying action language of PLATO.

The EFP version presented is the result of several scientific productions where its internal structure has been optimized and enriched; in chronological order, these are Le et al. [2018], Fabiano [2019], Fabiano et al. [2020, 2021a]. PLATO, instead, has been formalized and implemented initially in Burigana et al. [2020].

6. Chapter six discusses the integration of a famous cognitive theory, that is "thinking fast and slow" by Kahneman [2011], into the modern concept of AI.

This chapter stems from a collaboration with a research group from the IBM Thomas J. Watson Research Center. While the joint project aims to analyze cognitive theories to widen the AI capabilities, in this chapter we focus on how this research can affect the epistemic planning setting. In particular, the chapter will identify what it means to think *fast* and *slow* in Multi-agent Epistemic Planning, examining also the role of the meta-cognition in an architecture that employs the aforementioned paradigm. The chapter also introduces an architecture that, following the schema proposed by Kahneman, is able to tackle epistemic planning problems. This final contribution is based, alongside countless hours of discussion with the IBM research group, on the scientific contributions by Booch et al. [2021], Fabiano et al. [2021b], Ganapini et al. [2021, 2022].

7. The last chapter concludes the thesis with some final remarks on the various contributions and with a brief description of possible future works.

An investment in knowledge pays the best interest.

— Benjamin Franklin
Poor Richard's Almanac
[Franklin, 1750]

1 Introduction & Preliminaries

Contents

1.1 Motivation

Artificial Intelligence (AI for short) is a term, coined by McCarthy, Minsky, Rochester, and Shannon in 1955, used to capture the idea of autonomous machines which have capabilities that allow them "to think". While exploring what it means "to think" deserves a dissertation on its own, we make use of this term in a loose and non-formal way to provide the reader with a general intuition of what an autonomous agent should accomplish. This concept stems from one of the most important scientific figures of the last century, considered to be the founding father

of computer science and Artificial Intelligence, Alan Turing. In the first sentence of his publication "*Computing Machinery and Intelligence*" [Turing, 1950], Turing proposed "*to consider the question, 'Can machines think?'*". He also provided meaning to what it means for a machine "to think" introducing the well-known *Turing's test*, which states that machines can be considered intelligent when they can mimic the behavior of humans.

With his contribution, Turing effectively initiated the scientific quest of formalizing and constructing agents that are able to act on the world out of their own volition. After McCarthy, Minsky, Rochester, and Shannon proposed and organized the first meeting on Artificial Intelligence in 1956 [McCarthy et al., 2006], researchers started to investigate this topic with several revolutionary accomplishments in both theoretical and practical aspects of AI. In particular, the idea of intelligence has been refined to incorporate the fundamental aspect of *rationality* that does not always (nor often) coincide with human behavior. That is why, nowadays, the idea of machines' intelligence is not limited to the sole human behavior imitation but also considers the possibility of agents that reason, or act, following logic, as elegantly summarized by Russell and Norvig in their book [Russell and Norvig, 2010, chapter 1].

Alongside the novelties introduced at the conceptual level, the AI community formalized and developed several techniques that permit to model agents which can solve intricate problems in autonomy. These techniques range from the use of various formal logics, and in particular *modal logic*, to the creation of neural-based structures. The former is an area of study that stems from the field of philosophy which, after the initial efforts of Clarence Irving Lewis, evolved rapidly [Ballarin, 2021] and has become an essential tool to define rational behavior for our systems. The latter is a technology that, imitating the physiology of our brain, allows the agents to perform reasoning tasks emulating (to some degree) the human behavior. While this idea was firstly studied in the early days of AI, only recently, thanks to figures such as Geoffrey Hinton, Yoshua Bengio, and Yann LeCun, *neural networks* are adopted to solve a wide spectrum of problems, as reported by Bengio et al. [2021].

While the field of AI has constantly evolved after the intriguing question posed by Turing in 1950, over the last few years the concept of Artificial Intelligence has become more and more prominent in our life, whether we are computer science researchers or not. The concept of autonomous agents, often identified by software processes, performing tasks of different nature has been accepted and embraced in both our daily life and in the industry. That is why, AI-driven solutions are, nowadays, frequently used to tackle problems that range from mundane ones—*e.g.*, teaching how to play Sudoku [Hanson, 2021]—to very intricate tasks that require a high-level of expertise—*e.g.*, analyzing CT scans to help radiologists in identifying anomalies [Chu et al., 2019, Fabiano and Dal Palù, 2022]. Not only AI techniques are widely deployed, but it is becoming essential for the majority of the real-world scenarios, *e.g.*, Industry 4.0, to exploit tools derived from the fields of automated reasoning and knowledge representations [Lasi et al., 2014].

Even if AI is gaining popularity, most of the research efforts are not directed to this topic as a whole but to specialized sub-areas; *e.g.*, *natural language recognition*, *knowledge representation/manipulation*, and *formal verification*. In particular, the field of *automated planning* is one of the most important and most studied branches of AI. As said by Russell and Norvig [2010, chapter 10], "*we have defined AI as the study of rational action, which means that planning—devising a plan of action to achieve one's goals—is a critical part of AI*". That is why we decided to focus our research on the planning problem and, in particular, on those situations where multiple entities interact with each other. These scenarios, known as *multi-agent* for the presence of multiple active entities, are ubiquitous in everyday life and represent the majority of the "real-world" problems.

To correctly address multi-agent problems, a solving process needs to reason not only on the state of the world but also on its information flows. As said by [Van Ditmarsch et al., 2007] "*information is something that is relative to a subject who has a certain perspective on the world, called an agent, and that is meaningful as a whole, not just loose bits and pieces. This makes us call it knowledge*

and, to a lesser extent, belief". That is why *epsitemic*[1] and *doxastic*[2] reasoning come into play in formalizing such scenarios. These types of automated reasoning are used to capture the knowledge or beliefs relations among multiple agents and provide a tool to formalize those settings where the information flows must be considered by the solving process, *e.g.*, economy [Aumann et al., 1995], security [Balliu et al., 2011], justice [Prakken, 2013] and politics [Carbonell Jr, 1978].

1.2 Planning: Notation and Concepts

According to Cambridge Dictionary [2021], a *plan* is "*a method for doing or achieving something, usually involving a series of actions* [...]" implying that *planning* permeates every thought-out process performed by humans, animals, or even machines. To plan is, in fact, to devise a way of reaching an objective, whether this goal is to have enough food for the day or to build a skyscraper. The ability to divide processes, independently of their complexity, into "smaller" and more manageable ones is paramount to accomplish one's objectives and, ultimately, to manipulate the environment to her/his advantage. That is why, designing autonomous agents that incorporate the ability to select the best course of action to achieve their goals, is of the utmost importance in Artificial Intelligence.

While the concept of automated planning has several variations (*e.g.*, *classical*, *conformant*, *epistemic*, etc.) that are used to describe different real-world scenarios, all of them share the same objective: given an initial configuration of the environment, find a sequence of permitted actions to reach the desired configuration of the same environment.

Given its importance inside the AI community, the *planning problem* is a long-studied and researched topic. That is why, we will not provide a comprehensive introduction of this field addressing the interested readers to the book of Russell and Norvig [2010, chapters 10 and 11] for a much more complete and elegant description of automated planning.

[1] From the ancient Greek term 'episteme' (ἐπιστήμη) that means 'knowledge'.
[2] From the ancient Greek term 'doxasía' (δοξασία) that means 'belief, opinion'.

1.2.1 Basic Concepts

In what follows we will introduce the basic terminology and concepts, related to the planning environment, that will be essential to present the contributions of this thesis. The well-known *Block World* domain, presented in Planning Domain 1.1, will be used as a running example to better explain the concepts introduced throughout this section. In particular, this domain will be used to describe the key features of planning.

Planning Domain 1.1: Block World

The Block World, due to its simplicity, is one of the most employed domains when it comes to explaining the basics of planning. This domain consists of a few simple elements:

- *blocks* of the same size that can be placed either: on the table, or on top of another block; and
- a *mechanical arm* that can move the blocks and can determine whether it is holding a block or not.

Moreover, there are some constraints that regulate the Block World:

- the mechanical arm can only hold, and therefore move, one block at the same time; and
- a block can only be placed on top of a *clear* block—a block with no blocks on top of it and that is not held by the mechanical arm—or on the table.

In what follows, we will provide a series of definitions, each followed by an example based on the Block World domain, to formalize the ideas of *state*, *agent*, *action*, *planning problem*, *transition function*, and *solution* in planning.

The first fundamental concept that we need to introduce is the idea of *planning state*. This concept, formally introduced in Definition 1.1, is used to define a static "picture" of the environment expressing its properties thanks to *fluent literals*—Boolean propositional variables that can change their truth value over time—that represent different aspects of the state itself.

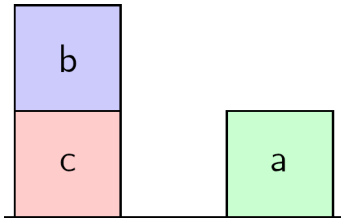


Figure 1.1: The World Block domain.

Definition 1.1: Planning State

A *state* of the domain is a configuration of the environment described by the domain, referred to as *world*, represented as a conjunction of *ground fluent literals*.

Example 1.1: Planning State Figure 1.1 represents a state in the Block World domain. This state is defined by the following positive fluent literals: {`onTable_A`, `onTable_C`, `onC_B`, `Clear_Arm`}. In this description and in what follows, the negative fluent literals, *e.g.*, ¬`onTable_B` or ¬`onA_C`, might be omitted for the sake of readability.

Next, let us introduce the idea of *agent*. Intuitively, an agent is an entity that acts upon the domain interacting with its elements and/or with other agents to achieve her/his goal.

Definition 1.2: Planning Agent [McNeill and Bundy, 2010]

An agent is an entity that responds to goals through forming plans to achieve them and then, possibly, enacts these plans through interacting with the domain.

Example 1.2: Agent In the Block World domain an agent is the (simulated) mechanical arm that moves the blocks to find the desired configuration.

After defining the idea of agents as entities that change the states trying to reach the goal executing some actions, we need to formalize the concept of *planning action*. Let us note that execution in a software environment is just a simulation of the actions.

Definition 1.3: Action

An *action* in planning is an operation, made by some agent, that changes the actual world or its perception. Actions can have *executability conditions* that express when an action is, as the name suggests, executable and when it is not.

Example 1.3: Action Given the state in Figure 1.1 we can give an example of executable and not-executable actions.

- An executable action is `take(B)`. This action states that the mechanical arm takes block `B` and keeps it. The executability conditions of this action are {`ClearArm`, ¬`onB_A`, ¬`onB_C`} which are respected in the current state. These conditions read as "The action `take(B)` is executable if: *(i)* the mechanical arm is not holding any block; *(ii)* block `A` is not on top of block `B`; and *(iii)* block `C` is not on top of block `B`."
- An example of not-executable action is `take(C)`, which demands to mechanical arm to take block `C`. This action is not executable because block `C` is not clear. More formally, the executability conditions of this action are {`ClearArm`, ¬`onC_A`, ¬`onC_B`} and, since ¬`onC_B` is not respected, the action cannot be executed.

The *planning problem* (Definition 1.4) formally describes *(i)* the scenario in which $n \geq 1$ agents act upon; *(ii)* the starting point; and *(iii)* the desired configuration. The combination of these descriptions, alongside the formalization of how an action affects the world (Definition 1.5), is what allows the agents to find the *plan* (Definition 1.6).

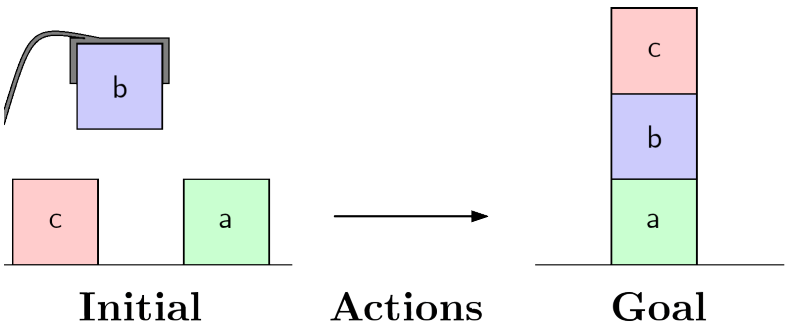


Figure 1.2: The Planning Problem in the World Block domain.

Definition 1.4: Planning Problem

A *planning problem* is a tuple $\langle D, I, G \rangle$ where:

- D is an *action domain* expressed in some language: *i.e.*, D describes the properties of interest of the environment in which the agents are acting upon and also specifies how the agents themselves can manipulate these properties through actions;
- I is a set of states of the domain—called *Initial state*—that describes the diverse (possible) starting configurations of the world. The example in Figure 1.2, comprised of single state, is described as: {`onTable_A`, `onTable_C`, `holding_B`, ¬`ClearArm`, ¬`onA_B`, ¬`onA_C`, ¬`onB_A`, ¬`onB_C`, ¬`onC_A`, ¬`onC_B`, ¬`holding_A`, ¬`holding_C`}.
- G is a set of states of the domain—called *Goal state*—that describes the desired configurations of the domain. The example in Figure 1.2, once again composed of a single state, is described as: {`onTable_A`, `onA_B`, `onB_C`, `ClearArm`}.

Before introducing the concept of transition function let us note that since the action execution may be non-deterministic, and therefore contemplates multiple states where only one should be intuitively created, we must consider the update to be capable of handling sets of states. These sets intuitively represent all the possible states that can be reached considering the various non-deterministic effects. The special case where all the actions are deterministic simply considers these sets to be singletons.

Definition 1.5: Transition Function

A *transition function* Φ is a function that, given a starting state and an action, returns a set of states in which the world can be after the execution of the action in the starting state. More formally, $\Phi : 2^{\Sigma} \times A \to 2^{\Sigma}$ where: Σ is the set of all the possible states and A the set of all the possible actions in the domain. If an action a is not executable in a state $s \in \Sigma$ then we will have $\Phi(a, s) = \emptyset$. Finally, we consider $\Phi : \Sigma \times A \to \Sigma$ when the actions are constrained to have deterministic effects.

Definition 1.6: Plan/Solution

A *plan/solution* for the generic planning problem $\langle D, I, G \rangle$ is a sequence of actions $\in D$ that, when executed, transforms the the given initial state I into one of the desired configurations $\in G$.

Once again, assuming the actions to be constrained to have deterministic effects, a *plan/solution* is a sequence of actions $[a_1, \ldots, a_n]$ such that:

- a_1 is executable in every state s belonging to I,
- a_i is executable in every state s belonging to $\Phi_D(a_{i-1}, \ldots, \Phi_D(a_1, I))$ for $i = 2, \ldots, n$
- G is true in every state s belonging to $\Phi_D(a_n, \ldots, \Phi_D(a_1, I))$.

Even if the presented terminology is shared across the whole planning community, as already mentioned, the research in planning is differentiated into several categories. A brief explanation of all the planning problem types, that are relevant to this thesis (except for *epistemic planning* that will have a section on its own, *i.e.*, Section 1.3), will be presented next.

1.2.2 Planning Problem Categories

Classical Planning

The idea of *classical planning* has been present since the birth of AI and it has been widely explored in the computer science community ever since. That is why, we believe that introducing this concept with an already existing description, that has been thought by far more experienced researchers, would be most appropriate. In particular, the brief yet elegant description written by Bolander and Andersen

does an excellent job in explaining what classical planning is: "*For most of its early life in the '60s and '70s, the field of automated planning was concerned with ways in which the problem of creating long-term plans for achieving goals could be formulated, such that solving problems of non-trivial size, would be computationally feasible. The type of planning that arose from this early work, is what is known today as Classical Planning*".

The success of classical planning is partially due to the several restrictions imposed on the world description—*i.e.*, the domain has to be *(i) static*; *(ii) deterministic*; and *(iii) fully observable* [Ghallab et al., 2004]—that make this kind of problems more tractable and approachable. In particular, *(i)* a *static* problem is represented by a domain that is not modified by elements that are external to the domain itself. *(ii)* A domain is *deterministic* when, for each state s and action a, the transition function $\Phi(a, s)$ has at most one element—*i.e.*, there is no ambiguity in which state will be the world after the execution of the action. Determinism also implies that the plan will be in one, and only one, state at each and every step of the planning computation. *(iii) Fully observable* means that an agent knows the complete description of the world, that is, she/he knows the state of every fluent literal in the domain. Moreover, to maintain its simplicity the classical planning domains are, most of the time, single-agent—*i.e.*, domains where only one agent can perform the actions and has to reach the goal.

A good example of classical planning can be, once again, the World Block domain described in Planning Domain 1.1. Here the single agent—namely, the mechanical arm—knows everything about the world (if a block or itself is clear or not, for example), and every action has only one possible outcome.

Several automated tools to solve classical planning problems, known as *planners* or *solvers* (Definition 1.7), have been developed for both "scientific" and industrial purposes. These tools [Richter and Westphal, 2010, Lipovetzky and Geffner, 2014, 2017], that improve each year in terms of performance and accuracy, are the foundation of all the instruments developed by the planning community.

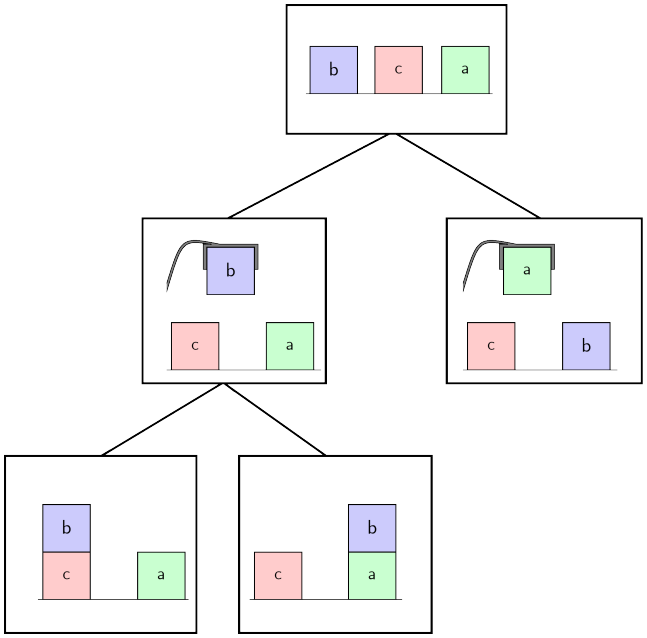


Figure 1.3: A planning tree for the World Block domain.

Definition 1.7: Planner/Solver

A *planner* (or *solver*) is a program that computes the solution of any given planning problem within a compatible domain.

During the years, to improve the performances of the various planners, different ways of representing the so-called *search-space*—an abstract representation of the paths to reachable states in a domain—have been developed. The most common and used one is referred to as *tree* and it is shown in Figure 1.3. Along with the study of how to represent the search space the planning community also studies ways to explore these spaces defining different strategies that allow the planning process to find the right balance between resources consumption and accuracy.

Classical planning is often seen as the basic form of all the other kinds of planning that, usually, consider more intricate problems or allow for a less strict description of the world. Moreover, since classical planning problems consider more constrained environments with respect to other types of planning problems, the existing solvers are the most efficient. In fact, it is not unusual to reduce, when possible, problems from more complex domains to the classical one—*i.e.*,

to elaborate the problem itself so it can be solved by a classical *planner*—even if sometimes reducing a domain may cause loss of expressiveness.

Conformant/Contingent Planning

Unfortunately, most of the real-world problems that we want to solve with planning methods do not comply with the restrictions posed by classical planners. In particular, agents may not have complete information about some properties of the world, meaning that they may not be able to retrieve the missing information until the actual execution of the plan. This type of domain is known as *conformant planning*. This "ignorance" leads to a substantial difference, with respect to classical planning, when it comes to the transition function Φ. While in classical planning applying an action produces one and only one successor, in conformant planning Φ produces a set of possible successor states. The most notable consequence is that—given that the solution for a problem must be true in all the reachable states (Definition 1.6)—the plan must be valid for all the possible configurations of the initial state.

In conformant planning the uncertainty derives from the initial state and is carried on by the actions. This means that the actions do not generate non-determinism themselves, for example through *if-else* conditions, but inherit the uncertainty from the state on which they are executed generating multiple outcomes. On the other hand, when actions do generate non-determinism we talk about *contingent* planning. To better explain this difference we will use two different descriptions of the same domain, the well-known *Vacuum-Cleaner*, introduced in Russell and Norvig [2010, chapter 2].

Planning Domain 1.2: Vacuum-Cleaner

In this domain (represented in Figure 1.4) an agent, the vacuum-cleaner, has to clean two rooms from the dirt using a sequence made from four actions `Left`, `Right`, `Clean`, `NoOperation` that do what their names suggest (described in detail in Russell and Norvig [2010, chapter 2]).

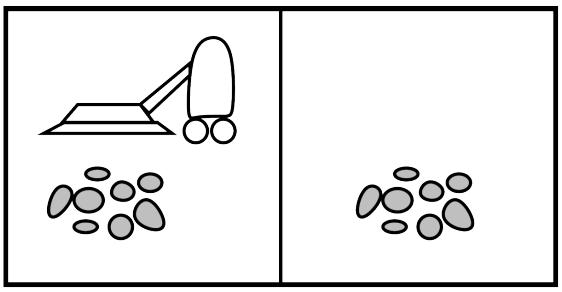


Figure 1.4: The Vacuum-Cleaner domain with both dirty rooms.

To differentiate conformant and contingent planning let us define two slightly different versions of the domain presented in Planning Domain 1.2.

The first one, used for the example of conformant planning, uses the normal interpretation of the actions but we assume that the vacuum-cleaner cannot perceive whether the rooms are dirty or clean. This implies that all the states where the agent is in the left room (accordingly to Figure 1.4) are possible initial states and the successor states of the action `Clean` are shown in Figure 1.5. A solution for this problem is ⟨`Clean`, `Right`, `Clean`⟩; this sequence of actions reaches the goal from every possible initial state.

On the other hand, the variation devised to present contingent planning uses a modification of the action `Clean`. In particular, whenever this action is applied to a room with dirt in it, the vacuum-cleaner effectively cleans the room but, sometimes, cleans the other room too. On the other hand, when the same action is applied to a clean room it sometimes deposits dirt on the carpet. A plan for the initial state, as described in Figure 1.4, is described by the *AND-OR tree* in Figure 1.6—a special data structure that is used to express the search-space in contingent planning. Without going into detail, we can see that the action `Clean` forms the so-called *OR nodes* (the ones with exiting arrows connected by an edge) that intuitively capture the idea of non-determinism.

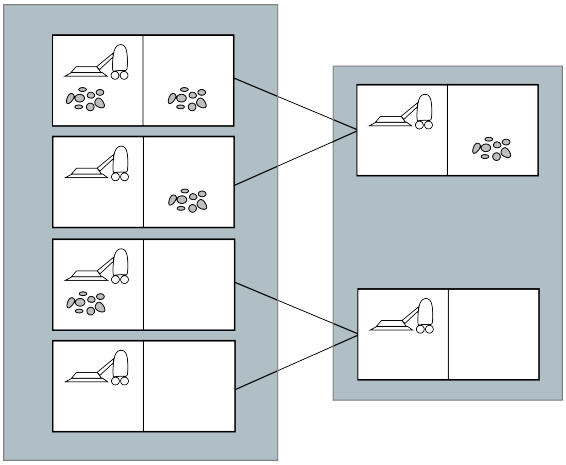


Figure 1.5: The execution of the action `Clean` in a conformant domain.

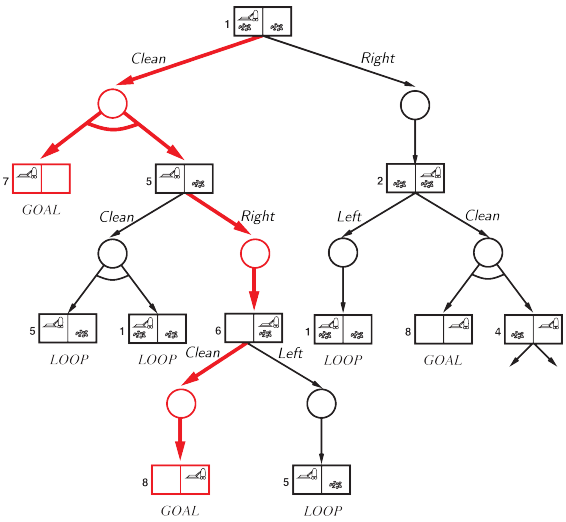


Figure 1.6: AND-OR tree for the Vacuum-Cleaner domain with non-deterministic actions. In red highlighted the solution ⟨`Clean`, `Right`, `Clean`⟩.

Multi-Agent Planning

Real-world scenarios, often, require more than a single agent that can act upon the domain. This family of planning problems that considers multiple entities is called *multi-agents planning* and it is used to model all those domains where agents' interactions are fundamental. Even if more agents acting in the same domain can, initially, seem a more efficient way to solve problems—for example, when we envision multi-agent as a means of parallelization—in reality, most of the time, having multiple entities increases the inherent complexity of the solving process.

The family of multi-agent problems engulfs several configurations of domains

that demand multiple acting entities. Next, we will list briefly some of the most known and studied configurations in the literature to provide the reader with an idea of the type of problems tackled by the multi-agent community.

The first way of classifying planning problems derives from the *agents-goal* relations. A basic subdivision is given by Bowling et al. [2005]:

- *Not-deterministic*: Each agent doesn't know which action, nor when, the other agents will perform. This implies that agents cannot accurately predict in which state the world will be in the future.
- *Cooperative*: The agents try to cooperate to reach the same goal (Planning Domain 1.3).
- *Adversary*: Under this specification, we find the most known multi-agent scenarios; *i.e.*, competitive *games*. Agents might have, in fact, opposite goals and try to reach theirs penalizing the others.
- *Overlapping Goals*: The agents just happen, without willing it, to help each other to reach their own goal.

Another distinction is based on the type of communications between agents. We can simplify the subdivision described by Fornara [2003], Katewa [2017] in two different categories of communication:

- *Free*: The agents are allowed to freely share their knowledge about the world (Planning Domain 1.3).
- *Privacy limited*: Agents can share only certain information with others. It is also possible that some agents get to share specific types of information with only a subset of the other agents.

Finally, another distinction, based on the solving process, is introduced by De Weerdt and Clement [2009] and Fornara [2003]. A planning system can therefore be:

- *Centralized*: A *master* agent coordinates the action of the others.
- *Decentralized*: Each agent acts independently (Planning Domain 1.3).

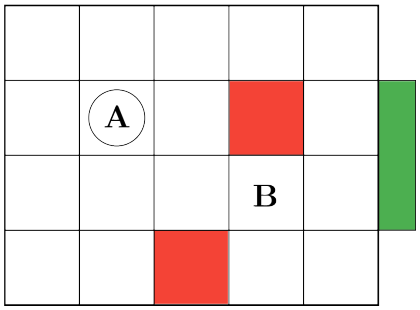


Figure 1.7: The Soccer domain.

Planning Domain 1.3: Soccer

An example of a *decentralized* multi-agent planning domain with *cooperative agents* and *without restrictions* about sharing information is a variation of the *Soccer domain* presented by Littman [1994].

In this game, each agent is placed in a single cell of the grid (Figure 1.7) and can move following the compass directions (`N`, `S`, `E`, and `W`) or `wait`. Agent **A** starts conventionally with the ball but whenever an agent tries to move in an already occupied cell she/he has to "pass" the ball to the agent that occupies that cell. The goal of this domain is to bring the ball inside the goal zone (green in Figure 1.7) without hitting the obstacles (red in Figure 1.7) in the fastest way possible.

1.3 Reasoning about Knowledge and Beliefs

Logicians have always been interested in describing *the state of the world* through formalism that would allow reasoning on the world with logic. This interest has led, among other things, to the formalization of the aforementioned planning problem and the introduction of several *modal logics* [Smullyan, 1968, Chagrov and Zakharyaschev, 1997, Van Ditmarsch et al., 2007] used to describe different types of scenarios. The difference between these logics is not merely syntactical, rather it carries implications in both expressiveness and complexity. Let us take for example, without going into details, the *Boolean propositional logic* and the *linear temporal logic* (LTL). The first one, being one of the simplest logic, is mostly used to encode the world as a set of facts that can be true or not and, therefore, allows to "reduce" properties of the domain to *Boolean formulae*. The latter instead, even if it is based on propositional logic, introduces *modal operators* that allow reasoning

about time (with a little abuse of the term). The absence of these operators in the first one makes propositional logic, adopted to represent problems in the complexity class NP such as *SAT*, not expressive enough to encode problems that LTL can deal with. So, in general, we have that different logics have diverse operators and therefore are suitable for different kinds of automated reasoning.

Nevertheless, even if different, the two logics introduced above are limited to reason only on the state of the world—*i.e.*, on its "physical" properties and their changes—and since this thesis aims to tackle the planning problem while considering the beliefs of the agents, it is clear that neither propositional logic nor LTL suffice to formalize the domains that we want to explore. *Epistemic Logic*, on the other hand, is used to reason not only on the state of the world but also on the *agents' knowledge* about the world or the others' knowledge. Similarly, the logic that addresses the problem of reasoning on the *agents' beliefs*—on both the physical world and on the others' beliefs—is referred to as *Doxastic Logic* [Meyer, 2003]. The idea behind epistemic and doxastic logic is, therefore, to have a formalization that allows to reason on domains where, not only the state world is taken into consideration, but also the knowledge/beliefs that the agents have about the world and about the knowledge/beliefs of each other are considered. That is why we used these logics as the foundation for our research.

In what follows we will briefly describe the fundamental concepts that are shared between epistemic and doxastic logic. This introduction is not to be intended as a complete survey of the vast area of epistemic and doxastic logic but, rather, as a way of presenting concepts that are paramount to illustrate the contributions of this thesis. During this work, for the sake of readability, we will make several (intuitive) assumptions to avoid the need to investigate "mind-twisting" aspects of epistemology that would complicate the design of autonomous agents—*e.g.*, we assume that the agents are perfect logicians. For a far more complete, compelling and informative introduction on this topic we refer the reader to Fagin et al. [1995], Van Ditmarsch et al. [2007], van Ditmarsch et al. [2015], Rendsvig and Symons [2021].

For the sake of readability let us identify both epistemic and doxastic logic with the term "epistemic logic" when there is no need to differentiate between them (differences in reasoning on knowledge or beliefs will be further explored later in this section). Moreover, for brevity, we will make use of the term "belief" to encapsulate both the notions of an agent's knowledge and beliefs about some information when the context permits it.

1.3.1 Epistemic Logic

Epistemology is the field of study that is concerned about knowledge and beliefs. Since its early days, Philosophy has always been intertwined with the concepts of knowledge and beliefs given that they play a central role in the development of cognitive theories as well as in the understanding of the human reasoning processes. While Aristotle is considered, among other things, to have initiated the discussion on epistemology [Rendsvig and Symons, 2021], the first logic formalization of this field is attributed to Ralph Strode, in 1387 [Boh, 1993]. This formalization, refined over the years, is what led modern philosophers in the fifties and sixties to define a complete axiomatization of the logic of knowledge and beliefs that resulted, in 1962, in the book "*Knowledge and Belief: An Introduction to the Logic of the Two Notions*" by Hintikka [1962].

While this formalization stems from the area of Philosophy, its application—*i.e.*, formally representing knowledge and/or beliefs—rapidly captured the interest of researchers of diverse areas. Notably, in the 1990s the computer science community started to embrace the idea of "reasoning about knowledge" and devised several ways to employ epistemic logic to model scenarios where autonomous agents could analyze knowledge/belief relations to, for example, better assess winning strategies. In particular, this thesis explores the interplay between (dynamic) epistemic logic and the field of planning—the so-called *Multi-agent Epistemic Planning problem*—inheriting its research scope from one of the most important works on this topic, *i.e.* "*Reasoning About Knowledge*" by Fagin et al. [1995].

Let us now introduce epistemic logic, namely the logic that allows to reason on the agents' knowledge/beliefs in static domains. In what follows we will make use of a simple instance of the *Coin in the Box domain* as a running example to present the main concepts of epistemic logic.

Planning Domain 1.4: Coin in the Box (Simplified)

Three agents, A, B, and C, are in a room where in the middle there is a box. The box has a lock that can only be opened with a key. Inside the box, there is a coin that lies *heads* up. In the initial configuration of this domain we have that everybody knows that:

- none of the agents know whether the coin lies heads (identified by `heads`) or tails (identified by the negation of `heads`, *i.e.*, ¬`heads`) up;
- the box is locked (identified by the negation of `opened`, *i.e.*, ¬`opened`); and
- only agent A has the key (identified by `haskey_A` and ¬`haskey_B`, ¬`haskey_C`).

In Planning Domain 1.4 we are presenting an example of an automated planning environment, and therefore, we should also specify the possible actions and the desired goals. Nevertheless, since we will use this example to better explain concepts of epistemic logic (which refers to static domains), for the moment we will not add any other specification to avoid unnecessary clutter. The ideas of action, transition function, and plan are, in fact, directly derived by the interaction between the field of planning and *Dynamic Epistemic Logic*. Since this combination is of great interest for our work, it will be the subject of a dedicated section, *i.e.*, Section 1.4.

Epistemic Logic Terminology

Let us consider a set $\mathcal{AG}$ of $n \geq 1$ agents and let $\mathcal{F}$ be a set of $m \geq 1$ propositional variables, *i.e.*, the fluent literals. With the term *epistemic world*, or simply *world* (Definition 1.8), we identify a subset of elements of $\mathcal{F}$—intuitively, only those that are *true* in that world. This means that a world describes a certain configuration of the environment identifying which properties hold (and, consequently, which do not). Furthermore, we use the term *pointed world* or *real world* to identify

the set of fluent literals that represents the actual configuration of the domain we are reasoning on.

Definition 1.8: Epistemic World

An *epistemic world* $\mathtt{w}$ is a set of propositional variables of $\mathcal{F}$ ($\mathtt{w} \subseteq \mathcal{F}$) which are interpreted as *true* in $\mathtt{w}$ ($\forall \mathtt{f} \in \mathtt{w}, \mathtt{f} \models_{\mathtt{w}} \top$; where $\top$ indicates *true*). The remaining elements of $\mathcal{F}$, *i.e.*, the ones that are not in $\mathtt{w}$, are considered to be false in $\mathtt{w}$ ($\forall \mathtt{f} \in \mathcal{F} \setminus \mathtt{w}, \mathtt{f} \models_{\mathtt{w}} \bot$; where $\bot$ indicates *false*).

Example 1.4: Epistemic World The description of the real world of Planning Domain 1.4 is expressed by the following set of true fluent literals: {`heads`, `haskey_A`}. The remaining fluent literals, *i.e.*, `opened`, `haskey_B`, and `haskey_C` are considered false.

During this thesis, we will, sometimes, make use of the more "complete" representation that explicitly presents both positive and negative (preceded by the symbol ¬) fluents to strengthen the clarity of the presentation. Following this schema, the world taken into consideration would be represented as {`heads`, `haskey_A`, ¬`opened`, ¬`haskey_B`, ¬`haskey_C`}.

In epistemic logic, as already said, we are not only concerned with the environment properties and that is why each agent $\mathsf{i} \in \mathcal{AG}$ is associated with an epistemic modal operator $\mathbf{B}_{\mathsf{i}}$. This operator, intuitively, represents the beliefs of the agent i. While the operator $\mathbf{B}_{\mathsf{i}}$ captures the direct beliefs of i, we also consider the *group operator* $\mathbf{C}_{\alpha}$. $\mathbf{C}_{\alpha}$ represents the *common beliefs* of a group of agents $\alpha \subseteq \mathcal{AG}$, *i.e.*, every agent in α believes a fact and believes that the others believe it too. The operators $\mathbf{B}_{\mathsf{i}}$ and $\mathbf{C}_{\alpha}$ allow to "enrich" the traditional definition of a *fluent formula* (Definition 1.9) and obtain the concept of *belief formula* (Definition 1.10). Several other operators, not considered by this thesis, that delineate far more complex beliefs relations—*e.g.*, the *Only Knowing* operator presented by Gerhard and Hector J. [2015]—have been devised.

Definition 1.9: Fluent Formula [Baral et al., 2015]

A *fluent formula* is a propositional formula built using the propositional variables in $\mathcal{F}$ and the traditional propositional operators $\wedge, \vee, \Rightarrow, \neg$. A *fluent atom* is a formula composed of just an element $\mathtt{f} \in \mathcal{F}$, instead a *fluent literal* is either a fluent atom $\mathtt{f} \in \mathcal{F}$ or its negation $\neg\mathtt{f}$. During this work, we will refer to fluent literals simply as *fluents*.

Definition 1.10: Belief Formula [Baral et al., 2015]

A *belief formula* is defined as follow:

- A fluent formula (Definition 1.9) is a belief formula;
- let φ be belief formula and $\mathsf{i} \in \mathcal{AG}$, then $\mathbf{B}_{\mathsf{i}}(\varphi)$ is a belief formula;
- let φ_1, φ_2, and φ_3 be belief formulae, then $\neg\varphi_3$ and $\varphi_1 \wedge \varphi_2$ are belief formulae (the connective $\vee$ is derived as a combination of $\neg$ and $\wedge$);
- the formulae of the form $\mathbf{C}_{\alpha}\varphi$ are belief formulae, where φ is itself a belief formula and $\emptyset \neq \alpha \subseteq \mathcal{AG}$.

The language $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ of well-formed belief formulae with *common belief*, over the sets $\mathcal{F}$ and $\mathcal{AG}$, can be defined compactly way by:

$$\varphi ::= \mathtt{f} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{B}_{\mathsf{i}}(\varphi) \mid \mathbf{C}_{\alpha}(\varphi),$$

where $\mathtt{f} \in \mathcal{F}$, $\mathsf{i} \in \mathcal{AG}$ and $\emptyset \neq \alpha \subseteq \mathcal{AG}$. We read the formula $\mathbf{B}_{\mathsf{i}}(\varphi)$ as "agent i believes that φ" and $\mathbf{C}_{\alpha}(\varphi)$ as "it is common belief between the agents in α that φ". In what follows, we will simply talk about "formulae" instead of "belief formulae", whenever there is no risk of confusion.

Example 1.5: Belief Formulae Considering Planning Domain 1.4, we can express "agent B believes that agent A has the key" with $\mathbf{B}_{\mathsf{B}}(\mathbf{B}_{\mathsf{A}}(\mathtt{haskey\_A}))$ and "it is common belief (between all the agents) that the box is closed" with $\mathbf{C}_{\{\mathsf{A},\mathsf{B},\mathsf{C}\}}(\neg\mathtt{opened})$.

Finally, from the ideas of "world" and agents' beliefs, we can informally define a state in epistemic logic, *i.e.*, an *e-state*.

Definition 1.11: Epistemic State (e-State)

An *epistemic state* is a collection of *epistemic worlds* believed to be possible by some agent in the domain. Moreover, an epistemic state captures the agents' beliefs about both the "physical properties" and others' beliefs.

Example 1.6: Epistemic State (e-State) The *e-state* that encapsulates Planning Domain 1.4, is made of two worlds. The first, the pointed one, is the one expressed in Example 1.4 and is described as: {`heads`, `haskey_A`}, while the latter is identified by {`haskey_A`}. These two worlds are considered possible by all the agents (A, B, and C) as they are not able to distinguish between the case in which the coin is heads or tails up. Nonetheless, no world that contains `opened` is found in the e-state as this property is known to be false by all the agents.

Let us note that Definition 1.11 does not clearly state how the agents' beliefs are represented. To do so we will need a much more formal definition of e-state that will be provided in the next paragraph.

Epistemic Logic Semantic

In the previous paragraph, we introduced the main concepts that are involved in epistemic logic, providing for them loose and intuitive meanings. Nevertheless, if we want to adopt these notions to define autonomous reasoners we must provide formal semantics for the proposed language, supporting the ideas introduced above. In particular, in this chapter we will explore the *Kripke structures* [Kripke, 1963], a data structure widely used in literature (for instance in Fagin et al. [1995], Van Ditmarsch et al. [2007], Baral et al. [2022]) to model the semantics of epistemic logic. These structures will allow us to provide a formal meaning for: the aforementioned idea of "world"; the concept of *epistemic state* (e-state); and for the entailment of belief formulae.

Definition 1.12: Kripke Structure [Kripke, 1963]

A *Kripke structure* (Figure 1.8) is a tuple $\langle \mathsf{W}, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_\mathsf{n} \rangle$, where:

- W is a set of *worlds*,
- $\pi : \mathsf{W} \mapsto 2^{\mathcal{F}}$ is a function that associates an interpretation of a set of propositional variables $\mathcal{F}$ to each element of W,
- $\mathcal{B}_\mathsf{i} \subseteq \mathsf{W} \times \mathsf{W}$, for $\mathsf{i} = 1, \ldots, \mathsf{n}$, is a binary relation over W.

Let us observe how Definition 1.12 deals with the terminology introduced in the previous paragraphs. In fact, we have that each element of W, thanks to its interpretation (described by π), identifies what we defined above as an "epistemic world", *i.e.*, a configuration of the environment. As mentioned above, each e-world contains only the positive propositional variables, and this is true also in the worlds of a Kripke structure. For example, the e-world presented in Example 1.4, let us call it w, is identified in both cases by $\mathsf{w} = \{\texttt{heads}, \texttt{haskey\_A}\}$ (represented by the left circle in Figure 1.8).

From now on whenever we consider Kripke structures we will be referring to a small variation of the structures: the *pointed Kripke structures* (Definition 1.13) that simply add an entry point. This entry point represents what we previously called the *pointed/real world*—the actual configuration of the environment on which we are planning.

Definition 1.13: Pointed Kripke Structure

A *Pointed Kripke structure* is a pair (M, w) where $M = \langle \mathsf{W}, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_\mathsf{n} \rangle$ is a Kripke structure and $\mathsf{w} \in \mathsf{W}$. In a pointed Kripke structure (M, w), we refer to w as the *pointed* (or *real*) *world* (represented by the bold circle in Figure 1.8).

For the sake of readability, we will make use of $M[\mathsf{W}]$, $M[\pi]$, $M[\mathsf{i}]$ and $M[\mathcal{B}]$ to denote the components $\mathsf{W}, \pi, \mathcal{B}_\mathsf{i}$ and $\mathcal{B} = \{M[\mathcal{B}_\mathsf{i}] \mid 1 \leq \mathsf{i} \leq \mathsf{n}\}$ of M, respectively. We write $M[\pi](\mathsf{w})$ to denote the interpretation associated to the world w via π and $M[\pi](\mathsf{w})(\phi)$ to denote the truth value of a fluent formula ϕ with respect to the interpretation $M[\pi](\mathsf{w})$. Moreover, we will often refer to a Kripke structure

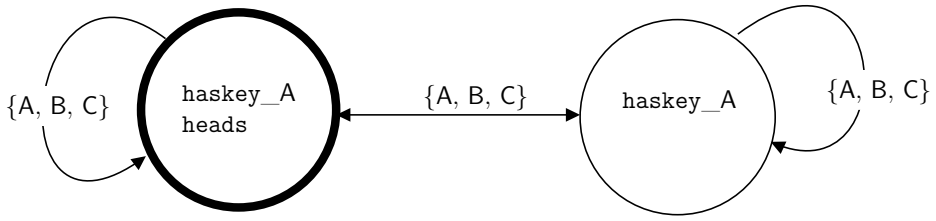


Figure 1.8: The Kripke structure that represents Planning Domain 1.4.

as a directed labeled graph, whose set of nodes is $M[\mathsf{W}]$ and whose set of edges contains $(\mathsf{w}_1, \mathsf{i}, \mathsf{w}_2)$[3] if and only if $(\mathsf{w}_1, \mathsf{w}_2) \in \mathcal{B}_\mathsf{i}$. $(\mathsf{w}_1, \mathsf{i}, \mathsf{w}_2)$ is referred to as an edge coming out of (resp. into) the world w_1 (resp. w_2).

Intuitively, a Kripke structure describes the possible worlds envisioned by the agents where the presence of multiple worlds identifies uncertainty. The relation $(\mathsf{w}_1, \mathsf{w}_2) \in \mathcal{B}_\mathsf{i}$ denotes that the beliefs of agent i about the characteristics of the domain are insufficient for her/him to distinguish between the configuration described by w_1 and the one described by w_2. This can be seen in Figure 1.8 where the two worlds are reachable from one to another by all the agents, meaning that agents A, B, and C are not able to distinguish between the worlds where the coin is heads or tails up. This results in agents' ignorance and we can say that A (and, similarly, B and C) *does not know* the coin position $(\neg\mathbf{B}_\mathsf{A}(\texttt{heads}) \wedge \neg\mathbf{B}_\mathsf{A}(\neg\texttt{heads}))$. On the other hand, since from the pointed world agent A (and, similarly, B and C) only reaches worlds where `haskey_A` is true we can say that A believes `haskey_A` ($\mathbf{B}_\mathsf{A}(\texttt{haskey\_A})$). Following the informal Definition 1.11, it is clear that the information contained in a Kripke structure suffices to represent an "e-state". In particular, the set of the possible worlds is captured by $M[\mathsf{W}]$ and the agents' beliefs can be derived by exploring the worlds' accessibility relations (starting from the real world).

More formally, in Definition 1.14, following Baral et al. [2015], we present how we can derive the truth value of belief formulae from an epistemic state representation, *i.e.*, a pointed Kripke structure. This definition allows us to provide semantics for

[3] $(\mathsf{w}_1, \mathsf{i}, \mathsf{w}_2)$ denotes the edge from node w_1 to node w_2, labeled with i.

the epistemic modal operators $\mathbf{B}_{\mathsf{i}}$ and $\mathbf{C}_{\alpha}$, where $\mathsf{i} \in \mathcal{AG} \supseteq \alpha$. To better express the semantics of the operator $\mathbf{C}_{\alpha}$ we will make use of an additional operator $\mathbf{E}_{\alpha}$. While $\mathbf{E}_{\alpha}$ does not add any expressiveness to the language it allows us to express the idea of common belief more elegantly. In fact, iterating on $\mathbf{E}^{k}_{\alpha}\varphi$ easily encodes the intuitive meaning of $\mathbf{C}_{\alpha}(\varphi)$; that is the conjunction of the following belief formulae: *(i)* every agent in α knows φ; *(ii)* every agent in α knows that every agent in α knows φ; *(iii)* and so on *ad infinitum*.

Definition 1.14: Entailment w.r.t. a Kripke structure

Given, a fluent $\mathtt{f}$, the belief formulae $\varphi, \varphi_1, \varphi_2$, an agent i, a group of agents α, and a pointed Kripke structure $(M = \langle \mathsf{W}, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_{\mathsf{n}} \rangle, \mathsf{w})$:

- $(M, \mathsf{w}) \models \mathtt{f}$ if $\mathtt{f} \in \pi(\mathsf{w})$ (or, alternatively, $\mathtt{f} \models_{\pi(\mathsf{w})} \top$);
- $(M, \mathsf{w}) \models \varphi$ if φ is a fluent formula and $\pi(\mathsf{w}) \models \varphi$ following the usual semantics of $\neg$ and $\wedge$;
- $(M, \mathsf{w}) \models \mathbf{B}_{\mathsf{i}}(\varphi)$ if for each t such that $(\mathsf{w}, \mathsf{t}) \in \mathcal{B}_{\mathsf{i}}$ it holds that $(M, \mathsf{t}) \models \varphi$;
- $(M, \mathsf{w}) \models \neg\varphi$ if $(M, \mathsf{w}) \not\models \varphi$;
- $(M, \mathsf{w}) \models \varphi_1 \wedge \varphi_2$ if $(M, \mathsf{w}) \models \varphi_1$ and $(M, \mathsf{w}) \models \varphi_2$; and
- $(M, \mathsf{w}) \models \mathbf{E}_{\alpha}\varphi$ if $(M, \mathsf{w}) \models \mathbf{B}_{\mathsf{i}}(\varphi)$ for all $\mathsf{i} \in \alpha$;
- $(M, \mathsf{w}) \models \mathbf{C}_{\alpha}\varphi$ if $(M, \mathsf{w}) \models \mathbf{E}^{k}_{\alpha}\varphi$ for every $k \geq 0$, where $\mathbf{E}^{0}_{\alpha}\varphi = \varphi$ and $\mathbf{E}^{k+1}_{\alpha}\varphi = \mathbf{E}_{\alpha}(\mathbf{E}^{k}_{\alpha}\varphi)$.

Axioms Systems

Following the works by Hintikka [1962], Fagin et al. [1995] let us now "quickly" define an axioms system for $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$—*i.e.*, the language of well-formed formulae over $\mathcal{F}$ and $\mathcal{AG}$. Such axiomatization will allow us to better categorize the properties of the language in relation to the structural constraints that an epistemic state representation, *e.g.*, a Kripke structure, must respect. In particular, we will provide the description of some properties, or axioms, on the e-states' relations—*i.e.*, the $\mathcal{B}_1, \ldots, \mathcal{B}_{\mathsf{n}}$ components of a pointed Kripke structure—or, more simply, its edges. These axioms, when respected, assure that the fundamental concepts of what we call *knowledge* and *beliefs* are preserved. To better understand what we are referring

Axiom	Property of $\mathcal{B}$
T	$\mathcal{B}_\mathsf{i}\varphi \Rightarrow \varphi$
4	$\mathcal{B}_\mathsf{i}\varphi \Rightarrow \mathcal{B}_\mathsf{i}\mathcal{B}_\mathsf{i}\varphi$
5	$\neg\mathcal{B}_\mathsf{i}\varphi \Rightarrow \mathcal{B}_\mathsf{i}\neg\mathcal{B}_\mathsf{i}\varphi$
D	$\neg\mathcal{B}_\mathsf{i}\bot$
K	$(\mathcal{B}_\mathsf{i}\varphi \wedge \mathcal{B}_\mathsf{i}(\varphi \Rightarrow \psi)) \Rightarrow \mathcal{B}_\mathsf{i}\psi$

Table 1.1: Knowledge and beliefs axioms [Fagin et al., 1995, chapter 3].

to, let us provide both the name and the formal definition of these axioms (the Left and Right column of Table 1.1, respectively).

Now we will give a brief description of the five axioms that we introduced in Table 1.1. More details on the axioms and their properties can be found in the work by Fagin et al. [1995, chapter 3].

- **T**: Has been introduced to capture the difference between knowledge and belief. When this axiom holds the real world must reflect the agents' knowledge, otherwise the agents might believe something that is not true in the actual configuration of the environment.

- **4**: Models the concept of positive introspection; this means that an agent must be aware of her/his beliefs.

- **5**: Models the concept of negative introspection; similarly to **4** an agent must be aware of what she/he does not believe.

- **D**: Introduced to ensure that an agent cannot believe "False".

- **K**: Expresses that the agent's beliefs are closed under logical consequence.

From now on, with $\mathbf{KD45}_n$-state we will indicate e-states that consider n agents and respect the axioms **4**, **5**, **D**, and **K**. Similarly we will refer to the e-states on n agents that respect all the aforementioned axioms (**T**, **4**, **5**, **D**, and **K**) as $\mathbf{S5}_n$-state.

Knowledge or Belief

As pointed out in the previous paragraphs the modal operator $\mathbf{B}_{\mathsf{i}}$ represents $M[\mathsf{i}]$—the world relations in a Kripke structure—and, as expected, different relations' properties imply different meanings for $\mathbf{B}_{\mathsf{i}}$. In particular, in our work, we are interested in representing the knowledge or the beliefs of the agents. The problem of formalizing these two concepts has been studied in depth bringing to an accepted formalization for both [Fagin et al., 1995]. If a relation[4] respects all the axioms presented in Table 1.1 it is called an **S5** relation and encodes the concept of knowledge, while when it respects all the axioms but **T** characterizes the concept of belief. That is, when reasoning about knowledge we must guarantee that the underlying representation is an $\mathbf{S5}_n$-state, while when we consider beliefs we need a $\mathbf{KD45}_n$-state. Following these characterizations, we will refer to knowledge and belief as **S5** and **KD45** logic, respectively.

Intuitively, the difference between the two logics is that an agent cannot *know* something that is not true in **S5** but she/he can *believe* it in **KD45**. While this difference may seem superficial, designing a planning system that deals with beliefs instead of knowledge requires much more attention and increases the difficulty of the solving process. In fact, a planner that reasons about knowledge can rely on a very important property that a solver that deals with beliefs cannot exploit. That is, a planning process based on knowledge can safely assume that the agents' information is always correct. In other words, once any agent knows a property she/he will maintain her/his knowledge even if the property changes its truth value. Moreover—since agents have to know the true nature of the information that they have—agents can also exploit the **T** axiom to reason about others' knowledge; *e.g.*, if an agent i knows that another agent j knows a property $\mathtt{p}$, then i knows $\mathtt{p}$. Furthermore, these properties, combined with the most commonly used epistemic actions (introduced in the next section), ensure that the agents' information increase monotonically. This implies that, when an agent learns something, that something can never be "unlearned" by that agent.

[4] In our case the relation between the worlds of a Kripke structure.

Contrarily, when we "drop" the **T** axiom, all of these assumptions do not hold anymore. This means that, when planning with beliefs, the solving process must account for the possibility that agents may:

- believe something that is not true in the real world;
- not being aware of changes about the truth value of already believed properties;
- believe something different from other agents;
- become ignorant about certain properties;
- announce/perceive something that does not correspond to the reality; and
- derive chains of beliefs of arbitrary length that cannot be collapsed into the same information.

All of these points make a planning system that takes into consideration beliefs, more intricate than one that is based on the **S5** logic. Nevertheless, planning on **KD45** presents the opportunity to model much more realistic (and interesting) scenarios and that is why, in this thesis, we tackle the problem of planning on beliefs.

1.4 Multi-agent Epistemic Planning

As already mentioned, reasoning about actions and information has always been one of the prominent interests since the beginning of AI [Russell and Norvig, 2010]. In particular, the continuous research effort that has characterized the field of autonomous planning is what ensured its rapid evolution. The "simple" task of reasoning in the *classical planning* environments rapidly evolved into more complex problems [Torreño et al., 2014]. This evolution, dictated both by research interests and real-world needs, developed interesting families of problems that vary in multiple aspects such as: *(i)* the number of *agents*; *(ii)* the determinism of the actions; *(iii)* the agent's communication policies; etc.

In particular, in this thesis, we are interested in the combination of the planning field and epistemic logic. While both of these research areas have been studied and

formalized since the early sixties, their combination, *i.e.*, *Multi-agent Epistemic Planning* (MEP), is a somewhat recent introduction in the Artificial Intelligence community [Van Ditmarsch et al., 2007]. Epistemic planners, differently from most of the other solvers, are not only interested in the state of the world but also in the *knowledge* or *beliefs* of the agents. This could also be viewed, as said by Gerbrandy [1999], as "*the process of reasoning on the information itself*". It is easy to see that an efficient autonomous reasoner that could exploit both the knowledge on the world and about other agents' information could provide an important tool in several scenarios, *e.g.*, economy, security, justice, or politics.

Nevertheless, reasoning about knowledge and beliefs is not as direct as reasoning on the "physical" state of the world. That is because expressing, for example, belief relations between agents often implies considering *nested* and *group* beliefs that are not easily extracted from the state description by a human reader. Even if several studies [Van Ditmarsch et al., 2007, Wan et al., 2015, Muise et al., 2015, Huang et al., 2017, Le et al., 2018] have been conducted on this topic, some fundamental complications remain while characterizing MEP. In particular, the inherent complexity of reasoning on beliefs is reflected in computational overhead that brings, most of the time, infeasibility to the solving process. Moreover, modeling subtle nuances of complex ideas—*e.g.*, *trust*, *lies*, *misconception*, and so on—that are necessarily present when we reason on beliefs, is a very intricate task that makes MEP even more difficult to completely grasp. These are some of the reasons why we deem it necessary to explore the field of Multi-agent Epistemic Planning.

1.4.1 Epistemic Actions

Before exploring what it means to plan on epistemic domains, let us briefly introduce the idea of *action* in epistemic logic. As said in Moss [2015], the formalization of various types of actions and, consequently of formal languages that incorporate them, is what originated the field of *Dynamic Epistemic Logic* (DEL). While DEL is not directly used in our thesis, the formalization provided by the works on this area [Fagin et al., 1995, van Eijck, 2004, Van Ditmarsch et al., 2007, Moss, 2015]

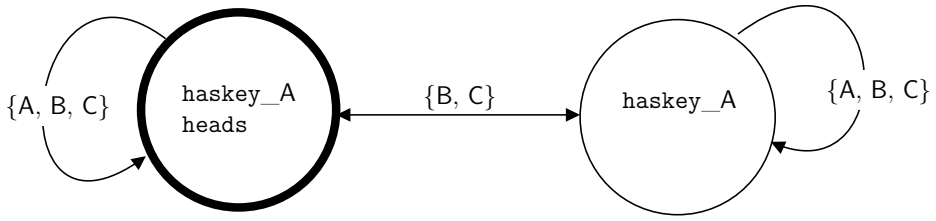


Figure 1.9: The Kripke structure that represents the Planning Domain 1.4 variation.

is what profoundly inspired our definition of *epistemic action languages* and their transition functions (presented in Chapter 2), fundamental components of our system. To better explain the concept of epistemic actions let use a variation of Planning Domain 1.4, as a running example, where we assume that: *(i)* agent A believes that the coin position is `heads` (Figure 1.9); and *(ii)* agents B and C believe that A knows the coin position without knowing it themselves (Figure 1.9); and *(iii)* the agents have the ability to *announce* publicly (*i.e.*, to all the other agents) some property (*i.e.*, a fluent) of the physical world. We capture the agents' capability with $\texttt{announce}\langle \mathsf{i} \rangle(\texttt{f})$, where $\mathsf{i} \in \mathcal{AG}$ is an agent that executes the action[5]; and `f` is the announced physical property. This action type is identified by the term *public announcement* and its informal semantics is: "the announcing agent tells everyone a property that she/he believes, making the other agents believe it too". This simple semantics does not consider concepts such as lying agents, or degrees of trust. Let us note that each action description (independently from the type) is associated with an *executability condition*, that is, a belief formula that when entailed permits the action itself to be executable.

To present the public announcements formal semantics we will follow Moss [2015]. Let us note that the execution of any action implies the possible modification of the underlying e-state. In particular, to model the semantics of a public announcement, agents must believe whatever has been announced. To do so, following the notion of entailment (Definition 1.14), we must ensure that no agent can reach a world where

[5]Distinguishing between the acting agent and the others is not necessary here, but let use this notation to be consistent with the rest of the thesis.

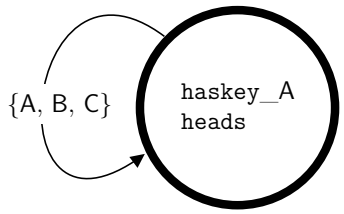


Figure 1.10: e-State of Figure 1.9 after the execution of $\texttt{announce}\langle \mathsf{A} \rangle(\texttt{heads})$.

the announced property is false. That is, the execution of $\texttt{announce}\langle \mathsf{A} \rangle(\texttt{heads})$ on the e-state depicted in Figure 1.9, must generate an e-state that contains only worlds that entails `heads`. This is accomplished by simply eliminating all the worlds that contain the negation of `heads` as shown in Figure 1.10. Let us note that the executability condition for this action is $\mathbf{B}_{\mathsf{A}}(\texttt{heads})$. Following Moss [2015], the formalization of the language that also contains the aforementioned semantics for public announcements is as follows:

$$\varphi ::= \mathtt{f} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{B}_{\mathsf{i}}(\varphi) \mid \mathbf{C}_{\alpha}(\varphi) \mid [!\mathtt{f}]\varphi,$$

where $\mathtt{f} \in \mathcal{F}$, φ is belief formula over $\mathcal{AG}$ and $\mathcal{F}$, $\mathsf{i} \in \mathcal{AG}$ and $\emptyset \neq \alpha \subseteq \mathcal{AG}$. While the epistemic operators $\mathbf{B}$ and $\mathbf{C}$ have already been formalized, the newly introduced operator $[!\mathtt{f}]\varphi$ needs to be defined. Contrarily to $\mathbf{B}_{\mathsf{i}}(\varphi)$ and $\mathbf{C}_{\alpha}(\varphi)$, that operate on a static e-state, the operator $[!\mathtt{f}]\varphi$ must take into account also the updated version of the e-state (and that is what transforms epistemic logic into *dynamic* epistemic logic). In particular, we read the operator as "if φ is respected in the current state then, after the execution of the announcement `f` must be believed by every agent"[6]. The axiomatization of this operator, following Moss [2015, equation 6.6], is:

$$(M, \mathtt{w}) \models [!\mathtt{f}]\varphi \text{ iff } (M, \mathtt{w}) \not\models \varphi, \text{ or else } (M_{\mathtt{f}}, \mathtt{w}) \models \mathtt{f}.$$

where $(M_{\mathtt{f}}, \mathtt{w})$ is the epistemic state $(M, \mathtt{w})$ updated after the execution of the announcement.

[6]The case when an agent announces $\neg f$ is similar.

Public announcement is just one of the possible actions formalized in DEL. Since we will rely on action languages (typical of planning domains), we will make use of a formalization that is akin to the one used in the planning area. Therefore, we will not further explore DEL, addressing the interested reader to Van Ditmarsch et al. [2007], Moss [2015] for more examples of epistemic actions and a much more detailed introduction on dynamic epistemic logic.

1.4.2 Multi-agent Epistemic Planning Problem

Bolander and Andersen [2011] define epistemic planning as the generation of plans for multiple agents to achieve goals which can refer to the state of the world, the beliefs of agents, and/or the knowledge of agents. After the introduction of the classical planning problem, in the early days of Artificial Intelligence, several studies have provided the foundations for several successful approaches to automated planning. However, the main focus of these research efforts has been about reasoning within single-agent domains. In single-agent domains, reasoning about actions and change mainly involves reasoning about what is true in the world, what the agent knows about the world, how the agent can manipulate the world (using world-changing actions) to reach particular states, and how the agent (using sensing actions) can learn unknown aspects of the world.

In multi-agent domains an agent's action may not just change the world and the agent's beliefs about the world, but also may change other agents' beliefs about the world and their beliefs about other agents' beliefs. Similarly, the goals of an agent in a multi-agent world may involve manipulating the beliefs of other agents.

Although there is a large body of research on multi-agent planning Fagin et al. [1995], Durfee [2001], Bernstein et al. [2002], Guestrin et al. [2001], De Weerdt et al. [2003], Goldman and Zilberstein [2004], De Weerdt and Clement [2009], Allen and Zilberstein [2009], Muise et al. [2015], Baral et al. [2015, 2022], very few efforts address the above aspects of epistemic domains which pose several research challenges in representing and reasoning about actions and change.

Let us now formally introduce the notion of Multi-agent Epistemic Planning domain in the following definition.

Definition 1.15: MEP Domain

A *Multi-agent Epistemic Planning domain* is a tuple $D = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}, \varphi_{ini}, \varphi_{goal} \rangle$, where $\mathcal{F}$, $\mathcal{AG}$, $\mathcal{A}$ are the sets of *fluents*, *agents*, *actions* of D, respectively; φ_{ini} and φ_{goal} are DEL formulae that must be entailed by the *initial* and *goal e-state*, respectively. The former e-state describes the domain's initial configuration while the latter encodes the desired one.

A MEP domain contains the information needed to describe a planning problem in a multi-agent epistemic setting. Given a domain D we refer to its elements through the parenthesis operator; *e.g.*, the fluent set of D will be denoted by $D(\mathcal{F})$. An *action instance* $\mathtt{a}\langle\alpha\rangle \in D(\mathcal{AI}) = D(\mathcal{A}) \times 2^{D(\mathcal{AG})}$ identifies the execution of action $\mathtt{a}$ by a set of agents α. Multiple executors are needed in certain types of actions, for example in the so-called sensing actions (introduced in detail in the next chapters). On the other hand, actions like the public announcement introduced above, only require one executor ($|\alpha| = 1$). The *transition function* $\Phi : D(\mathcal{AI}) \times D(\mathcal{S}) \to D(\mathcal{S}) \cup \{\emptyset\}$ formalizes the semantics of action instances (the result is the empty set if the action instance is not executable). Formal definitions of this concept will be introduced in Chapters 2 to 4 where we will analyze in detail diverse transition functions. Intuitively, the features of Planning Domain 1.5 (the Planning Domain 1.4 completed with actions and goal descriptions) are:

- $\mathcal{F} = \{\mathtt{heads},\ \mathtt{haskey\_X},\ \mathtt{opened}\}$ where $\mathsf{X} \in \mathcal{AG}$;
- $\mathcal{AG} = \{\mathsf{A},\ \mathsf{B},\ \mathsf{C}\}$;
- $\mathcal{A} = \{\mathtt{open},\ \mathtt{peek},\ \mathtt{announce}\}$;
- $\varphi_{ini} = \mathtt{heads} \wedge \mathtt{haskey\_A} \wedge \mathbf{C}_{\mathcal{AG}}(\mathtt{haskey\_A}) \wedge \mathbf{C}_{\mathcal{AG}}(\neg\mathtt{haskey\_B}) \wedge \mathbf{C}_{\mathcal{AG}}(\neg\mathtt{haskey\_C}) \wedge \mathbf{C}_{\mathcal{AG}}(\neg\mathtt{opened})$;
- $\varphi_{goal} = \mathbf{B}_{\mathsf{A}}(\mathtt{heads}) \wedge \mathbf{B}_{\mathsf{B}}(\mathtt{heads}) \wedge \mathbf{B}_{\mathsf{C}}(\mathtt{heads})$;

Planning Domain 1.5: Coin in the Box with Actions (Simplified)

Three agents, A, B, and C, are in a room where in the middle there is a box. The box has a lock that can only be opened with a key. Inside the box, there is a coin that lies *heads* up. In the initial configuration of this domain we have that everybody knows that:

- none of the agents know whether the coin lies heads or tails up;
- the box is locked; and
- only agent A has the key.

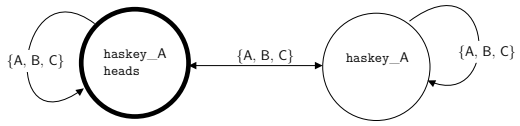
Moreover, we have that each agent can execute one of the following actions:

- `open`: an agent, if she/he has the key, can open the box. This results in all the agents believing that the box is open.
- `peek`: to learn whether the coin lies heads or tails up, an agent can peek into the box, but this requires the box to be open. This will result in the peeking agents knowing the coin position while the other agents are aware of this without knowing the coin position themselves.
- `announce`: following the public announcement semantics this will result in all the agents believing the announced coin position. As before, this action is only executable by an agent who believes the coin position to be `heads`.

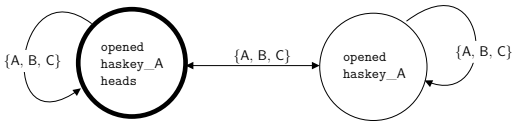
Finally, the desired configuration, *i.e.*, the goal, of this instance is that all the agents (A, B, and C) believe that the coin is `heads` up, *i.e.*, $\mathbf{B}_{\mathsf{A}}(\texttt{heads}) \wedge \mathbf{B}_{\mathsf{B}}(\texttt{heads}) \wedge \mathbf{B}_{\mathsf{C}}(\texttt{heads})$.

The correct solution (or plan) that permits the instance presented in Planning Domain 1.5 to reach its desired goal is the sequence of action instances $\langle \texttt{open}\langle \mathsf{A}\rangle, \texttt{peek}\langle \mathsf{X}\rangle, \texttt{announce}\langle \mathsf{X}\rangle(\texttt{heads})\rangle$ where $\mathsf{X} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. In Figure 1.11 we present the plan execution, representing the e-state resulting after the execution of each action.

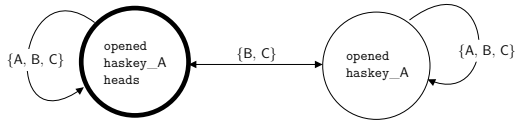
We can see that in Figure 1.11b the execution of $\texttt{open}\langle \mathsf{A}\rangle$ modified some property of the physical world, namely, the fluent `opened` became true. We will refer to this type of action, *i.e.*, the ones who modify some physical property, with the term *ontic* or, sometimes, *world-altering*. Ontic actions resemble the actions that we can find in classical planning. On the other hand, the actions that only deal with agents' beliefs, *e.g.*, `peek` and `announce`, are referred to as *epistemic actions*. We will see



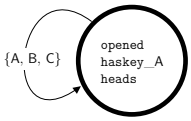
(a) The initial e-state described in Planning Domain 1.5.



(b) The e-state obtained after the execution of $\texttt{open}\langle\mathsf{A}\rangle$.



(c) The e-state obtained after the execution of $\texttt{peek}\langle\mathsf{A}\rangle$.



(d) The e-state obtained after the execution of $\texttt{announce}\langle\mathsf{A}\rangle(\texttt{heads})$.

Figure 1.11: The execution of the plan $\langle\texttt{open}\langle\mathsf{A}\rangle,\ \texttt{peek}\langle\mathsf{A}\rangle,\ \texttt{announce}\langle\mathsf{A}\rangle(\texttt{heads})\rangle$.

in Chapter 2 how these two types of actions have different transition functions.

In Figure 1.11c we assume that the executor is agent A. This means that A must believe that the coin is heads up. To ensure this we simply remove all the edges that, from the pointed world, allow A to reach a world where `heads` is not true. We leave the edge that allows A to loop on the right world of Figure 1.11c as this is used to capture that the other agents do not know which coin position A believes to be true.

Finally, Figure 1.11d is derived following the public announcement semantics presented in Section 1.4.1. We can see how the final e-state (Figure 1.11d) respects the given goal, *i.e.*, $\mathbf{B}_{\mathsf{A}}(\texttt{heads}) \wedge \mathbf{B}_{\mathsf{B}}(\texttt{heads}) \wedge \mathbf{B}_{\mathsf{C}}(\texttt{heads})$.

Let us remember that this paragraph is supposed to only provide an introduction to the field of Multi-agent Epistemic Planning. We will explore more in detail these concepts later, when we will present the contributions of this thesis.

1.4.3 Complexity Overview

Finally, as the last note on MEP, we will summarize the complexity results in the epistemic logic and in the epistemic planning fields. We will not present details on such results as we introduced them only to provide the reader with a general idea on "how hard" the problem of reasoning on information change is.

Let us start by providing some basic notions that we will use throughout this paragraph in Definitions 1.16 to 1.18.

Definition 1.16: Satisfiability of a Formula

Given a formula φ, φ is satisfiable if it is possible to find an interpretation, in our case an e-state, that makes the formula true.

Definition 1.17: Model Checking of a Formula

Given a formula φ and a model M (in our case an e-state), the model checking problem consists in determining if φ is true in M.

Definition 1.18: Plan Existence Problem

The plan existence problem consists of determining, if it exists, a solution, as defined in definition 1.6, for a planning domain D.

These definitions identify the problems of interest when planning on beliefs. Assuming that we make use of Kripke structures—other representations have the same results—to represent the e-states then: *(i)* Definition 1.16 is the problem of verifying whether there exists or not a Kripke structure that entails a formula; *(ii)* Definition 1.16 identifies the entailment of a formula over a given a Kripke structure; and *(iii)* Definition 1.16 represents the complete planning process. This means that identifying the complexity of these problems will allow us to characterize the MEP domain and provide us with a rough idea of how intricate is to tackle this setting. We now present a series of results that summarize the complexity of the aforementioned problems (Proposition 1.1, Tables 1.2 and 1.3). All the presented results are derived by Fagin et al. [1995], Bolander et al. [2015].

Proposition 1.1: Model Checking Complexity [Fagin et al., 1995]

There is an algorithm that, given a pointed Kripke structure (M, w)*, and a formula* $\varphi \in \mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$*, determines, in* POLYNOMIAL *time* $\mathcal{O}(||M|| \times |\varphi|)$ *whether* $(M, \mathsf{w}) \models \varphi$*, where* $||M|| = |M[\mathsf{W}]| + \sum_{i=0}^{n} |M[\mathsf{i}]|$ *with* $\mathsf{i} \in \mathcal{AG}$*, and* $|\varphi|$ *is the number of nested operators in* φ*.*

SAT Complexity	Epistemic logic
NP-COMPLETE	$\mathbf{S5}_1$, $\mathbf{KD45}_1$
PSPACE-COMPLETE	$\mathbf{S5}_n$, $\mathbf{KD45}_n$ with $n \geq 2$
EXPTIME-COMPLETE	$\mathbf{S5}_n^{\mathbf{C}}$, $\mathbf{KD45}_n^{\mathbf{C}}$ with $n \geq 2$

Table 1.2: Complexity of the satisfiability problem with respect to the underlying Kripke structure constraints [Fagin et al., 1995].

Let us note that to analyze the *plan existence problem* we need to categorize action types into four distinct subsets. Depending on which subset an action type belongs to, the action type itself impacts differently the e-state update. This means that certain subsets of action types may increase the complexity of the plan existence problem (as we can see in Table 1.3) when taken into consideration. In MEP, as we will see in more detail in Chapter 2, the actions are often represented through graphs. These action-graphs may collapse in more "simple" data structures, *i.e.*, *singletons*, *chains*, or *trees*, for some actions and that is what makes that action part of a subset rather than another.

In particular, Bolander et al. [2015] distinguish between three different types of action-structures:

- *singletons*: that corresponds to public announcements of propositional facts;
- *chains* and *trees*: that corresponds to different types of private announcements; and
- *graphs*: that capture any propositional epistemic actions.

Moreover, for each one of these classes of structures, Bolander et al. analyze the complexity for the plan existence problem considering also the effects and preconditions expressive power:

- non-factual actions (changing only beliefs) with propositional preconditions;
- factual actions (changing beliefs and fluents) with propositional preconditions; and
- factual actions with epistemic preconditions.

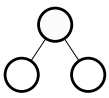
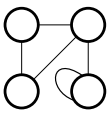
Underlying **Action Structure**	**Effects/Preconditions** types		
	Non-Factual Propositional	Factual Propositional	Factual Epistemic
Singleton 	NP-complete [Bolander et al., 2015]	PSPACE-hard [Jensen, 2014]	PSPACE-hard [Jensen, 2014]
Chain 	NP-complete [Bolander et al., 2015]	**Open Question**	**Open Question**
Tree 	PSPACE-complete [Bolander et al., 2015]	**Open Question**	**Open Question**
Graph 	EXPSPACE [Bolander et al., 2015]	NON-Elementary [Yu et al., 2013]	Undecidable [Bolander and Andersen, 2011]

Table 1.3: Complexity of the *plan existence problem* [Bolander et al., 2015].

In Table 1.3 the complexity of the plan existence problem, depending on the action type and on the underlying action structure, is summarised. As expected, the complexity of the problem increases as we loosen the restrictions on the underlying structure. Unfortunately, we are interested in the subset of actions that are represented through graphs without restrictions and that have factual epistemic preconditions (the right-bottom cell of Table 1.3). While these results are on a general Kripke structure, *i.e.*, not constrained by any **S5** axiom, Bolander et al. [2015] show that even the plan existence problem on $\mathbf{S5}_n$-states (more limited than $\mathbf{KD45}_n$-states, in which we are mostly interested) is reducible to the *halting problem* that it is well known to be undecidable.

[...] the (unobserved) past, like the future, is indefinite and exists only as a spectrum of possibilities.

— Stephen Hawking
The Grand Design
[Hawking and Mlodinow, 2010]

2

Possibilities-Based MEP Action Language

Contents

2.1 Background

While in Chapter 1 we presented a general introduction of the topics of this thesis, in this section we will explore in more detail the concepts that are required to describe the first contribution of our work, *i.e.*, the *multi-agent epistemic action language* $m\mathcal{A}^\rho$. We will start in Section 2.1.1 where we will present the MEP action language $m\mathcal{A}^*$ [Baral et al., 2015, Le et al., 2018, Baral et al., 2022], which has served as the foundation for our work. In Section 2.1.2 we will, then, introduce the theory of *non-well-founded* sets [Aczel, 1988] that is required to better understand the data structure that is used as a basis for $m\mathcal{A}^\rho$. Finally, again in Section 2.1.2, we will define formally the aforementioned data structure, referred to as *possibility*

in literature [Gerbrandy and Groeneveld, 1997, Gerbrandy, 1999], highlighting important properties that make it well-suited for representing e-states.

2.1.1 The Epistemic Action Language $m\mathcal{A}^*$

With the introduction of the classical planning problem, languages for representing actions and their effects were also proposed [Fikes and Nilsson, 1971]. These languages are referred to as *action languages* [Gelfond and Lifschitz, 1998].

Over the years, several action languages for single-agent scenarios (*e.g.*, STRIPS [Fikes and Nilsson, 1971], ADL [Pednault, 1994] and SAS+ [Bäckström, 1995]) have been developed providing the foundation for several successful approaches to automated planning. The effort of defining languages for classical planning domains culminated in the well-known *Planning Domain Description Language* (PDDL) [McDermott et al., 1998, Fox and Long, 2003] that standardized the notations and that is routinely adopted by planners. Nonetheless, as said by Baral et al. [2015]: "*in single-agent domains, reasoning about actions and change mainly involves reasoning about what is true in the world, what the agent knows about the world, how the agent can manipulate the world (using world-changing actions) to reach particular states, and how the agent (using sensing actions) can learn unknown aspects of the world.*"

On the other hand, multi-agent epistemic domains—the type of domain we are considering—need more careful consideration when it comes to actions effects. In particular, a MEP action language should be able to model how actions affect both the environment and the agents' beliefs (about the environment or others' beliefs). Similarly, the description of the states (be it an initial or a goal state) may involve the agents' beliefs. Few studies directly address the challenges derived by domains in which information flows must be taken into consideration.

To the best of our knowledge, two works, *i.e.* Baral et al. [2015] and Muise et al. [2015], firstly tackled the problem of providing formal action languages for Multi-agent Epistemic Planning domains. In particular, in this section, we will

illustrate $m\mathcal{A}^*$ [Le et al., 2018, Baral et al., 2022][1]—the evolution of the language $m\mathcal{A}+$ provided by "*An Action Language for Multi-Agent Domains: Foundations*" by Baral et al. [2015]—as it is the foundation of our newly introduced language $m\mathcal{A}^\rho$.

We decided to develop a language starting from $m\mathcal{A}^*$, rather than PDKB-PDDL [Muise et al., 2015], because of the nature of the planning process employed by the planners related to these languages. In fact, we thought that $m\mathcal{A}^*$ to be more in line with our objective of defining a comprehensive epistemic environment that reasons on the full extent of $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$. While both planners can achieve these results, $m\mathcal{A}^*$ plans on a search space where each state is effectively a complete e-state, while PDKB-PDDL makes use of a conversion into classical planning. Intuitively, the latter *transforms* e-states properties into classical states to obtain a faster solving process but renouncing to reason on "full-fledged" epistemic models. We, therefore, preferred to define a system that, even if with a more resource-heavy procedure, is able to reason and update complete e-states representation, *e.g.*, Kripke structures.

Before formally introducing $m\mathcal{A}^*$ we need to provide some notations that are paramount to describe the language semantics. This introduction is supposed to provide the reader with enough information to understand, at an intuitive level, the characteristics of $m\mathcal{A}^*$ and, therefore, does not provide all the details of the language. For a complete analysis of the language, we address the interested reader to Baral et al. [2022] where $m\mathcal{A}^*$ is extensively analyzed.

The first idea that is necessary to introduce is the notion of *event model* (also called *update model*) [Baltag and Moss, 2004, Van Benthem et al., 2006]. In $m\mathcal{A}^*$, the event models are used to define how the execution of actions impacts an e-state, that is, they provide a formal way of defining how an action execution updates the epistemic states. Let us now define the concept of update models, along with the idea of *substitution* (necessary to define update models).

[1]Let us note that we will use Baral et al. [2015] and Baral et al. [2022] as main references for $m\mathcal{A}^*$ as they define its syntax and semantics exhaustively. In fact, Le et al. [2018] only illustrate the additions to the language with respect to $m\mathcal{A}+$ redirecting the readers to Baral et al. [2015] for further information on the language.

Definition 2.1: $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$-substitution [Baral et al., 2015]

Let $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ be a language defined over a set $\mathcal{AG}$ of n agents and a set $\mathcal{F}$ of k fluents. An $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$-*substitution* is a set $\{\mathtt{f}_1 \to \varphi_1, \ldots, \mathtt{f}_k \to \varphi_k\}$, where each $\mathtt{f}_i$ is a distinct proposition in $\mathcal{F}$ and each $\varphi_i \in \mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$. We will implicitly assume that for each $\mathtt{f} \in \mathcal{F} \setminus \{\mathtt{f}_1, \ldots, \mathtt{f}_k\}$, the substitution contains $\mathtt{f} \to \mathtt{f}$. $SUB_{(\mathcal{F},\mathcal{AG})}$ denotes the set of all $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$-substitutions.

Definition 2.2: Event Model [Baral et al., 2015]

Given a language $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$, defined over a set $\mathcal{AG}$ of n agents and a set $\mathcal{F}$ of k fluents, an *event model* $\mathbf{\Sigma}$ is a tuple $\langle \mathcal{E}, Q, pre, sub \rangle$ where:

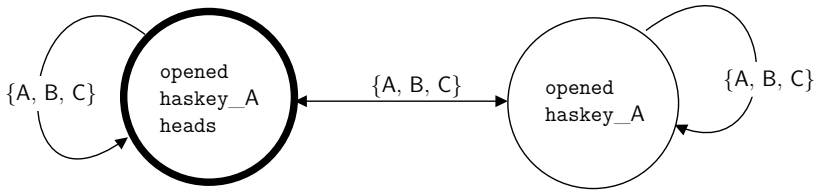
- $\mathcal{E}$*:* is a set, whose elements are called *events*;
- Q*:* $\mathcal{AG} \to 2^{\mathcal{E}\times\mathcal{E}}$ assigns an accessibility relation to each agent $\mathsf{i} \in \mathcal{AG}$;
- *pre*: $\mathcal{E} \to \mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ is a function mapping each event $e \in \mathcal{E}$ to a formula in $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$; and
- *sub:* $\mathcal{E} \to SUB_{(\mathcal{F},\mathcal{AG})}$ is a function mapping each event $e \in \mathcal{E}$ to a substitution in $SUB_{(\mathcal{F},\mathcal{AG})}$.

The idea behind event models is to provide a way to formally define an action that can correctly alter the underlying e-state representation (let us imagine a Kripke structure). Definition 2.3 illustrates how the information encoded in an update model (Figure 2.1b) is used to obtain the correct e-state update.

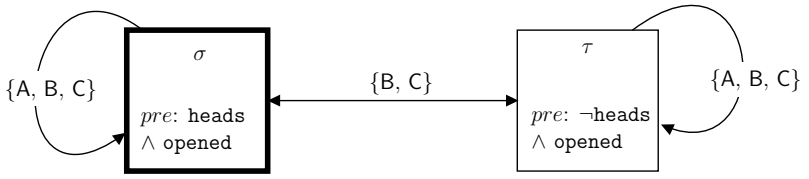
Definition 2.3: Update by Event Models [Baral et al., 2022]

Let $(\mathbf{\Sigma}, \gamma)$ be an *update template*, where $\mathbf{\Sigma} = \langle \mathcal{E}, Q, pre, sub \rangle$ is an event model and $\gamma \in \mathcal{E}$, and lrt (M, w) be an epistemic state. The execution of $(\mathbf{\Sigma}, \gamma)$ in (M, w) results in an epistemic state $(M', \mathsf{w}) = (M, \mathsf{w}) \otimes (\mathbf{\Sigma}, \gamma)$, where:

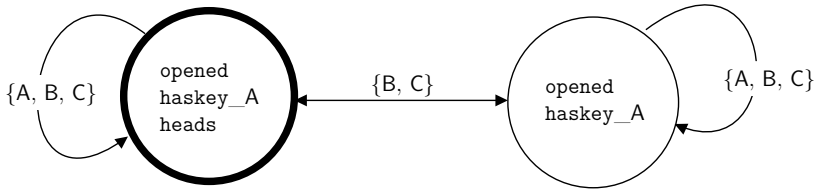
- $M'[\mathsf{W}] = \{(\mathsf{t}, e) \in M[\mathsf{W}] \times \mathcal{E} \mid (M, \mathsf{t}) \models pre(e)\}$;
- $M'[\mathsf{w}] = (M[\mathsf{w}], \gamma)$;
- $((\mathsf{t}_1, e_1), (\mathsf{t}_2, e_2)) \in M'[\mathsf{i}]$ iff $(\mathsf{t}_1, e_1), (\mathsf{t}_2, e_2) \in M'[\mathsf{W}], (\mathsf{t}_1, \mathsf{t}_2) \in M[\mathsf{i}]$ and $(e_1, e_2) \in Q$;
- For all $(\mathsf{w}, e) \in M'[W]$ and $\mathtt{f} \in \mathcal{F}$, $M'[\pi]((\mathsf{w}, e)) \models \mathtt{f}$ iff $\mathtt{f} \to \varphi \in sub(e)$ and $(M, \mathsf{w}) \models \varphi$.



(a) The e-state that represents the configuration where the box is opened



(b) The representation of the update template $(\mathbf{\Sigma}, \sigma)$ relative to $\texttt{peek}\langle \mathsf{A} \rangle$. The substitutions are not indicated as they are equal to $\emptyset$.



(c) The updated e-state after the execution of $\texttt{peek}\langle \mathsf{A} \rangle$.

Figure 2.1: The execution of an action instance through the application of Definition 2.3.

Let us note that, for simplicity, we assume that the event of interest, *i.e.*, γ, is exactly one and that the given e-state has one pointed world. These restrictions do not affect the definition of the language's properties needed in our introduction. Nonetheless, having a single-pointed world at each step means that the planning process implicitly discards the idea of executing conformant planning (where multiple unknown initial states should be kept into account). Since the language envisioned by Baral et al. is able to tackle also incomplete descriptions of the world, these assumptions are relaxed in their work [Baral et al., 2022].

Multiple events in a single update model correspond to multiple degrees of

observability; in particular Figure 2.1b represents the update model of the action instance $\texttt{peek}\langle\mathsf{A}\rangle$, introduced in Planning Domain 1.5. Here, only agent A becomes aware of the status of the coins, while the others learn that A knows the coin position without knowing it themselves—this corresponds to *partial observability*. We, therefore, have that the event σ corresponds to agent A learning the coin status, while event τ represents the other agents being aware of the peeking action. Figure 2.1 illustrates the result of applying the procedure described in Definition 2.3, with the update template in Figure 2.1b, to the Kripke structure that represents the state where the box is opened (Figure 2.1a) obtaining the correctly updated e-state (Figure 2.1c).

In what follows, we will introduce the syntax and the semantics of $m\mathcal{A}^*$ that will make use of more complex event models and observability relations. Once again, we will make use of the Coin in the Box domain. In particular, the example in Planning Domain 2.1 is a more complete version of the ones present in the previous chapter.

Each one of the actions presented in Planning Domain 2.1 falls into one of the three types distinguished by Baral et al. [2022]. In particular, these action types are:

- *World-altering* actions (also called *ontic*): used to modify certain properties (*i.e.*, fluents) of the world, *e.g.*, the actions `open` or $\texttt{distract\_}\mathsf{X}$ of Planning Domain 2.1.

- *Sensing* action: used by an agent to refine her/his beliefs about the world, *e.g.*, the action `peek` of Planning Domain 2.1.

- *Announcement* action: used by an agent to affect the beliefs of other agents. *e.g.*, in Planning Domain 2.1 the action `announce`.

Planning Domain 2.1: Three Agents and the Coin in the Box

Three agents, A, B, and C, are in a room where in the middle there is a box. The box has a lock that can only be opened with a key. Inside the box, there is a coin that lies *heads* up. In the initial configuration of this domain we have that everybody knows that:

- none of the agents know whether the coin lies heads or tails up;
- the box is locked;
- only agent A has the key;
- if an agent is attentive (identified by `look_X` with $\mathsf{X} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$) she/he is aware of the execution of the actions; and
- agents A and C are attentive while B is not.

Moreover, we have that each agent can execute one of the following actions:

- `open`: an agent, if she/he has the key, can open the box. This results in all the *attentive* agents believing that the box is open, while the others would not be aware of any change in the environment.
- `peek`: to learn whether the coin lies heads or tails up, an agent can peek into the box, but this requires the box to be open. This will result in the peeking agents believing the coin position while the other attentive agents are aware of this without knowing the coin position themselves.
- `announce`: this will result in all the listening (*i.e.*, attentive) agents to believe that the coin lies heads or tails up depending on the announced value. As before, for our configuration, this action is only executable by an agent who believes the coin position to be heads.
- `distract_X`/`signal_X`: these actions will make an attentive agent X no more attentive or vice-versa, respectively.

Finally, in the desired configuration, A would like to know whether the coin lies heads or tails up. She/He would also like to make agent B aware of this fact. However, A would like to keep this information secret from C.

A series of action instances—that is, a plan—to achieve the goal may be **(1)** `distract_C`$\langle\mathsf{A}\rangle$: to distract C from looking at the box; **(2)** `signal_B`$\langle\mathsf{A}\rangle$: to tell B to look at the box; **(3)** `open`$\langle\mathsf{A}\rangle$: to open the box; and **(4)** `peek`$\langle\mathsf{A}\rangle$: to make A peek into the box.

Given a domain D, an action instance $\mathtt{a} \in D(\mathcal{AI})$, a fluent literal $\mathtt{f} \in D(\mathcal{F})$, a fluent formula $\phi \in \mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ and a belief formula $\varphi \in \mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$, where $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ is defined

Action type	Full observers	Partial Observers	Oblivious
World-altering	✓		✓
Sensing	✓	✓	✓
Announcement	✓	✓	✓

Table 2.1: Action types and observability relations Baral et al. [2015].

over $D(\mathcal{AG})$ and $D(\mathcal{F})$, we can "briefly" introduce the syntax adopted in $m\mathcal{A}^*$. Executability conditions are captured by statements of the form:

executable a **if** φ;

for ontic actions we have:

a **causes** f **if** φ;

sensing actions statements have the form expressed by:

a **determines** f;

finally, announcement actions are expressed as follows:

a **announces** ϕ.

In multi-agent domains, the execution of an action might change or not the beliefs of an agent. This is because, in such domains, each action instance associates an observability relation to each agent. For example, agent C—that becomes oblivious after being distracted by A—is not able to see the execution of the action open$\langle$A$\rangle$. On the other hand, any agent who is watching a sensing or an announcement action can change her/his beliefs; *e.g.*, agent B, who is watching agent A sensing the status of the coin, will know that A knows the status of the coin without knowing it her/him-self. Table 2.1 summarizes the possible observability relations for each type of action. Partial observability for world-altering action is not admitted as, whenever an agent is aware of the execution of an ontic action, she/he must know its effects on the world as well. To indicate the set of agents that belong to the **F**ull, **P**artial, and **O**blivious observers we will use **F**, **P**, and **O**, respectively. The

idea of observability is captured in $m\mathcal{A}^*$ with specific statements. In particular, to state that an agent i is **F**ully observant, with respect to an action a, it is used:

$$\mathsf{i} \textbf{ observes } \mathsf{a} \textbf{ if } \varphi;$$

while to identify an agent i as **P**artially observant, with respect to an action a, it is used:

$$\mathsf{i} \textbf{ aware\_of } \mathsf{a} \textbf{ if } \varphi.$$

Notice that if we do not state otherwise, an agent will be considered oblivious. Finally, statements of the form

$$\textbf{initially } \varphi;$$

and

$$\textbf{goal } \varphi$$

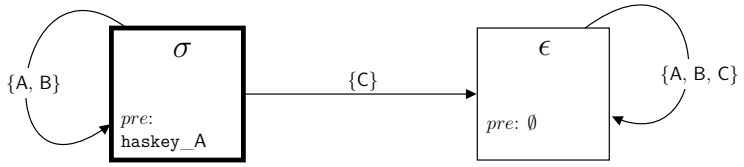
capture the initial and goal conditions, respectively.

The core of the language semantics is the transition function, and as already mentioned, it is defined using the concept of update models. In Figure 2.2a, Figure 2.2b, and Figure 2.2c we illustrate the update templates for ontic, sensing, and announcement actions, using $\mathtt{open}\langle\mathsf{A}\rangle$, $\mathtt{peek}\langle\mathsf{A}\rangle$, $\mathtt{announce}\langle\mathsf{A}\rangle$, respectively. The starting state is the one where A and B are attentive while C is not. That is, we can see A as representative for the fully observant, B as partially observant, and C as oblivious. For the sake of the presentation let us assume that B is partially observant also for the announcement action execution.

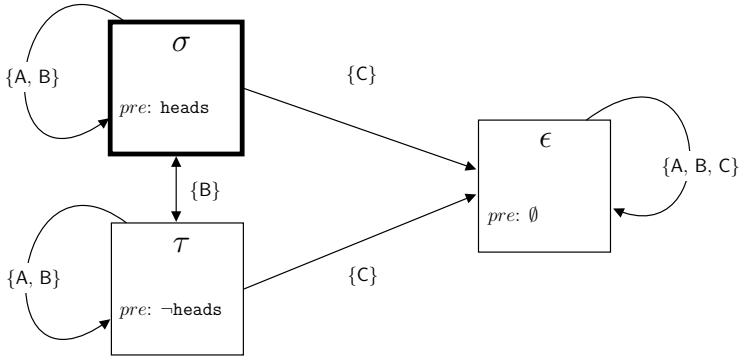
Finally, we can define the $m\mathcal{A}^*$ transition function Φ. Let (M, w) be an e-state and let $\mathsf{a} \in D(\mathcal{AI})$. The result of executing a on (M, w) is the e-state, denoted by $\Phi(\mathsf{a}, (M, \mathsf{w}))$ defined as follow:

- If a is not executable in (M, w) then $\Phi(\mathsf{a}, (M, \mathsf{w})) = \emptyset$
- If a is executable in (M, w) and $(\mathbf{\Sigma}, \sigma)$ is the representation of the occurrence of a on (M, w) then $(M', w) = (M, \mathsf{w}) \otimes (\mathbf{\Sigma}, \sigma)$.

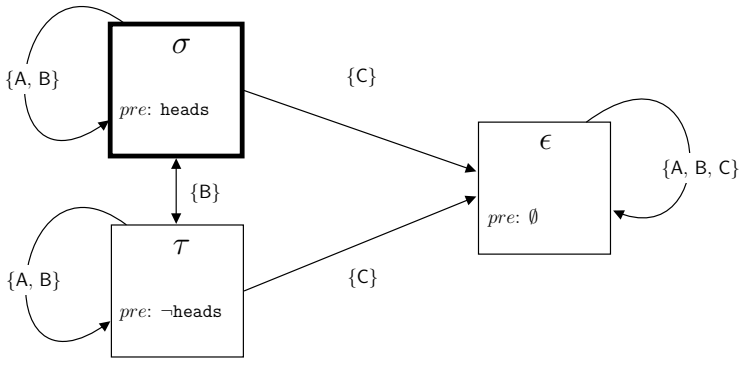
For more details and examples we address, once again, the reader to Baral et al. [2022].



(a) The update template $(\mathbf{\Sigma}, \sigma)$ of the ontic action instance $\texttt{open}\langle\mathsf{A}\rangle$. The substitution in σ intuitively add the fluent `haskey_A`, and removes its negation. In ϵ the substitution is equal to $\emptyset$.

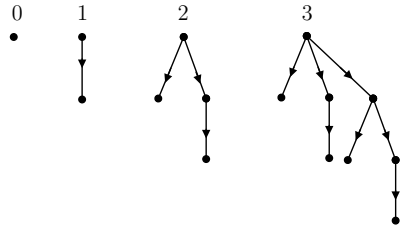


(b) The update template $(\mathbf{\Sigma}, \sigma)$ of the sensing action instance $\texttt{peek}\langle\mathsf{A}\rangle$. The substitutions are equal to $\emptyset$.

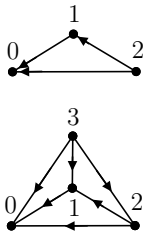


(c) The update template $(\mathbf{\Sigma}, \sigma)$ of the announcement action instance $\texttt{announce}\langle\mathsf{A}\rangle$. For the sake of the presentation we assume B to be partially observant. The substitutions are equal to $\emptyset$.

Figure 2.2: Examples of update templates for each action type described by Baral et al. [2022].



(a) Pictures of von Neumann ordinals where $0 = \emptyset$; $1 = \{\emptyset\}$; $2 = \{\emptyset, \{\emptyset\}\}$; $3 = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$.



(b) Alternative Pictures of von Neumann ordinals 2 and 3.

Figure 2.3: Well-founded sets represented through graphs [Aczel, 1988].

2.1.2 Possibilities

We are now ready to define the concept of *possibility* (originally introduced by Gerbrandy and Groeneveld [1997]). This section aims to provide the reader with enough information to understand what possibilities are, without providing all the details behind this topic. More on possibilities can be found in the works by Gerbrandy and Groeneveld [1997], Gerbrandy [1999], while we refer the reader to Aczel [1988] for a complete introduction on non-well-founded set theory and to Dovier [2015] for an introduction of non-well-founded sets and their equivalence in logic programming.

Non-well-founded Set Theory Fundamentals Let us start by giving some fundamental definitions of set theory. According to Aczel [1988], a *well-founded* and a *non-well-founded set* are defined as follows:

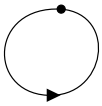
Definition 2.4: Well-founded Set

Let E be a set, E' one of its elements, E'' any element of E', and so on. A descent is the sequence of steps from E to E', E' to E'', etc. ... A set is *well-founded* (or *ordinary*) when it only gives rise to finite descents.

Definition 2.5: Non-well-founded Set [Aczel, 1988]

A set is *non-well-founded* (or *extraordinary*) when among its descents there are some which are infinite.

All sets, in the sense of Definition 2.4, can be represented in the form of graphs, called *pictures*, as shown in Figure 2.3. The concept of *picture of a set* is introduced, alongside the definition of *decoration*, in Definition 2.6.



(a) Standard picture Ω.

(b) Unfolding of the picture of Ω.

Figure 2.4: Representation of the non-well-founded set $\Omega = \{\Omega\}$ [Aczel, 1988].

Definition 2.6: Decoration and Picture

- A *decoration* of a graph $\mathcal{G} = (V, E)$ is a function δ that assigns to each node $\mathsf{n} \in V$ a set δ_{n} in such a way that the elements of δ_{n} are exactly the sets assigned to successors of n, *i.e.*, $\delta_{\mathsf{n}} = \{\delta_{\mathsf{n}'} \mid (\mathsf{n}, \mathsf{n}') \in E\}$. Therefore, the edges denote the membership relations.
- If δ is a decoration of a pointed graph $(\mathcal{G}, \mathsf{n})$, then $(\mathcal{G}, \mathsf{n})$ is a *picture* of the set δ_{n}.

These concepts are essential to investigate the differences between well-founded and non-well-founded set theories. We know that in well-founded set theory, it holds Mostovski's lemma: "*each well-founded graph*[2] *is a picture of exactly one set*" [Mostowski, 1949]. On the other hand, when the **Foundation Axiom**, expressed by Gerbrandy [1999] as "*Only well-founded graphs have decorations*", is substituted with the **Anti-Foundation Axiom** (**AFA**), expressed by Aczel [1988] as "*Every graph has a unique decoration*", the following consequences become true:

- every graph is a picture of exactly one set (**AFA** as is formulated by Gerbrandy [1999]);
- non-well-founded sets exist given that a non-well-founded pointed graph has to be a picture of a non-well-founded set.

Aczel [1988], Gerbrandy [1999] point out how non-well-founded sets can also be expressed through systems of equations. This concept will help us to formalize the notion of state in our action language $m\mathcal{A}^{\rho}$. A quick example of this representation can be derived by the set $\Omega = \{\Omega\}$ (Figure 2.4). We can informally define this set

[2] A well-founded graph is a graph that doesn't contain an infinite path $\mathsf{n} \to \mathsf{n}' \to \mathsf{n}'' \to \ldots$ of successors.

as the (singleton) system of equations $x = \{x\}$. Systems of equations and their solutions are described more formally in Definition 2.7.

Definition 2.7: System of Equations [Gerbrandy, 1999]

For each class of atoms, *i.e.*, objects that are not sets and have no further set-theoretic structure, $\mathcal{X}$ a *system of equations* in $\mathcal{X}$ is a class τ of equations $\mathtt{x} = \mathtt{X}$, where $\mathtt{x} \in \mathcal{X}$ and $\mathtt{X} \subseteq \mathcal{X}$, such that τ contains exactly one equation $\mathtt{x} = \mathtt{X}$ for each $\mathtt{x} \in \mathcal{X}$. A solution to a system of equations τ is a function δ that assigns to each $\mathtt{x} \in \tau(\mathcal{X})$[a] a set $\delta_{\mathtt{x}}$ such that $\delta_{\mathtt{x}} = \{\delta_{\mathtt{y}} \mid \mathtt{y} \in \mathtt{X}\}$, where $\mathtt{x} = \mathtt{X}$ is an equation of τ. If δ is the solution to a system of equations τ, then the set $\{\delta_{\mathtt{x}} \mid \mathtt{x} \in \tau(\mathcal{X})\}$ is called the solution set of that system.

[a] $\tau(\mathcal{X})$ denotes the class of atoms $\mathcal{X}$ in which τ is described.

Since both graphs and systems of equations are representations for non-well-founded sets, it is natural to investigate their relationships. In particular, it is interesting to point out how from a graph $\mathcal{G} = (V, E)$ it is possible to construct a system of equations τ and vice versa. The nodes in $\mathcal{G}$, in fact, can be the set of atoms $\tau(\mathcal{X})$ and, for each node $\mathtt{v} \in V$, an equation is represented by $\mathtt{v} = \{\mathtt{v}' \mid (\mathtt{v}, \mathtt{v}') \in E\}$. Since each graph has a unique decoration, each system of equations has a unique solution. Nonetheless, different graphs can represent the same set, and the notion that can help to identify this equivalence is known as *bisimulation*.

Bisimulation As mentioned before the idea of *bisimulation* can be exploited to characterize graphs, and therefore Kripke models, with "the same behavior". In particular, it can be proved that there is a unique minimum graph bisimilar to a given one, and it can be found by computing the *maximum bisimulation*. Bisimilar labeled graphs (or Kripke structures) have therefore a unique solution as well since we collapse their representations into the minimal one. While this topic is of the utmost importance in modal logic, and has been studied and used in several fields, it is not the aim of this thesis to study it in depth. Let us, therefore, only provide some basic notions referring the interested readers to much more complete works such as Gerbrandy [1999], Bolander et al. [2015], Dovier [2015]. In particular, Definition 2.8 formally introduces the concepts of bisimulation and Figure 2.5

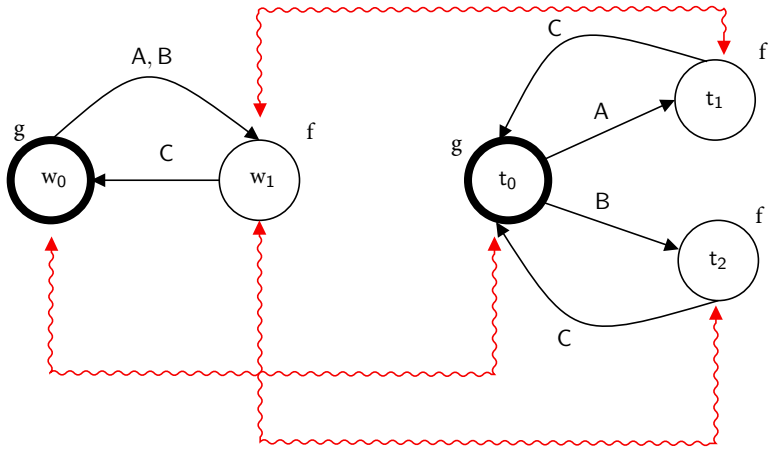


Figure 2.5: In Figure are represented two different pointed Kripke structures (M_1, w_0) (left) and (M_2, t_0) (right). It is easy to see that these two Kripke structures are structurally different but bisimilar since there exists a relation $\{(\mathsf{w}_0, \mathsf{t}_0), (\mathsf{w}_1, \mathsf{t}_1), (\mathsf{w}_1, \mathsf{t}_2)\}$ that is a bisimulation [Riouak, 2019].

presents a graphical example of such concept. Finally, Corollary 2.1 states how the concept of entailment and bisimulation are intertwined in Kripke structures.

Definition 2.8: Bisimulation [Bolander et al., 2015]

Given two pointed Kripke structures (M_1, w_0) and (M_2, t_0) let $\mathsf{W} = M_1[\mathsf{W}]$ and $\mathsf{T} = M_2[\mathsf{W}]$, and let $\mathcal{F}$ be a set of propositional variables. The relation $\mathcal{R} \subseteq \mathsf{W} \times \mathsf{T}$ is a *bisimulation* if and only if for all $\mathsf{w} \in \mathsf{W}$ and $\mathsf{t} \in \mathsf{T}$, if $(\mathsf{w}, \mathsf{t}) \in \mathcal{R}$ then:

- **Atom:** $\mathsf{w} \models \mathtt{f} \iff \mathsf{t} \models \mathtt{f}$, for all $\mathtt{f} \in \mathcal{F}$;
- **Forth:** For all w' such that $(\mathsf{w}, \mathsf{w}') \in M_1[\mathcal{B}]$ there exists a t' such that $(\mathsf{t}, \mathsf{t}') \in M_2[\mathcal{B}]$ and $(\mathsf{w}', \mathsf{t}') \in \mathcal{R}$;
- **Back:** For all t' such that $(\mathsf{t}, \mathsf{t}') \in M_2[\mathcal{B}]$ there exists a w' such that $(\mathsf{w}, \mathsf{w}') \in M_1[\mathcal{B}]$ and $(\mathsf{t}', \mathsf{w}') \in \mathcal{R}$;

Two pointed Kripke structures (M_1, w_0) and (M_2, t_0) are *bisimilar*, indicated with $(M_1, \mathsf{w}_0) \simeq (M_2, \mathsf{t}_0)$, if and only if there exists a bisimulation between M_1 and M_2 such that $(\mathsf{s}_0, \mathsf{t}_0) \in \mathcal{R}$.

Corollary 2.1: Truth in Bisimilar Kripke Structures

Given two pointed Kripke structures (M_1, w_0) *and* (M_2, t_0) *and a language* $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$*, we have that:*

$$(M_1, \mathsf{w}_0) \simeq (M_2, \mathsf{t}_0) \iff (\forall \varphi \in \mathcal{L}^{\mathbf{C}}_{\mathcal{AG}} : (M_1, \mathsf{w}_0) \models \varphi \iff (M_2, \mathsf{t}_0) \models \varphi).$$

Possibilities Let us now introduce the notion of possibility, following Gerbrandy and Groeneveld [1997]:

Definition 2.9: Possibilities [Gerbrandy and Groeneveld, 1997]

Let $\mathcal{AG}$ be a set of agents and $\mathcal{F}$ a set of propositional variables:

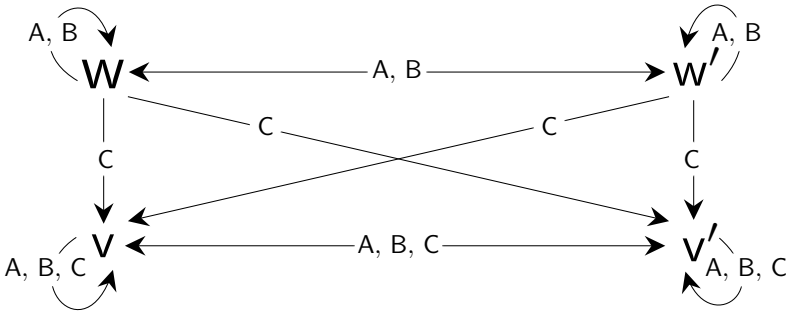
- A *possibility* u is a function that assigns to each propositional variable $\mathtt{f} \in \mathcal{F}$ a truth value $\mathsf{u}(\mathtt{f}) \in \{0, 1\}$ and to each agent $\mathsf{i} \in \mathcal{AG}$ an information state $\mathsf{u}(\mathsf{i}) = \sigma$.
- An *information state* σ is a set of possibilities.

In Section 2.2 we will use this concept to describe an epistemic state. The intuition behind this is that a possibility u is a "possible" interpretation of the world and of the agents' beliefs. That is, $\mathsf{u}(\mathtt{f})$ specifies the truth value of the fluent $\mathtt{f}$ in u and $\mathsf{u}(\mathsf{a})$ is the set of all the interpretations that agent a considers possible in u.

Moreover, a possibility can be represented as a decoration (Definition 2.6) of a labeled graph and, therefore, as a unique solution to a system of equations for possibilities (Definition 2.10), as shown in Figure 2.6. A possibility represents the solution to the minimal system of equations in which all bisimilar systems of equations are collapsed; namely, the possibilities that represent decorations of bisimilar labeled graphs are bisimilar and can be represented by the minimal one. This shows that the class of bisimilar labeled graphs and, therefore, of bisimilar Kripke structures, used by $m\mathcal{A}^*$ as e-states, can be represented by a single possibility.

Definition 2.10: System of Equations for Possibilities [Gerbrandy, 1999]

Given a set of agents $\mathcal{AG}$ and a set of propositional variables $\mathcal{F}$, a *system of equations for possibilities* in a class of possibilities $\mathcal{X}$ is a set of equations such that for each $\mathsf{x} \in \mathcal{X}$ there exists exactly one equation of the form $\mathsf{x}(\mathtt{f}) = b$, where $b \in \{0, 1\}$, for each $\mathtt{f} \in \mathcal{F}$, and of the form $\mathsf{x}(\mathsf{i}) = \mathtt{X}$, where $\mathtt{X} \subseteq \mathcal{X}$, for each $\mathsf{i} \in \mathcal{AG}$.
A solution to a system of equations for possibilities is a function δ that assigns to each atom x a possibility δ_{x} in such a way that if $\mathsf{x}(\mathtt{f}) = i$ is an equation then $\delta_{\mathsf{x}(\mathtt{f})} = i$, and if $\mathsf{x}(\mathsf{i}) = \sigma$ is an equation, then $\delta_{\mathsf{x}(\mathsf{i})} = \{\delta_{\mathsf{y}} \mid \mathsf{y} \in \sigma\}$.



(a) Picture of w.

$$\begin{cases} \mathsf{w} & = \{(\mathsf{A}, \{\mathsf{w}, \mathsf{w}'\}), (\mathsf{B}, \{\mathsf{w}, \mathsf{w}'\}), (\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \mathsf{f}, \mathsf{g}, \mathsf{h}\} \\ \mathsf{w}' & = \{(\mathsf{A}, \{\mathsf{w}, \mathsf{w}'\}), (\mathsf{B}, \{\mathsf{w}, \mathsf{w}'\}), (\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \mathsf{g}, \mathsf{h}\} \\ \mathsf{v} & = \{(\mathsf{A}, \{\mathsf{v}, \mathsf{v}'\}), (\mathsf{B}, \{\mathsf{v}, \mathsf{v}'\}), (\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \mathsf{f}, \mathsf{h}\} \\ \mathsf{v}' & = \{(\mathsf{A}, \{\mathsf{v}, \mathsf{v}'\}), (\mathsf{B}, \{\mathsf{v}, \mathsf{v}'\}), (\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \mathsf{h}\} \end{cases}$$

(b) System of equations of w.

Figure 2.6: Representation of a generic possibility w. The possibility is expanded for clarity.

Finally, in Proposition 2.1, we show some interesting relations between labeled graphs and possibilities, while in Proposition 2.2 we summarize important properties that capture the relations between Kripke structures and possibilities.

Proposition 2.1: Labeled Graphs and Possibilities
[Gerbrandy and Groeneveld, 1997]

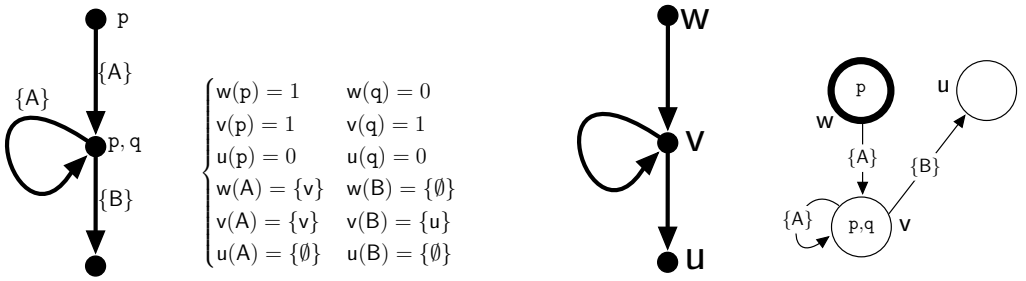
The relations between labeled graphs and possibilities are summarized as follows:

- *each possibility can be pictured by a labeled graph;*
- *each labeled graph has a unique decoration;*
- *two labeled graphs have the same decoration if and only if are bisimilar.*

Proposition 2.2: Kripke Structures and Possibilities

Given a Kripke structure (M, w)*, a possibility* u*, and a language* $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$*, we have that:*

- *each pointed Kripke structure has exactly one set as its solution;*
- *two models are bisimilar if and only if they are the picture of the same set; and*
- *If* (M, w) *is a picture of* u*, then for each* $\varphi \in \mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ *it holds* $(M, \mathsf{w}) \models \varphi \iff \mathsf{u} \models \varphi$.



(a) A possibility. **(b)** Its system of equation. **(c)** The solution. **(d)** The corresponding Kripke structure.

Figure 2.7: An e-state represented through a possibility **(a)** and then converted to a Kripke structure **(d)**.

2.2 The Epistemic Action Language $m\mathcal{A}^{\rho}$

The aforementioned idea of possibilities is central in $m\mathcal{A}^{\rho}$. This language, instead of using Kripke structures, exploits possibilities as e-states (Figure 2.7). That is, $m\mathcal{A}^{\rho}$, while keeping the same syntax of $m\mathcal{A}^{*}$, changes the way of representing an epistemic state. The modification of the underlying structure implies also a different formalization of the transition function. This allowed us to define a new planning environment that outperforms the state-of-the-art comprehensive epistemic planner presented by [Le et al., 2018] by orders of magnitude in the experiments.

We will now briefly explain how a possibility can be used to represent an e-state. The main idea is to identify with each possibility u both an interpretation of the world and of each agent's beliefs. That is, the component $\mathsf{u}(\mathtt{f})$ assigns a truth value to the fluent $\mathtt{f}$ in u while $\mathsf{u}(\mathsf{i})$ represents the (non-well-founded) set of possibilities that are considered by agent i.

The choice of possibilities over Kripke structures as e-state representation provides several advantages. One of these is, as said by Gerbrandy and Groeneveld [1997], that: "*a possibility represents the solution to the minimal system of equations in which all bisimilar Kripke structures are collapsed*". Intuitively, this means that a class of bisimilar Kripke structures, that in $m\mathcal{A}^{*}$ represents different e-states, is easily represented by a single possibility and therefore, by a single e-state in $m\mathcal{A}^{\rho}$. That is, thanks to possibilities and the newly introduced transition function

it has been possible to maintain e-states with smaller size, with respect to the planner EFP 1.0 presented by Le et al. [2018], during the solving process. From a more concrete point of view, implementing $m\mathcal{A}^{\rho}$ allowed us to work on e-states of reduced dimension[3] without having to rely on minimization techniques, such as the algorithms presented by Paige and Tarjan [1987], Dovier et al. [2004], during the solving process. Another advantage of using possibilities derives from their non-well-founded aspect. Since a possibility is a non-well-founded graph, whose nodes are themselves possibilities, the solving process can store each calculated possibility; and, whenever needed, it can retrieve the stored possibilities to reuse them as "nodes" inside a new e-state. To summarize, although possibilities and Kripke structures are tightly connected (Figure 2.7), the advantages of using $m\mathcal{A}^{\rho}$ are: *(i)* the reduced size of the e-states that does not depend on external procedures; and *(ii)* the fact that possibilities can be stored and easily reused thanks to their non-well-founded nature. In this sense, we can see possibilities as a more compact representation, with respect to Kripke structures, that allows us to save computational resources.

2.2.1 The Language Specification

As the first main contribution, we present the language $m\mathcal{A}^{\rho}$[4]. $m\mathcal{A}^{\rho}$ borrows the syntax from $m\mathcal{A}^{*}$ but changes the underlying e-state representation from Kripke structures to possibilities. After rapidly introducing the concept of entailment we will describe an improved transition function for $m\mathcal{A}^{\rho}$ along with some important properties.

Let us start with the concept of entailment for possibilities. Definition 2.11 combines the concept of Gerbrandy [1999] with the action language $m\mathcal{A}^{*}$.

[3]With respect to the e-states generated following $m\mathcal{A}^{*}$.

[4]The original version of $m\mathcal{A}^{\rho}$ was presented by Fabiano et al. [2019]. Instead, here we will introduce a newer version that maintains the same core while optimizing some details.

Definition 2.11: Entailment in Possibilities

Given, a fluent $\mathtt{f}$, the belief formulae $\varphi, \varphi_1, \varphi_2$, an agent i, a group of agents α, and a and a possibility u:

(i) $\mathsf{u} \models \mathtt{f}$ if $\mathsf{u}(\mathtt{f}) = 1$;

(ii) $\mathsf{u} \models \varphi$ if φ is a fluent formula and $\mathsf{u} \models \varphi$ following the standard semantics for $\neg$ and $\wedge$;

(iii) $\mathsf{u} \models \mathbf{B}_{\mathsf{i}}(\varphi)$ if for each $\mathsf{v} \in \mathsf{u}(\mathsf{i})$, $\mathsf{v} \models \varphi$;

(iv) $\mathsf{u} \models \neg\varphi$ if $\mathsf{u} \not\models \varphi$;

(v) $\mathsf{u} \models \varphi_1 \wedge \varphi_2$ if $\mathsf{u} \models \varphi_1$ and $\mathsf{u} \models \varphi_2$;

(vi) $\mathsf{u} \models \mathbf{E}_{\alpha}\varphi$ if $\mathsf{u} \models \mathbf{B}_{\mathsf{i}}(\varphi)$ for all $\mathsf{i} \in \alpha$;

(vii) $\mathsf{u} \models \mathbf{C}_{\alpha}(\varphi)$ if $\mathsf{u} \models \mathbf{E}^k_{\alpha}\varphi$ for every $k \geq 0$, where $\mathbf{E}^0_{\alpha}\varphi = \varphi$ and $\mathbf{E}^{k+1}_{\alpha}\varphi = \mathbf{E}_{\alpha}(\mathbf{E}^k_{\alpha}\varphi)$.

We are now ready to introduce the transition function. This new transition function is,in our opinion, more compact and therefore, more approachable than the one introduced by Le et al. [2018]. Moreover, the "simplicity" of the e-states update formalization is reflected in a much cleaner and faster implementation, as we will see in Chapter 5. Let a domain D, its set of action instances $D(\mathcal{AI})$, and the set $\mathcal{S}$ of all the possibilities reachable from $D(\varphi_{ini})$ with a finite sequence of action instances be given. Moreover let us identify the observability groups $\mathbf{F}$, $\mathbf{P}$, and $\mathbf{O}$ with respect to an action instance $\mathtt{a}$ with $\mathbf{F}_{\mathtt{a}}$, $\mathbf{P}_{\mathtt{a}}$, and $\mathbf{O}_{\mathtt{a}}$, respectively. The transition function $\Phi : D(\mathcal{AI}) \times \mathcal{S} \to \mathcal{S} \cup \{\emptyset\}$ for $m\mathcal{A}^{\rho}$ relative to D is formalized following Definition 2.12.

Definition 2.12: $m\mathcal{A}^{\rho}$ transition function

Allow us to use the compact notation $\mathsf{u}(\mathcal{F}) = \{\mathtt{f} \mid \mathtt{f} \in D(\mathcal{F}) \wedge \mathsf{u} \models \mathtt{f}\} \cup \{\neg\mathtt{f} \mid \mathtt{f} \in D(\mathcal{F}) \wedge \mathsf{u} \not\models \mathtt{f}\}$ for the sake of readability. Let an action instance $\mathtt{a} \in D(\mathcal{AI})$, a possibility $\mathsf{u} \in \mathcal{S}$ and an agent $\mathsf{i} \in D(\mathcal{AG})$ be given. If $\mathtt{a}$ is not executable in u, then $\Phi(\mathtt{a}, \mathsf{u}) = \emptyset$ otherwise $\Phi(\mathtt{a}, \mathsf{u}) = \mathsf{u}'$, where:

- Let us consider the case of an *ontic* action instance $\mathtt{a}$. We then define u'

such that:

$$e(\mathsf{a},\mathsf{u}) = \{\ell \mid (\mathsf{a}\ \textbf{causes}\ \ \ell) \in D\};\ \text{and}$$
$$\overline{e(\mathsf{a},\mathsf{u})} = \{\neg\ell \mid \ell \in e(\mathsf{a},\mathsf{u})\}\ \text{where}\ \neg\neg\ell\ \text{is replaced by}\ \ell.$$

$$\mathsf{u}'(\mathtt{f}) = \begin{cases} 1 & \text{if } \mathtt{f} \in (\mathsf{u}(\mathcal{F}) \setminus \overline{e(\mathsf{a},\mathsf{u})}) \cup e(\mathsf{a},\mathsf{u}) \\ 0 & \text{if } \neg\mathtt{f} \in (\mathsf{u}(\mathcal{F}) \setminus \overline{e(\mathsf{a},\mathsf{u})}) \cup e(\mathsf{a},\mathsf{u}) \end{cases}$$
$$\mathsf{u}'(\mathsf{i}) = \begin{cases} \mathsf{u}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_\mathsf{a} \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{i})} \Phi(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{F}_\mathsf{a} \end{cases}$$

- if a is a *sensing* action instance, used to determine the fluent $\mathtt{f}$. We then define u' such that:

$$\begin{aligned} e(\mathsf{a},\mathsf{u}) =& \{\mathtt{f} \mid (\mathsf{a}\ \textbf{determines}\ \ \mathtt{f}) \in D \wedge \mathsf{u} \models \mathtt{f}\} \\ & \cup \{\neg\mathtt{f} \mid (\mathsf{a}\ \textbf{determines}\ \ \mathtt{f}) \in D \wedge \mathsf{u} \not\models \mathtt{f}\} \end{aligned}$$

$$\mathsf{u}'(\mathcal{F}) = \mathsf{u}(\mathcal{F})$$
$$\mathsf{u}'(\mathsf{i}) = \begin{cases} \mathsf{u}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_\mathsf{a} \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{i})} \Phi(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{P}_\mathsf{a} \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{i}):\ e(\mathsf{a},\mathsf{w})=e(\mathsf{a},\mathsf{u})} \Phi(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{F}_\mathsf{a} \end{cases}$$

- if a is an *announcement* action instance of the fluent formula ϕ. We then define u' such that:

$$e(\mathsf{a},\mathsf{u}) = \begin{cases} 0 & \text{if } \mathsf{u} \models \phi \\ 1 & \text{if } \mathsf{u} \models \neg\phi \end{cases}$$

$$\mathsf{u}'(\mathcal{F}) = \mathsf{u}(\mathcal{F})$$
$$\mathsf{u}'(\mathsf{i}) = \begin{cases} \mathsf{u}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_\mathsf{a} \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{i})} \Phi(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{P}_\mathsf{a} \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{i}):\ e(\mathsf{a},\mathsf{w})=e(\mathsf{a},\mathsf{u})} \Phi(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{F}_\mathsf{a} \end{cases}$$

2.2.2 The Language Properties

The newly introduced transition function allowed us to reason about fundamental properties that, as said by Baral et al. [2015], each multi-agent epistemic action language should respect. In particular, each epistemic reasoner should ensure that:

- if an agent is fully aware of the execution of an action instance then her/his beliefs will be updated with the effects of such action execution;
- an agent who is only partially aware of the action occurrence will believe that the agents who are fully aware of the action occurrence are certain about the effects of the actions; and
- an agent who is oblivious of the action occurrence will also be ignorant about its effects.

Propositions 2.3 to 2.5 capture the concept of beliefs update and ensure that, when satisfied, the action language can be soundly used for multi-agent epistemic reasoning. For the sake of readability, their complete proofs are reported in Appendix A.2. In the following, we will use p' instead of $\Phi(\mathsf{a}, \mathsf{p})$ when possible to avoid unnecessary clutter.

Proposition 2.3: Ontic Action Properties

Assume that a *is an* ontic action instance *executable in* u *s.t.* a ***causes*** ℓ *if* ψ *belongs to* D*. In* $m\mathcal{A}^{\rho}$ *it holds that:*

(1) *for every agent* $\mathsf{x} \in \mathbf{F}_{\mathsf{a}}$*, if* $\mathsf{u} \models \mathbf{B}_{\mathsf{x}}(\psi)$ *then* $\mathsf{u}' \models \mathbf{B}_{\mathsf{x}}(\ell)$*;*

(2) *for every agent* $\mathsf{y} \in \mathbf{O}_{\mathsf{a}}$ *and a belief formula* φ*,* $\mathsf{u}' \models \mathbf{B}_{\mathsf{y}}(\varphi)$ *iff* $\mathsf{u} \models \mathbf{B}_{\mathsf{y}}(\varphi)$*; and*

(3) *for every pair of agents* $\mathsf{x} \in \mathbf{F}_{\mathsf{a}}$ *and* $\mathsf{y} \in \mathbf{O}_{\mathsf{a}}$ *and a belief formula* φ*, if* $\mathsf{u} \models \mathbf{B}_{\mathsf{x}}(\mathbf{B}_{\mathsf{y}}(\varphi))$ *then* $\mathsf{u}' \models \mathbf{B}_{\mathsf{x}}(\mathbf{B}_{\mathsf{y}}(\varphi))$*.*

Proposition 2.4: Sensing Action Properties

Assume that $\mathtt{a}$ *is a* sensing action instance *and D contains the statement* $\mathtt{a}$ ***determines*** $\mathtt{f}$. *In* $m\mathcal{A}^{\rho}$ *it holds that:*

(1) *if* $\mathsf{u} \models \mathtt{f}$ *then* $\mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathtt{a}}\mathtt{f}$;

(2) *if* $\mathsf{u} \models \neg\mathtt{f}$ *then* $\mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathtt{a}}\neg\mathtt{f}$;

(3) $\mathsf{u}' \models \mathbf{C}_{\mathbf{P}_\mathtt{a}}(\mathbf{C}_{\mathbf{F}_\mathtt{a}}\mathtt{f} \vee \mathbf{C}_{\mathbf{F}_\mathtt{a}}\neg\mathtt{f})$;

(4) $\mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathtt{a}}(\mathbf{C}_{\mathbf{P}_\mathtt{a}}(\mathbf{C}_{\mathbf{F}_\mathtt{a}}\mathtt{f} \vee \mathbf{C}_{\mathbf{F}_\mathtt{a}}\neg\mathtt{f}))$;

(5) *for every agent* $\mathsf{y} \in \mathbf{O}_\mathtt{a}$ *and a belief formula* φ, $\mathsf{u}' \models \mathbf{B}_\mathsf{y}(\varphi)$ *iff* $\mathsf{u} \models \mathbf{B}_\mathsf{y}(\varphi)$; *and*

(6) *for every pair of agents* $\mathsf{x} \in \mathbf{F}_\mathtt{a}$ *and* $\mathsf{y} \in \mathbf{O}_\mathtt{a}$ *and a belief formula* φ, *if* $\mathsf{u} \models \mathbf{B}_\mathsf{x}(\mathbf{B}_\mathsf{y}(\varphi))$ *then* $\mathsf{u}' \models \mathbf{B}_\mathsf{x}(\mathbf{B}_\mathsf{y}(\varphi))$.

Proposition 2.5: Announcement Action Properties

Assume that $\mathtt{a}$ *is a* announcement action instance *and D contains the statement* $\mathtt{a}$ ***announces*** φ. *If* $\mathsf{u} \models \phi$ *in* $m\mathcal{A}^{\rho}$ *it holds that:*

(1) $\mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathtt{a}}\phi$;

(2) $\mathsf{u}' \models \mathbf{C}_{\mathbf{P}_\mathtt{a}}(\mathbf{C}_{\mathbf{F}_\mathtt{a}}\phi \vee \mathbf{C}_{\mathbf{F}_\mathtt{a}}\neg\phi)$;

(3) $\mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathtt{a}}(\mathbf{C}_{\mathbf{P}_\mathtt{a}}(\mathbf{C}_{\mathbf{F}_\mathtt{a}}\phi \vee \mathbf{C}_{\mathbf{F}_\mathtt{a}}\neg\phi))$;

(4) *for every agent* $\mathsf{y} \in \mathbf{O}_\mathtt{a}$ *and a belief formula* φ, $\mathsf{u}' \models \mathbf{B}_\mathsf{y}(\varphi)$ *iff* $\mathsf{u} \models \mathbf{B}_\mathsf{y}(\varphi)$; *and*

(5) *for every pair of agents* $\mathsf{x} \in \mathbf{F}_\mathtt{a}$ *and* $\mathsf{y} \in \mathbf{O}_\mathtt{a}$ *and a belief formula* φ, *if* $\mathsf{u} \models \mathbf{B}_\mathsf{x}(\mathbf{B}_\mathsf{y}(\varphi))$ *then* $\mathsf{u}' \models \mathbf{B}_\mathsf{x}(\mathbf{B}_\mathsf{y}(\varphi))$.

Baral et al. [2015] show how the above-listed properties capture the concept of update in an epistemic environment. Therefore, we consider the epistemic action languages, that respect all of the aforementioned properties, to be correct with respect to the knowledge/belief update. That is the case with both $m\mathcal{A}^{*}$ and $m\mathcal{A}^{\rho}$.

2.2.3 $m\mathcal{A}^{*}$ and $m\mathcal{A}^{\rho}$ Comparison

Finally, for a clearer understanding on differences between the two languages, let us show *(i)* the execution, on both $m\mathcal{A}^{*}$ and $m\mathcal{A}^{\rho}$, of the action instances sequence Δ;

	distract_C⟨A⟩	open⟨A⟩	peek⟨A⟩
F_D	A, B, C	A, B	A
P_D	-	-	B
O_D	-	C	C

Table 2.2: Observability relations of the actions instances in Δ.

and *(ii)* a direct comparison of the number of worlds and edges created by $m\mathcal{A}^*$ and $m\mathcal{A}^\rho$ when executing a sequence of action instances.

Plan Execution $\Delta = \texttt{distract\_C}\langle \mathsf{A}\rangle, \texttt{open}\langle \mathsf{A}\rangle, \texttt{peek}\langle \mathsf{A}\rangle$ is the sequence that leads to the desired goal in Planning Domain 2.1 if we assume that B is already looking at the box. With this, we want to give a graphical explanation of both the transition functions and state-space defined by the two languages (Figures 2.8 to 2.11). Each state in $m\mathcal{A}^*$ will be represented by a Kripke structure while in $m\mathcal{A}^\rho$ will be a possibility (expanded to its respective system of equations for clarity). The observability relations of each action instance in Δ are expressed in Table 2.2.

Assuming that $\alpha = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, then the initial state, based on a small variation of Planning Domain 2.1 where we assume B to be already attentive, is defined by the conditions:

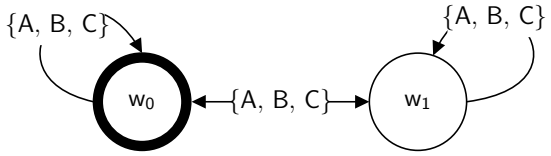
- **initially** $\mathbf{C}_\alpha(\texttt{haskey\_A}) \wedge \mathbf{C}_\alpha(\neg\texttt{haskey\_B}) \wedge \mathbf{C}_\alpha(\neg\texttt{haskey\_C})$
- **initially** $\mathbf{C}_\alpha(\neg\texttt{opened})$
- **initially** $\mathbf{C}_\alpha(\neg\mathbf{B}_\mathsf{i}(\texttt{heads}) \wedge \neg\mathbf{B}_\mathsf{i}(\neg\texttt{heads}))$ for $\mathsf{i} \in \alpha$
- **initially** $\mathbf{C}_\alpha(\texttt{look\_i})$ for $\mathsf{i} \in \alpha$
- **initially** heads

Finally, the goal of Planning Domain 2.1 is expressed with the following formulae:

$$\mathbf{B}_\mathsf{A}(\texttt{heads}) \wedge \mathbf{B}_\mathsf{A}(\mathbf{B}_\mathsf{B}((\mathbf{B}_\mathsf{A}(\texttt{heads}) \vee \mathbf{B}_\mathsf{A}(\neg\texttt{heads}))))$$

$$\mathbf{B}_\mathsf{B}(\mathbf{B}_\mathsf{A}(\texttt{heads}) \vee \mathbf{B}_\mathsf{A}(\neg\texttt{heads})) \wedge (\neg\mathbf{B}_\mathsf{B}(\texttt{heads} \wedge \neg\mathbf{B}_\mathsf{B}(\neg\texttt{heads})))$$

$$\mathbf{B}_\mathsf{C}([\bigwedge_{\mathsf{i}\in\{\mathsf{A},\mathsf{B},\mathsf{C}\}} (\neg\mathbf{B}_\mathsf{i}(\texttt{heads}) \wedge \neg\mathbf{B}_\mathsf{i}(\neg\texttt{heads}))])$$



$M_0[\pi](\mathsf{w}_0) = \{\texttt{look\_i}, \texttt{haskey\_A}, \texttt{heads}\}$
$M_0[\pi](\mathsf{w}_1) = \{\texttt{look\_i}, \texttt{haskey\_A}\}$
where $\mathsf{i} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$.

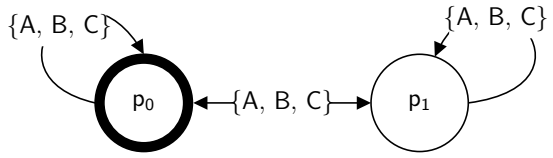
(a) The initial Kripke structure (M_0, w_0).

$$\begin{cases} \mathsf{u} = & \{(\mathsf{i}, \{\mathsf{u}, \mathsf{u}'\}), \texttt{look\_i}, \\ & \texttt{haskey\_a}, \texttt{heads}\} \\ \mathsf{u}' = & \{(\mathsf{i}, \{\mathsf{u}, \mathsf{u}'\}), \texttt{look\_i}, \\ & \texttt{haskey\_a}\} \end{cases}$$

where $\mathsf{i} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$.

(b) The initial possibility u.

Figure 2.8: The initial state.



$M_1[\pi](\mathsf{p}_0) = \{\texttt{look\_A}, \texttt{look\_B}, \texttt{haskey\_A}, \texttt{heads}\}$
$M_1[\pi](\mathsf{p}_1) = \{\texttt{look\_A}, \texttt{look\_B}, \texttt{haskey\_A}\}$

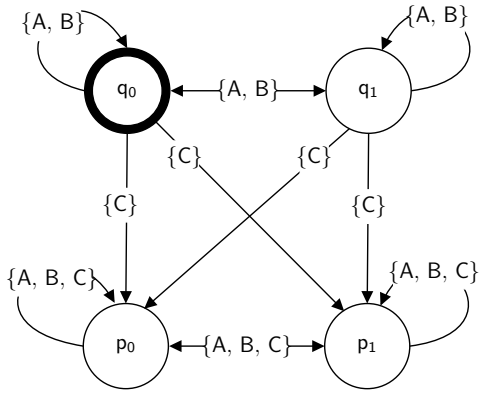
(a) The Kripke structure (M_1, p_0), obtained after the execution of $\texttt{distract\_C}\langle\mathsf{A}\rangle$ in (M_0, w_0) (Figure 2.8a).

$$\begin{cases} \mathsf{v} = & \{(\mathsf{i}, \{\mathsf{v}, \mathsf{v}'\}), \texttt{look\_A}, \\ & \texttt{look\_B}, \texttt{haskey\_A}, \texttt{heads}\} \\ \mathsf{v}' = & \{(\mathsf{i}, \{\mathsf{v}, \mathsf{v}'\}), \texttt{look\_A}, \\ & \texttt{look\_B}, \texttt{haskey\_A}\} \end{cases}$$

where $\mathsf{i} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$

(b) Possibility v, obtained after the execution of $\texttt{distract\_C}\langle\mathsf{A}\rangle$ in u (Figure 2.8b).

Figure 2.9: Execution of $\texttt{distract\_C}\langle\mathsf{A}\rangle$.



$M_2[\pi](\mathsf{q}_0) = \{\texttt{look\_i}, \texttt{haskey\_A}, \texttt{opened}, \texttt{heads}\}$
$M_2[\pi](\mathsf{q}_1) = \{\texttt{look\_i}, \texttt{haskey\_A}, \texttt{opened}\}$
where $\mathsf{i} \in \{\mathsf{A}, \mathsf{B}\}$.

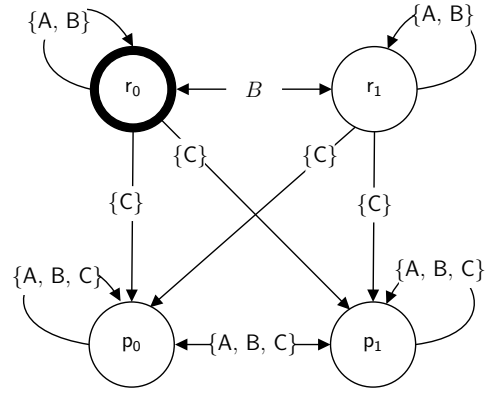
(a) The Kripke structure (M_2, q_0), obtained after the execution of $\texttt{open}\langle\mathsf{A}\rangle$ in (M_1, p_0) (Figure 2.9a).

$$\begin{cases} \mathsf{w} = & \{(\mathsf{i}, \{\mathsf{w}, \mathsf{w}'\}), (\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \texttt{look\_i}, \\ & \texttt{haskey\_a}, \texttt{opened}, \texttt{heads}\} \\ \mathsf{w}' = & \{(\mathsf{i}, \{\mathsf{w}, \mathsf{w}'\}), (\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \texttt{look\_i}, \\ & \texttt{haskey\_a}, \texttt{opened}\} \end{cases}$$

where $\mathsf{i} \in \{\mathsf{A}, \mathsf{B}\}$ and v, v', are the possibilities of Figure 2.9b.

(b) Possibility w, obtained after the execution of $\texttt{open}\langle\mathsf{A}\rangle$ in v (Figure 2.9b).

Figure 2.10: Execution of $\texttt{open}\langle\mathsf{A}\rangle$.



$M_3[\pi](\mathsf{r}_0) = \{\texttt{look\_i}, \texttt{haskey\_A}, \texttt{opened}, \texttt{heads}\}$
$M_3[\pi](\mathsf{r}_1) = \{\texttt{look\_i}, \texttt{haskey\_A}, \texttt{opened}\}$
where $\mathsf{i} \in \{\mathsf{A}, \mathsf{B}\}$.

(a) The Kripke structure (M_3, q_0), obtained after the execution of $\texttt{peek}\langle\mathsf{A}\rangle$ in (M_2, q_0).

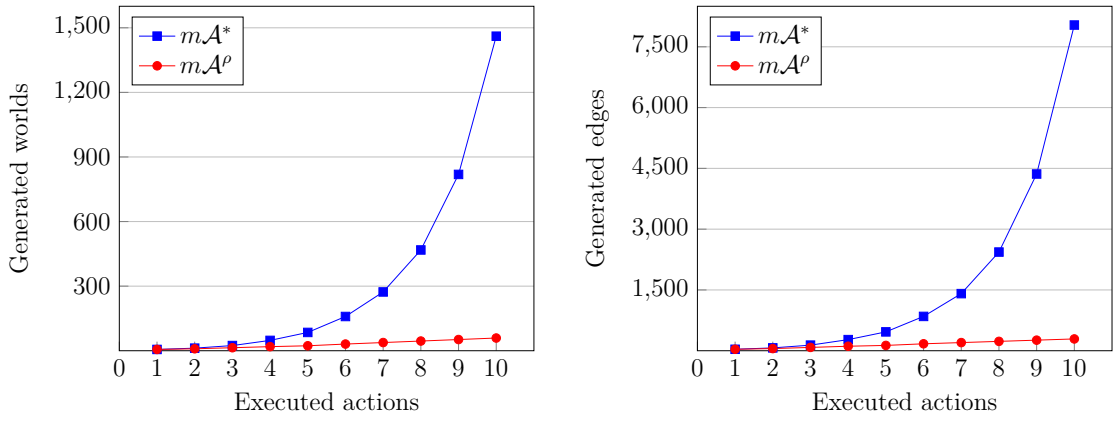
$$\begin{cases} \mathsf{z} = & \{(\mathsf{A}, \{\mathsf{z}\}), (\mathsf{B}, \{\mathsf{z}, \mathsf{z}'\})(\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \\ & \texttt{look\_i}, \texttt{haskey\_a}, \texttt{opened}, \texttt{heads}\} \\ \mathsf{z}' = & \{(\mathsf{A}, \{\mathsf{z}'\}), (\mathsf{B}, \{\mathsf{z}, \mathsf{z}'\})(\mathsf{C}, \{\mathsf{v}, \mathsf{v}'\}), \\ & \texttt{look\_i}, \texttt{haskey\_a}, \texttt{opened}\} \end{cases}$$

where $\mathsf{i} \in \{\mathsf{A}, \mathsf{B}\}$ and the possibilities v, v' are represented in Figure 2.9b.

(b) Possibility z, obtained after the execution of $\texttt{peek}\langle\mathsf{A}\rangle$ in w (Figure 2.10b).

Figure 2.11: Execution of $\texttt{peek}\langle\mathsf{A}\rangle$.

e-States Size Comparison To conclude, let us summarize, in Figure 2.12, the difference in size of the e-states of created by $m\mathcal{A}^{\rho}$ and $m\mathcal{A}^{*}$. This figure gives an intuitive idea of how $m\mathcal{A}^{\rho}$ helps in reducing the e-state size and its relative overhead. We note, in fact, that the size of the e-states, generated after ten consecutive action instances execution, varies immensely between the one generated by $m\mathcal{A}^{\rho}$, that is comprised of 59 worlds and 291 edges, and the one produced by $m\mathcal{A}^{*}$ that has 1461 worlds and 8037 edges. Let us stress that with a smaller e-state size is associated a faster solving process as we will show in the experimental evaluation, later in this thesis (Chapter 5).



(a) Worlds generated by $m\mathcal{A}^{\rho}$ and $m\mathcal{A}^{*}$.

(b) Edges generated by $m\mathcal{A}^{\rho}$ and $m\mathcal{A}^{*}$.

Figure 2.12: Comparison between the number of worlds and edges generated by $m\mathcal{A}^{\rho}$ and $m\mathcal{A}^{*}$ on the Coin in the Box domain with a sequence of ten action instances.

Fidarsi è bene, non fidarsi è meglio.

To trust is good, not to trust is better.

— Italian proverb

3 Communication with Trust

Contents

3.1 Trust in $m\mathcal{A}^\rho$

As already mentioned, multi-agent planning and epistemic reasoning have recently gained attention from several research communities. Efficient autonomous systems that can reason in these domains could lead to winning strategies in various fields such as economy [Aumann et al., 1995], security [Balliu et al., 2011], justice [Prakken, 2013], politics [Carbonell Jr, 1978] and can be exploited by autonomous devices, *e.g.*, self-driving cars, that can control several aspects of our daily life. We already explained why epistemic reasoners are not only interested in the state of the world, but also in the knowledge/beliefs of the agents. Nonetheless, the existing epistemic action languages [Bolander and Andersen, 2011, Muise et al., 2015, Baral et al., 2015, Fabiano et al., 2020, Baral et al., 2022] are able to model several

families of problems and study their information flows but cannot comprehensively reason on aspects like *trust*, *dishonesty*, *deception*, and other subtle epistemic concepts. To exploit epistemic reasoning in complex real-world scenarios it is, then, necessary to introduce the aforementioned epistemic nuances in the formalism used to express epistemic domains.

In this first section, we present an "expansion" of $m\mathcal{A}^{\rho}$ that allows us to formalize the notion of *Trust*. We do so by introducing two new actions that model the information exchange between agents when the idea of trust is involved:

(i) un-trustworthy announcement; and

(ii) mis-trustworthy announcement.

In particular, *(i) un*-trustworthy announcement formalizes the situation when the *untrusty* agents will not change their beliefs about the world no matter what the announcer says; and *(ii) mis*-trustworthy announcement captures the scenarios where the announcer, when not trusted, is believed to have a systematic faulty perception of the announced environment's properties. This leads the untrusty agents to believe the opposite of what has been announced.

3.1.1 *un/mis*-Trustworthy Announcement

We are now ready to provide a formal definition of the actions *un*-trustworthy announcement and *mis*-trustworthy announcement. That is, an agent can or cannot trust what another agent is telling her/him and act consequently. The expressions

$$\mathsf{i}\ \textbf{t\_announces}\ \mathsf{a}\ \textbf{if}\ \varphi;$$

and

$$\mathsf{i}\ \textbf{m\_announces}\ \mathsf{a}\ \textbf{if}\ \varphi;$$

express that the agent i is executing an *un*-trustworthy announcement or a *mis*-trustworthy announcement, respectively.

In defining the actions let us consider a static and globally visible version of "trust" that can be formalized with a simple function $\mathcal{T} : \mathcal{AG} \times \mathcal{AG} \mapsto \{0, 1\}$ and that enriches the definition of MEP domain (Definition 3.1).

Definition 3.1: MEP Domain with Trust

A *MEP domain with Trust* is a tuple $D = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}, \mathcal{T}, \varphi_{ini}, \varphi_{goal} \rangle$, where the additional (with respect to Definition 1.15) element $\mathcal{T}$ contains the trust relations between agents:

$$\mathcal{T}(\mathsf{i}, \mathsf{j}) = \begin{cases} 1 & \text{if } [\mathsf{j} \textbf{ trust } \mathsf{i}] \\ 0 & \text{otherwise} \end{cases}$$

where i and $\mathsf{j} \in D(\mathcal{AG})$.

We will consider only the case where $\mathcal{T}$ is a static and globally visible function. Let us notice that having $\mathcal{T}$ to be dynamic is easily achievable. In particular, we just need to define how $\mathcal{T}$ may vary, *e.g.*, making the function depending on some fluents value. For the sake of simplicity, let us imagine $\mathcal{T}$ to be fixed and not dependent on the plan execution. On the other hand, making $\mathcal{T}$ not globally visible—*i.e.*, each agent knows her/his own version of the *trust* function—is not straightforward. The problem arises when two agents have different views of the same trust relation leading to the generation of different e-states (*i.e.*, each agent must preserve its separate view of the domain). We leave the investigation of this scenario as future work.

To clarify the e-state update after the execution of the new actions we will also present a graphical representation of the transition function application. The examples of execution will be based on a variation of the *Grapevine* domain, firstly introduced by Kominis and Geffner [2015], described later in Planning Domain 3.1. Let us now provide a formal definition of e-state update of the two new actions of $m\mathcal{A}^{\rho}$.

un-Trustworthy Announcement

We can now introduce the transition function for an *un*-trustworthy announcement. Intuitively, this action models an announcement where the listening agents can or cannot trust the announcer. That is:

- the *trusty* agents will update their belief consistently with what has been announced; and

- the *untrusty*[1] ones will maintain their beliefs about the world and will only update their perspective on the beliefs of the trusty agents.

Let us recall that the sets $\mathbf{F}_{\mathsf{a}}, \mathbf{P}_{\mathsf{a}}, \mathbf{O}_{\mathsf{a}}$ represent the set of fully observant, partially observant, and oblivious agents with respect to the execution of an action instance $\mathsf{a}\langle \mathsf{i} \rangle$, respectively.

Let a domain D, its set of action instances $D(\mathcal{AI})$, and the set $\mathcal{S}$ of all the possibilities reachable from $D(\varphi_{ini})$ with a finite sequence of action instances be given. The transition function $\Phi : D(\mathcal{AI}) \times \mathcal{S} \to \mathcal{S} \cup \{\emptyset\}$ for the *un*-trustworthy announcement relative to D is presented in Definition 3.2. Intuitively, this transition function allows, through the use of Υ and Ψ, to model the idea that the untrusty agents maintain their beliefs while knowing that the trusty ones updated their point of view of the "physical world" (and vice versa).

[1]The agents that are fully observant with respect to the announcement but that do not trust the announcer.

Definition 3.2: $m\mathcal{A}^{\rho}$ *un*-Trustworthy Announcement

Allow us to use the compact notation $\mathsf{u}(\mathcal{F}) = \{\mathtt{f} \mid \mathtt{f} \in D(\mathcal{F}) \wedge \mathsf{u} \models \mathtt{f}\} \cup \{\neg\mathtt{f} \mid \mathtt{f} \in D(\mathcal{F}) \wedge \mathsf{u} \not\models \mathtt{f}\}$ for the sake of readability. Let an action instance $\mathtt{a}\langle\mathsf{i}\rangle \in D(\mathcal{AI})$ where agent $\mathsf{i} \in D(\mathcal{AG})$ announces the fluent formula ϕ, an and a possibility $\mathsf{u} \in \mathcal{S}$ be given.
If $\mathtt{a}$ is not executable in u, then $\Phi(\mathtt{a}, \mathsf{u}) = \emptyset$ otherwise $\Phi(\mathtt{a}, \mathsf{u}) = \mathsf{u}'$, where:

$$e(\mathtt{a}, \mathsf{u}) = \begin{cases} 0 & \text{if } \mathsf{u} \models \phi \\ 1 & \text{if } \mathsf{u} \models \neg\phi \end{cases}$$

$$\mathsf{u}'(\mathcal{F}) = \mathsf{u}(\mathcal{F})$$

$$\mathsf{u}'(\mathsf{j}) = \begin{cases} \mathsf{u}(\mathsf{j}) & \text{if } \mathsf{j} \in \mathbf{O}_{\mathtt{a}} \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{j})} \Upsilon(\mathtt{a}, \mathsf{w}) \cup \Psi(\mathtt{a}, \mathsf{w}) & \text{if } \mathsf{j} \in \mathbf{P}_{\mathtt{a}} \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{j})} \Upsilon(\mathtt{a}, \mathsf{w}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathtt{a}} \wedge e(\mathtt{a}, \mathsf{u}) = 1 \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{j})} \Psi(\mathtt{a}, \mathsf{w}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathtt{a}} \wedge e(\mathtt{a}, \mathsf{u}) = 0 \end{cases}$$

with $\Upsilon(\mathtt{a}, \mathsf{w}) = \mathsf{w}'$ such that

$$\mathsf{w}'(\mathcal{F}) = \mathsf{w}(\mathcal{F})$$

$$\mathsf{w}'(\mathsf{j}) = \begin{cases} \mathsf{w}(\mathsf{j}) & \text{if } \mathsf{j} \in \mathbf{O}_{\mathtt{a}} \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{j})} \Phi(\mathtt{a}, \mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{P}_{\mathtt{a}} \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{j})} \Upsilon(\mathtt{a}, \mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathtt{a}} \wedge \mathcal{T}(\mathsf{j}, \mathsf{i}) = 0 \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{j}) : e(\mathtt{a}, \mathsf{v}) = 1} \Upsilon(\mathtt{a}, \mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathtt{a}} \wedge \mathcal{T}(\mathsf{j}, \mathsf{i}) = 1 \end{cases}$$

and $\Psi(\mathtt{a}, \mathsf{w}) = \mathsf{w}'$ such that

$$\mathsf{w}'(\mathcal{F}) = \mathsf{w}(\mathcal{F})$$

$$\mathsf{w}'(\mathsf{j}) = \begin{cases} \mathsf{w}(\mathsf{j}) & \text{if } \mathsf{j} \in \mathbf{O}_{\mathtt{a}} \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{j})} \Phi(\mathtt{a}, \mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{P}_{\mathtt{a}} \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{j})} \Psi(\mathtt{a}, \mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathtt{a}} \wedge \mathcal{T}(\mathsf{j}, \mathsf{i}) = 0 \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{j}) : e(\mathtt{a}, \mathsf{v}) = 0} \Psi(\mathtt{a}, \mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathtt{a}} \wedge \mathcal{T}(\mathsf{j}, \mathsf{i}) = 1 \end{cases}$$

for each agent $\mathsf{j} \in D(\mathcal{AG})$.

An Example of Execution As mentioned above, we will provide a graphical representation of the newly introduced transition function. Let us remember that

we will represent a possibility as a graph where the nodes correspond to the possible worlds while the edges encode the beliefs of the agents. The thicker node represents the pointed possibility. Moreover, to extract the point of view of the agents from a graph we need to follow the entailment rules (Definition 2.11) starting from the pointed possibility. In Planning Domain 3.1 we briefly describe the instance that will be used as a running example. Since we are only interested in showing how to e-state update works we will omit the actions and goal description.

Planning Domain 3.1: Trust-Grapevine

$n \geq 2$ agents are located in $k \geq 2$ rooms. Each agent knows $j \geq 0$ secrets. An agent can move freely to other rooms, and she/he can share a "secret" with the agents that are in the room with her/him. Moreover, the agents will be aware of the execution of announcements made in adjacent rooms without actually knowing the truth value of the announced fluent. Each agent can or cannot *trust* what another agent shares[a].

Our running example considers five agents: A, B, C, D, and E. Initially we have that:

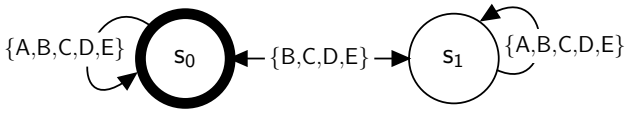
- A, B, C are located in the same room (`room_1`) while D is in a room (`room_2`) adjacent to `room_1` and E is located in `room_3`, not adjacent to `room_1`;
- agents B and D trust A while C and E do not;
- only agent A knows `secret_a`;
- the value of `secret_a` is $\top$; and
- initially everyone knows the position of each agent and that only A knows the value of `secret_a`.

In Figure 3.1 we present a graphical representation of the above described initial state.

[a]Let us notice that since the idea of trust is involved, each agent, in order to learn a secret, needs to witness an announcement of agents that she/he trusts, making the newly presented domain slightly more intricate than the original Grapevine.

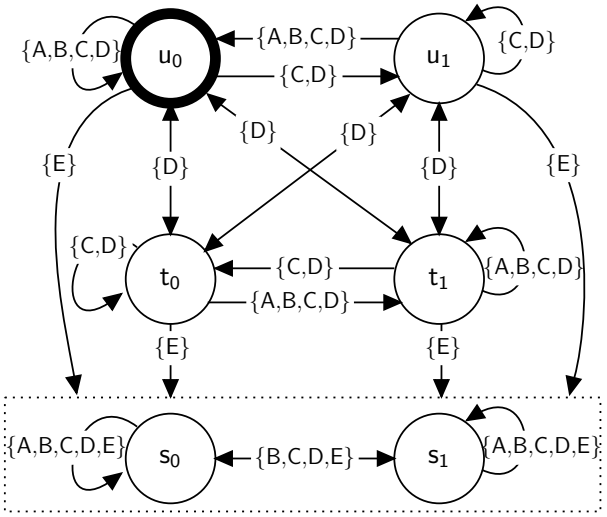
Figure 3.2 represents the e-state generated after the execution of the *un*-trustworthy announcement action instance $\texttt{announce\_secret\_a}\langle \mathsf{A} \rangle$ (`ann_a` for brevity). With `ann_a`, A announces the value of `secret_a`. Let us note that from the position of the agents we know that $\mathsf{A}, \mathsf{B}, \mathsf{C} \in \mathbf{F}_{\texttt{ann\_a}}$, $\mathsf{D} \in \mathbf{P}_{\texttt{ann\_a}}$, and $\mathsf{E} \in \mathbf{O}_{\texttt{ann\_a}}$.

Once again, from Figure 3.2, we can derive that B—a trusty fully observant—believes `secret_a` to be true ($\top$) while C—an untrusty fully observant—did not change her/his direct belief about `secret_a`, but changed her/his beliefs on other agents', *e.g.*, B, beliefs. More intricate relations, described in Proposition 3.1, can also be derived from Figure 3.2.



s_0	`secret_a, at_1_A, t_1_B, at_1_C, at_2_D, at_3_E`
s_1	`at_1_A, t_1_B, at_1_C, at_2_D, at_3_E`

Figure 3.1: The initial e-state described in Planning Domain 3.1. The bottom Table presents the fluents interpretation of each possibility (as usual, only the positive fluents are reported).



$u_0/t_0/s_0$	`secret_a, at_1_A, t_1_B, at_1_C, at_2_D, at_3_E`
$u_1/t_1/s_1$	`at_1_A, t_1_B, at_1_C, at_2_D, at_3_E`

Figure 3.2: The e-state obtained after executing the *un*-trustworthy announcement action `ann_a` in the e-state represented in Figure 3.1.

***mis*-Trustworthy Announcement**

In Definition 3.2 we assumed that an agent j, when does not trust the announcer, will not change her/his beliefs about what has been announced. That is, an untrusty agent will not change her/his perspective on the "physical" state of the world. Let us notice that this type of trust captures the idea that, for the untrusty agents, the announcer is not reliable and the information she/he is providing is not worth taking into consideration as it can be not accurate.

Depending on the scenario it could be necessary to model a stronger concept of untrust. In particular, it could be required to design an *un*-trustworthy announcement such that the untrusty agents will believe the contrary of what has been announced (while still believing that the announcer believes what she/he announced). We will call this type of action *mis*-trustworthy announcement. The formalization of such variation is presented in Definition 3.3.

Let us note that the transition functions introduced in Definition 3.2 and Definition 3.3 only differ in the specification of Υ and Ψ for the untrusty fully observant agents. This difference is needed to represent the fact that in the case of an *un*-trustworthy announcement the untrusty agents maintain their beliefs while in the mis-trustworthy one they will believe the opposite of what has been announced.

Definition 3.3: $m\mathcal{A}^{\rho}$ *mis*-Trustworthy Announcement

Let an action instance $\mathsf{a}\langle\mathsf{i}\rangle \in D(\mathcal{AI})$ where agent $\mathsf{i} \in D(\mathcal{AG})$ announces the fluent formula ϕ and a possibility $\mathsf{u} \in \mathcal{S}$ be given.
If a is not executable in u, then $\Phi(\mathsf{a},\mathsf{u}) = \emptyset$ otherwise $\Phi(\mathsf{a},\mathsf{u}) = \mathsf{u}'$, where:

$$e(\mathsf{a},\mathsf{u}) = \begin{cases} 0 & \text{if } \mathsf{u} \models \phi \\ 1 & \text{if } \mathsf{u} \models \neg\phi \end{cases}$$

$$\mathsf{u}'(\mathcal{F}) = \mathsf{u}(\mathcal{F})$$

$$\mathsf{u}'(\mathsf{j}) = \begin{cases} \mathsf{u}(\mathsf{j}) & \text{if } \mathsf{j} \in \mathbf{O}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{j})} \Upsilon(\mathsf{a},\mathsf{w}) \cup \Psi(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{j} \in \mathbf{P}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{j})} \Upsilon(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathsf{a}} \wedge e(\mathsf{a},\mathsf{u}) = 1 \\ \bigcup\limits_{\mathsf{w}\in\mathsf{u}(\mathsf{j})} \Psi(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathsf{a}} \wedge e(\mathsf{a},\mathsf{u}) = 0 \end{cases}$$

with $\Upsilon(\mathsf{a},\mathsf{w}) = \mathsf{w}'$ such that

$$\mathsf{w}'(\mathcal{F}) = \mathsf{w}(\mathcal{F})$$

$$\mathsf{w}'(\mathsf{j}) = \begin{cases} \mathsf{w}(\mathsf{j}) & \text{if } \mathsf{j} \in \mathbf{O}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{j})} \Phi(\mathsf{a},\mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{P}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{j}):e(\mathsf{a},\mathsf{v})=0} \Upsilon(\mathsf{a},\mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathsf{a}} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 0 \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{j}):e(\mathsf{a},\mathsf{v})=1} \Upsilon(\mathsf{a},\mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathsf{a}} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 1 \end{cases}$$

and $\Psi(\mathsf{a},\mathsf{w}) = \mathsf{w}'$ such that

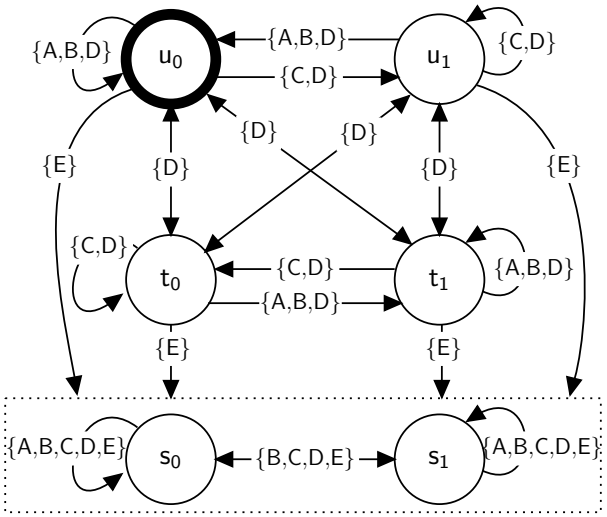
$$\mathsf{w}'(\mathcal{F}) = \mathsf{w}(\mathcal{F})$$

$$\mathsf{w}'(\mathsf{j}) = \begin{cases} \mathsf{w}(\mathsf{j}) & \text{if } \mathsf{j} \in \mathbf{O}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{j})} \Phi(\mathsf{a},\mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{P}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{j}):e(\mathsf{a},\mathsf{v})=1} \Psi(\mathsf{a},\mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathsf{a}} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 0 \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{j}):e(\mathsf{a},\mathsf{v})=0} \Psi(\mathsf{a},\mathsf{v}) & \text{if } \mathsf{j} \in \mathbf{F}_{\mathsf{a}} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 1 \end{cases}$$

for each agent $\mathsf{j} \in D(\mathcal{AG})$.

An Example of Execution As for the *un*-trustworthy announcement, we will provide an example of *mis*-trustworthy announcement execution. The initial state is identical to the one introduced in Planning Domain 3.1 . The only difference is that

now the action announce_secret_a⟨A⟩ (or ann_a for brevity) is a *mis*-trustworthy announcement instead of an *un*-trustworthy announcement. The initial state is, therefore, represented in Figure 3.1 while the e-state obtained after the execution of the *mis*-trustworthy announcement is shown in Figure 3.3. From Figure 3.3, we can extrapolate that B—a trusty fully observant—believes secret_a to be true (⊤) while C—an untrusty fully observant—believes the opposite, *i.e.*, secret_a = ⊥. As for the previous actions, also for *mis*-trustworthy announcement more intricate relations, described in Proposition 3.2, can also be derived from Figure 3.3.



$u_0/t_0/s_0$	secret_a, at_1_A, t_1_B, at_1_C, at_2_D, at_3_E
$u_1/t_1/s_1$	at_1_A, t_1_B, at_1_C, at_2_D, at_3_E

Figure 3.3: The e-state obtained after executing the *mis*-trustworthy announcement action ann_a in the e-state represented in Figure 3.1.

3.1.2 Desired Properties

In Section 2.2.2 some useful properties that correctly capture certain intuitions concerning the effects of the various types of actions in $m\mathcal{A}^{\rho}$ are listed. Similarly, in what follows, we will provide some properties that the e-state update, after executing the *un/mis*-trustworthy announcement, meets. In Appendix A.3 we will show the formal proofs that these properties hold. As usual, we will indicate the sets of partially observant and oblivious agents (with respect to the action

instance $\mathsf{a}\langle\mathsf{i}\rangle$) with $\mathbf{P}_{\mathsf{a}}$ and $\mathbf{O}_{\mathsf{a}}$, respectively. Moreover, we will indicate the set of trusty fully observant agents with $\mathbf{F}_{\mathsf{a}}$ while will indicate the set of untrusty fully observant with $\mathbf{U}_{\mathsf{a}}$.

Proposition 3.1: *un*-Trustworthy Announcement Properties

Let $\mathsf{a}\langle\mathsf{i}\rangle$ *be an* un-*trustworthy announcement* action instance *where an agent* i **t_announces** ϕ. *Let* e *be an e-state and let* e' *be its updated version (that is,* $\Phi(\mathsf{a},\mathsf{e}) = \mathsf{e}'$*), then in* $m\mathcal{A}^{\rho}$ *it holds that:*

(1) $\mathsf{e}' \models \mathbf{C}_{\mathbf{F}_{\mathsf{a}}}\phi$;

(2) $\mathsf{e}' \models \mathbf{C}_{\mathbf{U}_{\mathsf{a}}}(\mathbf{C}_{\mathbf{F}_{\mathsf{a}}}\phi)$;

(3) $\mathsf{e}' \models \mathbf{C}_{\mathbf{P}_{\mathsf{a}}}(\mathbf{C}_{\mathbf{F}_{\mathsf{a}}}\phi \vee \mathbf{C}_{\mathbf{F}_{\mathsf{a}}}\neg\phi)$;

(4) $\mathsf{e}' \models \mathbf{C}_{\mathbf{F}_{\mathsf{a}}\cup\mathbf{U}_{\mathsf{a}}}(\mathbf{C}_{\mathbf{P}_{\mathsf{a}}}(\mathbf{C}_{\mathbf{F}_{\mathsf{a}}}\phi \vee \mathbf{C}_{\mathbf{F}_{\mathsf{a}}}\neg\phi))$;

(5) *for every agent* $\mathsf{y} \in \mathbf{O}_{\mathsf{a}}$ *and a belief formula* φ, $\mathsf{e}' \models \mathbf{B}_{\mathsf{y}}(\varphi)$ *iff* $\mathsf{e} \models \mathbf{B}_{\mathsf{y}}(\varphi)$; *and*

(6) *for every pair of agents* $\mathsf{x} \in \mathbf{F}_{\mathsf{a}} \cup \mathbf{U}_{\mathsf{a}} \cup \mathbf{P}_{\mathsf{a}}$ *and* $\mathsf{y} \in \mathbf{O}_{\mathsf{a}}$ *and a belief formula* φ, *if* $\mathsf{e} \models \mathbf{B}_{\mathsf{x}}(\mathbf{B}_{\mathsf{y}}(\varphi))$ *then* $\mathsf{e}' \models \mathbf{B}_{\mathsf{x}}(\mathbf{B}_{\mathsf{y}}(\varphi))$.

(7) *for every agent* $\mathsf{y} \in \mathbf{U}_{\mathsf{a}}$, $\mathsf{e}' \models \mathbf{B}_{\mathsf{y}}(\phi)/\mathbf{B}_{\mathsf{y}}(\neg\phi)/(\neg\mathbf{B}_{\mathsf{y}}(\phi) \wedge \neg\mathbf{B}_{\mathsf{y}}(\neg\phi))$ *iff* $\mathsf{e} \models \mathbf{B}_{\mathsf{y}}(\phi)/\mathbf{B}_{\mathsf{y}}(\neg\phi)/(\neg\mathbf{B}_{\mathsf{y}}(\phi) \wedge \neg\mathbf{B}_{\mathsf{y}}(\neg\phi))$;

The properties presented in Proposition 3.1 capture some fundamental aspects of an *un*-trustworthy announcement action. Intuitively:

(1) Captures the idea that all the trusty fully observant agents should believe *(i)* what has been announced; and *(ii)* that all the other trusty fully observant agents believe what has been announced and so on *ad infinitum* (that is why we use the $\mathbf{C}$ operator).

(2) Models the fact that all the untrusty agents believe that all the trusty ones have common belief of what has been announced.

(3) Captures that the partially observants believe that the trusty fully observants have common knowledge of what has been announced while the partially observants themselves do not know the announced value.

(4) States that the fully observant agents have common knowledge of the previous property.

(5) Captures the fact that the oblivious agents do not change their beliefs.

(6) States that the observant agents (trusty, untrusty, and partial) believe that the oblivious agents did not change their beliefs.

(7) Models the idea that all the untrusty agents do not modify their beliefs about the announced values.

As we did for the *un*-trustworthy announcement, let us identify some properties also for the *mis*-trustworthy announcement action.

Proposition 3.2: *mis*-Trustworthy Announcement Properties

Let $\mathsf{a}\langle\mathsf{i}\rangle$ *be a* mis-*trustworthy announcement* action instance *where an agent* i **m_announces** ϕ. *Let* e *be an e-state and let* e' *be its updated version (that is,* $\Phi(\mathsf{a},\mathsf{e}) = \mathsf{e}'$*), then in* $m\mathcal{A}^{\rho}$ *Items* **(1)** *to* **(6)** *of Proposition 3.1 hold. In addition,*

(8) $\mathsf{e}' \models \mathbf{C}_{\mathbf{U}_\mathsf{a}} \neg\phi$;

(9) $\mathsf{e}' \models \mathbf{C}_{\mathbf{F}_\mathsf{a}}(\mathbf{C}_{\mathbf{U}_\mathsf{a}} \neg\phi)$;

(10) $\mathsf{e}' \models \mathbf{C}_{\mathbf{P}_\mathsf{a}}(\mathbf{C}_{\mathbf{U}_\mathsf{a}} \phi \vee \mathbf{C}_{\mathbf{U}_\mathsf{a}} \neg\phi)$;

Proposition 3.2 describes the core ideas behind a *mis*-trustworthy announcement action. While Items **(1)** to **(6)** of Proposition 3.1 have already been described, the intuitive meaning of the remaining ones is as follows.

(8) Captures the idea that all the untrusty fully observant agents should believe *(i)* the contrary of what has been announced; and *(ii)* that all the other untrusty fully observant agents believe the negation of what has been announced and so on *ad infinitum* (that is why we use the **C** operator).

(9) Models the fact that all the trusty agents believe that all the untrusty ones have common belief of the negation of what has been announced.

(10) Captures that the partially observants believe that the untrusty fully observants have common belief of what has been announced, while the partially observant themselves do not know the announced value.

3.2 Capturing Trust with Update Models

Since most of the work in dynamic epistemic planning revolves around the concepts of Kripke structures and update models, *e.g.*, Bolander and Andersen [2011], Baral et al. [2022], we believe that formalizing the aforementioned actions using these concepts would provide interesting insights for the community. In this paragraph we, therefore, succinctly present the update models that define the announcement actions where the trust relation $\mathcal{T}$ is taken into consideration.

3.2.1 $m\mathcal{A}^*$ *un*-Trustworthy Announcement

Let us begin by describing the update model related to the *un*-trustworthy announcement for the language $m\mathcal{A}^*$. We will follow the scheme presented by Baral et al. [2015, 2022] to introduce a new action's update model. The transition function is derived by applying the update model to a Kripke structure. Let us also recall that the sets $\mathbf{F}_{\mathsf{a}}, \mathbf{P}_{\mathsf{a}}, \mathbf{O}_{\mathsf{a}}$ represent the set of fully observant, partially observant, and oblivious agents with respect to an action instance $\mathsf{a}\langle \mathsf{i} \rangle$, respectively. Definition 3.4 presents formally the update model of an *un*-trustworthy announcement. Let us notice that an update instance of an *un*-trustworthy announcement action occurrence has five events to capture the idea of both updating the beliefs of the trusty agents and maintaining the beliefs of the untrusty ones. Figure 3.4 provides a graphical representation of this update template. Each event is associated with sets of states and edges where the truth value of the announced fluent formula is either known to be true, known to be false, or unknown.

Definition 3.4: $m\mathcal{A}^*$ *un*-trustworthy announcement Update Model

Let $\mathsf{a}\langle\mathsf{i}\rangle$ be an *un*-trustworthy announcement action instance where agent i announces the fluent formula ϕ with the precondition ψ, a function $\mathcal{T} : \mathcal{AG} \times \mathcal{AG} \mapsto \{0,1\}$ and a frame of reference $\rho = (\mathbf{F}_\mathsf{a}, \mathbf{P}_\mathsf{a}, \mathbf{O}_\mathsf{a})$. The update model for a, $\mathcal{T}$, and ρ, $\omega(\mathsf{a}, \mathcal{T}, \rho)$, is defined by $\langle \Sigma, \mathcal{R}_1, \ldots, \mathcal{R}_\mathsf{n} \rangle$ where:

- $\Sigma = \{\sigma, \tau, \eta, \mu, \epsilon\}$;
- $\mathcal{R}_\mathsf{j}$ is defined by:

$$\mathcal{R}_\mathsf{j} = \begin{cases} \{(\sigma,\sigma),(\tau,\tau),(\epsilon,\epsilon)\} & \text{if } \mathsf{i} \in \mathbf{F}_\mathsf{a} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 1 \\ \{(x,x'),(y,y'),(\epsilon,\epsilon)\} & \text{if } \mathsf{i} \in \mathbf{F}_\mathsf{a} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 0 \\ \quad \text{where } x,x' \in \{\sigma,\eta\} \wedge y,y' \in \{\mu,\tau\} & \\ \{(x,y),(\epsilon,\epsilon)\} & \text{if } \mathsf{j} \in \mathbf{P}_\mathsf{a} \\ \quad \text{where } x,y \in \{\sigma,\tau,\eta,\mu\} & \\ \{(x,\epsilon),(\epsilon,\epsilon)\} & \text{if } \mathsf{j} \in \mathbf{O}_\mathsf{a} \\ \quad \text{where } x \in \{\sigma,\tau,\eta,\mu\} & \end{cases}$$

- The preconditions are are defined by:

$$pre(x) = \begin{cases} \psi \wedge \phi & \text{if } x \in \{\sigma,\mu\} \\ \psi \wedge \neg\phi & \text{if } x \in \{\eta,\tau\} \\ \top & \text{if } x = \epsilon \end{cases}$$

- *sub* is defined by $sub(x) = \emptyset$ for each $x \in \Sigma$.
- We identify σ as the *pointed event*.

The update instance for the *un*-trustworthy announcement action occurrence a and the frame of reference ρ is $(\omega(\mathsf{a}, \mathcal{T}, \rho), \{\sigma, \tau, \eta, \mu\})$.

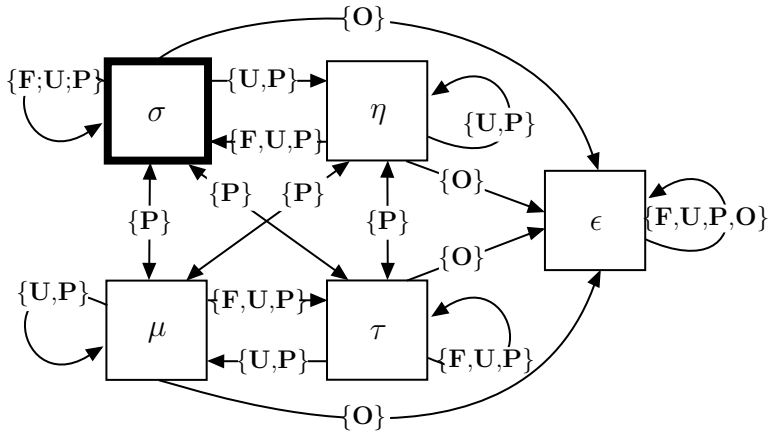


Figure 3.4: The update template (Σ, σ) for the *un*-trustworthy announcement. **F**, **U**, **P**, **O** represent the trusty fully observant, untrusty fully observant, partially observant, and oblivious agents respectively.

3.2.2 $m\mathcal{A}^*$ *mis*-Trustworthy Announcement

Let us now introduce the transition function related to the *mis*-trustworthy announcement action. As before, we will follow the scheme of Baral et al. [2015, 2022] to introduce a new action's update model, and the sets $\mathbf{F}_{\mathsf{a}}, \mathbf{P}_{\mathsf{a}}, \mathbf{O}_{\mathsf{a}}$ represent the set of fully observant, partially observant, and oblivious agents with respect to an action instance $\mathsf{a}\langle\mathsf{i}\rangle$, respectively.

Let us note that Definition 3.5 only varies, with respect to Definition 3.4, in the behavior of $\mathcal{R}_{\mathsf{j}}$ for the untrusty fully observant agents (*i.e.*, $\mathsf{j} \in \mathbf{F}_{\mathsf{a}} \wedge \mathcal{T}(\mathsf{j}, \mathsf{i}) = 0$).

Definition 3.5: $m\mathcal{A}^*$ *mis*-trustworthy announcement Update Model

Let $\mathsf{a}\langle\mathsf{i}\rangle$ be an *mis*-trustworthy announcement action instance where agent i announces the fluent formula ϕ with the precondition ψ, a function $\mathcal{T} : \mathcal{AG} \times \mathcal{AG} \mapsto \{0,1\}$ and a frame of reference $\rho = (\mathbf{F}_\mathsf{a}, \mathbf{P}_\mathsf{a}, \mathbf{O}_\mathsf{a})$. The update model for a, $\mathcal{T}$, and ρ, $\omega(\mathsf{a}, \mathcal{T}, \rho)$, is defined by $\langle \Sigma, \mathcal{R}_1, \ldots, \mathcal{R}_\mathsf{n} \rangle$ where:

- $\Sigma = \{\sigma, \tau, \eta, \mu, \epsilon\}$;
- $\mathcal{R}_\mathsf{i}$ is defined by:

$$\mathcal{R}_\mathsf{j} = \begin{cases} \{(\sigma,\sigma),(\tau,\tau),(\epsilon,\epsilon)\} & \text{if } \mathsf{i} \in \mathbf{F}_\mathsf{a} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 1 \\ \{(\eta,\eta),(\mu,\mu),(\epsilon,\epsilon)\} & \text{if } \mathsf{i} \in \mathbf{F}_\mathsf{a} \wedge \mathcal{T}(\mathsf{j},\mathsf{i}) = 0 \\ \quad \text{where } x, x' \in \{\sigma,\eta\} \wedge y, y' \in \{\mu,\tau\} & \\ \{(x,y),(\epsilon,\epsilon)\} & \text{if } \mathsf{j} \in \mathbf{P}_\mathsf{a} \\ \quad \text{where } x, y \in \{\sigma,\tau,\eta,\mu\} & \\ \{(x,\epsilon),(\epsilon,\epsilon)\} & \text{if } \mathsf{j} \in \mathbf{O}_\mathsf{a} \\ \quad \text{where } x \in \{\sigma,\tau,\eta,\mu\} & \end{cases}$$

- The preconditions are are defined by:

$$pre(x) = \begin{cases} \psi \wedge \phi & \text{if } x \in \{\sigma,\mu\} \\ \psi \wedge \neg\phi & \text{if } x \in \{\eta,\tau\} \\ \top & \text{if } x = \epsilon \end{cases}$$

- *sub* is defined by $sub(x) = \emptyset$ for each $x \in \Sigma$.
- We identify σ as the *pointed event*.

The update instance for the *mis*-trustworthy announcement action occurrence a and the frame of reference ρ is $(\omega(\mathsf{a}, \mathcal{T}, \rho), \{\sigma, \tau, \eta, \mu\})$.

The update template for *mis*-trustworthy announcement is presented in Figure 3.5.

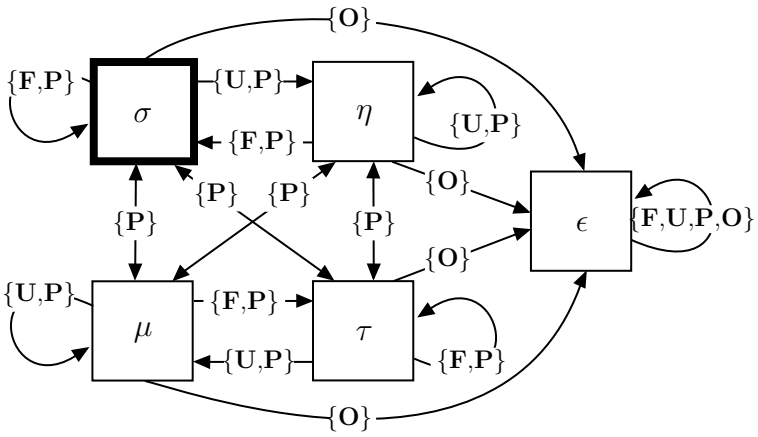


Figure 3.5: The update template (Σ, σ) for the *mis*-trustworthy announcement. **F**, **U**, **P**, **O** represent the trusty fully observant, untrusty fully observant, partially observant, and oblivious agents respectively.

People get built different. We don't need to figure it out, we just need to respect it.

— Princess Bubblegum
in *"Bonnie and Neddy"*
Adventure Time

4

Trust, Misconception, and Lies in MEP

Contents

4.1 Agents' Attitudes and Inconsistent Beliefs

In the previous chapter, we presented a formalization for announcements when the concept of trust is taken into consideration. While this introduction expands the range of scenarios that $m\mathcal{A}^{\rho}$ may represent, it still does not fully capture aspects like dishonesty, deception, and incongruent beliefs.

In this chapter, we further enrich the language $m\mathcal{A}^{\rho}$ with the concept of agents' *attitudes*. Our idea of attitudes stems from the concept of *dynamic attitudes* that "*represent the agent's assessment of the reliability of the source*" introduced by Rodenhäuser [2014]. We define basic attitudes that capture how an agent reacts when another agent is informing her/him about something. In the real world, in fact, it is often the case that we associate an idea of reliability with

an information source. This work captures this idea by having agents behave accordingly to the following attitudes: *doubtful*, *impassive*, *trustful*, *mistrustful*, or *stubborn* (detailed descriptions are given later).

In addition to the agents' relation with the information source, we also consider the scenario when agents learn a fact that discords with their previous beliefs. When such a discrepancy arises, we talk about *inconsistent belief*. Since, as said in Chapter 2, we consider $\mathbf{KD45}_n$-states, inconsistencies are relative only to the beliefs of an agent (and not to the actual world). Let us assume that agent i believes that $\neg\varphi$ is the case in the e-state u (*i.e.*, $\mathsf{u} \models \mathbf{B}_\mathsf{i}(\neg\varphi)$); in $m\mathcal{A}^\rho$ there are two main sources of inconsistencies: *(i)* i observes the real world—performing a *sensing* action—and learns φ (the opposite of what she/he believed); *(ii)* i learns φ as a result of an *announcement* performed by another agent j. In both scenarios, we must account for the belief of i after the action execution. In particular, the resulting e-state u' must be consistent with the axiom **D** of Table 1.1. In the former case, *(i)* we resolve the inconsistency by having i believing φ; *i.e.*, we make sure that $\mathsf{u}' \models \mathbf{B}_\mathsf{i}(\varphi)$. This is a reasonable solution, as we assume that agents trust their senses when observing the world. In the latter, *(ii)* we must take into account the *attitude* of the agent with respect to the announcer j. As said by Rodenhäuser [2014], "*we are not only interested in the acceptance of new information (based on trust), but also in its rejection (based on distrust)*". For instance, the listener may be skeptical or credulous, and thus she/he would change her/his belief according to her/his attitude. Let us notice that inconsistent beliefs are different from *false beliefs*. An agent has a false belief about a property φ if she/he believes φ to be true even if such property does not hold in the actual world. False beliefs are already allowed in $m\mathcal{A}^\rho$ as a result of the presence of oblivious agents in action instances.

Going back to the attitudes of agents, the notion of *trust* naturally arises. It is reasonable to assume that the listener i believes the announcer j if i trusts j. In particular, let us consider three attitudes[1] for fully observant agents that listen to an announcement: *trustful*, *mistrustful*, and *stubborn*. *Trustful* agents believe

[1]We only consider basics attitudes, we leave the exploration of more complex ones as future work.

what the announcer tells them; *mistrustful* agents believe the opposite of what is announced; and *stubborn* agents do not modify their beliefs. Considering the case of *semi-private* announcements, we need to introduce the concept of attitude for partially observant agents as well. Specifically, we consider *impassive* and *doubtful* agents. *Impassive* agents keep their current beliefs, while *doubtful* agents believe neither what is being announced nor the opposite, regardless of their previous beliefs. Note that *stubborn* and *impassive* agents are different, as the former kind is aware of what is being announced—*i.e.*, the truth value of the property φ. Let us note that such attitudes are named to capture our personal idea of the behavior they represent and they are not meant to wholly describe the nuances of complex social attitudes such as, for example, stubbornness.

When communicating with their peers, agents might announce something that is *false* relative to their own point of view. We call *lies* such announcements. Similar to the notion of inconsistent belief, the truthfulness of announcements depends on the point of view of the announcer i—*i.e.*, i truthfully announces φ iff $\mathsf{u} \models \mathbf{B}_{\mathsf{i}}(\varphi)$.

Introducing these novel concepts enriches $m\mathcal{A}^{\rho}$ in terms of what the actions may describe/perform. The language can describe a much broader set of real-life scenarios where different degrees of trust relations between agents are needed. In summary, in this chapter we present, to the best of our knowledge, the first transition function that can update an *epistemic state*—*i.e.*, the knowledge/belief-graph of the agents—when considering: *(i) inconsistent beliefs*, *i.e.*, discrepancies between the beliefs currently held by an agent and some new information that she/he acquires; *(ii) trust* relations between agents; and *(iii)* the possibility for an agent to *lie*. In the next section, we formally incorporate such features in the semantics of $m\mathcal{A}^{\rho}$.

4.1.1 Enriched Domains

Let us start by providing some definitions necessary to introduce the updated transition function for $m\mathcal{A}^{\rho}$. In particular, let us formally introduce the idea of *frame of reference* in Definition 4.1, the concept of *attitude* in Definition 4.2 and finally the MEP domain enriched with attitudes in Definition 4.3. In what

follows, when clear from the context, we will make use of $\mathtt{a}$ to indicate the action instance $\mathtt{a}\langle\alpha\rangle$, with $\alpha \subseteq D(\mathcal{AG})$.

Definition 4.1: Frame of Reference [Baral et al., 2015]

The *frame of reference* of an action instance $\mathtt{a}\langle\alpha\rangle$ is a partition $\rho_{\mathtt{a}\langle\alpha\rangle} = \langle \mathbf{F}_\mathtt{a}, \mathbf{P}_\mathtt{a}, \mathbf{O}_\mathtt{a}\rangle$ of the set $D(\mathcal{AG})$, denoting the **F**ully observant, **P**artially observant, and **O**blivious agents of $\mathtt{a}\langle\alpha\rangle$, respectively.

The concept of attitude is strictly related to announcements. Therefore, in what follows, $\mathtt{a}\langle\alpha\rangle$ is assumed to be an announcement action. We recall that announcement action instances are assumed to have a single executor ($|\alpha| = 1$), referred to as the *announcer*. In this case, we make use of the short notation $\mathtt{a}\langle\mathtt{j}\rangle$ in place of $\mathtt{a}\langle\{\mathtt{j}\}\rangle$.

Definition 4.2: Attitude

The *attitude* of an agent determines how she/he updates her/his beliefs when new information is announced. Attitudes induce a refined partition of the frame of reference $\rho_{\mathtt{a}\langle\mathtt{j}\rangle} = \langle \mathbf{F}_\mathtt{a}, \mathbf{P}_\mathtt{a}, \mathbf{O}_\mathtt{a}\rangle$ as follows:

- $\mathbf{F}_\mathtt{a} = \{\mathtt{j}\} \cup \mathbf{T}_\mathtt{a} \cup \mathbf{M}_\mathtt{a} \cup \mathbf{S}_\mathtt{a}$: fully observant agents may be *the executor*, *Trustful*, *Mistrustful*, or *Stubborn*;
- $\mathbf{P}_\mathtt{a} = \mathbf{I}_\mathtt{a} \cup \mathbf{D}_\mathtt{a}$: partially observant agents may be *Impassive* or *Doubtful*.

Attitudes are specified with $m\mathcal{A}^\rho$ statements of the form "**has_attitude** $\mathtt{i}$ **wrt** $\mathtt{j}$ $\mathtt{att}$ **if** φ" (where $\mathtt{att}$ is one of the attitudes of Definition 4.2) and they define the *trust relations* among agents. Such a statement asserts that $\mathtt{i}$ bears the attitude $\mathtt{att}$ towards $\mathtt{j}$ if the condition φ is met. We assume that the attitudes of the agents are publicly visible, *except for the attitude that the announcer has with respect to her/him-self*. That is, the announcer knows whether she/he is being truthful, lying or announcing something that she/he is unaware of, while other agents do not. Instead, trustful and stubborn agents believe that the announcer is truthful (*i.e.*, they believe that the executor believes the announced property), whereas mistrustful agents believe the announcer to be lying (*i.e.*, they believe that the announcer believes the negation of the announced property). In what follows we

assume this schema of trust with respect to the executor, although it can be easily adapted to best represent different scenarios. Finally, we assume that the announcer does not modify her/his own beliefs about the property being announced. The considered attitudes provide agents with a simple set of possible behaviors. More sophisticated attitudes can be built upon the ones introduced in Definition 4.2.

Definition 4.3: MEP Domain with Attitudes

A *MEP domain with attitudes* is a tuple $D = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}, \mathcal{AT}, \varphi_{ini}, \varphi_{goal} \rangle$, where the additional element $\mathcal{AT}$ contains the attitudes relations of agents:

$$\mathcal{AT} = \{(\mathsf{i}, \mathsf{j}, \mathtt{att}, \varphi) \mid [\textbf{has\_attitude } \mathsf{i} \textbf{ wrt } \mathsf{j} \ \mathtt{att} \textbf{ if } \varphi]\}.$$

4.2 Updated Transition Function

In this section, we provide a formalization of the transition function Φ of $m\mathcal{A}^{\rho}$ that captures the aspects that we previously discussed in this section. Let a MEP domain with attitudes $D = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}, \mathcal{AT}, \varphi_{ini}, \varphi_{goal} \rangle$, an agent $\mathsf{j} \in D(\mathcal{AG})$, an e-state $\mathsf{u} \in D(\mathcal{S})$, and an action instance $\mathsf{a} \in D(\mathcal{AI})$ be given. The frame of reference ρ_{a} and the attitudes of the agents are determined by confronting the elements of the attitudes relation $\mathcal{AT}$ with the possibility u. If a is not executable in u, then $\Phi(\mathsf{a}, \mathsf{u}) = \emptyset$. Otherwise, we distinguish between ontic and epistemic actions.

Ontic Actions Since ontic actions are not affected by the introduction of inconsistent beliefs, or attitudes, the previous formalization described in Definition 2.12 is maintained.

Epistemic Actions Sensing and announcement actions modify the beliefs of agents. Since agents might acquire information that discords with previous beliefs, we must resolve the discrepancies. In the case of sensing actions, we consider all fully observant agents as executors. Since each agent trusts her/his senses, we have $\mathbf{F}_{\mathsf{a}} = \mathbf{T}_{\mathsf{a}}$. Similarly, we assume partially observant agents to keep their beliefs about

the physical features of the world unchanged, *i.e.*, $\mathbf{P_a} = \mathbf{I_a}$. Hence, the refined frame of reference of sensing actions is $\rho_{\mathtt{a}\langle\mathbf{T_a}\rangle} = \langle \mathbf{T_a}, \mathbf{I_a}, \mathbf{O_a} \rangle$.

In the case of announcement actions, it is necessary to state both the executor $\mathtt{j} \in D(\mathcal{AG})$ and the attitudes to resolve inconsistent beliefs. Therefore, the frame of reference of announcement actions is $\rho_{\mathtt{a}\langle\mathtt{j}\rangle} = \langle (\{\mathtt{j}\}, \mathbf{T_a}, \mathbf{M_a}, \mathbf{S_a}), (\mathbf{I_a}, \mathbf{D_a}), \mathbf{O_a} \rangle$. During the computation of the update, the attitude of the announcer j is set to match the perspective of the agent being currently handled by the transition function. In particular, as mentioned before, the announcer considers her/him-self stubborn; given that the announcement does not intact her/his beliefs about the truth value of the announced property. On the other hand, trustful and stubborn agents consider the announcer to be truthful; and mistrustful agents consider the announcer to be lying. Notice that the announcer is aware of the perspectives of others on her/his attitude, and so are the remaining agents. Assuming private points of view on the agents' attitudes brings an extra overhead to the problem and, therefore, we will address this issue in future works.

Let us note that the actions' effects are assumed to be deterministic. This assumption can be relaxed, as shown by Kuter et al. [2008]. Nonetheless, this would generate a significant performance overhead that would render the planning process unfeasible, most of the time. Given that the interest of the epistemic planning community lies in trying to capture the agents' information relations, we leave the formalization of non-deterministic actions as future work.

We assume the presence of a unique statement that describes the effects of an epistemic action. Namely, we allow to sense/announce a single literal at a time. Therefore, we assume the presence of a unique fluent literal that describes the effects of epistemic actions to further avoid non-determinism. This limitation is necessary as the presented transition function of epistemic actions considers the negation of the effects that, if defined as a conjunction, would generate a disjunctive form (*i.e.*, non-deterministic effects). As mentioned above, we decided to avoid non-determinism considering the already poor scalability of MEP problems even without it. Nonetheless, relaxing this restriction would allow to represent domains in which

sensing/announcing fluent formulae might be central. Conversely, ontic actions are not subject to this restriction and, therefore, can affect conjunctions of literals.

Let ℓ be the (unique) fluent literal such that [a **senses/announces** ℓ] $\in D$. With a slight abuse of notation, we define the *value* of ℓ in a possibility w as $val(\mathsf{a}, \mathsf{w}) = \mathsf{w}(\ell)$. The *effect* $e(\mathsf{a})$ of action a is equal to 1 if ℓ is a positive fluent literal ($e(\mathsf{a}) = 0$, otherwise). We use the following simplifications: given a possibility p, *(i)* p' denotes the updated version of p; and *(ii)* if not stated otherwise, we consider $\mathsf{p}'(\mathcal{F}) = \mathsf{p}(\mathcal{F})$. For clarity, we briefly describe each component of the transition function after its definition.

Definition 4.4: Epistemic Actions with Attitude in $m\mathcal{A}^{\rho}$

Let i be an agent (*i.e.*, $\mathsf{i} \in D(\mathcal{AG})$). Applying an epistemic action instance a on the pointed possibility u results in the updated pointed possibility $\Phi(\mathsf{a}, \mathsf{u}) = \mathsf{u}'$ such that:

$$\mathsf{u}'(\mathsf{i}) = \begin{cases} \mathsf{u}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_{\mathsf{a}} \\ \mathsf{P}(\mathsf{a}, \mathsf{u}) & \text{if } \mathsf{i} \in \mathbf{P}_{\mathsf{a}} \\ \mathsf{F}(\mathsf{a}, \mathsf{u}, 1) & \text{if } \mathsf{i} \in \mathbf{T}_{\mathsf{a}} \\ \mathsf{F}(\mathsf{a}, \mathsf{u}, 0) & \text{if } \mathsf{i} \in \mathbf{M}_{\mathsf{a}} \\ \mathsf{S}(\mathsf{a}, \mathsf{u}, e(\mathsf{a}), 1) & \text{if } \mathsf{i} \in \mathbf{S}_{\mathsf{a}} \\ \mathsf{S}(\mathsf{a}, \mathsf{u}, e(\mathsf{a}), 0) & \text{if } \mathsf{i} = \mathsf{j} \end{cases}$$

where $\mathtt{P}, \mathtt{F}, \mathtt{S}$ are defined below.

Description: Φ modifies the beliefs of each agent on the announced fluent with respect to her/his attitude. For instance, the beliefs of trustful agents are updated by the sub-function $\mathtt{F}$. Each sub-function ($\mathtt{P}, \mathtt{F}, \mathtt{S}$) updates the nested beliefs of the agents, *i.e.*, the beliefs that the agents have of others' perspectives.

Helper functions χ and $\bar{\chi}$

We first define the *helper functions* χ and $\bar{\chi}$. Let $\mathsf{w}'_{\mathtt{x}} = \chi(\mathsf{a}, \mathsf{w}, \mathtt{x})$ and $\bar{\mathsf{w}}'_{\mathtt{x}} = \bar{\chi}(\mathsf{a}, \mathsf{w}, \bar{\mathtt{x}})$ where: *(i)* $\mathsf{w}'_{\mathtt{x}}$ and $\bar{\mathsf{w}}'_{\mathtt{x}}$ represent the possibility w updated with χ and $\bar{\chi}$, respectively; *(ii)* $\mathtt{x}$ and $\bar{\mathtt{x}}$ represent opposite Boolean values s.t. $\mathtt{x} = \neg\bar{\mathtt{x}}$; and *(iii)* let b be 1 and 0 when executing χ and $\bar{\chi}$, respectively. Then $\mathsf{w}'_{\mathtt{x}}$ and $\bar{\mathsf{w}}'_{\mathtt{x}}$ are defined as follows:

$$\mathsf{w}'_{\mathtt{x}}(\ell) = \begin{cases} \mathtt{x} & \text{if } \ell = \mathtt{f} \\ \mathsf{u}(\ell) & \text{otherwise} \end{cases} \qquad \bar{\mathsf{w}}'_{\mathtt{x}}(\ell) = \begin{cases} \bar{\mathtt{x}} & \text{if } \ell = \mathtt{f} \\ \mathsf{u}(\ell) & \text{otherwise} \end{cases}$$

$$\left.\begin{matrix} \mathsf{w}'_{\mathsf{x}}(\mathsf{i}) \\ \bar{\mathsf{w}}'_{\mathsf{x}}(\mathsf{i}) \end{matrix}\right\} = \begin{cases} \mathsf{w}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_{\mathsf{a}} \\ \mathsf{P}(\mathsf{a},\mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{P}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{i})} \chi(\mathsf{a},\mathsf{v},\mathsf{x}) & \text{if } \mathsf{i} \in \mathbf{T}_{\mathsf{a}} \ \vee (\mathsf{i}=\mathsf{j} \wedge =1) \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{i})} \bar{\chi}(\mathsf{a},\mathsf{v},\bar{\mathsf{x}}) & \text{if } \mathsf{i} \in \mathbf{M}_{\mathsf{a}} \vee (\mathsf{i}=\mathsf{j} \wedge =0) \\ \mathsf{S}(\mathsf{a},\mathsf{w},\mathsf{x},1) & \text{if } \mathsf{i} \in \mathbf{S}_{\mathsf{a}} \end{cases}$$

DESCRIPTION: Functions χ and $\bar{\chi}$ recursively update the nested beliefs of trustful and mistrustful agents (the cases of other attitudes are delegated to the respective sub-functions). However, they do not correspond to *trustful* and *mistrustful* attitudes. The functions χ and $\bar{\chi}$ are exploited by `P` and `F`/`S` by specifying the correct value of $\mathtt{x}$ to guarantee the correct update of the beliefs of partially and fully observant agents, respectively. We make use of two Boolean variables: *(i)* $\mathtt{x}$ encodes the truth value of ℓ believed by i; *(ii)* b is a flag that keeps track of whether i is trustful ($b = 1$) or mistrustful ($b = 0$) with respect to the announcer.

Sub-function `P`

Let $\mathsf{w}'_{\mathsf{p}} = \mathsf{P}(\mathsf{a},\mathsf{w})$. Then:

$$\mathsf{w}'_{\mathsf{p}}(\mathsf{i}) = \begin{cases} \mathsf{w}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{i})} \mathsf{P}(\mathsf{a},\mathsf{v}) & \text{if } \mathsf{i} \in \mathbf{I}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{i})} \chi(\mathsf{a},\mathsf{v},0) \cup \chi(\mathsf{a},\mathsf{v},1) & \text{if } \mathsf{i} \in \mathbf{D}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{i})} \chi(\mathsf{a},\mathsf{v},\mathit{val}(\mathsf{a},\mathsf{v})) & \text{if } \mathsf{i} \in \mathbf{T}_{\mathsf{a}} \cup \mathbf{M}_{\mathsf{a}} \cup \{\mathsf{j}\} \\ \bigcup\limits_{\mathsf{v}\in\mathsf{w}(\mathsf{i})} \mathsf{S}(\mathsf{a},\mathsf{v},\mathit{val}(\mathsf{a},\mathsf{v}),1) & \text{if } \mathsf{i} \in \mathbf{S}_{\mathsf{a}} \end{cases}$$

DESCRIPTION: Function `P` updates the beliefs of partially observant agents. It updates their "direct beliefs" (*i.e.*, that represent their point of view) on ℓ and the nested beliefs of fully observant agents (by calling χ with $\mathtt{x} = \mathit{val}(\mathsf{a},\mathsf{w})$). This guarantees that agents in $\mathbf{P}_{\mathsf{a}}$ believe that (mis)trustful agents are aware of the action's effect. For doubtful agents χ is executed with both $\mathtt{x} = 0$ and $\mathtt{x} = 1$, forcing them to be ignorant about ℓ.

Sub-function F

Let $\mathsf{w}'_{\mathsf{f}} = \mathsf{F}(\mathsf{a}, \mathsf{w}, \mathsf{b})$. Then:

$$\mathsf{w}'_{\mathsf{f}}(\mathsf{i}) = \begin{cases} \mathsf{w}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_{\mathsf{a}} \\ \mathsf{P}(\mathsf{a}, \mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{P}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{i})} \chi(\mathsf{a}, \mathsf{v}, e(\mathsf{a})) & \text{if } \mathsf{i} \in \mathbf{T}_{\mathsf{a}} \vee (\mathsf{i} = \mathsf{j} \wedge {=}1) \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{i})} \bar{\chi}(\mathsf{a}, \mathsf{v}, \neg e(\mathsf{a})) & \text{if } \mathsf{i} \in \mathbf{M}_{\mathsf{a}} \vee (\mathsf{i} = \mathsf{j} \wedge {=}0) \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{i})} \mathsf{S}(\mathsf{a}, \mathsf{v}, e(\mathsf{a}), 1) & \text{if } \mathsf{i} \in \mathbf{S}_{\mathsf{a}} \end{cases}$$

Description: Function F updates the point of views on ℓ of trustful and mistrustful agents, calling χ and $\bar{\chi}$, respectively. Moreover, F deals with the beliefs of other agents with respect to to (mis)trustful agents. The flag b keeps track of whether F is executed from the perspective of a trustful ($b = 1$) or a mistrustful ($b = 0$) agent allowing to update i's perspective on the beliefs of the announcer.

Sub-function S

Let $\mathsf{w}'_{\mathsf{s}} = \mathsf{S}(\mathsf{a}, \mathsf{w}, \mathtt{x}, \mathtt{s})$. Then:

$$\mathsf{w}'_{\mathsf{s}}(\mathsf{i}) = \begin{cases} \mathsf{w}(\mathsf{i}) & \text{if } \mathsf{i} \in \mathbf{O}_{\mathsf{a}} \\ \mathsf{P}(\mathsf{a}, \mathsf{w}) & \text{if } \mathsf{i} \in \mathbf{P}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{i})} \chi(\mathsf{a}, \mathsf{v}, \mathtt{x}) & \text{if } \mathsf{i} \in \mathbf{T}_{\mathsf{a}} \vee (\mathsf{i} = \mathsf{j} \wedge \mathtt{s} = 1) \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{i})} \bar{\chi}(\mathsf{a}, \mathsf{v}, \neg\mathtt{x}) & \text{if } \mathsf{i} \in \mathbf{M}_{\mathsf{a}} \\ \bigcup\limits_{\mathsf{v} \in \mathsf{w}(\mathsf{i})} \mathsf{S}(\mathsf{a}, \mathsf{v}, \mathtt{x}, \mathtt{s}) & \text{if } \mathsf{i} \in \mathbf{S}_{\mathsf{a}} \vee (\mathsf{i} = \mathsf{j} \wedge \mathtt{s} = 0) \end{cases}$$

Description: Function S keeps the "direct" beliefs of the executor and stubborn agents unchanged and it updates their perspective on other agents' beliefs. Here, we make use of two Boolean variables: *(i)* x is defined as in $\chi/\bar{\chi}$; *(ii)* s is used to identify whether the function has been called by a stubborn agent ($\mathtt{s} = 1$) or if it is updating the "direct" beliefs of the executor ($\mathtt{s} = 0$).

While Definition 4.4 formally defines how an e-state is updated after the execution of an epistemic action when agents' attitudes are considered, let us present its intuitive meaning. Let a be an announcement action (a sensing action can be thought of as a special case of an announcement). The point of view of oblivious agents remains untouched. Since a is an epistemic action, the fluents of the pointed world u' are unchanged with respect to its previous version u. On the other hand, trustful agents' points of view are changed to fit the announced property ℓ; mistrustful

agents believe the opposite of what is announced; stubborn and impassive agents do not change their belief on ℓ. The perspective of doubtful agents is built by including also the opposite point of view with respect to ℓ. Higher-order beliefs are also correctly updated as stated in Proposition 4.1. Fully observant agents are aware of how each agent updates her/his beliefs. That is, they update the information states of other agents according to the attitudes of the other: *(i)* impassive agents do not change their belief on ℓ; *(ii)* doubtful agents no longer hold any belief on ℓ; and *(iii)* fully observant agents change their belief on ℓ as described above.

The information states of partially observant agents are updated similarly. The main difference lies in the fact that they are not aware of how fully observant agents update their points of view. Hence, partially observant agents maintain their perspective on the beliefs of fully observant agents unchanged.

Finally, the announcer considers her/him-self stubborn, since the announcement does not intact her/his beliefs, while other agents derive the attitude of the announcer depending on their own. As mentioned before, trustful and stubborn agents consider the announcer to be truthful, while mistrustful agents consider the announcer to be lying. Notice that the announcer is aware of the other agents' perspective on her/his attitude.

4.2.1 Examples of Actions Execution

In this section, we will show some examples of execution to better illustrate how the newly introduced transition function works. When needed we will describe the agents' attitudes. We will present, through labeled graphs, the e-state before and after the update for each example. While to capture all the combinations of attitudes we would need a far larger number of examples, we decided to provide only those that show the fundamental attitudes behavior and interactions. All the examples will be based on a simple variation of the Coin in the Box domain. Before presenting the examples of execution let us introduce the *Rigged Coin in the Box* in Planning Domain 4.1.

Planning Domain 4.1: Rigged Coin in the Box

Five agents, $\mathsf{l}, \mathsf{m}, \mathsf{r}, \mathsf{s}, \mathsf{c}$, are inside a room containing a box. Agents l, m stand to the left of the box; r, s are at the box's right, and c is positioned in front of the box. Inside the box, agent c placed a rigged coin. Any agent might `peek` (*sensing* action) inside the box to learn the coin position. Since the coin is rigged, the actual position (either tails or heads up) is only visible when an agent is standing in front of the box (*i.e.*, c) or at its right (*i.e.*, r and s). On the other hand, any agent that stands at the box's left (*i.e.*, l and m) will always see the coin facing tails up. All the agents can share information through the action `announce(position)` (*announcement* action) that allows them to announce the position of the coin.

For the sake of readability, in all the following examples, we will use the same initial e-state while varying the agents' attitudes and the executed action. Let us now explain the initial configuration in Example 4.1 and then illustrate the corresponding e-state, in Figure 4.1, that from now on will be identified with u. A reminder on how to "read" an e-state graphical representation is presented right after the following initial e-state description.

Example 4.1: The Initial Configuration In our initial configuration we assume that is common belief that agents l, m, r, and s do not know the coin position, *i.e.*, $\mathbf{C}_{D(\mathcal{AG})}(\neg\mathbf{B}_{\mathsf{i}}(\mathtt{heads}) \wedge \neg\mathbf{B}_{\mathsf{i}}(\neg\mathtt{heads}))$ with $\mathsf{i} \in \{\mathsf{l}, \mathsf{m}, \mathsf{r}, \mathsf{s}\}$. On the other hand, agent c is aware of the coin position and the other agents know this, that is, in our initial e-state it holds $\mathbf{C}_{D(\mathcal{AG})}(\mathbf{B}_{\mathsf{c}}(\mathtt{heads}) \vee \mathbf{B}_{\mathsf{c}}(\neg\mathtt{heads}))$. Assuming that the coin position is heads up the graphical representation of this e-state is as follows.

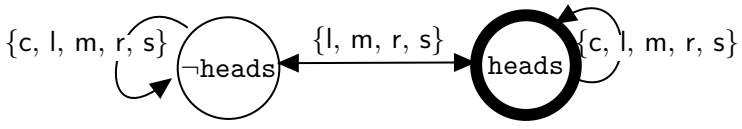


Figure 4.1: The initial e-state u of Planning Domain 4.1.

Before exploring the examples, let us briefly recall how to interpret the graphical representation of e-states. Consider Figure 4.1. The *bold-lined* world represents the actual world. If a world u is connected by an edge labeled with agent i to a world v, this means that in the world u agent i believes v to be possible.

In the initial state, each agent, except c, admits both the worlds where heads holds and where ¬heads holds. This means that such agents are uncertain about the coin position. On the other hand, agent c does know the actual configuration of the coin. We can understand this because in the actual world c admits only the world where heads hold.

Finally, observe that the remaining agents do not know what c knows. In fact, the formula $\mathbf{B}_{\mathsf{m}}(\mathbf{B}_{\mathsf{c}}(\mathtt{heads})) \vee \mathbf{B}_{\mathsf{c}}(\neg\mathtt{heads})$ is true. On the other hand, the formula $\mathbf{B}_{\mathsf{m}}(\mathbf{B}_{\mathsf{c}}(\mathtt{heads}))$ does not hold in the initial state.

Example 4.2: Correct Sensing This example shows how u is updated after the execution of the action instance $\mathtt{peek}\langle\{\mathsf{r},\mathsf{s}\}\rangle$. As said in Planning Domain 4.1 both the agents r and s are able to correctly determine whether the coin lies tails or heads up. Since we are executing a sensing action we are only interested in defining the oblivious, the fully, and the partially observant agents. In particular, for this action instance, we assume r and s to be fully observant, l and m to be partially observant and c to be oblivious. As we can see in the resulting e-state (Figure 4.2) r and s believe that the coin lies heads up. Moreover, l and m still do not know the coin position but believe that r and s know it. Finally, being c oblivious, she did not change her beliefs about anything.

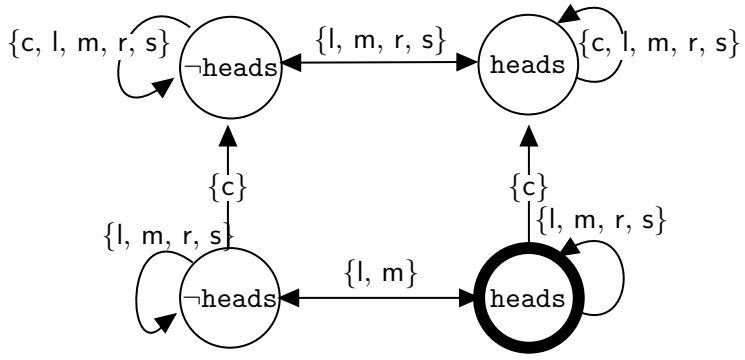


Figure 4.2: The e-state u′ obtained after the execution of correct sensing on u.

Example 4.3: Wrong Sensing This example shows how u is updated after the execution of the action instance $\mathtt{peek}\langle\{\mathsf{l},\mathsf{m}\}\rangle$. As said in Planning Domain 4.1 both the agents l and m always see the coin lying tails up. Since we are executing a sensing action we are only interested in defining the oblivious, the fully, and the partially observant agents. In particular, for this action instance, we assume l and m to be fully observant, r and s to be partially observant and c to be oblivious. As we can see in the resulting e-state (Figure 4.3) l and m believe that the coin lies tails up. Moreover, r and s still do not know the coin position but believe that l and m know it. Finally, being c oblivious, she did not change her beliefs about anything.

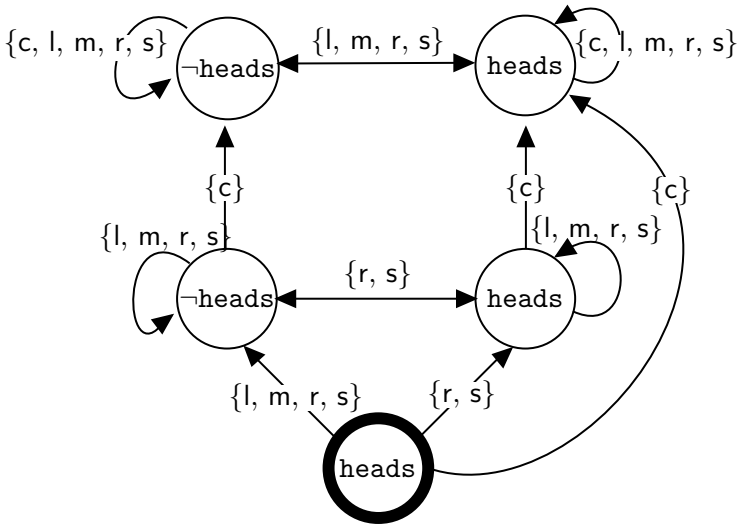


Figure 4.3: The e-state u' obtained after the execution of wrong sensing on u.

Example 4.4: Trust & Mistrust This example shows how u is updated after the execution of the action instance announce⟨c⟩ where c announces heads. In particular, for this action instance, we assume:

- c to be the *executor*;
- l to be *trustful*;
- m to be *mistrustful*;
- r to be *impassive*; and
- s to be *doubtful*;

As we can see in the resulting e-state (Figure 4.4) l and m believe that the coin lies heads and tails up, respectively. Moreover, l and m believe that c shares their beliefs on the coin position. Finally, agents r and s, still do not know the coin position but believe that c, l, and m know it.

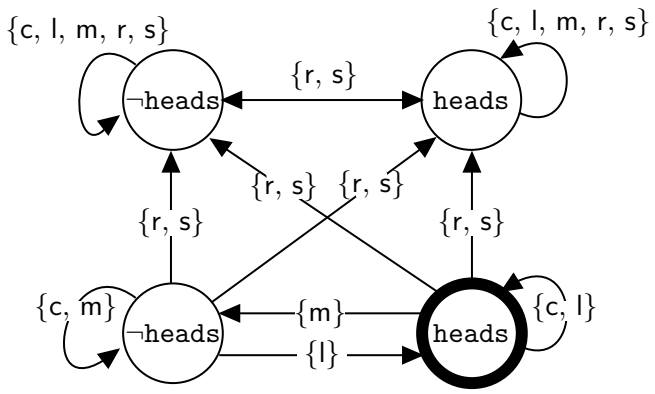


Figure 4.4: The e-state u′ obtained after the announcement of heads in u with trustful & mistrustful listeners.

Example 4.5: (Mis)Trust & Stubbornness This example shows how u is updated after the execution of the action instance announce⟨c⟩ where c announces heads. Differently from the previous example, the agents' attitudes are as follows:

- c to be the *executor*;
- l to be *trustful*;
- m to be *mistrustful*;
- r to be *doubtful*; and
- s to be *stubborn*;

As we can see in the resulting e-state (Figure 4.5) agents c and l believe that the coin lies heads up while m thinks that it lies tails up. Even if s did not change her beliefs on the coin position she knows what c, l believe that the coin is heads up while m think that it is tails up. Agent r, instead still does not know the coin position but believes that c, l, and m know it. We will use a dotted square to indicate that the edges that reach such a square, transitively reach all the worlds contained.

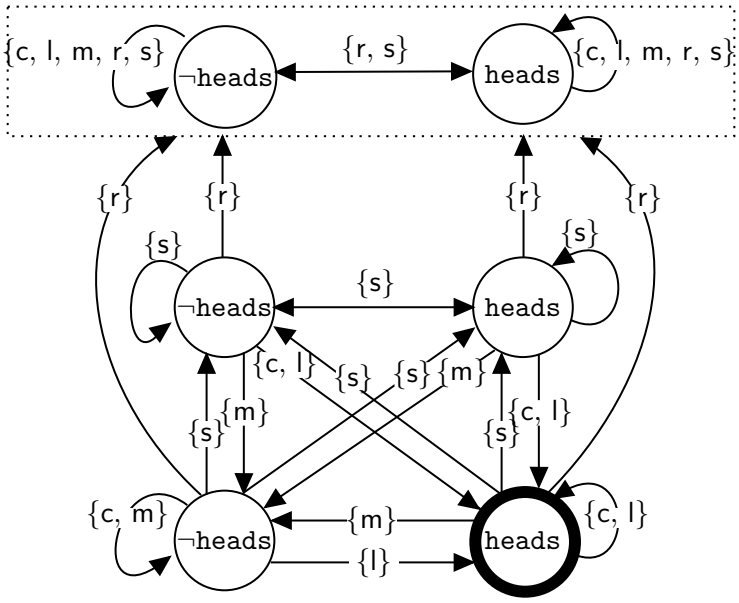


Figure 4.5: The e-state u′ obtained after the announcement of heads in u with (mis)trustful & stubborn listeners.

Example 4.6: Lie This example shows how u is updated after the execution of the action instance announce⟨c⟩ where c announces ¬heads. Let us note that this announcement, since it is performed by c that believes heads, is a lie. For this action instance, we assume:

- c to be the *executor*;
- l to be *trustful*;
- m to be *mistrustful*;
- r to be *doubtful*; and
- s to be *oblivious*;

As we can see in the resulting e-state (Figure 4.6) agent l believed to the lie and now has a wrong belief about the coin position. On the other hand and m did not believe the announcer and, therefore, now correctly think that the coin lies heads up. Being the executor, c knew that she was lying and, therefore, still believes that the coin is heads up. Moreover, l and m believe that c shares their beliefs on the coin position. Agents r, still does not know the coin position but believe that c, l, and m know it. Finally, being s oblivious, she did not change her beliefs about anything.

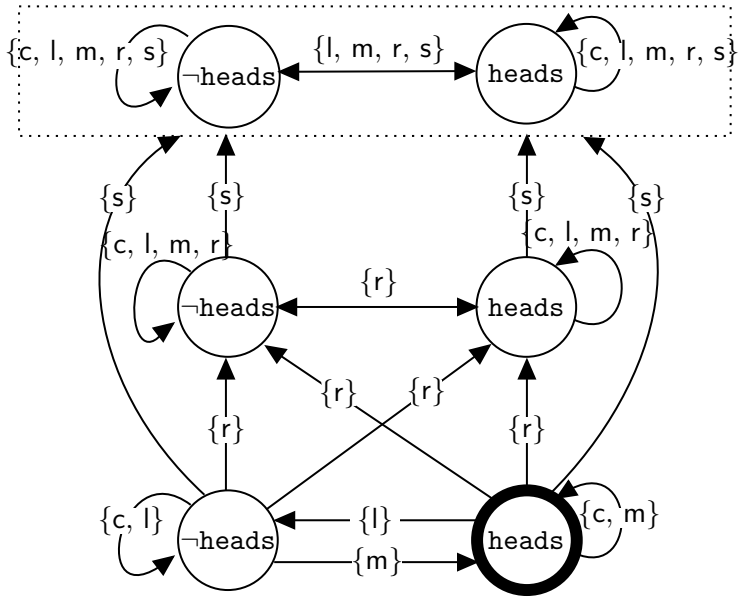


Figure 4.6: The e-state u′ obtained after the execution of a lie (*i.e.*, c announce ¬heads) in u with trustful & mistrustful listeners.

4.2.2 Desired Properties

Following the usual schema, we list some properties concerning the new actions that consider attitudes. Complete proofs are available in Appendix A.4.

Proposition 4.1: Epistemic Actions Properties

Let $\mathsf{a}\langle\mathsf{j}\rangle$ *be an* epistemic action instance *such that* j ***announces*** ℓ *(where* ℓ *is either* $\mathtt{f}$ *or* $\neg\mathtt{f}$*). Let* u *be an e-state and let* u' *be its updated version,* i.e.*,* $\Phi(\mathsf{a},\mathsf{u}) = \mathsf{u}'$*, then it holds that:*

(1) $\mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathsf{a}}(\mathbf{C}_{\mathbf{T}_\mathsf{a}}(\ell \wedge \mathbf{B}_\mathsf{j}(\ell)))$*;*

(2) $\mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathsf{a}}(\mathbf{C}_{\mathbf{M}_\mathsf{a}}(\neg\ell \wedge \mathbf{B}_\mathsf{j}(\neg\ell)))$*;*

(3) $\forall \mathsf{i} \in (\mathbf{S}_\mathsf{a} \cup \{\mathsf{j}\}),\ \mathsf{u}' \models \varphi$ *if* $\mathsf{u} \models \varphi$ *with* $\varphi \in \{\mathbf{B}_\mathsf{i}(\ell);\ \mathbf{B}_\mathsf{i}(\neg\ell);\ (\neg\mathbf{B}_\mathsf{i}(\ell) \wedge \neg\mathbf{B}_\mathsf{i}(\neg\ell))\}$*;*

(4) $\forall \mathsf{i} \in \mathbf{F}_\mathsf{a},\ \mathsf{u}' \models \mathbf{C}_{\mathbf{P}_\mathsf{a}}(\mathbf{B}_\mathsf{i}(\ell) \vee \mathbf{B}_\mathsf{i}(\neg\ell))$*;*

(5) $\forall \mathsf{i} \in \mathbf{D}_\mathsf{a},\ \mathsf{u}' \models \mathbf{C}_{\mathbf{F}_\mathsf{a}\cup\mathbf{P}_\mathsf{a}}(\neg\mathbf{B}_\mathsf{i}(\ell) \wedge \neg\mathbf{B}_\mathsf{i}(\neg\ell))$*;*

(6) *for every pair of agents* $\mathsf{i} \in D(\mathcal{AG})$ *and* $\mathsf{o} \in \mathbf{O}_\mathsf{a}$*, and a belief formula* φ*,* $\mathsf{u}' \models \mathbf{B}_\mathsf{i}(\mathbf{B}_\mathsf{o}(\varphi))$ *if* $\mathsf{u} \models \mathbf{B}_\mathsf{i}(\mathbf{B}_\mathsf{o}(\varphi))$*.*

The features presented in Proposition 4.1 capture fundamental aspects of the updated e-state after the execution of an announcement. Intuitively, they model the following properties:

(1) Fully observant agents think that trustful agents believe that the announced property holds *and* that the announcer believes such property;

(2) Fully observant agents think that mistrustful agents believe that the announced property does not hold *and* that the announcer does not believe such property;

(3) Stubborn agents and the announcer do not modify their beliefs about the announced property.

(4) Partially observant agents believe that fully observant agents (including the announcer) are certain of the value of the announced property;

(5) Non-oblivious agents believe that doubtful agents are uncertain on the truth value of the announced property;

(6) Every agent (even oblivious agents) knows that oblivious agents do not change their beliefs.

4.3 Related Work

The enriched semantics of $m\mathcal{A}^{\rho}$ has been implemented in the C++ solver EFP (presented in chapter 5) that is now able to tackle families of problems that consider complex aspects such as doxastic reasoning, lying agents, faulty perception, etc.

To the best of our knowledge, in the literature, only one other solver, RP-MEP [Muise et al., 2015], can tackle such domains. This solver firstly encodes a MEP problem into a classical planning problem and then handles the solving phase with a "classical" planner. The key difference between EFP and RP-MEP is that while RP-MEP grounds the agents' beliefs and reasons on them as if they were "static facts", EFP builds and interprets e-states, and it updates them using a full-fledged epistemic transition function. For this reason, the latter constitutes a more comprehensive framework. In fact, given the effects of an action instance (a single literal/conjunction of literals), the transition function of $m\mathcal{A}^{\rho}$ *propagates* the effects and updates the nested beliefs of agents automatically. Conversely, RP-MEP needs the *propagated* effects to be explicit. Nonetheless, the "implicit beliefs update" of EFP makes this approach less performing with respect to RP-MEP. The latter, in fact, with a little extra effort on the input description, is able to solve the same domains as EFP, outperforming it whenever the depth of the formulae is set to a reasonable number.

Other theoretical approaches explore the idea of trust between agents [Castelfranchi and Falcone, 1998, Herzig et al., 2010, Rodenhäuser, 2014]. For example, following Castelfranchi and Falcone [1998], Herzig et al. devised a logic to capture the "*truster's belief about certain relevant properties of the trustee with respect to a given goal*". While the ideas of Castelfranchi and Falcone are elegantly captures

by this logic, Herzig et al. do not actively use the notion of trust to modify the outcome of an action's execution with respect to an agent's perspective, that is what we are trying to accomplish with our idea of attitudes. Conversely, Rodenhäuser [2014] proposes a theoretical framework where agents make use of the reliability of the source (using the so-called *dynamic attitudes*) to correctly update their beliefs. While our idea of attitudes stems from such work, we only introduced attitudes that are intuitively derived from real-world scenarios without considering more complex ones. In the future, we plan to expand our formalization and the planner with the attitudes presented by Rodenhäuser [2014] along with the idea of "*plausibility*".

Belief revision [Baltag and Smets, 2016] in presence of inconsistent/false beliefs has been explored by Baral et al. and Herzig et al.. These works focus on the introduction of a theoretical framework for resolving inconsistencies. Hence, we only compare their approaches with our formalization. Baral et al. [2015] mainly focus on *false beliefs*, dividing their approach into two steps. First, they remove the incorrect beliefs of fully observant agents from the current e-state; and next, they apply the action on the revised state. Their solution correctly accounts for false beliefs, but it is not sufficient to resolve inconsistent beliefs. On the other hand, Herzig et al. [2005] propose a multi-agent extension of AGM-style belief revision [Gärdenfors and Makinson, 1988]. Once a new property is acquired with a sensing action, the agents update their beliefs according to their view of the observation. The revision procedure makes use of a preference-based revision operator. While revising the agents' beliefs could be a viable solution we believe that having to decouple the belief revision from the e-state update for each action execution would generate an excessive overhead in the solving process.