Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)

EFP 2.0: A Multi-Agent Epistemic Solver with Multiple E-State Representations

Francesco Fabiano
DMIF Department
University of Udine
I-33100 Udine, Italy
francesco.fabiano@uniud.it

Alessandro Burigana
DMIF Department
University of Udine
I-33100 Udine, Italy
burigana.alessandro@spes.uniud.it

Agostino Dovier
DMIF Department
University of Udine
I-33100 Udine, Italy
agostino.dovier@uniud.it

Enrico Pontelli
Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
epontell@cs.nmsu.edu

Abstract

Multi-agent systems have been employed to model, simulate and explore a variety of real-world scenarios. It is becoming more and more important to investigate formalisms and tools that would allow us to exploit automated reasoning in these domains. An area that has received increasing attention is the use of multi-agent systems which allow an agent to reason about the knowledge and beliefs of other agents. This type of reasoning, *i.e.*, about agents' perception of the world and also about agents' knowledge of her and others' knowledge, is referred to as *epistemic reasoning*.

This paper presents an updated formalization and implementation of a multi-agent epistemic planner, called EFP. In particular, the paper explores the advantages of using alternative state representations that deviate from the commonly used Kripke structures. The paper explores such alternatives in the context of an action language for multi-agent epistemic planning. The paper presents also an actual implementation of a planner that uses the novel ideas, demonstrating concrete performance improvements on benchmarks collected from the literature.

Motivation

Artificial Intelligence, or AI, has recently gained attention in several communities. It is, in fact, becoming essential for the majority of the real-world scenarios, *e.g.*, Industry 4.0, to exploit techniques derived from the fields of *automated reasoning* and *knowledge representations*. In particular, the field of *automated planning* is one of the most important branches of AI. That is why we decided to focus our researches on the planning problem.

Reasoning about actions and information has been one of the prominent interests since the beginning of the AI (McCarthy 1959). The "simple" task of reasoning in the *classical planning* environments rapidly evolved into more complex problems (Torreño, Onaindia, and Sapena 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, this paper studies one of these settings that, even if formalized by studies of philosophy/logic in the early sixties, is a somewhat recent introduction in the planning scenario (Van Ditmarsch, van Der Hoek, and Kooi 2007). That is, the *Multi-agent Epistemic Planning problem* (MEP) or, as usually it is called in literature, *epistemic planning*. 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 in 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 to consider *nested* and *group* beliefs that are not easily extracted from the state description by a human reader. This inherent complexity is reflected in computational overhead that brings, most of the time, infeasibility to the solving process. That is why it is necessary to advance in the study of the epistemic planning problem (Fabiano 2019; Le et al. 2018; Huang et al. 2017; Wan et al. 2015; Muise et al. 2015).

Therefore, in this work, we present an updated version of the *Epistemic Forward Planner* (EFP) presented in Le et al. (2018). As main contribution we integrated the planner with a new *e-state* (epistemic state) representation, based on the concept of *possibilities* (Gerbrandy and Groeneveld 1997), along with a new transition function derived by Fabiano et al. (2019). Finally, in the experimental evaluations, we will show how this new implementation outperforms the state-of-the-art planner, especially when combined with the removal of already *visited e-states*.

Epistemic Planning

Dynamic Epistemic Logic Epistemic reasoning was initially formalized by logicians in the early sixties. This formalization rapidly evolved from allowing to reason on the knowledge/beliefs of agents in a static environment into *Dy-*

namic Epistemic Logic (DEL), a formalism used to reason not only on the state of the world but also on *information change* in dynamic domains. As discussed in Van Ditmarsch, van Der Hoek, and Kooi (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*." Due to space limits we will provide only the fundamental definitions and intuitions of DEL; the interested reader is referred to Fagin and Halpern (1994) for further details.

Let $\mathcal{AG}$ be a set of agents s.t. $|\mathcal{AG}| = n$ with $n \geq 1$ and let $\mathcal{F}$ be a set of propositional variables, called *fluents*. Each *world* is described by a subset of elements of $\mathcal{F}$ (intuitively, those that are "true"). Moreover, in epistemic logic each agent $\mathsf{ag} \in \mathcal{AG}$ is associated to an epistemic modal operator $\mathbf{B}_{\mathsf{ag}}$ that represents the knowledge/belief of ag herself. Finally, epistemic *group operators* $\mathbf{E}_\alpha$ and $\mathbf{C}_\alpha$ are also introduced in epistemic logic. Intuitively, $\mathbf{E}_\alpha$ and $\mathbf{C}_\alpha$ represent the knowledge/belief of a group of agents α and the *common knowledge/belief* of α, respectively. To be more precise, as in Baral et al. (2015), we have that:

Definition 1 (Fluent formula) A fluent formula *is a propositional formula built using fluents in $\mathcal{F}$ as propositional variables and the propositional operators $\wedge, \vee, \Rightarrow, \neg$. A* fluent atom *is a formula composed of just an element $\mathsf{f} \in \mathcal{F}$; a* fluent literal *is either a fluent atom $\mathsf{f} \in \mathcal{F}$ or its negation $\neg\mathsf{f}$.*

With a slight abuse of notation, we will refer to fluent literals simply as *fluents*.

Definition 2 (Belief formula) *A* belief formula *is defined as follows:*

- *A fluent formula is a belief formula;*
- *If φ is a belief formula and $\mathsf{ag} \in \mathcal{AG}$, then $\mathbf{B}_{\mathsf{ag}}\varphi$ is a belief formula;*
- *If φ_1, φ_2 and φ_3 are belief formulae, then $\neg\varphi_3$ and $\varphi_1 \;\mathsf{op}\; \varphi_2$ are belief formulae, where $\mathsf{op} \in \{\wedge, \vee, \Rightarrow\}$;*
- *If φ is a belief formula and $\emptyset \neq \alpha \subseteq \mathcal{AG}$ then $\mathbf{E}_\alpha\varphi$ and $\mathbf{C}_\alpha\varphi$ are belief formulae.*

From now on we will denote with $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ the language of the belief formulae over the sets $\mathcal{F}$ and $\mathcal{AG}$.

Example 1 *Let us consider the formula $\mathbf{B}_{\mathsf{ag}_1}\mathbf{B}_{\mathsf{ag}_2}\varphi$. This formula expresses that the agent ag_1 believes that the agent ag_2 believes that φ is true. The formula $\mathbf{B}_{\mathsf{ag}_1}\neg\varphi$ expresses that the agent ag_1 believes that φ is false.*

The classical way of providing a semantics for the language of epistemic logic is in terms of *pointed Kripke structures* (Kripke 1963).

Definition 3 (Kripke structure) *Let $|\mathcal{AG}| = n$ with $n \geq 1$. A* Kripke structure *is a tuple $\langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_\mathsf{n}\rangle$, such that:*

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

Definition 4 (Pointed Kripke structure) *A* pointed Kripke structure *is a pair (M, s) where M is a Kripke structure as defined above, and $\mathsf{s} \in S$, where s points at the real world.*

Following the notation of Baral et al. (2015), we will indicate with $M[S], M[\pi]$, and $M[\mathsf{i}]$ the components S, π, and $\mathcal{B}_\mathsf{i}$ of M, respectively. Intuitively, $M[S]$ captures all the worlds that the agents believe to be possible and $M[\mathsf{i}]$ encodes the beliefs of each agent. More formally the semantics on pointed Kripke structures is as follows:

Definition 5 (Entailment w.r.t. a Kripke structure) *Given a fluent f, a belief formula φ, a set of agents $\mathcal{AG}$ s.t. $|\mathcal{AG}| = n$, an agent $\mathsf{ag_i} \in \mathcal{AG}$ with $1 \leq \mathsf{i} \leq n$, a group of agents $\alpha \subseteq \mathcal{AG}$, a pointed Kripke structure (M, s) with $M = \langle S, \pi, \mathcal{B}_1, \ldots, \mathcal{B}_\mathsf{n}\rangle$:*

1. *$(M, \mathsf{s}) \models \mathsf{f}$ if $M[\pi](\mathsf{s}) \models \mathsf{f}$;*
2. *$(M, \mathsf{s}) \models \mathbf{B}_{\mathsf{ag_i}}\varphi$ if for each t such that $(\mathsf{s}, \mathsf{t}) \in M[\mathsf{i}]$ it holds that $(M, \mathsf{t}) \models \varphi$;*
3. *$(M, \mathsf{s}) \models \mathbf{E}_\alpha\varphi$ if $(M, \mathsf{s}) \models \mathbf{B}_{\mathsf{ag_i}}\varphi$ for all $\mathsf{ag_i} \in \alpha$;*
4. *$(M, \mathsf{s}) \models \mathbf{C}_\alpha\varphi$ if $(M, \mathsf{s}) \models \mathbf{E}^k_\alpha\varphi$ for every $k \geq 0$, where $\mathbf{E}^0_\alpha\varphi = \varphi$ and $\mathbf{E}^{k+1}_\alpha\varphi = \mathbf{E}_\alpha(\mathbf{E}^k_\alpha\varphi)$;*
5. *the semantics of the traditional propositional operators is defined as usual.*

Epistemic Planning Domains Let us introduce the notion of *multi-agent epistemic planning domain*. Intuitively, an epistemic planning domain contains all the necessary information to define a planning problem in a multi-agent epistemic scenario.

Definition 6 (Multi-agent epistemic planning domain) *We define a multi-agent epistemic domain as the tuple $D = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}, \varphi_i, \varphi_g\rangle$ where:*

- *$\mathcal{F}$ is the set of all the* fluents *of D;*
- *$\mathcal{AG}$ is the set of the* agents *of D;*
- *$\mathcal{A}$ represents the set of all the* actions *of D;*
- *φ_i is the belief formula that describes the* initial conditions *of the planning process; and*
- *φ_g is the belief formula that represents the* goal *condition.*

Moreover, from now on, with the term *action instance* we will indicate an element of the set $\mathcal{AI} = \mathcal{A} \times \mathcal{AG}$. Intuitively, an action instance $\mathsf{a}\langle\mathsf{ag}\rangle$ identifies the execution of the action a by the agent ag.

Given a domain D we will refer to its components through the *parenthesis* operator. For instance to access the elements $\mathcal{F}$ and $\mathcal{AG}$ of D we will use the more compact notation $D(\mathcal{F})$ and $D(\mathcal{AG})$, respectively.

Furthermore, we will indicate a state of an epistemic planning domain as *e-state*. Therefore, an e-state, that in our case can be represented by both a Kripke structure or by a Possibility[1], captures a configuration of the world and of the agents' knowledge/belief.

[1]Concept introduced in the next Section.

Epistemic Action Languages

The action language $m\mathcal{A}^*$ Even though automated planning and DEL are both vastly explored fields of study, their combination, *i.e.*, epistemic planning, has gained attention only recently in the AI community. In the last few years epistemic planning has been tackled using different techniques, such as:

1. reducing the epistemic planning to a classical planning problem (Muise et al. 2015; Kominis and Geffner 2015);
2. adapting algorithms from other planning domains, *e.g.*, contingent planning (Huang et al. 2017); or
3. addressing the problem with already existing solvers supported by domain-specific external epistemic procedures to derive the agent's knowledge status (Hu, Miller, and Lipovetzky 2019).

Nevertheless, all the previously mentioned approaches are not suitable to reason on the agents' beliefs[2] on the full extent of $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$. For instance: i) the reduction to classical planning implies bounded nested-knowledge; ii) the system presented in Huang et al. (2017) cannot deal with dynamic common knowledge; and finally iii) the use of domain-specific procedures implies a loss of generality that is a limit in a solver design. Moreover the approach in Hu, Miller, and Lipovetzky (2019) cannot reason about agents' beliefs (*i.e.*, on **KD45** logic) but only on the agents' knowledge (*i.e.*, **S5** logic). Hence it is important to find strategies to address the planning problem on the full extent of $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$ where the underlying e-states are able to capture the concept of belief.

To the best of our knowledge, the first formalization of a comprehensive action language for multi-agent epistemic planning is $m\mathcal{A}^*$ (Baral et al. 2015). $m\mathcal{A}^*$ is high-level action language that allows to reason about agents' beliefs on $\mathcal{L}^{\mathbf{C}}_{\mathcal{AG}}$where states are represented as Kripke structures. In particular, $m\mathcal{A}^*$ has an English-like syntax and exploits the concepts of *events* to define the transition function. The entailment, on the other hand, is defined following Definition 5.

In Baral et al. (2015), the authors distinguish between three types of actions:

1. *world-altering* actions (also called *ontic*): used to modify certain properties (*i.e.*, fluents) of the world;
2. *sensing* actions: used by an agent to refine her beliefs about the world; and
3. *announcement* actions: used by an agent to affect the beliefs of other agents.

Given a domain D and an action instance $\mathsf{a} \in D(\mathcal{AI})$, a fluent literal $\mathsf{f} \in D(\mathcal{F})$, a fluent formula ϕ and the belief formulae φ, ψ we can introduce the syntax of $m\mathcal{A}^*$.

- **executable** a **if** φ captures the *executability conditions*;
- a **causes** f **if** ψ captures the *ontic* actions;
- a **determines** f **if** ψ captures the *sensing* actions;
- a **announces** ϕ **if** ψ captures the *announcement* actions.

In multi-agent domains another important concept is the *action observability*. That is, the execution of an action might change or not the beliefs of an agent depending on whether or not she is aware of the action's occurrence. $m\mathcal{A}^*$ identifies three levels of action observability given an action a, an agent ag:

- *fully observant* (denoted by $\mathsf{ag} \in F_{\mathsf{a}}$) if ag knows about the execution of a and about its effects on the world;
- *partially observant* (denoted by $\mathsf{ag} \in P_{\mathsf{a}}$) if ag knows about the execution of a but she does not know how a affected the world;
- *oblivious* (denoted by $\mathsf{ag} \in O_{\mathsf{a}}$) if ag does not know about the execution of a.

Let us observe that partial observability for world-altering actions is not admitted as, whenever an agent is aware of the execution of an ontic action, she must know its effects on the world as well. A final remark has to be done about the actions' determinism. In both Le et al. (2018) and our approach the actions' effects are assumed to be deterministic. This assumption can be relaxed allowing non-deterministic effects. From the planning prospective this can be done, for instance, following the approach presented in Kuter et al. (2008). For the sake of readability we will not explore $m\mathcal{A}^*$ in more detail and we refer the interested reader to Le et al.; Baral et al. (2018; 2015) for a complete description.

The action language $m\mathcal{A}^\rho$ The main contribution of this paper is an improved transition function and an implementation for $m\mathcal{A}^\rho$ (Fabiano et al. 2019). $m\mathcal{A}^\rho$ is an epistemic action language based on $m\mathcal{A}^*$ that, instead of using Kripke structures as e-states, uses *possibilities* (Gerbrandy and Groeneveld 1997). Before introducing our contributions it is therefore necessary to quickly introduce the notion of *possibility*.

In this section we will briefly present the main concepts related to $m\mathcal{A}^\rho$ and its e-states representation: possibilities. For a complete survey on possibilities we recommend Gerbrandy (1999).

Let us start by introducing some notions from the *non-well-founded set* theory.

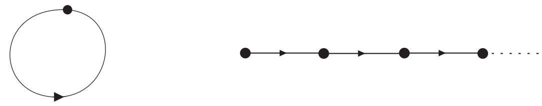
Definition 7 (Non-well-founded set (Aczel 1988)) *Let E^0 be a set, E^1 one of its elements, E^2 any element of E^1, and so on. A* descent *is the sequence of steps from E^0 to E^1, E^1 to E^2, etc. ... A set is* non-well-founded *when among its descents there are some which are infinite.*

A simple example of non-well-founded set is the set $\Omega = \{\Omega\}$ represented in Figure 1.

Definition 8 (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.*, $\delta_{\mathsf{n}} = \{\delta_{\mathsf{n}'} \mid (\mathsf{n}, \mathsf{n}') \in E\}$.*

Given a pointed graph $(\mathcal{G}, \mathsf{n})$ *(*i.e.*, a graph with a particular node $\mathsf{n} \in V$ identified), if δ is a decoration of $\mathcal{G}$, then $(\mathcal{G}, \mathsf{n})$ is a* picture *of the set δ_{n}.*

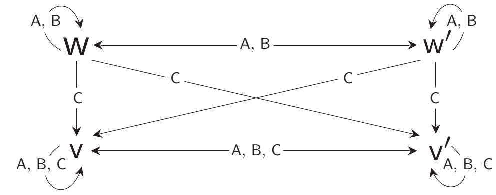
[2]As mentioned in Fagin and Halpern (1994) the concept of knowledge and belief are encoded by two different logics: **S5** and **KD45**, respectively.



(a) Standard picture of Ω.

(b) Unfolding of Ω.

Figure 1: The non-well-founded set $\Omega = \{\Omega\}$ (Aczel 1988).



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

In particular, in Aczel (1988) is shown that, in non-well-founded theory, every graph has a unique decoration and every decoration can be converted to a unique *system of equations*.

We are now ready to introduce the concept of possibility.

Definition 9 (Possibility) *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* $\mathsf{f} \in \mathcal{F}$ *a truth value* $\mathsf{u}(\mathsf{f}) \in \{0, 1\}$ *and to each agent* $\mathsf{ag} \in \mathcal{AG}$ *an information state* $\mathsf{u}(\mathsf{ag}) = \sigma$.
- *An* information state σ *is a (non-well-founded) set of possibilities.*

The idea of possibilities is central in $m\mathcal{A}^\rho$. In fact this language, instead of using Kripke structures, exploits possibilities as e-states. That is, $m\mathcal{A}^\rho$, while keeping the same syntax of $m\mathcal{A}^*$, changes the way of representing an epistemic state. Changing the underlying structure implies also a different formalization of the transition function (introduced in the following Section). The differences in the e-state representation and in the transition function are what allowed us to outperform the state-of-the-art comprehensive epistemic planner presented in (Le et al. 2018) by orders of magnitude in most of the experiments.

Possibilities in MEP Following Fabiano et al. (2019) we will now briefly explain how a possibility can be used to represent an e-state (Figure 2). 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 $\mathsf{u}(\mathsf{f})$ assigns a truth value to the fluent f in u while $\mathsf{u}(\mathsf{ag})$ represents the (non-well-founded) set of possibilities that could be true w.r.t. the agent ag.

The choice of possibilities over Kripke structures as e-state representation provides several advantages. The most important is that, as said in Gerbrandy and Groeneveld (1997), *a possibility represents the solution to the minimal system of equations in which all bisimilar Kripke structures are collapsed*. More 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 to the newly introduced transition function it has been possible to maintain e-states with smaller size, w.r.t. EFP 1.0, 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 in Paige and Tarjan; Dovier, Piazza, and Policriti (1987; 2004), during the solving process. Another advantage in using possibilities derived from their non-well-founded aspect. In fact, since a possibility is a non-well-founded graph, which nodes are themselves possibilities, our planner stores each calculated possibility; and, whenever needed, EFP 2.0 retrieve the stored possibilities (through pointers) to reuse it as "node" inside a new e-state. To summarize, although possibilities and Kripke structures are tightly connected, 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, w.r.t. Kripke structure, that allows us to save computational resources.

[3]W.r.t. the e-states generated following $m\mathcal{A}^*$.

An Updated Transition Function for $m\mathcal{A}^\rho$

As first main contribution we present the formalization of a new transition function for the action language $m\mathcal{A}^\rho$, an epistemic action language initially introduced in Fabiano et al. (2019). $m\mathcal{A}^\rho$ 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. The following definition combines the concept of Gerbrandy (1999) with the action language $m\mathcal{A}^*$.

Definition 10 (Entailment w.r.t. possibilities) *Let the belief formulae $\varphi, \varphi_1, \varphi_2$, a fluent f, an agent ag, a group of agents α, and a possibility u be given.*

1. $\mathsf{u} \models \mathsf{f}$ *if* $\mathsf{u}(\mathsf{f}) = 1$*;*
2. $\mathsf{u} \models \mathbf{B}_{\mathsf{ag}}\varphi$ *if for each* $\mathsf{v} \in \mathsf{u}(\mathsf{ag})$, $\mathsf{v} \models \varphi$*;*
3. $\mathsf{u} \models \neg\varphi$ *if* $\mathsf{u} \not\models \varphi$*;*
4. $\mathsf{u} \models \varphi_1 \vee \varphi_2$ *if* $\mathsf{u} \models \varphi_1$ *or* $\mathsf{u} \models \varphi_2$*;*
5. $\mathsf{u} \models \varphi_1 \wedge \varphi_2$ *if* $\mathsf{u} \models \varphi_1$ *and* $\mathsf{u} \models \varphi_2$*;*
6. $\mathsf{u} \models \mathbf{E}_\alpha\varphi$ *if* $\mathsf{u} \models \mathbf{B}_{\mathsf{ag}}\varphi$ *for all* $\mathsf{ag} \in \alpha$*;*

7. $\mathsf{u} \models \mathbf{C}_\alpha\varphi$ *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 an updated version of the transition function. The new transition function is more compact and therefore, more understandable than the original. Moreover, the "simplicity" of the e-states update formalization is reflected in a much cleaner and faster implementation. Let a domain D, its set of action instances $D(\mathcal{AI})$, and the set $\mathcal{S}$ of all the possibilities reachable from $D(\varphi_i)$ 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 $m\mathcal{A}^\rho$ relative to D is defined as follows.

Definition 11 ($m\mathcal{A}^\rho$ transition function) *Allow us to use the compact notation* $\mathsf{u}(\mathcal{F}) = \{\mathsf{f} \mid \mathsf{f} \in D(\mathcal{F}) \wedge \mathsf{u} \models \mathsf{f}\} \cup \{\neg\mathsf{f} \mid \mathsf{f} \in D(\mathcal{F}) \wedge \mathsf{u} \not\models \mathsf{f}\}$ *for the sake of readability. Let an action instance* $\mathsf{a} \in D(\mathcal{AI})$, *a possibility* $\mathsf{u} \in \mathcal{S}$ *and an agent* $\mathsf{ag} \in D(\mathcal{AG})$ *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:*

- *Let us consider the case of an* ontic *action instance* a. *We then define* u' *such that:*

  $e(\mathsf{a}, \mathsf{u}) = \{\ell \mid (\mathsf{a}\ \textit{\textbf{causes}}\ \ell) \in D\}$; *and*

  $\overline{e(\mathsf{a}, \mathsf{u})} = \{\neg\ell \mid \ell \in e(\mathsf{a}, \mathsf{u})\}$ *where* $\neg\neg\ell$ *is replaced by* ℓ.

$$\mathsf{u}'(\mathsf{f}) = \begin{cases} 1 & \textit{if}\ \mathsf{f} \in (\mathsf{u}(\mathcal{F}) \setminus \overline{e(\mathsf{a}, \mathsf{u})}) \cup e(\mathsf{a}, \mathsf{u}) \\ 0 & \textit{if}\ \neg\mathsf{f} \in (\mathsf{u}(\mathcal{F}) \setminus \overline{e(\mathsf{a}, \mathsf{u})}) \cup e(\mathsf{a}, \mathsf{u}) \end{cases}$$

$$\mathsf{u}'(\mathsf{ag}) = \begin{cases} \mathsf{u}(\mathsf{ag}) & \textit{if}\ \mathsf{ag} \in O_\mathsf{a} \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{ag})} \Phi(\mathsf{a}, \mathsf{w}) & \textit{if}\ \mathsf{ag} \in F_\mathsf{a} \end{cases}$$

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

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

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

$$\mathsf{u}'(\mathsf{ag}) = \begin{cases} \mathsf{u}(\mathsf{ag}) & \textit{if}\ \mathsf{ag} \in O_\mathsf{a} \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{ag})} \Phi(\mathsf{a}, \mathsf{w}) & \textit{if}\ \mathsf{ag} \in P_\mathsf{a} \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{ag}):\ e(\mathsf{a},\mathsf{w}) = e(\mathsf{a},\mathsf{u})} \Phi(\mathsf{a}, \mathsf{w}) & \textit{if}\ \mathsf{ag} \in F_\mathsf{a} \end{cases}$$

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

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

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

$$\mathsf{u}'(\mathsf{ag}) = \begin{cases} \mathsf{u}(\mathsf{ag}) & \textit{if}\ \mathsf{ag} \in O_\mathsf{a} \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{ag})} \Phi(\mathsf{a}, \mathsf{w}) & \textit{if}\ \mathsf{ag} \in P_\mathsf{a} \\ \bigcup\limits_{\mathsf{w} \in \mathsf{u}(\mathsf{ag}):\ e(\mathsf{a},\mathsf{w}) = e(\mathsf{a},\mathsf{u})} \Phi(\mathsf{a}, \mathsf{w}) & \textit{if}\ \mathsf{ag} \in F_\mathsf{a} \end{cases}$$

$m\mathcal{A}^\rho$ Properties The newly introduced transition function allowed us to reason about fundamental properties that, as said in 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 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 actions effects; and
- an agent who is oblivious of the action occurrence will also be ignorant about its effects.

These propositions fully capture the concept of beliefs update and ensure that, when satisfied, the action language can be soundly used for multi-agent epistemic reasoning. Due to space limits we will only list these properties without presenting their proofs that can be found in the Supplementary Documents (available upon request).

In the following we will use p' instead of $\Phi(\mathsf{a}, \mathsf{p})$ when possible to avoid unnecessary clutter.

Proposition 1 (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* $\mathsf{x} \in F_\mathsf{a}$, *if* $\mathsf{u} \models \mathbf{B}_\mathsf{x}\psi$ *then* $\mathsf{u}' \models \mathbf{B}_\mathsf{x} l$;
2. *for every agent* $\mathsf{y} \in O_\mathsf{a}$ *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* $\mathsf{x} \in F_\mathsf{a}$ *and* $\mathsf{y} \in O_\mathsf{a}$ *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 (Sensing Action Properties) *Assume that* a *is a* sensing action instance *and* D *contains the statement* a ***determines*** f. *In* $m\mathcal{A}^\rho$ *it holds that:*

1. *if* $\mathsf{u} \models \mathsf{f}$ *then* $\mathsf{u}' \models \mathbf{C}_{F_\mathsf{a}}\mathsf{f}$;
2. *if* $\mathsf{u} \models \neg\mathsf{f}$ *then* $\mathsf{u}' \models \mathbf{C}_{F_\mathsf{a}}\neg\mathsf{f}$;
3. $\mathsf{u}' \models \mathbf{C}_{P_\mathsf{a}}(\mathbf{C}_{F_\mathsf{a}}\mathsf{f} \vee \mathbf{C}_{F_\mathsf{a}}\neg\mathsf{f})$;
4. $\mathsf{u}' \models \mathbf{C}_{F_\mathsf{a}}(\mathbf{C}_{P_\mathsf{a}}(\mathbf{C}_{F_\mathsf{a}}\mathsf{f} \vee \mathbf{C}_{F_\mathsf{a}}\neg\mathsf{f}))$;
5. *for every agent* $\mathsf{y} \in O_\mathsf{a}$ *and a belief formula* φ, $\mathsf{u}' \models \mathbf{B}_\mathsf{y}\varphi$ *iff* $\mathsf{u} \models \mathbf{B}_\mathsf{y}\varphi$; *and*
6. *for every pair of agents* $\mathsf{x} \in F_\mathsf{a}$ *and* $\mathsf{y} \in O_\mathsf{a}$ *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 3 (Announcement Action Properties) *Assume that* a *is a* announcement action instance *and* D *contains the statement* a ***announces*** φ. *If* $\mathsf{u} \models \phi$ *in* $m\mathcal{A}^\rho$ *it holds that:*

1. $\mathsf{u}' \models \mathbf{C}_{F_\mathsf{a}}\phi$;
2. $\mathsf{u}' \models \mathbf{C}_{P_\mathsf{a}}(\mathbf{C}_{F_\mathsf{a}}\phi \vee \mathbf{C}_{F_\mathsf{a}}\neg\phi)$;
3. $\mathsf{u}' \models \mathbf{C}_{F_\mathsf{a}}(\mathbf{C}_{P_\mathsf{a}}(\mathbf{C}_{F_\mathsf{a}}\phi \vee \mathbf{C}_{F_\mathsf{a}}\neg\phi))$;
4. *for every agent* $\mathsf{y} \in O_\mathsf{a}$ *and a belief formula* φ, $\mathsf{u}' \models \mathbf{B}_\mathsf{y}\varphi$ *iff* $\mathsf{u} \models \mathbf{B}_\mathsf{y}\varphi$; *and*
5. *for every pair of agents* $\mathsf{x} \in F_\mathsf{a}$ *and* $\mathsf{y} \in O_\mathsf{a}$ *and a belief formula* φ, *if* $\mathsf{u} \models \mathbf{B}_\mathsf{x}\mathbf{B}_\mathsf{y}\varphi$ *then* $\mathsf{u}' \models \mathbf{B}_\mathsf{x}\mathbf{B}_\mathsf{y}\varphi$.

In Baral et al. (2015) is shown how the above listed properties capture the concept of update in epistemic environment. Therefore, we consider two epistemic action languages that respect all of the above mentioned properties correct w.r.t. the knowledge/belief update. That is the case with $m\mathcal{A}^*$ and $m\mathcal{A}^\rho$.

Epistemic Forward Planner

Along with the new transition function formalization in this work we also present an updated version of the epistemic planner EFP (Le et al. 2018), called EFP 2.0[4]. This new solver redesigned every element of EFP 1.0 to introduce multiple e-states representations and, therefore, multiple transition functions. On the other hand, our implementation keeps the same modular structure of EFP 1.0.

We will now introduce how EFP 2.0 works and how it differs from its predecessor.

EFP 2.0 Structure

The planning process executed by EFP 2.0 is a *breadth-first search* with duplicate checking. Let us note that the computation of the initial state is not a trivial task in MEP. In particular, given a belief formula φ_i it is, in general, possible to generate infinite e-states that respect φ_i. To overcome this problem EFP 1.0 imposes that the initial state description should be a *finitary **S5**-theory* (Son et al. 2014). In EFP 2.0 we still require the initial description to be a finitary **S5**-theory but we allow φ_i to be less specific. In particular, without going into details of finitary **S5**-theories, whenever a fluent f is not considered by φ_i, EFP 2.0 assumes that is common knowledge between all the agents that f is unknown.

Another remark that has to be done is about the e-states. EFP 2.0 has a "templatic" e-state definition. This means that each solving process can be executed using the desired e-state representation with its relative transition function. Currently EFP 2.0 implements two e-states representations, *i.e.*, Kripke structures and possibilities, and two transition functions: i) the one introduced in Baral et al. (2015) (for Kripke structures); ii) the transition function for possibilities introduced above. Another important concept that EFP 2.0 integrates is the Kripke structures size reduction. We, in fact, implemented two algorithms (Paige and Tarjan 1987; Dovier, Piazza, and Policriti 2004) that starting from a generic Kripke structure compute its bisimilar, and therefore semantically equivalent, correspondent with minimal size.

Finally EFP 2.0 introduces the concept of "*already visited e-state*". Excluding the already visited states during the planning is a common practice and it is done in the majority of the solving processes. Nevertheless, EFP 1.0 did not implement the visited states comparison. That is because comparing two e-states is not as trivial as comparing, for instance, two sets of fluents. In fact, being each e-state in $m\mathcal{A}^*$ a Kripke structure, comparing two e-states means to check for *isomorphism* between them. Given the inherent complexity of the isomorphism algorithm the e-states visited check could, therefore, results in a even less efficient solving process. That is why in EFP 1.0 the comparison for already visited states was left as future development. On the other hand, with possibilities the equality check should be faster since, thanks to the non-well-foundeness, we can collapse each possibility in a small system of equations and exploit the already calculated possibilities information. That is why in EFP 2.0 we implemented the visited e-state check initially for possibilities and later for Kripke structures. As shown in Table 1, we found that all the solver's executions (with both possibilities and Kripke structures as e-states) were faster when the check was active.

As future work we also plan to exploit the *bisimulation* algorithm for the equality check on possibilities that, having faster implementations than isomorphism, *e.g*. (Dovier, Piazza, and Policriti 2004), should make EFP 2.0 even more efficient.

[4]Source code available upon request.

Experimental Evaluation

In this Section we compare the new multi-agent epistemic planner EFP 2.0 with, to the best of our knowledge, the only other comprehensive multi-agent epistemic solver in literature, *i.e.*, the planner presented in Le et al. (2018). All the experiments were performed on 3.60GHz Intel Core i7-4790 machine with 32GB of memory.

From now on, to avoid unnecessary clutter, we will make use of the following notations:

- L to indicate the (optimal) length of the plan;
- `WP` to indicate that the solving process returned an Wrong Plan;
- `TO` to indicate that the solving process did not return any solution before the timeout (25 minutes);
- EFP 1.0 to denote the Breadth-First search planner presented in Le et al. (2018). We chose the Breadth-First solver because we wanted to focus on the base of the solving process so that all the future optimizations could benefit from this research.
- K-MAL to identify our solver while using Kripke structures as e-state representation and the transition function of Baral et al. (2015).
- K-BIS to identify our solver while using Kripke structures as e-state representation and the algorithm to find the coarsest refinement, presented in Paige and Tarjan (1987), to minimize the e-states size. We also tried to compact the e-states using the algorithm presented in Dovier, Piazza, and Policriti (2004) but the performances were almost identical. This is probably because the Kripke structures we are considering are relatively small in size.
- P-MAR to identify our solver while using possibilities as e-state with the transition function introduced in the previous Section.

All the configurations K-MAL, K-BIS, and P-MAR check for already visited states. To indicate the same configurations without the visited states check we will use K-MAL-`NV`, K-BIS-`NV`, and P-MAR-`NV`.

We evaluate EFP 2.0 on benchmarks collected from the literature (Kominis and Geffner 2015; Huang et al. 2017). In particular, these domains are:

Grapevine										
$\lvert\mathcal{AG}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{A}\rvert$	L	EFP 1.0	K-MAL-NV	K-MAL	K-BIS-NV	K-BIS	P-MAR-NV	P-MAR
3	9	24	2	WP	.09	.09	.19	.20	.03	**.02**
			4	WP	9.19	8.13	13.54	12.76	1.34	**1.25**
			5	WP	94.49	75.32	111.38	84.46	8.67	**7.71**
			6	WP	372.64	278.93	398.10	232.54	27.63	**20.26**
4	12	40	2	WP	1.85	1.786	1.95	2.08	**.17**	.18
			4	WP	403.11	274.53	178.52	111.38	13.49	**7.31**
			5	WP	TO	TO	TO	775.63	123.54	**36.54**
			6	WP	TO	TO	TO	TO	427.97	**108.64**

Table 1: Runtimes for the Grapevine domain. We compare the configurations with and without the visited e-states check. EFP 1.0 errors are caused by a wrong initial e-state generation.

1. *Collaboration and Communication* (**CC**). In this domains $n \geq 2$ agents move along a corridor with $k \geq 2$ rooms in which $m \geq 1$ boxes can be located. Whenever an agent enters a room, she can determine if a certain box is in the room. Moreover, agents can communicate information about the boxes' position to the another *attentive* agents. The goals consider agents' positions and their beliefs about the boxes (Table 2).
2. *Selective Communication* (**SC**). **SC** has $n \geq 2$ agents that start in one of the $k \geq 2$ rooms in a corridor. An agent can tell some information and all the agents in her room or the neighboring ones can hear what was told. Every agent is free to move from one room to its adjacent. The goals usually require some agents to know certain properties while other agents ignore them (Figure 3).

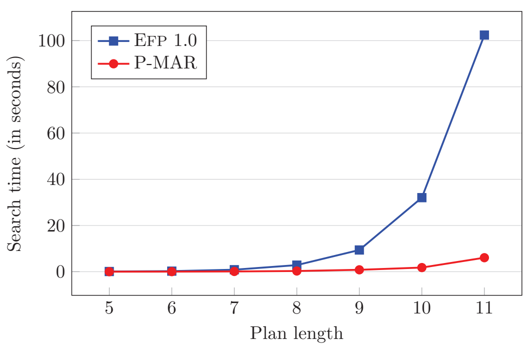


Figure 3: Comparison between EFP 1.0 and P-MAR on **SC** instances with $k = 11$ rooms and $|\mathcal{AG}| = 9$.

3. *Grapevine*. $n \geq 2$ agents are located in $k \geq 2$ rooms. An agent can move freely to each other room and she can share a "secret" with the agents that are in the room with her. This domain supports different goals, from sharing secrets with other agents to having misconceptions about agents' beliefs (Table 1).
4. *Coin in the Box* (**CB**). $n \geq 3$ agents are in a room where in the middle there is a box containing a coin. None of the agents know whether the coin lies heads or tails up and the box is locked. One agent has the key to open the box. The goals usually consist in some agents knowing whether the coin lies heads or tails up while other agents know that she knows or are ignorant about this (Table 3).
5. *Assembly Line* (**AL**). In this problem there are two agents, each responsible for processing a different part of a product. Each agent can fail in processing her part and can inform the other agent of the status of her task. Two agents decide to *assemble* the product or *restart*, depending on their knowledge about the product status. The goal in this domain is fixed, *i.e.*, the agents must assemble the product, but what varies is the *depth* of the belief formulae used as executability conditions (Table 4).

All our experiments (Tables 1–4, Figure 3) show that EFP 2.0, if used with its fastest configuration P-MAR, performs significantly better than EFP 1.0. We believe that these results derive from several factors.

First and foremost the choice of using possibilities as e-states and $m\mathcal{A}^\rho$ as action language ensured that every e-state generated during the planning process had always smaller or equal size w.r.t. the same state generated in EFP 1.0. In particular, EFP 1.0, generating e-states with non-minimal size, introduces extra (always increasing) overhead at each action application w.r.t. EFP 2.0. Moreover the implementation of P-MAR exploits already calculated e-states information when it creates new ones reducing even more the e-states generation time. From our results it is clear that EFP 1.0 and P-MAR perform similarly on very small instances of the problems but as soon as the problem grows the two solvers have different behaviors. In fact, while EFP 1.0 search time increases very rapidly P-MAR stays relatively stable. That is because when the problems become more complex the planner, generally, has to generate more e-states. Regarding the other configurations of EFP 2.0, namely K-MAL and K-BIS, we note that they generally outperforms EFP 1.0. Nevertheless, in some cases (Tables 2 and 4), we note some exceptional peeks in these configuration's performances. These peeks are the results of: i) the use of the visited-state check that in some configurations may add an extra overhead that in EFP 1.0 was not present; and ii) a less optimized *entailment-check* function, w.r.t. EFP 1.0, in the configurations of EFP 2.0 that are based on Kripke structures. A remark has to be done on the K-BIS configuration. From the results (Tables 1–4) it is clear how this configuration, even if ex-

L	EFP 1.0	K-MAL	K-BIS	P-MAR	EFP 1.0	K-MAL	K-BIS	P-MAR
	CC_1: $\|\mathcal{AG}\| = 2$, $\|\mathcal{F}\| = 10$, $\|\mathcal{A}\| = 16$				**CC_3**: $\|\mathcal{AG}\| = 3$, $\|\mathcal{F}\| = 14$, $\|\mathcal{A}\| = 24$			
3	.08	.05	.08	**.02**	.12	.07	.13	**.03**
4	.16	.09	.16	**.03**	.56	.31	.54	**.10**
5	1.31	.79	1.14	**.16**	6.55	3.25	4.89	**.60**
6	6.99	3.58	4.42	**0.64**	25.11	9.09	12.66	**1.71**
7	49.44	15.95	16.06	**2.61**	TO	92.37	142.06	**12.37**
	CC_2: $\|\mathcal{AG}\| = 2$, $\|\mathcal{F}\| = 14$, $\|\mathcal{A}\| = 28$				**CC_4**: $\|\mathcal{AG}\| = 3$, $\|\mathcal{F}\| = 14$, $\|\mathcal{A}\| = 42$			
3	.31	.21	.37	**.07**	.62	.54	.81	**.15**
4	1.54	.98	1.77	**.26**	3.22	2.84	5.40	**.87**
5	22.14	12.55	18.80	**1.68**	104.97	106.02	152.38	**7.41**
6	171.19	72.92	102.97	**7.71**	473.03	246.08	313.70	**25.47**
7	TO	437.91	592.48	**38.81**	TO	TO	TO	**174.67**

Table 2: Runtimes for the Collaboration and Communication domain.

CB with $\|\mathcal{AG}\| = 3$, $\|\mathcal{F}\| = 8$, $\|\mathcal{A}\| = 21$				
L	EFP 1.0	K-MAL	K-BIS	P-MAR
2	.003	.003	.006	**.001**
3	.048	.077	.097	**.016**
5	WP	5.546	1.438	**.367**
6	WP	108.080	14.625	**2.932**
7	WP	317.077	38.265	**6.996**

Table 3: Runtimes for the Coin in the Box domain.

AL with $\|\mathcal{AG}\| = 2$, $\|\mathcal{F}\| = 4$, $\|\mathcal{A}\| = 6$				
d	EFP 1.0	K-MAL	K-BIS	P-MAR
2	.43	.32	.42	**.07**
4	.96	.75	.64	**.11**
6	26.20	27.85	13.51	**2.44**
8	TO	TO	883.87	**150.92**
C	.44	.32	.43	**.08**

Table 4: Runtimes for the Assembly Line domain. The last row identify the instance where the executability conditions are expressed through common knowledge.

ecutes the solving process on minimal sized e-states, it is still outperformed by P-MAR. The reasons to this are essentially two: i) thanks to their non-well-founded nature possibilities allow to re-use already generated information during the planning process; and ii) the use of external algorithms to minimize the size of the e-states introduces an extra overhead w.r.t. P-MAR.

Another important factor that makes EFP 2.0 faster than EFP 1.0 is the concept of visited e-states. As we can see in Table 1 the planner takes advantage from this check even when the e-states are represented as Kripke structures. The fact that the visited-state check increases the performances of EFP 2.0 proves that, even if this check relies on 'heavy' algorithms, the epistemic planning process benefits from the duplicates elimination.

Finally, the complete refactoring of the code helped us to implement a more efficient solver. In fact, even if EFP 2.0 is based on EFP 1.0, the remodeling of the solver allowed us: i) to correct bugs related to the initial e-state generation (Table 1) and to the transition function (Table 3); and ii) to optimize the code. This optimization is reflected by the comparison between K-MAL and EFP 1.0. In fact, these two configurations both use Kripke structures as e-states and implements $m\mathcal{A}^*$ (Baral et al. 2015). Nevertheless K-MAL generally outperforms EFP 1.0 as shown in Table 2.

Conclusion and Future Works

In this paper we introduced an updated formalization of the transition function for the multi-agent epistemic action language $m\mathcal{A}^\rho$. In particular, the newly introduced transition function allowed us to prove some desirable properties of $m\mathcal{A}^\rho$ and helped us in deriving a clean and efficient implementation for a comprehensive epistemic planner based on possibilities, namely EFP 2.0.

We also provided some experimental results by comparing the state-of-the-art comprehensive epistemic planner EFP 1.0 (Le et al. 2018) with EFP 2.0 on benchmarks collected from the literature. As shown in the reported tables, the employment of possibilities as representation for epistemic state achieves better results, especially when coupled with the visited-state check. By conducting an analysis on the algorithms we believe that this higher efficiency is due to i) the employment of the dynamic programming paradigm that allowed us to re-use already calculated information about e-states; and ii) the reduced size of the e-states w.r.t. EFP 1.0. Finally the complete refactoring of the solver presented in Le et al. (2018) allowed to apply corrections and optimizations to the original solving process.

An immediate development to this work will be the implementation of a visited-state check based on *bisimulation* to reduce even more the planning times. Moreover, as future continuations to this work, we plan to: i) consider other alternatives to Kripke structures and possibilities; ii) formalize the concept of *non-consistent* belief in $m\mathcal{A}^\rho$; iii) implement the notion of *distributed knowledge* in EFP 2.0; iv) introduce the concept of epistemic static laws; v) further investigate the connection between Kripke structures and non-well–founded sets. vi) derive heuristics, as in Le et al. (2018), to prune the search space; and vii) consider symbolic e-states representations such as BDDs.

Acknowledgments

This research is partially supported by: the NSF HRD 1914635 and NSF HRD 1345232 projects, the University of Udine PRID ENCASE project, and the GNCS-INdAM 2017–2020 projects.

References

Aczel, P. 1988. Non-well-founded sets. CSLI Lecture Notes, 14.

Baral, C.; Gelfond, G.; Pontelli, E.; and Son, T. C. 2015. An action language for multi-agent domains: Foundations. *CoRR* abs/1511.01960.

Dovier, A.; Piazza, C.; and Policriti, A. 2004. An efficient algorithm for computing bisimulation equivalence. *Theoretical Computer Science* 311(1-3):221–256.

Fabiano, F.; Riouak, I.; Dovier, A.; and Pontelli, E. 2019. Non-well-founded set based multi-agent epistemic action language. In *Proceedings of the 34th Italian Conference on Computational Logic*, volume 2396 of *CEUR Workshop Proceedings*, 242–259.

Fabiano, F. 2019. Design of a solver for multi-agent epistemic planning. In *Proceedings 35th International Conference on Logic Programming (Technical Communications), ICLP 2019 Technical Communications, Las Cruces, NM, USA, September 20-25, 2019*, 403–412.

Fagin, R., and Halpern, J. Y. 1994. Reasoning about knowledge and probability. *Journal of the ACM (JACM)* 41(2):340–367.

Gerbrandy, J., and Groeneveld, W. 1997. Reasoning about information change. *Journal of Logic, Language and Information* 6(2):147–169.

Gerbrandy, J. 1999. *Bisimulations on planet Kripke*. Inst. for Logic, Language and Computation, Univ. van Amsterdam.

Hu, G.; Miller, T.; and Lipovetzky, N. 2019. What you get is what you see: Decomposing epistemic planning using functional strips.

Huang, X.; Fang, B.; Wan, H.; and Liu, Y. 2017. A general multi-agent epistemic planner based on higher-order belief change. In *IJCAI International Joint Conference on Artificial Intelligence*, 1093–1101.

Kominis, F., and Geffner, H. 2015. Beliefs in multiagent planning: From one agent to many. In *Proceedings of the International Conference on Automated Planning and Scheduling, ICAPS*, 147–155.

Kripke, S. A. 1963. Semantical considerations on modal logic. *Acta Philosophica Fennica* 16(1963):83–94.

Kuter, U.; Nau, D. S.; Reisner, E.; and Goldman, R. P. 2008. Using classical planners to solve nondeterministic planning problems. In Rintanen, J.; Nebel, B.; Beck, J. C.; and Hansen, E. A., eds., *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, 190–197. AAAI.

Le, T.; Fabiano, F.; Son, T. C.; and Pontelli, E. 2018. EFP and PG-EFP: Epistemic forward search planners in multiagent domains. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling*, 161–170. Delft, The Netherlands: AAAI Press.

McCarthy, J. 1959. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, 75–91.

Muise, C. J.; Belle, V.; Felli, P.; McIlraith, S. A.; Miller, T.; Pearce, A. R.; and Sonenberg, L. 2015. Planning over multiagent epistemic states: A classical planning approach. In *Proc. of AAAI*, 3327–3334.

Paige, R., and Tarjan, R. E. 1987. Three partition refinement algorithms. *SIAM Journal on Computing* 16(6):973–989.

Son, T. C.; Pontelli, E.; Baral, C.; and Gelfond, G. 2014. Finitary s5-theories. In *European Workshop on Logics in Artificial Intelligence*, 239–252. Springer.

Torreño, A.; Onaindia, E.; and Sapena, Ó. 2014. Fmap: Distributed cooperative multi-agent planning. *Applied Intelligence* 41(2):606–626.

Van Ditmarsch, H.; van Der Hoek, W.; and Kooi, B. 2007. *Dynamic epistemic logic*, volume 337. Springer Science & Business Media.

Wan, H.; Yang, R.; Fang, L.; Liu, Y.; and Xu, H. 2015. A complete epistemic planner without the epistemic closed world assumption. In *IJCAI International Joint Conference on Artificial Intelligence*, 3257–3263.